Fachbereich Informatik und Medien

Masterarbeit

Kontext-spezifische Analyse von Benutzerpräferenzen mittels Clustering für Musikempfehlungen auf Grundlage von Semantic-Web-Metadaten

Vorgelegt von: Christian Freye

am: 10.08.2011

zur

Erlangung des akademischen Grades

MASTER OF SCIENCE

(M.Sc.)

Betreuer: Prof. Dr.-Ing. Jochen Heinsohn

Zweitbetreuer: Dipl.-Inf. Rafael Schirru

Dipl.-Inf. Ingo Boersch

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit zum Thema

Kontext-spezifische Analyse von Benutzerpräferenzen mittels Clustering für Musikempfehlungen auf Grundlage von Semantic-Web-Metadaten

vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg/Havel, den 10.08.2011

Unterschrift

Zusammenfassung

In dieser Masterarbeit wird ein Verfahren vorgestellt, mit dem Benutzerprofile extrahiert werden können, die als Basis für Musikempfehlungen genutzt werden können. Um dieses Ziel zu erreichen werden die Künstler, die ein Nutzer gehört hat, mit Metadaten beschrieben, die im Semantic-Web gefunden werden können. Nach der Vorverarbeitung der Daten beginnt das Clustering der Künstler. Dadurch werden sie anhand ihres Musikstils in unterschiedliche Cluster unterteilt. Für jedes dieser Cluster wird dann ein Label gewählt, mit dem die zugrunde liegende Musikrichtung beschrieben werden kann. Diese Labels formen dann ein Benutzerprofil, welches ein breites Spektrum des Musikgeschmacks eines Nutzers widerspiegelt. Durch die Evaluation kann gezeigt werden, dass die extrahierten Benutzerprofile spezifisch für die unterschiedlichen Musikrichtungen der Nutzer sind. Somit können sie als Basis für Empfehlungssysteme dienen, oder in bereits bestehende Anwendungen integriert werden.

Abstract

This master thesis presents an approach extracting user profiles, which can be used for music recommendations. To achieve this goal, the artists, a user listened to, are described by meta data obtained from Semantic Web data sources. After preprocessing the data, the artists are clustered. For each of these clusters a label is extracted, which descibes the artists in the clusters in terms of their associated music styles. These labels represent the user profiles representing a broad range of a user's preferred music styles. The evaluation shows, that these extracted profiles are specific for the different music styles of a user. So they can serve as the basis for recommender systems or can be integrated into existing applications.

Inhaltsverzeichnis

1 Einleitung

Im Bereich der Musikempfehlungen arbeiten die meisten der heutigen Empfehlungssysteme entweder inhaltsbasiert, z.B. mit extrahierten Audio-Features oder händisch generierten Metadaten, kollaborativ, d.h. mittels Gemeinsamkeiten der Nutzer untereinander oder in einer Kombination von inhaltsbasierten und kollaborativen Verfahren. Diese Empfehlungssysteme sind dafür bekannt an einer Reihe von Schwächen zu leiden, wie z.B. der Überspezialisierung der Empfehlungen, da nicht das komplette Spektrum der Interessen eines Nutzers berücksichtigt wird ([BS01],[ZH09]). Das Semantic Web bietet dabei eine neue Möglichkeit Informationen über Künstler automatisiert abzurufen und diese dadurch beschreiben zu können. Anhand dieser Informationsgewinnung kann es möglich werden Schwächen bisheriger Empfehlungssysteme, wie z.B. die geringe Anzahl an Empfehlungen neuer Künstler, zu überwinden.

Ein zweites Novum der vorliegenden Arbeit ist die Modellierung von Benutzerprofilen unter Berücksichtigung unterschiedlicher Benutzerkontexte (z.B. ein Nutzer hört sowohl House als auch klassische Rockmusik). Es wird in dieser Arbeit gezeigt, dass Nutzer eine Vielzahl unterschiedlicher Musikrichtungen bevorzugen, die bei der Generierung von Musikempfehlungen berücksichtigt werden sollten. Diese unterschiedlichen Musikrichtungen werden extrahiert und in Form von Benutzerprofilen dargestellt. Diese Profile können dann Grundlage neuer Empfehlungssysteme bilden, oder in bereits existierende Empfehlungssysteme integriert werden.

1.1 Problem- und Zielstellung

Das Problem, welches im Zuge dieser Masterarbeit gelöst werden soll, kann im Bereich des Data-Mining angesiedelt werden. Das Hauptproblem besteht darin, dass aus einer Menge an Künstlern, die ein Nutzer zuvor gehört hat, ein Benutzerprofil erstellt werden soll, das die

Musikpräferenzen des Nutzers enthält.

Dieses Problem lässt sich in die folgenden kleineren Probleme zerlegen:

- Sammeln von Informationen von Semantic-Web-Diensten, mit deren Hilfe die Künstler beschrieben werden können,
- Clustern der Künstler anhand dieser Beschreibungen,
- Extrahieren von relevanten Merkmalen aus den Clustern.

Dies bedeutet, dass eine Abbildung gefunden werden soll, mit der es möglich ist eine Mengen von Künstlern durch ein Label zu beschreiben. Dieses Label soll dabei Informationen enthalten, mit denen es möglich ist die Musikrichtung zu identifizieren, mit der die Künstler assoziiert werden. Die Güte dieser Labels, soll dann mit Hilfe einer Evaluation bewertet werden.

1.2 Vorgehen

Zu Beginn des Verfahrens werden Meta-Daten zu einer großen Anzahl an Künstlern von den Semantic-Web-Diensten Freebase und Musicbrainz gesammelt. Diese werden für die spätere Verarbeitung in einer Datenbank abgelegt.

Im zweiten Schritt folgt das entwickelte Verfahren, welches pro Benutzer die Extraktion der Benutzerprofile vornimmt. Dazu werden die Playlisten von Last.fm-Nutzern geladen und alle Künstler extrahiert, die ein Nutzer gehört hat. Diese Künstler werden dann mit den Meta-Daten angereichert, welche sich in der Datenbank über sie finden lassen. Dadurch werden die Künstler und deren Eigenschaften in einer sogenannten Artist-Property-Matrix zusammengefasst, d.h. es erfolgt eine binäre Verknüpfung zwischen Künstlern und Eigenschaften.

Es folgt eine Datenvorverarbeitung, wobei die Matrix anhand der Häufigkeitsverteilung der Attribute gefiltert wird. Es werden somit alle Attribute entfernt, die eine bestimmte Anzahl an Vorkommen unterschreiten. Anschließend werden die verbleibenden Einträge in der Matrix gewichtet und dadurch für das Clustering vorbereitet. Die Künstler in der Matrix werden nun mittels k-Means in Cluster unterteilt. Das Clustering wird dabei mit verschiedenen Clusteranzahlen durchgeführt. Anschließend wird mittels RSS (engl. Residual Sum of Squares) die Clusteranzahl ausgewählt, mit der die Menge der Künstler geeignet

repräsentiert werden kann. Innerhalb der einzelnen Cluster erfolgt nun eine Bestimmung der Relevanz der enthaltenen Attribute. Entsprechend der Größe der Relevanz der einzelnen Terme erfolgt die Auswahl als beschreibendes Label für ein Cluster. Dadurch ist die Extraktion der Benutzerprofile abgeschlossen, da diese sich aus den einzelnen Labels für die entsprechenden Cluster zusammensetzen.

Der letzte Schritt befasst sich mit der Evaluation der Ergebnisse. Dazu werden die Messgrößen Precision, Recall und F-Maß der einzelnen Labels bestimmt. Somit kann festgestellt werden, ob die Labels spezifisch genug sind, um die in den Clustern enthaltenen Musikrichtungen zu beschreiben. Des Weiteren wird überprüft, für wie viele Nutzer sinnvolle Cluster gefunden werden können. Zusätzlich dazu werden stichprobenartig Nutzer ausgewählt und deren Labels dargestellt. Damit soll gezeigt werden, ob sich mit Hilfe der Labels die unterschiedlichen Musikrichtungen eines Nutzers identifizieren lassen.

1.3 Aufbau der Arbeit

Der Aufbau der weiteren Arbeit unterteilt sich in fünf Kapitel. Im Folgenden wird ein Überblick über das Umfeld der Arbeit gegeben. In diesem Kapitel wird auf die grundlegenden Prinzipien der kollaborativen, inhaltsbasierten, sowie hybriden Ansätze für Empfehlungssysteme eingegangen. Des Weiteren befasst sich dieses Kapitel mit dem Aufbau des Semantic-Web und stellt in diesem Zusammenhang einige wichtige Technologien daraus vor. Es wird auf das Prinzip der Linked-Data eingegangen und die zwei Semantic-Web-Dienste Freebase und MusicBrainz vorgestellt, da diese zur Realisierung der Arbeit genutzt werden. Daran anschließend wird in diesem Zusammenhang auf verwandte Arbeiten auf dem Gebiet der Kontext-sensitiven Musikempfehlungen, sowie der Empfehlungssysteme, die auf Meta-Daten aus dem Semantic-Web basieren, eingegangen. In Kapitel 3 werden die zur Durchführung des Ansatzes notwendigen theoretischen Grundlagen vorgestellt. Zu Beginn des Kapitels werden dazu die verwendeten Daten vorgestellt und auf die Beschaffung der Meta-Daten von Freebase und MusicBrainz eingegangen. Das Kapitel setzt sich fort mit der Beschreibung der Datenvorverarbeitung, sowie des Clusterings und endet mit der Merkmalsauswahl aus den entsprechenden Clustern. Das daran anschließende Kapitel befasst sich mit der Implementierung und stellt Klassen und Methoden vor, die bei der Ausarbeitung entstanden sind. In dem Kapitel 5 wird dann die Evaluation der Ergebnisse vorgenommen. Auch hierbei werden anfangs die theoretischen Grundlagen beschrieben, die dafür notwendig sind. Es folgt die Vorstellung, sowie die Auswertung der Ergebnisse. Abgeschlossen wird diese Masterarbeit

mit dem Kapitel „Fazit&Ausblick“ in dem die Arbeit resümiert wird und deren Ergebnisse zusammengefasst werden. Des Weiteren werden weiterführende Ideen gegeben, wie die hier entstandenen Benutzerprofile für Musikempfehlungen genutzt werden können.

2 Umfeld

In diesem Kapitel wird das Umfeld, in dem die Arbeit entstanden ist, vorgestellt. Dazu werden in dem ersten Abschnitt die Ansätze der Empfehlungssysteme näher erläutert und auf die verschiedenen Umsetzungsmöglichkeiten eingegangen, wobei als Hauptquelle eine Zusammenfassung von Adomavicius et al. [AT05] genutzt wird. In dem zweiten Abschnitt wird die Technologie des Semantic-Web erklärt und es werden die Online-Dienste Freebase und MusicBrainz, welche bei der Durchführung genutzt werden, vorgestellt.

2.1 Empfehlungssysteme

Empfehlungssysteme haben sich seit der Mitte der 1990er Jahre zu einem wichtigen Forschungsgebiet entwickelt [AT05]. In dieser Zeit sind die ersten Veröffentlichungen zu dem Thema des *Kollaborativen Filterns* ([SM95], [HSRF95], [RIS+94]) erschienen. Obwohl in den letzten Jahren viele Ansätze auf diesem Gebiet erforscht wurden, ist das Interesse an diesem problemreichen Forschungsgebiet unverändert hoch. Dies liegt vor allem an der Fülle der praktischen Anwendungen, die einem Nutzer bei der Handhabung mit der stetig wachsenden Informationsflut helfen können und ihm personalisierte Empfehlungen, Inhalte oder Dienste liefern. Als Beispiele für derartige Anwendungen können hierbei die Empfehlungen von Büchern, CDs und anderen Waren bei Amazon[1] [LSY03], Filme bei Moviepilot[2] und Nachrichten bei Google News[3] genannt werden.

Die Ursprünge der Empfehlungssysteme lassen sich in der Kognitionswissenschaft [Ric79], der Approximationstheorie [Pow81], dem Information Retrival [Sal88], der Vorhersagetheorie [Arm01], sowie der Verwaltungswissenschaft [MS03] und dem Modellieren der Verbraucherwahl innerhalb des Marketings [LKM92] finden. Seit der Mitte der 1990er Jahre haben sich

[1] www.Amazon.com
[2] www.moviepilot.de
[3] news.google.de

die Empfehlungssysteme zu einem selbstständigen Forschungsgebiet entwickelt, als Forscher den Fokus auf Probleme legten, die sich explizit auf Bewertungssysteme begründeten. Das Problem der Empfehlungen kann somit auf ein Problem reduziert werden, bei dem es um die Vorhersage von Bewertungen von Gegenständen geht, die ein Nutzer noch nicht gesehen hat. Diese Vorhersagen basieren normalerweise auf Bewertungen des Nutzers, die er vorher für andere Objekte getätigt hat. Sobald diese Bewertungen vorhergesagt werden können, können dem Nutzer Empfehlungen für Objekte mit den höchsten Bewertungen ausgesprochen werden.

Formal lässt sich dieses Problem folgendermaßen beschreiben: Sei C die Menge aller Nutzer und S die Menge aller Objekte, die empfohlen werden können. Hierbei kann sowohl S, als auch C unter Umständen sehr groß werden. Eine Nutzenfunktion u, welche den Wert eines Objektes s für einen Nutzer c bewertet, kann somit durch $u : C \times S \to R$ definiert werden, wobei R einer geordneten Menge aus z.B. Realzahlen innerhalb eines Intervalls entspricht. Wie in Formel 2.1 zu sehen, kann nun für jeden Nutzer $c \in C$ ein Objekt $s' \in S$ gewählt werden, durch welches die Nutzenfunktion maximiert wird.

$$\forall c \in C, \quad s'_c = \arg\max_{s \in S} u(c, s) \tag{2.1}$$

Innerhalb von Empfehlungssystemen wird mittels der Nutzenfunktion u eine Bewertung repräsentiert, durch die geschätzt werden kann, wie sehr einem Nutzer ein bestimmtes Objekt gefallen würde. Als Beispiel sei hier eine Bewertungsskala von 1 bis 10 genannt, mit deren Hilfe der Nutzer einzelne Objekte bewerten kann. Ein Nutzer aus der Menge C kann hierbei durch ein Profil beschrieben werden. Dieses Profil kann im einfachsten Fall lediglich eine Nutzer-ID enthalten, jedoch können darin auch weitere Informationen, wie z.B. Alter, Herkunft, Einkommen, Familienstand, etc. gespeichert werden. Auf die gleiche Weise können die zu empfehlenden Objekte aus S beschrieben werden. Handelt es sich bei S z.B. um eine Menge von Musikstücken, so können diese Informationen wie Künstler, Genre, Erscheinungsjahr, etc. enthalten.

Das zentrale Problem bei Empfehlungssystemen ist, dass die Nutzenfunktion meistens nicht für den gesamten Raum $C \times S$ definiert ist, sondern lediglich für eine Untermenge. Somit muss u extrapoliert werden, dass sie für alle Elemente aus $C \times S$ gilt. Die Nutzenfunktionen in Empfehlungssystemen werden im Allgemeinen anhand von Bewertungen erstellt, die von einem Nutzer in der Vergangenheit getätigt wurden. Bei der Empfehlung von Musikstücken könnte beispielsweise eine Bewertungsmatrix genutzt werden, wie sie in Tabelle 2.1 zu sehen

	Michael Jackson	Metallica	Nena	Moby
Alice	4	2	1	⊘
Bob	5	3	⊘	5
Charlie	1	5	3	4
David	4	⊘	1	3

Tabelle 2.1: Beispiel für eine Bewertungsmatrix

ist. Hierbei wurden von den Nutzern vor dem eigentlichen Empfehlungsprozess Künstler auf einer Skala von 1 bis 5 bewertet, die bereits gehört wurden. Künstler, die noch nicht bewertet wurden, werden durch ⊘ gekennzeichnet.

Auf Basis dieser Bewertungen soll nun ein Empfehlungssystem die fehlenden Bewertungen für Künstler, welche mit ⊘ gekennzeichnet sind, vorhersagen können. Es handelt sich dabei um eine Extrapolationen von den Bekannten zu den unbekannten Bewertungen. Dieser Vorgang kann auf zwei verschiedenen Wegen vorgenommen werden. Die erste Möglichkeit befasst sich mit dem Erstellen von Heuristiken. Es wird dabei eine Nutzenfunktion definiert, deren Vorhersagegenauigkeit überprüft werden muss. Bei der zweiten Möglichkeit kann versucht werden ein Modell zu erlernen, wodurch die Nutzenfunktion vorhergesagt werden kann. Bei dem Lernvorgang wird dann versucht diverse Messwerte für die Performanz des Modells, wie z.B. den mittleren quadratischen Fehler, zu minimieren. Wenn die fehlenden Bewertungen bestimmt sind, können im Anschluss Empfehlungen für den Nutzer generiert werden, wobei das Element mit der höchsten Bewertung, bzw. die N Elemente mit den höchsten Bewertungen ausgewählt werden.

Die Einträge der Matrix, welche nicht bewertet sind, können auf verschiedene Weise geschätzt werden. Dabei werden Methoden des Maschinellen Lernens und der Approximationstheorie, sowie verschiedene Heuristiken genutzt. Empfehlungssysteme können nach der Art und Weise beurteilt werden, wie die Ermittlung der Bewertungen durchgeführt wird. Diese Beurteilung kann in drei Kategorien eingeteilt werden:

- *Inhaltsbasierte Empfehlungssysteme*: Dem Nutzer werden Dinge empfohlen, die ähnlich zu denen sind, die er bereits in der Vergangenheit bevorzugt hat.
- *Kollaborative Empfehlungssysteme*: Dem Nutzer werden Dinge empfohlen, die andere (ihm ähnliche) Nutzer in der Vergangenheit favorisiert haben.
- *Hybride Ansätze*: Diese Methoden kombinieren inhaltsbasierte und kollaborative Empfehlungssysteme.

2.1.1 Inhaltsbasierte Methoden

Diese Art der Empfehlungssysteme bedient sich der Bewertungen $s_i \in S$, die ein Nutzer c in der Vergangenheit abgegeben hat. Wenn für den Nutzer eine Bewertung für das Element s vorhergesagt werden soll, basiert die Nutzenfunktion $u(c, s)$ auf jenen Bewertungen s_i, welche ähnlich zu s sind. Bei einem Empfehlungssystem für Bücher, wird dazu versucht die Gemeinsamkeiten von Büchern zu finden, die ein Nutzer hoch bewertet hat. Dies kann anhand verschiedener Faktoren, wie z.B. Autor, Thema, Genre, etc. beurteilt werden. Demnach werden dem Nutzer nur Bücher empfohlen, welche einen hohen Grad an Ähnlichkeit zu Büchern besitzen, die ihm bereits in der Vergangenheit gefielen.

Der inhaltsbasierte Ansatz hat seinen Ursprung im Information Retrieval [Sal88], [BYRN99] und in der Informationsfilterung [BC92]. Die Fortschritte, welche in diesen Forschungsgebieten gemacht wurden und deren Bedeutung für diverse Text-basierende Anwendungen, führten dazu, dass viele der heutigen inhaltsbasierten Empfehlungssysteme auf die Empfehlung von Texten, wie z.B. Dokumente, Internetseiten oder Nachrichten spezialisiert sind. Eine Erweiterung der traditionellen Ansätze des Information Retrieval bietet dabei die Verwendung von Benutzerprofilen. Ein solches Profil enthält Informationen über die Vorlieben, Bedürfnisse und den Geschmack eines Nutzers und es kann explizit über Bewertungen oder implizit über das Verhalten des Nutzers erstellt werden.

Die Menge der Attribute, die ein Objekt beschreiben, kann formal mittels $Content(s)$ zusammengefasst werden. Diese Merkmale werden aus dem Objekt s extrahiert und spiegeln somit dessen Inhalt wider. Dieses Profil eines Objekts kann nun genutzt werden, um dessen Eignung innerhalb des Empfehlungsprozesses zu bestimmen. Da in vielen Fällen textbasierte Objekte empfohlen werden, besteht das Profil meistens aus einer Menge von Stichwörtern, welche aus einem Text extrahiert werden. Bei der Empfehlung von Büchern kann $Content(s)$ z.B. aus den 100 wichtigsten Wörtern bestehen, die in einem Buch vorkommen. Die Wichtigkeit eines Wortes k_i in einem Dokument d_j kann dabei mittels einer Gewichtung w_{ij} ermittelt werden, die auf unterschiedliche Weise definiert werden kann.

Eines der bekanntesten Verfahren zum Ermitteln von Gewichten für solche Stichwörter im Information Retrieval ist das TF-IDF-Maß, auf das in Kapitel 3 näher eingegangen wird. Somit lässt sich der Inhalt eines Dokuments d_j wie in Formel 2.2 als gewichteter Termvektor

darstellen.

$$Content(d_j) = (w_{1j}, w_{2j}, \ldots, w_{kj}) \tag{2.2}$$

Das Profil eines Nutzers c wird formal mittels $ContentBasedProfile(c)$ definiert und enthält Informationen über dessen Vorlieben. Der Inhalt eines solchen Profils wird durch die Analyse der Objekte, die ein Nutzer bereits kennt und bewertet hat, ermittelt. Beispielsweise kann das Profil als ein Vektor von Gewichten $(w_{c1}, \ldots, w_{ck})$ definiert werden, wobei jedes dieser Gewichte w_{ci} die Relevanz eines Stichwortes k_i für den Nutzer c repräsentiert. Zur Erstellung dieser Gewichte können verschiedene Techniken zum Einsatz kommen, wie z.B. der Bayes-Klassifikator in [PBMW97], welcher die Wahrscheinlichkeit schätzt, dass dem Nutzer ein Dokument gefällt. Eine weitere Methode, die zur Erstellung der Profile genutzt werden kann, ist der Rocchio-Algorithmus [Roc71], mit dessen Hilfe es möglich ist, die Relevanz von Stichwörtern für den Nutzer zu bestimmen.

Demnach lassen sich sowohl der Inhalt eines Dokuments $Content(s)$, als auch das Profil eines Nutzers $ContentBasedProfile(c)$ durch Vektoren $\vec{w_c}$ und $\vec{w_s}$ darstellen, welche beispielsweise die TF-IDF-Gewichte für bestimmte Stichwörter enthalten. Mittels dieser beiden Vektoren kann dann die Nutzenfunktion $u(c, s)$, wie in Formel 2.3 durch das Kosinus-Ähnlichkeitsmaß ([BYRN99], [Sal88]) errechnet werden. Dabei entspricht K der Gesamtanzahl an Stichwörtern innerhalb des Verfahrens. $\vec{w_c} \cdot \vec{w_s}$ entspricht dabei dem Skalarprodukt dieser beiden Vektoren. Durch $\|\vec{w_c}\|_2$ wird die Euklidischen Norm dargestellt, was der Länge des Vektors entspricht.

$$\begin{aligned} u(c, s) = cos(\vec{w_c}, \vec{w_s}) &= \frac{\vec{w_c} \cdot \vec{w_s}}{\|\vec{w_c}\|_2 \cdot \|\vec{w_s}\|_2} \\ &= \frac{\sum_{i=1}^{K} w_{i,c} w_{i,s}}{\sqrt{\sum_{i=1}^{K} |w_{i,c}|^2} \sqrt{\sum_{i=1}^{K} |w_{i,s}|^2}} \end{aligned} \tag{2.3}$$

Neben dem traditionellen Verfahren mittels dem Kosinus-Ähnlichkeitsmaß Empfehlungen zu generieren, finden auch Techniken aus dem Bereich des Maschinellen Lernens Anwendung. Dabei werden unter anderem Clustering-Verfahren, Entscheidungsbäume oder Neuronale Netze [PBMW97] eingesetzt. Der Unterschied bei diesen Ansätzen besteht darin, dass nicht eine Heuristik zur Anwendung kommt, mit deren Hilfe eine Ähnlichkeit zwischen Objekten bestimmt werden kann, sondern ein Modell, das aus den zugrunde liegenden Daten mittels

Klassifikationsalgorithmen erlernt wird.

Wie in [SM95], [BS97] zu sehen besitzen inhaltsbasierte Methoden einige Einschränkungen, auf die im nachfolgenden Abschnitt näher eingegangen wird.

Begrenzte Inhaltsanalyse

Eine Einschränkung bei inhaltsbasierten Empfehlungssystemen bieten die Merkmale, die explizit mit den zu verwendenden Objekten in Verbindung gebracht werden können. Die Merkmale können dafür manuell mit einem Objekt in Verbindung gebracht werden oder sie werden automatisch durch die Analyse dessen Inhalts, wie es z.B. bei Text üblich ist, hinzugefügt. Bei anderer Medien, wie z.B. Bild, Ton, Video, etc. ist es zwar möglich automatisch Merkmale, wie Lautheit bei Musik oder Helligkeit bei Bildern zu extrahieren, jedoch können über diese Merkmale keine Informationen über den Inhalt des Mediums gewonnen werden. Aus diesem Grund gestaltet sich die automatische Analyse dieser Medien schwierig und es ist selten praktikabel die Merkmale über den Inhalt per Hand hinzuzufügen [SM95].

Ein weiteres Problem ergibt sich mit der Repräsentation selbst, d.h. dass zwei unterschiedliche Objekte, welche die gleiche Menge an Merkmalen besitzen nicht unterschieden werden können [SM95]. Beispielsweise können Texte, welche die gleichen Schlüsselwörter enthalten, aber in einem unterschiedlichen Stil geschrieben sind, nicht voneinander unterschieden werden. Des Weiteren ist es nicht möglich anhand dieser Merkmale eine Aussage über die Qualität einer Sache zu treffen.

Überspezialisierung

Ein weiteres Problem bei inhaltsbasierten Empfehlungssystemen kann dadurch entstehen, dass einem Nutzer nur Dinge empfohlen werden, die ähnlich zu denen sind, die er bereits kennt. Dadurch gestaltet es sich schwierig, dass der Nutzer etwas Neues kennen lernt, von dem er vorher noch nicht wusste, dass es ihm gefallen könnte. In [SM93] wird im Kontext der Filterung von Informationen versucht durch genetische Algorithmen mittels Zufall die Wahl der Empfehlungen zu beeinflussen. Des Weiteren sollte vermieden werden, dem Nutzer Dinge zu empfehlen, die sehr ähnlich zu denen sind die er bereits kennt. Dies könnten z.B. Artikel in einem Online-Shop sein, die einem Nutzer zwar gefallen würden, von denen er aber bereits einen Ähnlichen besitzt. Dazu werden in [ZCM02] fünf Messgrößen vorgestellt,

mit deren Hilfe es möglich ist zu Ermitteln, ob ein relevantes Dokument neue Informationen für den Nutzer enthält. Im Allgemeinen ist eine gewisse Vielfalt bei den Empfehlungen ein erstrebenswertes Kriterium bei der Entwicklung von Empfehlungssystemen. Daher sollten dem Nutzer eine Reihe von verschiedenen Auswahlmöglichkeiten angeboten werden, anstatt ihm beispielsweise alle Bücher eines Autors zu empfehlen, nur weil er ein Buch von diesem Autor hoch bewertet hat.

Problem des neuen Nutzers

Da die Empfehlungen aufgrund der Bewertungen eines Nutzers generiert werden, ist es schwierig verlässliche Empfehlungen an Nutzer zu geben, die erst sehr wenig, bzw. gar keine Bewertungen abgegeben haben. Neue Benutzer eines Systems müssen daher erst eine Weile selbstständig aktiv sein oder müssen explizit nach ihrem Geschmack und ihren Vorlieben befragt werden.

2.1.2 Kollaborative Methoden

Im Gegensatz zu den inhaltsbasierten Empfehlungssystemen wird bei dem kollaborativen Filtern versucht Empfehlungen aufgrund von Bewertungen anderer Nutzer zu erstellen. Formal lässt sich dies mit $u(c, s)$ ausdrücken, welche auf Basis von $u(c_j, s)$ erstellt wird, wobei $c_j \in C$ Nutzer sind, die ähnlich zu c sind.

Das erste Empfehlungssystem, welches nach dieser Methode arbeitete, war das Grundy System [Ric79], bei dem die Nutzer in verschiedene Stereotypen zusammengefasst wurden. Mittels dieser Stereotypen konnten die einzelnen Nutzer in bestimmte Nutzergruppen unterteilt werden, womit es anschließend möglich war, ihnen Bücher zu empfehlen, die für sie von Interesse sein könnten. Die ersten Systeme, bei denen das kollaborative Filtern eingesetzt wurde um automatisch Empfehlungen zu generieren waren GroupLens [KMM+97], Video Recommender [HSRF95] und Ringo [SM95]. Die Algorithmen, welche für das kollaborative Filtern eingesetzt werden, lassen sich nach [BHK98] in zwei Klassen (*speicherbasiert*, *modellbasiert*) unterteilen.

Speicherbasierte Algorithmen

Es handelt sich hierbei um Heuristiken, welche die gesamte Datenbasis der zuvor bewerteten Objekte nutzen, um Vorhersagen für nicht bewertete Objekte zu treffen. Dabei setzt sich

eine unbekannte Bewertung $r_{c,s}$ von Nutzer c für Objekt s meist aus den Bewertungen der N ähnlichsten Nutzer zusammen. Dies ist in Formel 2.4 dargestellt, wobei $\hat{C}$ den N ähnlichsten Nutzern entspricht, die eine Bewertung für s abgegeben haben.

$$r_{c,s} = \underset{c' \in \hat{C}}{aggr}\ r_{c',s} \tag{2.4}$$

Einige Beispiele für solche Vereinigungen der Nutzerbewertungen sind in den Formeln 2.5a und 2.5b zu sehen. Die einfachste Methode die Bewertungen der Nutzer zusammenzufassen ist dabei in der Formel 2.5a dargestellt und entspricht dem arithmetischen Mittel. Die am häufigsten verwendete Variante ist in Formel 2.5b zu finden. Dabei werden die einzelnen Bewertungen der Nutzer gewichtet, je nach dem wie ähnlich sich c und c' sind, was mittels $sim(c, c')$ bestimmt werden kann. Eine Normalisierung erfolgt dabei über den Faktor k. Die Funktion $sim(c, c')$ bildet im allgemeinen ein Ähnlichkeitsmaß zwischen den Nutzern, d.h. je ähnlicher sich zwei Nutzer sind, desto mehr Gewicht bekommt die Bewertung $r_{c',s}$ bei der Bestimmung von $r_{c,s}$.

$$r_{c,s} = \frac{1}{N} \sum_{c' \in \hat{C}} r_{c',s} \tag{2.5a}$$

$$r_{c,s} = k \sum_{c' \in \hat{C}} sim(c, c') \cdot r_{c',s} \tag{2.5b}$$

In kollaborativen Empfehlungssystemen existieren verschiedene Ansätze mit deren Hilfe die Ähnlichkeit $sim(c, c')$ zwischen Nutzern bestimmt werden kann. In den meisten Fällen richtet sich die Ähnlichkeit zwischen Nutzern nach der Menge der gemeinsam bewerteten Objekte S_{xy} . Zur Bestimmung der Ähnlichkeit können verschiedene Ansätze herangezogen werden, wobei Korrelation und Kosinus-Ähnlichkeitsmaß zu den bekanntesten zählen. In Formel 2.6 ist der Pearson-Korrelationskoeffizient zu sehen, der ein Maß für den Grad des linearen Zusammenhangs zwischen zwei Nutzern darstellt [SM95], [RIS+94]. Das Ergebnis $sim(x, y)$ liegt dabei in dem Intervall $[-1, 1]$, wobei ein Wert von 1, bzw. -1 einen vollständigen positiven, bzw. negativen linearen Zusammenhang zwischen den Nutzern repräsentiert. Wenn der Korrelationskoeffizient einen Wert von 0 annimmt besteht zwischen den betrachteten Nutzern kein linearer Zusammenhang.

$$sim(x,y) = \frac{\sum\limits_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)(r_{y,s} - \bar{r}_y)}{\sqrt{\sum\limits_{s \in S_{xy}} (r_{x,s} - \bar{r}_x)^2 \sum\limits_{s \in S_{xy}} (r_{y,s} - \bar{r}_y)^2}} \tag{2.6}$$

Bei dem Kosinus-Ähnlichkeitsmaß werden die zwei Nutzer x und y als zwei Vektoren im m-dimensionalen Raum behandelt, wobei $m = |S_{xy}|$ entspricht. Die Ähnlichkeit zwischen diesen Vektoren kann nun bestimmt werden, indem der Kosinus des Winkels zwischen diesen Vektoren errechnet wird [BHK98]. Formal dargestellt wird dies in Formel 2.3, wobei $\vec{w_c}$ und $\vec{w_s}$ durch die Vektoren $\vec{x}$ und $\vec{y}$ ersetzt werden. K entspricht in diesem Fall der Anzahl der gemeinsam bewerteten Elemente $|S_{xy}|$.

Die Ähnlichkeiten zwischen den Nutzern werden nur bestimmt, wenn diese mindestens eine bestimmte Anzahl an gleichen Objekten bewertet haben. Da die Bewertungsmatrix in den meisten Fällen nur dünn besetzt ist, wurden einige Verbesserungen entwickelt, welche die Leistung der kollaborativen Empfehlungssysteme steigern sollen. Eine dieser Verbesserungen ist die Standardbewertung [BHK98], d.h. fehlende Bewertungen werden mit Standardwerten gefüllt, um so mehr gemeinsame Bewertungen zwischen Nutzer x und y zu schaffen. Die Ähnlichkeiten zwischen den Nutzern müssen dann bei der Generierung von Empfehlungen neu bestimmt werden, woraus sich ein Performanzproblem ergibt.

Modellbasierte Algorithmen

Modellbasierte Algorithmen bilden die zweite Art der Methoden für das kollaborative Filtern. Hierbei wird versucht anhand der Bewertungen ein Modell zu erlernen, womit anschließend Vorhersagen für Bewertungen getroffen werden können. In [BHK98] wird dazu ein Ansatz vorgestellt, mit dem die Wahrscheinlichkeiten für bestimmte Bewertungen (0, . . . , n) ermittelt werden können, indem die vorher abgegebenen Bewertungen für Objekte s' eines Nutzers c analysiert werden. Der Erwartungswert einer Bewertung für ein Objekt s ergibt sich dann aus folgender Formel:

$$r_{c,s} = E(r_{c,s}) = \sum_{i=0}^{n} i \cdot Pr(r_{c,s} = i | r_{c,s'}, s' \in S_c). \tag{2.7}$$

Für die Vorhersage der Wahrscheinlichkeiten können verschiedene Methoden des Data-Minings eingesetzt werden, wie z.B. Clusteringverfahren, Bayes'sche Netze, lineare Regres-

sion, Neuronale Netze, etc. Die Modellparameter für die verschiedenen Verfahren werden dann mittels Lernalgorithmen ermittelt, welche auf die zugrunde liegenden Daten angewandt werden.

Kollaborative Empfehlungssysteme generieren Empfehlungen für Nutzer aufgrund der Bewertungen anderer Nutzer, weshalb sie problemlos mit jeglicher Art von Medien und Objekten umgehen können. Sie enthalten nicht die typischen Probleme der inhaltsbasierten Empfehlungssysteme und können daher einem Nutzer z.B. Empfehlungen anbieten, welche komplett unterschiedlich zu allen Sachen sind, die er vorher bereits gesehen und bewertet hat. Aber auch hier existieren einige Probleme und Einschränkungen [BS97], auf die im folgenden Abschnitt eingegangen wird.

Problem des neuen Nutzers

Dieses Problem ist identisch mit dem der inhaltsbasierten Empfehlungssysteme. Um nützliche Empfehlungen generieren zu können müssen ähnliche Nutzer gefunden werden. Dies ist allerdings erst möglich, wenn eine bestimmte Anzahl an Bewertungen abgegeben wurde. Es existieren verschiedene Ansätze um diesem Problem entgegen zu wirken. Die meisten davon nutzen hybride Empfehlungssysteme, auf die im späteren Verlauf näher eingegangen wird.

Problem des neuen Objekts

Es ist sehr häufig der Fall, dass neue Objekte zu einem System hinzugefügt werden. Das Problem, das bei diesen Objekten besteht ist, dass sie erst empfohlen werden können, wenn sie von einer bestimmten Anzahl an Nutzern bewertet wurden. Auch bei diesem Problem können hybride Ansätze dabei helfen die Qualität der Bewertungen zu steigern.

Anzahl der Bewertungen

Es wurde bereits die dünn besetzte Bewertungsmatrix angesprochen und die daraus resultierende geringe Anzahl an Nutzerbewertungen. In den meisten Fällen liegt diese Anzahl weit unter der Anzahl an Bewertungen, die vorhergesagt werden sollen. Die Güte der Empfehlungen hängt demnach davon ab, wie gut es möglich ist von einer kleinen Menge an Beispielen auf eine große Menge an Bewertungen zu schließen. Dabei ist es ebenfalls wichtig, dass eine bestimmte Mindestanzahl an Nutzern vorhanden sein muss, damit Vorhersagen über Bewertungen getroffen werden können. Dennoch existieren immer Nutzer, deren Geschmack

komplett unterschiedlich im Vergleich zu anderen Nutzern ist. Für diese Nutzer ergibt sich eine schlechte Auswahl an ähnlichen Nutzern, was wiederum zu schlechten Empfehlungen führt [BS97].

Das demographische Filtern [Paz99] versucht dem entgegen zu wirken. Dazu werden Profilinformationen, wie z.B. Geschlecht, Alter, Herkunft, etc. in die Berechnung der Ähnlichkeiten einbezogen. Somit können Nutzer nicht nur ähnlich sein, wenn sie die gleichen Objekte bewertet haben, sondern auch, wenn sie aus dem gleichen demographischen Segment stammen. Ein anderer Ansatz wird in [SKKR00] vorgestellt, bei dem durch Singulärwertzerlegung die Dimensionalität der dünn besetzten Bewertungsmatrix reduziert wird.

2.1.3 Hybride Methoden

Bei den hybriden Methoden werden Ansätze unterschiedlicher Empfehlungssysteme kombiniert, um so den Problemen der jeweiligen Methoden entgegen zu wirken. Dabei lassen sich die verschiedenen Kombinationen in folgende Klassen unterteilen [AT05]:

1. kollaborative und inhaltsbasierte Methoden werden separat implementiert und ihre Ergebnisse kombiniert,
2. Besonderheiten von inhaltsbasierten Methoden werden in kollaborative Systeme integriert,
3. Besonderheiten von kollaborativen Methoden werden in inhaltsbasierte Systeme integriert,
4. es wird ein Modell erstellt, welches die Eigenschaften beider Methoden kombiniert.

Diese vier Kombinationsmöglichkeiten werden im Folgenden näher erläutert und mit Beispielen ihre Anwendungsmöglichkeiten gezeigt.

Kombination der Ergebnisse separater Empfehlungssysteme

Die Kombination der Ergebnisse kollaborativer und inhaltsbasierter Methoden kann auf zwei verschiedenen Arten durchgeführt werden. Zum Einen können die Ergebnisse jedes einzelnen Empfehlungssystems zu einer endgültigen Bewertung zusammengefasst werden. Dies kann entweder über eine lineare Kombination der Ergebnisse [CGM+99] oder über ein Auswahlschema [Paz99] erfolgen. Zum Anderen kann das Ergebnis einer Methode gewählt werden, das aufgrund eines Gütemaßes als das „Bessere“ erachtet wird. Dies wird

beispielsweise bei dem *DailyLearner* [BP00] erreicht, indem das Ergebnis gewählt wird, welches ein höheres Maß an Vertrauen besitzt. In [TC00] hingegen wird die Bewertung gewählt, welche eine höhere Konsistenz zu den vorherigen Bewertungen eines Nutzers besitzt.

Integration inhaltsbasierter Methoden in kollaborative Systeme

Dieser Ansatz für hybride Empfehlungssysteme basiert auf den traditionellen kollaborativen Methoden, welche jedoch mit Eigenschaften erweitert werden, die den inhaltsbasierten Empfehlungssystemen zuzuordnen sind. Dies kann z.B. die Einbindung von Benutzerprofilen sein [Paz99], mit deren Hilfe die Ähnlichkeiten zwischen den Nutzern bestimmt werden können und somit dem Problem der dünn besetzten Bewertungsmatrix entgegengewirkt wird.

Integration kollaborativer Methoden in inhaltsbasierte Systeme

Bei dieser Art der hybriden Empfehlungssysteme bildet die Reduzierung der Dimensionalität einer Gruppe von inhaltsbasierten Profilen den bekanntesten Ansatz. In [SN99] wird für diese Reduzierung das *Latent Semantic Indexing* (LSI) verwendet. Es handelt sich dabei um ein Verfahren, das Hauptkomponenten, bzw. thematische Gruppen in Dokumenten findet. Dadurch wird versucht eine kollaborative Sicht auf eine Menge von Nutzerprofilen zu erzeugen. Die Nutzerprofile werden dabei in Form von Termvektoren dargestellt und es wird gezeigt, dass dadurch eine Leistungssteigerung gegenüber den reinen inhaltsbasierten Systemen erzielt werden konnte.

Vereinigung der Methoden in einem Modell

Bei diesem Ansatz können verschiedene Methoden genutzt werden, um die Merkmale der kollaborativen und inhaltsbasierten Empfehlungssysteme zu kombinieren. In [BHC98] werden z.B. Alter und Geschlecht der Nutzer, sowie Genre von Filmen genutzt um daraus einen einzigen regelbasierten Klassifikator zu erstellen. Einen anderen Ansatz verfolgt Burke in [Bur00], indem er hybride Empfehlungssysteme mit wissensbasierten Techniken, wie dem fallbasierten Schließen, anreichert. Dadurch wird versucht einige Grenzen traditioneller Ansätze, wie das Problem des neuen Nutzers oder des neuen Objekts, zu überwinden und die Qualität der Vorhersagen zu steigern. Obwohl bei solchen wissensbasierten Techniken die Datenbeschaffung ein Problem darstellt, sind bisher viele Anwendungen entwickelt worden, für die Informationen in maschinenlesbarer Form, z.B. als Ontologien vorhanden sind. So

nutzt beispielsweise das wissensbasierte Empfehlungssystem Entrée [Bur00] Wissen über Restaurants, Küche und Essen, um für die Nutzer Empfehlungen zu generieren.

Die Leistung und Qualität von hybriden Systemen im Gegensatz zu den reinen kollaborativen oder inhaltsbasierten Ansätzen wurde in diversen Publikationen [BS97], [Paz99], [SN99] empirisch verglichen. Dabei wurde festgestellt, dass hybride Systeme in der Lage sind präzisere Empfehlungen zu generieren als traditionelle Methoden.

2.2 Semantic-Web

Das Konzept des Semantic-Web wird von Tim Berners-Lee et al. in [BLHL01] vorgestellt. Darin heißt es „The Semantic-Web is not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation.“ In diesem Zitat kommt in wenigen Worten der Kerngedanke des Semantic-Web zum Ausdruck. Das heutige World Wide Web (WWW) soll um Strukturen erweitert werden, mit denen es möglich ist Informationen mit maschinenlesbaren Bedeutungen anzureichern. Dadurch soll es Computerprogrammen ermöglicht werden, eine wichtigere Rolle im Web zu übernehmen und den Menschen z.B. durch eine semantische Suche bei der Arbeit im Web zu unterstützen. Sie sollen stärker in das Geschehen einbezogen und ihre analytischen Fähigkeiten bei der alltäglichen Informationsverarbeitung besser genutzt werden. Während es einem Menschen ohne Probleme möglich ist im WWW Informationen zu erfassen und zu interpretieren, so ist es für eine Maschine nicht ohne Weiteres möglich die Bedeutung (Semantik) der erfassten Informationen zu ermitteln. Dies liegt vorwiegend daran, dass das WWW sich sehr schnell als Medium entwickelt hat, dass auf die Benutzung durch den Menschen abzielt und nicht auf die automatische Verarbeitung durch Maschinen. Um Maschinen die Möglichkeit zu geben, die Semantik von Informationen zu erfassen, gibt es zwei Möglichkeiten: Zum Einen könnte man sie mit Methoden der künstlichen Intelligenz ausstatten, wodurch sie, ähnlich wie der Mensch, in der Lage wären, selbstständig die Bedeutung von Informationen zu erfassen. Zum Anderen könnte man die Informationen mit Metadaten anreichern, um den Maschinen die Interpretation zu erleichtern, bzw. abzunehmen. Obwohl die künstliche Intelligenz ein spannendes Forschungsgebiet darstellt, lassen die bisher erreichten Resultate darauf schließen, dass eine zuverlässige Anwendung im gesamten WWW zum jetzigen Zeitpunkt sehr unwahrscheinlich ist. Daher bedient sich die Idee des Semantic-Web der möglichst umfangreichen Anreicherung der bisherigen Informationen des WWW mit Metadaten, um sie so in eine Form zu bringen, die durch Maschinen „verstanden“

werden kann. Dieses „Verständnis“ bezieht sich nicht nur auf die Erfassung der Bedeutung der (Meta-)Daten, sondern soll vielmehr ausdrücken, dass die Maschinen dadurch in der Lage sind, aus den vorhandenen Daten Schlüsse zu ziehen und somit neue Informationen zu erschaffen.

Der theoretische Hintergrund des Semantic-Web orientiert sich an Modellen der Semiotik, der Wissensrepräsentation, der Graphentheorie, sowie der Logik [Not05]. Mit Hilfe dieser theoretischen Grundlagen ist es nun möglich einheitliche und offene Standards für die Beschreibung von Informationen zu vereinbaren, wodurch es ermöglicht werden kann, dass Informationen auch zwischen verschiedenen Plattformen und Anwendungen ausgetauscht und zueinander in Beziehung gesetzt werden können. Eine weitere Anforderung an diese Standards, neben der klaren formalen Definition, ist die Flexibilität und Erweiterbarkeit, sodass auch zukünftige Anwendungsfälle gut behandelt werden können. Zu den bisher definierten Standards, welche alle vom *World Wide Web Consortium*[4] (W3C) entwickelt wurden, zählen u.a. XML[5] (Extensible Markup Language), RDF[6] (Resource Description Framework), RDFS[7] (RDF Schema) und OWL[8] (Web Ontology Language). Diese Standards erfüllen unterschiedliche Aufgaben im Semantic-Web, wobei XML eher dem WWW zuzuordnen ist. Bei RDF, RDFS und OWL handelt es sich jedoch um sogenannte Ontologiesprachen, welche speziell für die Verwendung im Semantic-Web entwickelt wurden. Der Begriff Ontologie beschreibt in der künstlichen Intelligenz eine eindeutige Spezifikation eines Konzepts, welches geteiltes Wissen enthält [Gru93].

[4] http://www.w3.org
[5] http://www.w3.org/TR/xml/
[6] http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/
[7] http://www.w3.org/TR/rdf-schema/
[8] http://www.w3.org/2004/OWL/

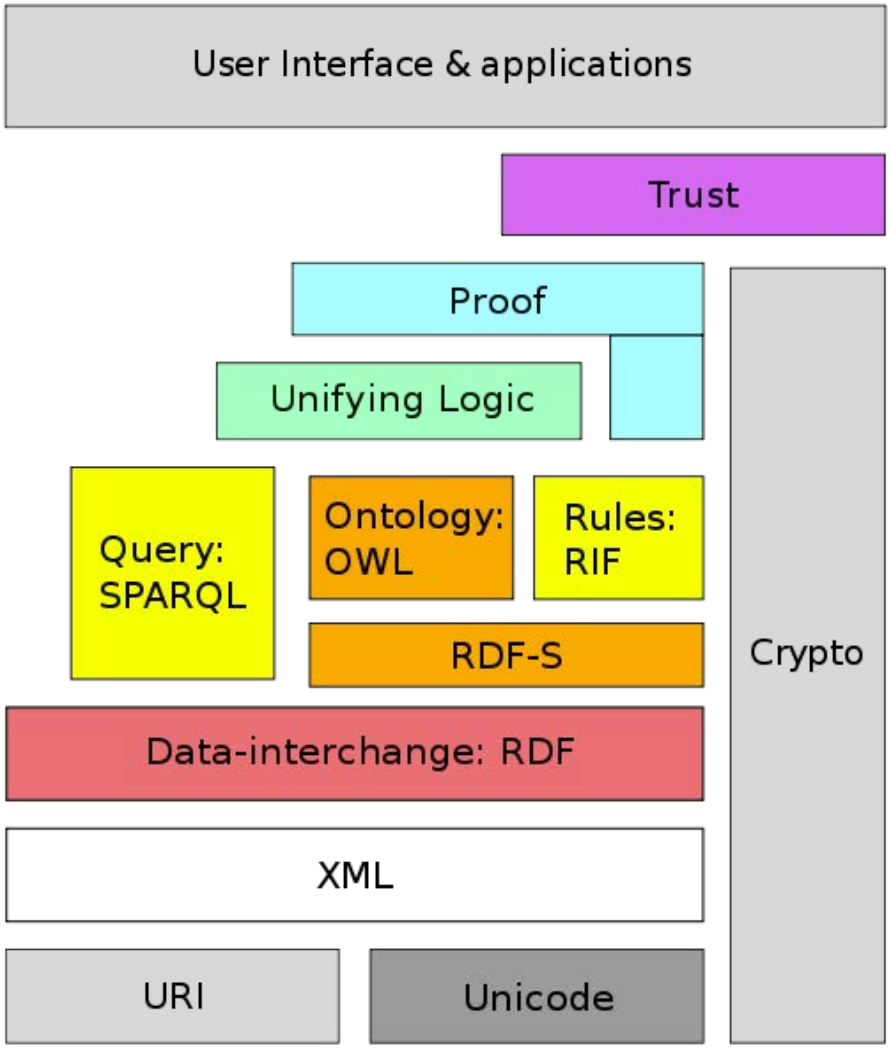


Abbildung 2.1: Semantic-Web Stack des W3C (Quelle: [Wik11d])

Die vom W3C entworfenen Standards bilden somit eine Grundlage für den Aufbau des Semantic-Web. Der komplette Semantic-Web Stack oder auch Semantic-Web Layercake genannte Aufbau, welcher ebenfalls vom W3C vorgeschlagen wurde, ist in Abbildung 2.1 zu sehen. Dabei dienen die unteren Elemente Unicode, URI, XML und RDF zur Beschreibung der Daten. Unicode stellt dabei einen internationalen Zeichensatz zur Verfügung, der Zeichensätze wie z.B. arabisch, japanisch und kyrillisch enthält. URI[9] (Uniform Ressource Identifier) ist bereits ein geltender Standard und dient zur Identifizierung von Objekten im Internet und der realen Welt. XML ist ebenfalls bereits ein geltender Standard im WWW mit dem Daten hierarchisch strukturiert werden können. Mit dem neu erschaffenen Standard RDF können Ressourcen im Semantic-Web beschrieben werden. Durch RDFS können Ressourcenklassen und ihre typischen Eigenschaften definiert und somit die Ressourcen

[9] `http://www.w3.org/Addressing/URL/uri-spec.html`

Abbildung 2.2: Beispiel eines RDF-Graphen

hierarchisch zu Taxonomien gegliedert werden. Darauf aufsetzend werden Ontologien und Regeln gebildet, mit deren Hilfe die Daten verarbeitet werden können. SPARQL dient dabei der Anfrage von Daten im Semantic-Web. Als übergeordnete Ebene folgt eine Logik-Schicht, mit der die gelieferten Ergebnisse durch logische Operatoren weiter verarbeitet werden können. Die Beweis-Schicht dient der Überprüfung der Ergebnisse und soll bei dem Nutzer ein gewisses Maß an Vertrauen in die gelieferten Ergebnisse schaffen, welche letztendlich an die oberste Schicht, der eigentlichen Anwendung, weitergegeben werden. Im Folgenden wird nun genauer auf einige wichtige Komponenten dieses Semantic-Web Stacks eingegangen.

2.2.1 Ressource Description Framework

Bei RDF handelt es sich um ein System zur Beschreibung von Daten im Semantic-Web. Da dies der erste neue Standard im Semantic-Web Stack ist, kann RDF als Grundlage des Semantic-Web angesehen werden. Es handelt sich bei diesem Standard um ein sehr offenes Modell, mit dem es lediglich möglich ist Daten zu beschreiben. Eine Beschreibung der Daten ist dabei als Tripel aufgebaut und enthält die folgenden 3 Elemente:

- die Ressource, die beschrieben werden soll (Subjekt),
- die Eigenschaft der Ressource (Prädikat) und
- den Wert dieser Eigenschaft (Objekt).

Diese Art der Aussagen erinnern dabei durch den Aufbau aus Subjekt, Prädikat und Objekt stark an die natürliche Sprache. Dadurch wird beispielsweise der Satz „Das Auto hat die Farbe Rot.“ in RDF folgendermaßen modelliert. Die Ressource, die beschrieben wird ist das „Auto“ (Subjekt), welches eine Eigenschaft „Farbe“ (Prädikat) besitzt. Das Objekt, bzw. der Wert dieser Eigenschaft ist in diesem Fall „Rot“. Grafisch können die Objekte, welche mit RDF beschrieben werden, durch Graphen dargestellt werden. Dabei bilden Subjekt und Objekt jeweils einen Knoten, welche durch eine Kante, die das Prädikat enthält, verbunden werden. Das oben genannte Beispiel ist auf diese Weise in Abbildung 2.2 dargestellt.

Dabei ist zu sehen, dass Subjekt und Prädikat als URIs notiert sind. Dabei handelt es sich um einen Grundsatz bei der Verwendung von RDF, der besagt, dass alle Elemente durch URIs beschrieben werden müssen. Lediglich Objekte können Literale, wie in diesem Fall die Zeichenkette „Rot“ enthalten. Durch die Verwendung der URIs kann das Problem der Homonyme überwunden werden und es ist einer Maschine möglich Gegenstände eindeutig zu identifizieren. Die momentan verbreitetste Form der Darstellung von RDF bedient sich XML. Der Vorteil dabei liegt natürlich in der Eignung der leichten maschinellen Verarbeitung und der schnellen Einbindung in HTML-Dokumente und somit der Bereitstellung im Web. Eine kürzere Form der Syntax wurde von Tim Berners-Lee entworfen und nennt sich Notation 3 (N3). Bei dieser Syntax werden die Tripel-Elemente in Tag-Klammern notiert und es wird auf die XML-typischen Element- und Attributbezeichnungen verzichtet. Das obige Beispiel könnte in N3 demnach durch <#http://www.foo.com/abc/auto><#http://www.bar.de/xyz/farbe><“Rot“> dargestellt werden. Da das reine Ablegen der Tripel in einer Datenbank nicht sehr effizient ist wurden für die Speicherung von RDF bereits verschiedene Konzepte dafür entworfen.

2.2.2 RDF Schema

Es handelt sich bei RDF Schema (RDFS) ebenso wie bei RDF um eine Empfehlung des W3C. Diese bietet die Möglichkeit die Ressourcen, welche durch RDF beschrieben und maschinell auswertbar sind, mit anderen Ressourcen in Beziehung zu setzen. Da es mit RDF nicht möglich ist Klassen für Objekte zu definieren, wurde für diesen Zweck RDFS entwickelt. Es bietet die Möglichkeit Begriffe und die damit verbundenen Eigenschaften semantisch in Beziehung zu setzen, wodurch gleichartige Ressourcen klassifiziert werden können. Somit werden einzelne RDF-Ressourcen in Klassen zusammengefasst, welche dann wiederum mit anderen Klassen in Beziehung gesetzt werden können. Die Klassen und Relationen, welche in RDFS definiert werden, lassen sich hierarchisch gliedern und werden wie bei RDF in Tripeln beschrieben. Die zugrunde liegende Idee von RDFS beruht auf einem mengentheoretischen Klassenmodell, wobei Klassen und Eigenschaften separat voneinander modelliert werden. Es wird für jede Eigenschaft festgelegt, welche Werte für diese erlaubt sind, was für eine Bedeutung die Eigenschaft hat, welche Beziehung zu anderen Eigenschaften bestehen und welche Arten von Ressourcen diese Eigenschaft verwenden dürfen. Dabei wurde vom W3C kein allgemein gültiges Schema definiert, sondern es werden die eigentlichen Schemata mittels einer *Scheme-Definition-Language* beschrieben. Dabei existieren verschiedene Konzepte, welche bei der Erstellung spezieller Schemata genutzt werden können. Die Konzepte für die Erstellung von Klassen sind in Tabelle 2.2 und die zur Erstellung von Eigenschaften in Tabelle 2.3 zu sehen. Zusätzlich zu diesen beiden Konzepten gibt es noch sogenannte Behelfs-

rdfs:Ressource	Dies ist die eine spezielle Klasse, welcher jede Entität im RDF-Modell angehört
rdfs:Class	Damit können Ressourcen als eine Klasse anderer Ressourcen deklariert werden
rdfs:Property	Es handelt sich dabei um die Basisklasse der Eigenschaften und um eine Unterklasse von rdfs:Ressource
rdfs:Literal	Dies ist die Klasse für Literalwerte, wie z.B. Zeichenketten oder Zahlen

Tabelle 2.2: Konzepte zur Erstellung von Klassen in RDFS

rdfs:subClassOf	Mittels dieser Eigenschaft können hierarchische Strukturen innerhalb der Klassen aufgebaut werden
rdfs:subPropertyOf	Dient zur Festlegung von hierarchischen Strukturen in Eigenschaften
rdfs:domain	Dient zur Festlegung des Datentyps einer Eigenschaft
rdfs:range	Damit wird der Datentyp des Objekts einer Eigenschaft festgelegt

Tabelle 2.3: Konzepte zur Erstellung von Eigenschaften in RDFS

eigenschaften. Dazu zählt rdfs:seeAlso, mit der ein Verweis zu einer Ressource erstellt werden kann, die evtl. zusätzliche Informationen enthält. Des Weiteren gibt es rdfs:isDefinedBy, wodurch eine Ressource angegeben werden kann, in der das aktuelle Subjekt definiert ist.

2.2.3 SPARQL

Es handelt sich bei SPARQL (Simple Protocol And RDF Query Language) um eine Anfragesprache für RDF. Sie wurde 2008 vom W3C als Empfehlung ausgesprochen [PS08] und ist der Nachfolger mehrerer anderer Anfragesprachen, wie z.B. RDQL (RDF Data Query Language) oder RDF Query Language, mit denen ebenfalls auf RDF-Daten zugegriffen werden kann [Wik11e]. Mittels SPARQL können Anfragen an sogenannte SPARQL Endpunkte gestellt werden. Unter SPARQL Endpunkten versteht man eine spezifische Adresse, welche durch eine URI identifiziert werden kann und einen Web-Service zur Verfügung stellt. Auf diesen Web-Service kann dann mittels eines bestimmten Protokolls und Datenformats zugegriffen werden, um Informationen von diesem Endpunkt abzufragen. Für die Anfragen stehen folgende Anfrage-Formen zur Verfügung [PS08]:

- **SELECT** - Extrahiert Rohdaten von einem SPARQL Endpunkt, welche in Tabellenform zurückgeliefert werden,
- **CONSTRUCT** - Extrahiert Informationen von einem SPARQL Endpunkt und wandelt diese in das RDF-Format um,

- **ASK** - Erstellt eine Anfrage, mit der ermittelt werden kann, ob für diese ein Ergebnis existiert oder nicht,
- **DESCRIBE** - Liefert einen RDF-Graphen, welcher Informationen über Ressourcen enthält.

Bei allen Anfrage-Formen muss eine WHERE-Klausel eingebunden werden, um die Ergebnisse genauer zu spezifizieren, mit der Ausnahme von DESCRIBE, bei der die WHERE-Klausel optional ist.

2.2.4 Web Ontology Language

Durch die Erstellung von Klassen und deren Beziehungen untereinander ist es nun möglich systematische Modelle von bestimmten Fach- oder Wissensgebieten zu erstellen. Um diese Ontologien zu entwickeln muss es eine Möglichkeit geben die Eigenschaften der Klassen und ihre Beziehungen untereinander detaillierter zu beschreiben, als es mit RDFS möglich ist.

Bei der Web Ontologie Language (OWL) handelt es sich um eine Spezifikation des W3C [MH04], welche 2004 herausgegeben wurden. Seit 2009 existiert eine weitere Version namens OWL2 [OWL09], bei der es sich ebenfalls um eine Empfehlung des W3C handelt. Auch bei der Web Ontology Language (OWL) handelt es sich um eine Spezifikation des W3C [MH04]. Sie dient als formale Beschreibungssprache, mit deren Hilfe es möglich ist Ontologien zu beschreiben, zu publizieren und zu verteilen. Des Weiteren beinhaltet sie Sprachkonstrukte, mit denen es möglich ist Ausdrücke zu formulieren, die ähnlich zur Prädikatenlogik sind. OWL existiert in den 3 unterschiedlichen Sprachebenen OWL Lite, OWL DL (Description Logic) und OWL Full, welche sich durch ihre Ausdruckskraft unterscheiden.

Die Basis dieser Web Ontology Sprachen bildet dabei die Beschreibungslogik, da sie ebenfalls Formalismen zur Wissensrepräsentation besitzt. Alle Sprachen der OWL-Familie bedienen sich dem Konzept der sogenannten „Open World Assumption“, d.h. wenn Wissen über einen Sachverhalt nicht in der Wissensbasis vorhanden ist, so kann auch keine Aussage darüber getroffen werden. Anders als bei der „Closed world assumption“ wird etwas, das nicht in der Wissensbasis vorhanden ist nicht als falsch interpretiert, sondern als unbekannt.

2.2.5 Beweiskraft und Vertrauen

In den beiden oberen Schichten des Semantic-Web Stacks in Abbildung 2.1 geht es darum, die Informationen, die ein Software-Agent gesammelt hat zu validieren und ein gewisses Maß an Vertrauen bei dem Nutzer zu schaffen. Dabei sollen die Agenten selbstständig in der Lage sein, Beweise, die sie für die gesammelten Informationen bekommen, nachzuvollziehen [KM01]. Hinzu kommt durch die Fülle der Informationen im Semantic-Web und deren dezentrale Struktur die Frage danach, welchem Anbieter der Informationen vertraut werden kann. Denn die bisher vorgestellten Techniken können nur korrekte Aussagen treffen, wenn sie auf korrekten Daten arbeiten. Da jedoch jeder im Semantic-Web in der Lage ist beliebige Informationen zu Ressourcen zu veröffentlichen, werden Kontrollstrukturen benötigt, mit denen sich ermitteln lässt, in welchem Maße einer Quelle vertraut werden kann. Dies ist auch hilfreich, wenn mehrere Quellen unterschiedliche Informationen zu dem selben Sachverhalt enthalten, da nun entschieden werden muss, welche Information aktueller, bzw. präziser ist.

Für das Semantic-Web werden dafür sogenannte *trusted rating services* vorgeschlagen. Dabei kann es sich z.B. um unabhängige Institutionen handeln, die Bewertungen für bestimmte Anbieter von Informationen verteilen oder es könnte sich um Bewertungen von Mitgliedern einer Community handeln, wie es z.B. bei Onlineshops üblich ist. Welcher der verschiedenen Bewertungsservices für einen Software-Agenten verwendet werden soll, liegt dann in der Entscheidung des Nutzers. Zusätzlich dazu soll einem Nutzer stets präsentiert werden, woher welche Informationen stammen und welche Informationen zur Entscheidungsfindung herangezogen wurden.

Ein weiterer Teil in der Schicht Vertrauen befasst sich mit der Vergabe von Rechten. Dadurch soll festgelegt werden, wo ein Software-Agent welche Informationen abfragen darf. Es sind dazu bereits diverse Konzepte entwickelt worden, jedoch ist noch wenig über die praktische Umsetzung in Erfahrung zu bringen.

2.2.6 Linked Data

Linked Data stellt den ersten Schritt in Richtung des Semantic-Web dar. Es sollen nicht nur Informationen in einem Format vorhanden sein, die maschinell verarbeitet werden können, sondern die Informationen sollen miteinander verknüpft werden, um sie noch nützlicher zu machen. Dadurch wird es möglich neue Informationen zu finden, die in Zusammenhang mit

bereits bekannten stehen. Es handelt sich dabei um das gleiche Prinzip, wie es bereits im WWW durch die Verwendung von Hyperlinks zur Anwendung kommt. Der Unterschied dabei besteht jedoch darin, dass nicht wie bisher HTML-Dokumente, sondern beliebige Objekte, die mittels RDF beschrieben werden, verknüpft werden können. Die Objekte werden mittels URIs identifiziert und können somit eindeutig referenziert werden. Auf Linked Data kann direkt mittels HTTP zugegriffen werden, wodurch die Informationen für den Menschen direkt erfasst werden können. Des Weiteren wird zusätzlich durch Techniken des Semantic-Web eine maschinelle Verarbeitung ermöglicht.

Sind diese Daten frei verfügbar, so spricht man auch von Linked Open Data. Das weltweite Netz, welches sich dann aus den verlinkten Daten ergibt wird als „Linked [Open] Data Cloud“ oder als „Giant Global Graph“ bezeichnet und ist in Abbildung 2.3 zu sehen. Der Durchmesser der Kreise repräsentiert dabei die Anzahl der Tripel, die in der jeweiligen Datenquelle vorhanden ist. Ist der Durchmesser klein, so entspricht dies 10.000-500.000 Tripeln. Datenquellen, welche durch einen sehr großen Kreis dargestellt sind, beinhalten mehr als eine Milliarde Tripel. Die Pfeile zwischen den Kreisen symbolisieren Links zwischen den Datenquellen. Ein Link ist in diesem Fall dadurch gekennzeichnet, dass sich die URIs von Subjekt und Objekt eines Tripels in den Namensräumen verschiedener Datenquellen befinden. Die Dicke der Pfeile entspricht der Anzahl an Links, wobei dünne Pfeile weniger als 1.000 Links und dicke Pfeile mehr als 100.000 Links darstellen [CJ10]. Die Erwartungen, welche sich laut [BL09] an die Bereitstellung der Daten ergeben lassen sich wie folgt formulieren:

1. Objekte werden durch URIs identifiziert,
2. es sollen HTTP URIs verwendet werden, sodass Nutzer diese nachschlagen können,
3. wenn jemand eine URI nachschlägt, müssen dazu stichhaltige Informationen mittels Standards (z.B. RDF, SPARQL) geliefert werden,
4. in diesen Informationen sollen weitere Links enthalten sein, sodass mehr Informationen verfügbar gemacht werden.

Auch die Bereitstellung der Daten sollte sich an bestimmte Regeln halten, weshalb Tim Berners-Lee dafür ein Bewertungssystem entwickelt hat [BL09]. Dieses Bewertungssystem soll Nutzern dabei helfen die Qualität ihrer bereitgestellten Daten zu bestimmen. Dadurch wird ersichtlich, welche Änderungen durchgeführt werden können, um die Qualität und damit auch den Nutzen der Daten zu steigern. Bei dem System von Berners-Lee werden bis zu fünf Sterne für die Daten verteilt, wenn sie die dafür nötigen Bedingungen erfüllen.

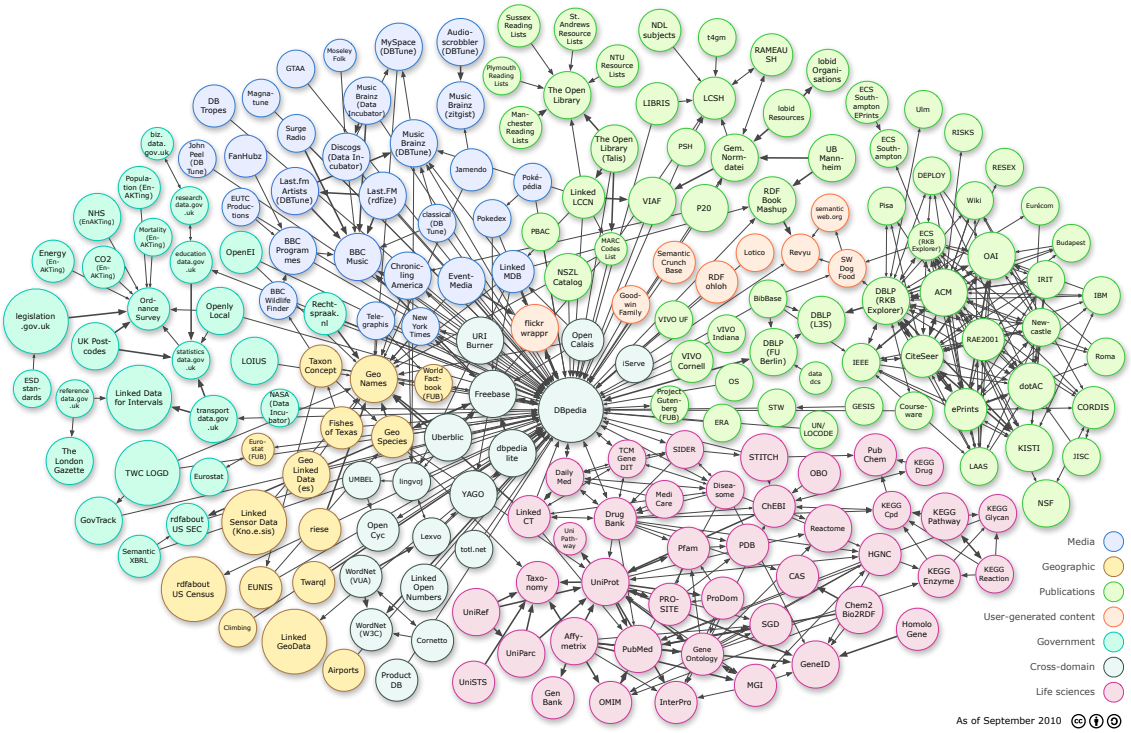


Abbildung 2.3: Diagramm der Linked Open Data Cloud (Quelle: [CJ10])

2.2.7 Freebase

Freebase ist ein Semantic-Web-Dienst, welcher im Zuge dieser Arbeit genutzt wird, um Informationen zu Künstlern und deren Veröffentlichungen zu sammeln. Es handelt sich dabei um eine Online-Datenbank, die strukturierte Daten enthält, welche aus verschiedenen Quellen gesammelt wurden. Der Dienst wurde von der amerikanischen Softwarefirma Metaweb entwickelt und ist seit März 2007 verfügbar [Wik11a]. Freebase bietet für Entwickler die Schnittstellen API, einen RDF Endpunkt und Datenbankdumps an. Die Veröffentlichung der Daten erfolgt unter der Creative Commons Attribution License[10] (CC-BY). Eine Ausnahme bilden die Beschreibungen, welche von Wikipedia stammen, da diese unter der GNU Free Documentation License[11] (GFDL) stehen und daher diese Lizenz übernommen werden muss. Wie bereits erwähnt, werden Daten im Semantic-Web in Form von Graphenmodellen gespeichert, in denen die Daten mittels Knoten und deren Verbindungen durch Kanten repräsentiert werden. Dieses Prinzip findet auch bei Freebase Anwendung, wodurch komplexere Beziehungen zwischen den Elementen modelliert werden können, als es mit konventionellen Datenbanken möglich wäre.

[10] `http://creativecommons.org/licenses/by/3.0/legalcode`
[11] `http://www.gnu.org/licenses/fdl.html`

Die einzelnen Einträge, welche bei Freebase gespeichert sind, werden *topics* genannt. Mit Stand von April 2010 besitzt Freebase ca. 11,5 Mio. solcher *topics*. Abhängig davon, welchen *type* sie besitzen, sind unterschiedliche Informationen zu den Objekten verfügbar. Dabei kann ein *topic* mehrere *types* beinhalten, wie z.B. Snoop Dogg, der eine Vielzahl an Typen besitzt, die ihn unter anderem als „Musiker“ und „Schauspieler“ beschreiben. Dies sind die Ontologien, die bei Freebase genutzt werden und die ebenfalls durch die Nutzer bearbeitet werden können. Dadurch können durch die Nutzer neue Typen erstellt werden, welche allerdings erst für die Öffentlichkeit freigegeben werden, wenn ein Mitarbeiter von Metaweb diese geprüft hat. Des Weiteren kann die Struktur existierender Ontologien nicht geändert werden, indem z.B. eine Eigenschaft gelöscht wird, da es sonst zu Problemen bei externen Applikationen kommen kann, die bestimmte Strukturen voraussetzen.

2.2.8 MusicBrainz

Bei MusicBrainz (MB) handelt es sich ebenfalls um einen Semantic-Web-Dienst, der in dieser Arbeit für die Beschaffung von Metadaten genutzt wird. Das Ziel von MusicBrainz ist die Erschaffung einer freien und offenen Musik-Datenbank. MB wurde am 9.März 1999 von Robert Kaye unter dem Namen *cdindex* gegründet [Wik11c]. Die Veröffentlichung der Daten erfolgt unter der Lizenz Creative Commons CC-BY-NC-SA[12]. Durch die verwendete Technik „MusicDNS“ ist es möglich Musikstücke mittels sogenannter PUIDs (Portable Unique IDentifier) eindeutig zu identifizieren. Dies ist hilfreich, wenn die Musikstücke mit Metadaten angereichert werden sollen.

Die einzelnen Einträge, wie z.B. Künstler, Labels, Lieder, etc. werden dabei mittels sogenannten MusicBrainz-IDs identifiziert. Eine solche ID besteht aus einem 36-stelligem Code und sorgt dafür, dass eine eindeutige Identifizierung stattfinden kann und somit gleichnamige Entitäten unterschieden werden können. Mit Stand vom Juli 2010 enthält MusicBrainz Informationen über ca. 560.000 Künstler, 830.000 Veröffentlichungen und 9,8 Mio. Lieder. Zu den MusicBrainz-IDs können dann Informationen hinzugefügt werden und diese mit anderen Elementen in Beziehung gesetzt werden. Beschrieben werden die Daten mittels RDF und XML. Zur automatisierten Verarbeitung steht eine REST-Schnittstelle zur Verfügung, über die die Daten angefragt werden können. Angemeldeten Nutzern ist es möglich Informationen zu dem System hinzuzufügen, die durch Kontrollen anderer Nutzer validiert werden können.

[12] `http://creativecommons.org/licenses/by-nc-sa/2.0/de/legalcode`

Dies trägt dazu bei, dass die Qualität der vorgefundenen Informationen meist höher ist, als bei vergleichbaren Diensten.

2.3 Verwandte Arbeiten

In diesem Kapitel wird der aktuelle Stand der Technik im Bereich der Kontext-sensitiven Musikempfehlungen und der Empfehlungssysteme, die auf Meta-Daten aus dem Semantic Web basieren, vorgestellt. Dazu wird im ersten Abschnitt auf einen Ansatz zum Ermitteln der semantischen Distanz innerhalb der Linked Data, die zur Generierung von Empfehlungen genutzt werden kann [Pas10], eingegangen. Im zweiten Abschnitt folgt ein kollaboratives Empfehlungssystem, welches in [WK07] vorgestellt wird und semantische Informationen nutzt, um somit das Clustering von Nutzern zu verbessern. Abschließend wird ein Verfahren ([BSS10]) präsentiert, in dem Semantic-Web-Daten genutzt werden, um ein Ähnlichkeitsmaß für Musik-Künstler zu erstellen, mit dem es möglich ist Empfehlungen zu generieren.

Bei dem Ansatz von Alexandre Passant, der in [Pas10] vorgestellt wird, handelt es sich um die Entwicklung von Messgrößen, mit denen die Entfernung zwischen Elementen der Linked Data Cloud (s. Kapitel 2) bestimmt werden kann. Somit soll es ermöglicht werden, die Distanzen der Elemente untereinander zu bestimmen, wodurch anschließend Empfehlungen generiert werden können. Aber auch andere Anwendungen, wie z.B. das Erkennen von Gruppen innerhalb sozialer Netzwerke oder das Vorschlagen von Web-Seiten für intelligentes Browsen seien möglich.

Passant geht dabei auf die Hintergründe der Linked Data ein und definiert einen Datensatz als einen Graph $G = (R, L, I)$, wobei $R = \{r_1, r_2, \ldots, r_n\}$ einer Menge von Ressourcen, $L = \{l_1, l_2, \ldots, l_n\}$ einer Menge von Links und $I = \{i_1, i_2, \ldots, i_n\}$ einer Menge von Instanzen der Links zwischen den Ressourcen entspricht. Die Ressourcen und Links werden dabei durch URIs identifiziert. Die Linked Data Cloud setzt sich dann aus der Vereinigung aller Graphen G_i zusammen, die im Web veröffentlicht und verlinkt sind ($LOD = \bigcup_i G_i$). Anhand dieser Definition des Graphen entwickelt Passant verschiedene Strategien, um die semantischen Entfernungen zwischen den Linked Data Ressourcen zu errechnen. Dadurch soll es möglich werden die Beziehung zwischen zwei Ressourcen, in diesem Fall zweier URIs, zu bestimmen.

Um dieses Ziel zu erreichen wird eine Menge von Messwerten mit dem Namen $LDSD$ (Linked Data Semantic Distance) definiert, wobei deren unterschiedliche Varianten Ergebnisse im Intervall $[0, 1]$ liefern. Die erste Variante ist in Formel 2.8 zu sehen und bestimmt die direkte Entfernung zwischen Ressourcen, d.h. mit Hilfe der Anzahl der direkten Links (eingehende und ausgehende) zwischen zwei Elementen. In [Pas10] wird dies wie folgt definiert:

$$LDSD_d(r_a, r_b) = \frac{1}{1 + C_d(n, r_a, r_b) + C_d(n, r_b, r_a)} \tag{2.8}$$

C_d stellt dabei eine Funktion dar, welche die Anzahl der direkten Links n zwischen den Ressourcen r_a und r_b liefert. n dient dabei als eine Art Variable, die das Ergebnis der Funktion enthält. Des Weiteren wird eine gewichtete Version $LDSD_{dw}$ vorgestellt, in der das Gewicht von der Anzahl an Vorkommen eines jeden Links in dem Graphen abhängt, um somit den Einfluss der populärsten Links zu verringern.

Mittels einer weiteren Variante von $LDSD$, die von Passant vorgestellt wird, ist es möglich die indirekte Distanz zwischen zwei Ressourcen zu bestimmen. Dies wird damit begründet, dass der Wert der Linked Data nicht nur in der direkten Verknüpfung von Ressourcen besteht, sondern auch über die Verbindung weiterer anderer Ressourcen. Mittels $LDSD_i$ ist es möglich die indirekte Distanz zwischen den Ressourcen r_a und r_b zu bestimmen, welche über eine weitere Ressource miteinander verknüpft sind. In Formel 2.9 ist dargestellt, wie $LDSD_i$ errechnet wird, wobei C_{ii} und C_{io} Funktionen sind, die die Anzahl der indirekten Links n (eingehende und ausgehende) zwischen den Elementen r_a und r_b bestimmen.

$$LSDS_i(r_a, r_b) = \frac{1}{1 + C_{io}(n, r_a, r_b) + C_{ii}(n, r_a, r_b)} \tag{2.9}$$

Auch hierfür wird eine gewichtete Version $LDSD_{iw}$ vorgestellt, bei der die weniger populären Links ein höheres Gewicht bekommen, da davon ausgegangen wird, dass zwischen zwei Ressourcen ein größerer Zusammenhang besteht, wenn nur wenige eine bestimmte Eigenschaften teilen [PR08].

Als dritter Messwert für die semantische Distanz wird eine Kombination der beiden vorhergehenden vorgestellt. Auch hierbei existiert eine einfache ($LDSD_c$) und eine gewichtete ($LDSD_{cw}$) Version. Zur Demonstration der Ergebnisse wird von Passant ein Empfehlungssystem für Musik-Künstler entwickelt. Die Entfernungen zwischen beliebigen Ressourcen r_a und r_b bilden dabei die Grundlage für das Generieren der Empfehlungen. Anfangs wird

anhand einer Nutzerbefragung eine der sechs vorgestellten Varianten der *LDSD* ausgewählt, welche die besten Ergebnisse verspricht. Die Datenbasis für dieses Empfehlungssystem bildet dabei der Datenbestand des Semantic-Web-Dienstes DBPedia[13], wobei Passant sich auf die Ressourcen des Typs „dbpedia-owl:MusicArtist“ und „dbpedia-owl:Band“ beschränkt. Dem Empfehlungssystem kann eine URI übergeben werden, zu der dann die semantischen Entfernungen zu allen anderen Ressourcen des Datenbestandes errechnet werden. Diese Resultate werden dann der Größe nach geordnet und die ersten x Elemente als Empfehlungen genutzt.

Im Gegensatz zu dem Ansatz, der in dieser Arbeit vorgestellt wird, werden bei Passant die Ressourcen der Linked Data direkt verglichen und somit ihre Distanz zueinander bestimmt. Währenddessen in dem hiesigen Ansatz die Daten genutzt werden, um Künstler zu beschreiben und somit das Clustering dieser Künstler zu ermöglichen.

Ein weiteres Verfahren, in dem Semantic-Web-Daten zur Generierung von Empfehlungen genutzt werden, wird von Wang und Kong in [WK07] vorgestellt. Das von ihnen entwickelte Empfehlungssystem nutzt z.B. Film-Genres, welche aus dem Semantic Web gewonnen werden. Weiterhin nutzt es demographische Informationen der Nutzer, sowie deren Bewertungen in Form einer *User-Item-Rating*-Matrix. Zu Beginn des Prozesses wird eine Ontologie zu den Genres geschaffen. Diese kann in Form einer Baum-ähnlichen, hierarchischen Struktur dargestellt werden. Anschließend werden die vorhandenen Objekte zu den jeweiligen Ontologie-Elementen zugewiesen und deren Ähnlichkeit untereinander bestimmt. Die Größe der Ähnlichkeit zwischen zwei Objekten t_i und t_j ergibt sich aus der Anzahl der gemeinsam geteilten Genres n_{ij}. In Formel 2.10 ist dies formal dargestellt, wobei N der Gesamtanzahl der Genres entspricht.

$$SIM(t_i, t_j) = \frac{n_{ij}}{N} \tag{2.10}$$

Traditionelle kollaborative Empfehlungssysteme besitzen unter anderem Probleme bezüglich der Skalierung, da die Berechnung der Ähnlichkeiten zwischen Nutzern in der Online-Phase des Empfehlungsprozesses durchgeführt wird. Aufgrund der meist großen Anzahl an Nutzern in einem System kann diese Berechnung sehr zeitintensiv sein. Um diesem Problem entgegen zu wirken, werden die Ähnlichkeiten zwischen den Nutzern in diesem Ansatz bereits in der Offline-Phase errechnet. Des Weiteren basieren traditionelle Empfehlungssysteme meist

[13] `http://dbpedia.org`

nur auf der *User-Item-Rating*-Matrix, ohne dass dieser weitere Informationen hinzugefügt werden. Dadurch ist sie sehr dünn besetzt, was zur Folge hat, dass die Empfehlungen meist unbefriedigend sind. Der Ansatz von Wang und Kong nutzt zur Bestimmung der Ähnlichkeiten zwischen den Nutzern neben der Bewertungsmatrix zusätzlich demographische Informationen über die Nutzer, semantische Informationen, sowie die Genre-Ähnlichkeit der Objekte.

Der Clustering-Prozess wird demnach in fünf Schritte unterteilt. In dem ersten Schritt wird die Ähnlichkeit zwischen zwei Nutzern u_i und u_j aufgrund deren vorhandenen Bewertungen ermittelt. Wang und Kong nutzen dazu die Pearson-Korrelation, welche in Formel 2.11 dargestellt ist. Dabei entspricht r_{ik} und r_{jk} den Bewertungen von Nutzer u_i, bzw. u_j bezüglich des Objekts k, $\overline{r_i}$ und $\overline{r_j}$ den durchschnittlichen Bewertungen der Nutzer u_i, bzw. u_j und k der Anzahl der Objekte, die von beiden Nutzern bewertet wurden.

$$sim(u_i, u_j)^1 = \frac{\sum\limits_k (r_{ik} - \overline{r_i}) \cdot (r_{jk} - \overline{r_j})}{\sqrt{\sum\limits_k (r_{ik} - \overline{r_i})^2} \cdot \sqrt{\sum\limits_k (r_{jk} - \overline{r_j})^2}} \tag{2.11}$$

In dem zweiten Schritt des Clusterings wird die demographische Ähnlichkeit der beiden Nutzer bestimmt. Dabei werden die nominalen Attribute auf Gleichheit getestet und entweder mit 0 oder 1 bewertet. Das numerische Attribut „Alter“ wird in zehn Klassen unterteilt anhand derer die Gleichheit bestimmt wird. Die demographische Ähnlichkeit der Nutzer u_i und u_j entspricht dann der Summe aller Ähnlichkeiten der einzelnen demographischen Merkmale, welche zusätzlich über einen Faktor gewichtet werden können. Der dritte Schritt befasst sich mit der Bestimmung der Ähnlichkeit der Nutzer anhand der semantischen Ähnlichkeit von Objekten, welche die Nutzer bewertet oder aufgerufen haben. Die Gleichung für dieses Ähnlichkeitsmaß ist in Formel 2.12 dargestellt, wobei u_i auf die Objekte t_1 bis t_{k1} und u_j auf die Objekte t'_1 bis t'_{k2} zugegriffen, bzw. diese bewertet hat.

$$sim(u_i, u_j)^3 = \max(sim(u_i \to u_j), sim(u_j \to u_i)) \tag{2.12}$$

$$sim(u_i \to u_j) = \frac{\sum\limits_{k1} \max\limits_{k2}(SIM(t_{k1}, t'_{k2}))}{k1}$$

$$sim(u_j \to u_i) = \frac{\sum\limits_{k2} \max\limits_{k1}(SIM(t'_{k2}, t_{k1}))}{k2}$$

In dem vierten Schritt wird dann aus den drei vorhergehenden Ähnlichkeitsmaßen eine endgültige Ähnlichkeit zwischen den Nutzern errechnet. Dargestellt wird dies in Formel 2.13, wobei α_k einem Gewichtungsfaktor entspricht. Der letzte Schritt befasst sich dann mit dem Clustering der Nutzer, welches mittels k-Means Algorithmus (s. Kapitel 3) durchgeführt wird.

$$sim(u_i, u_j) = \sum_{k=1}^{3} sim(u_i, u_j)^k \cdot \alpha_k \tag{2.13}$$

$$\sum_{k=1}^{3} \alpha_k = 1$$

Diese fünf Schritte des Clusterings werden, wie bereits erwähnt, offline durchgeführt und müssen somit nicht abgearbeitet werden, wenn eine Empfehlung generiert werden soll. Wenn einem aktiven Nutzer Empfehlungen vorgeschlagen werden sollen, so wird als erstes ein Vektor mit seinen bevorzugten Genres erstellt. Dies geschieht mit Hilfe der Objekte, auf die der Nutzer bereits zugegriffen, bzw. die er bewertet hat. Im zweiten Schritt wird dann das Cluster bestimmt, in dem sich der Nutzer befindet. Anhand der anderen Nutzer im Cluster werden dann kollaborativ die Bewertungen der Objekte vorhergesagt, auf die der Nutzer noch nicht zugegriffen hat. Dem Nutzer werden dann die Objekte vorgeschlagen, zu denen die höchsten Bewertungen vorhergesagt wurden.

Die Evaluation der Ergebnisse wird mit Daten durchgeführt, die von *MovieLens*[14] gewonnen wurden. Es wird dabei das vorgestellte Verfahren mit einem traditionellen kollaborativen Empfehlungssystem verglichen. Als Messwert für die Güte der Ergebnisse haben Wang und Kong die Precision (s. Kapitel 5) gewählt. Es werden dabei die vorhergesagten Bewertungen mit den vorhandenen Bewertungen in den Testdaten verglichen. Die Daten wurden dazu in fünf Untermengen aufgeteilt, wobei jede dieser Teilmengen in 75% Trainingsdaten und 25% Testdaten aufgeteilt wurde. Die vorgestellten Ergebnisse zeigen, dass dieser Ansatz eine ca. 12% höhere Vorhersagegenauigkeit gegenüber traditionellen kollaborativen Empfehlungssystemen besitzt. Dies wird damit begründet, dass semantische und demographische Informationen genutzt werden und somit die Vorhersagen nicht auf einer dünn besetzten Bewertungsmatrix beruhen. Des Weiteren wird die Zeit für die Erstellung der Vorhersagen verglichen. Da in diesem Ansatz die Ähnlichkeiten zwischen den Nutzern offline errechnet werden, liegen die Zeiten für die Berechnung der Vorhersagen mit steigender Nutzerzahl weit

[14] `http://www.movielens.org`

unter denen der traditionellen Methode.

In dem Ansatz von Wang und Kong werden die Semantic-Web-Daten genutzt, um das Clustering von Nutzern zu verbessern. Es werden dabei, ähnlich dem hier vorgestellten Ansatz, die zu empfehlenden Objekte mit diesen Daten angereichert. Dabei werden allerdings die Nutzer untereinander verglichen und nicht versucht, die bevorzugten Genres der einzelnen Nutzer zu ermitteln, wie es in diesem Ansatz geschieht.

Ein weiteres Verfahren, mit dessen Hilfe Ähnlichkeiten zwischen Künstlern bestimmt werden können und das Daten aus dem Semantic-Web nutzt wird von Baumann et al. in [BSS10] vorgestellt. Es wird dabei versucht Gemeinsamkeiten zwischen Künstlern in deren Biographien, bzw. deren musikalischen Aktivitäten zu finden. Dafür werden Daten, wie z.B. Genres, Veröffentlichungsjahre, etc. der Künstler von Freebase[15] extrahiert. Die verschiedenen Veröffentlichungsjahre werden dann zu Zeitabschnitten zusammengefasst, wie z.B. „1965-1975“ zu „1965s“. Die ermittelten Genres und Zeitabschnitte werden daraufhin miteinander verknüpft, sodass sich Kombinationen wie z.B. „pop-1980s“ ergeben. Die Künstler und deren extrahierte Eigenschaften werden anschließend in den Vektorraum überführt, wobei eine Eigenschaft einer Dimension im Vektorraum entspricht. Besitzt ein Künstler eine Eigenschaft, so wird diese durch eine „1“ mit dem Künstler verknüpft, andernfalls mit einer „0“.

Im folgenden Schritt wird eine Merkmalsauswahl durchgeführt, bei der Attribute entfernt werden, die zu selten vorkommen, als dass sie einen Künstler ausreichend charakterisieren könnten. Anschließend wird das Resultat mittels dem TF-IDF-Maß gewichtet, welches ebenfalls in dieser Arbeit verwendet wird (s. Kapitel 3). Die Ähnlichkeit zwischen den einzelnen Künstlern wird dann mittels dem Kosinus-Ähnlichkeitsmaß ermittelt, welches in Formel 2.3 formal dargestellt ist. Die Profile zweier Künstler a_i und a_j werden dabei durch die Vektoren $\vec{w_c}$ und $\vec{w_s}$ repräsentiert. K entspricht dabei der Anzahl der Eigenschaften, die ein Künstler besitzt.

Evaluiert wurden die Ergebnisse des Verfahrens in zwei Schritten. Zum Einen wurde überprüft, ob die ermittelten Ähnlichkeiten zwischen den Künstlern denen von Last.fm entspricht. Zum Anderen wurde eine Nutzerstudie mit Studenten und Mitarbeitern der Popakademie[16] in Mannheim durchgeführt. Dabei sollten die Nutzer die Qualität, sowie die Neuheit der vorgeschlagenen Künstler bewerten. Im ersten Experiment wurden die Ergebnisse norma-

[15] `http://www.freebase.com`
[16] `http://www.popakademie.de`

lisiert, damit sie in dem Intervall $[0, 1]$ liegen. Anschließend wurde der mittlere absolute Fehler, sowie der mittlere quadratische Fehler zwischen den Ergebnissen und den Daten von Last.fm errechnet. Im zweiten Experiment wurde dann die Nutzerstudie durchgeführt, bei der den Probanden für drei ihrer fünf Lieblingskünstler Empfehlungen vorgeschlagen wurden. Generiert wurden diese Empfehlungen von Last.fm, Echo Nest[17] und dem hier beschriebenen Ansatz. Den Testpersonen wurden zu jedem ihrer Lieblingskünstler 15 Empfehlungen präsentiert, wobei jeweils fünf Künstler durch einen der drei Ansätze bestimmt wurden. Es war den Testpersonen jedoch nicht bekannt, welche Empfehlungen durch welchen Ansatz generiert wurden. Die Teilnehmer mussten die Empfehlungen danach bewerten, ob ihnen die vorgeschlagenen Künstler bekannt sind, und wie sie die Qualität einschätzen.

Die Evaluation hat ergeben, dass die Qualität der gelieferten Empfehlungen hinter der von Last.fm oder Echo Nest bleibt. Viele dieser Empfehlungen waren den Nutzern allerdings bereits bekannt. Dadurch konnte gezeigt werden, dass durch diesen Ansatz eine größere Menge an hochwertigen Empfehlungen generiert werden kann, die den Nutzern unbekannt sind.

Im Unterschied zu dem hiesigen Ansatz werden in [BSS10] die Semantic-Web-Daten genutzt um ein Ähnlichkeitsmaß für verschiedene Künstler zu erstellen. Es werden dabei nicht die Nutzer und ihre unterschiedlichen bevorzugten Musikrichtungen berücksichtigt, wie es in dieser Arbeit der Fall ist.

[17] `http://the.echonest.com`

3 Durchführung

Dieses Kapitel behandelt den Data-Mining-Prozess von der Datensammlung über die Datenaufbereitung und das Clustering der Daten bis hin zur Extraktion von Labels, welche als Beschreibung für Cluster dienen. Der Prozess ist in Abbildung 3.1 dargestellt und spiegelt, mit Ausnahme der Evaluation, den Aufbau des aktuellen Kapitels wider.

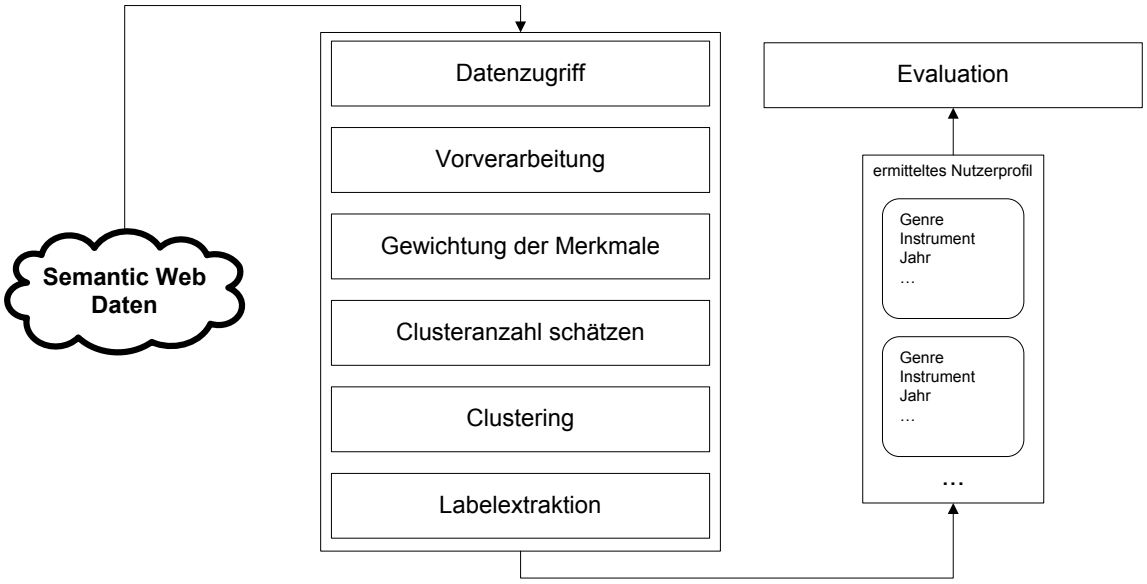


Abbildung 3.1: Schematische Darstellung des Data-Mining-Prozesses

3.1 Datenzugriff

Als Ausgangspunkt für den Data-Mining-Prozess sind verschiedene Daten notwendig. Zum Einen werden Metadaten über Musikkünstler und zum Anderen Nutzerdaten benötigt, welche im Zuge des Prozesses analysiert werden, um somit Benutzerprofile aus diesen Daten extrahieren zu können.

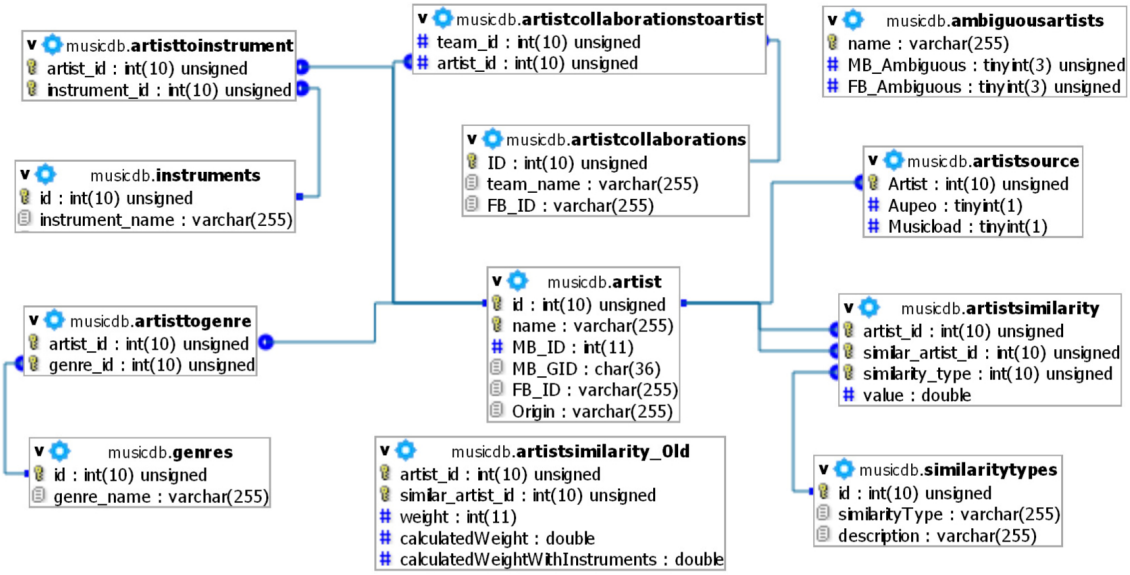


Abbildung 3.2: ERM der ursprünglichen Datenbank

3.1.1 Datenbasis

Als Datenbasis wird vom DFKI[1] eine Datenbank zur Verfügung gestellt, welche bereits Daten zu insgesamt 500.378 Künstlern enthält. Das Entity-Relationship-Modell (ERM) zu dieser Datenbank ist in Abbildung 3.2 dargestellt und zeigt die darin enthaltenen Datenstrukturen. Erstellt wurden die ERMs mit der PHP-Applikation *phpMyAdmin*[2], welches zur Administration von MySQL-Datenbanken genutzt werden kann. Zu einem einzelnen Künstler können sich folgende Daten in der Datenbank befinden:

- Name,
- Musicbrainz-ID,
- globale Musicbrainz-ID,
- Freebase-ID,
- Herkunft,
- Genres, die der Künstler bedient und
- Instrumente, die der Künstler spielt.

Unter der Eigenschaft „Herkunft“ befinden sich 3.902 unterschiedliche Einträge, welche auf 16.248 Künstler verteilt sind. Es existieren 920 verschiedene Genres in der Datenbank, mit denen 24.606 Künstler in Verbindung gebracht werden können. Bezüglich der Instrumente

[1] `http://www.dfki.de`
[2] `http://www.phpmyadmin.net`

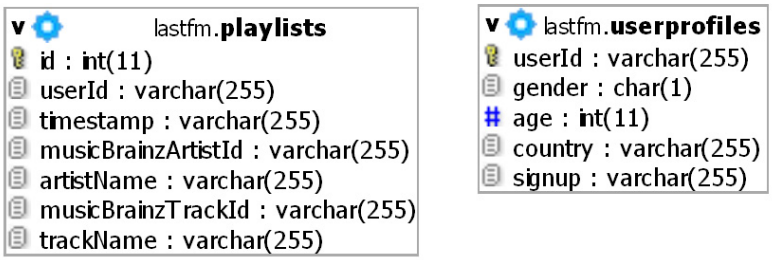


Abbildung 3.3: ERM der „lastfm“-Datenbank

befinden sich 273 unterschiedliche Einträge in der Datenbank. Diese können 10.510 Künstlern zugeordnet werden.

Des Weiteren wird das „Music Recommendation Datasets for Research“[3] von Òscar Celma genutzt, welches anonymisierte Nutzerdaten von 992 Last.fm[4]-Nutzern enthält. In den Daten sind Informationen wie Alter, Geschlecht, Herkunft und Anmeldedatum zu finden, welche unter einer eindeutigen ID zusammengefasst werden. Das ERM dieser Datenbank ist in Abbildung 3.3 zu sehen. Zudem existiert zu jedem Nutzer eine Liste mit den Liedern, die er auf Last.fm gehört hat (bis 05.05.2009). Dies ergibt eine Gesamtanzahl von 19.150.819 Playlist-Einträgen. In den Playlisten sind folgende Informationen zu den Liedern gespeichert:

- Nutzer-ID,
- Zeitstempel,
- Musicbrainz-ID des Künstlers,
- Künstlername,
- Musicbrainz-ID des Liedes,
- Liedername.

Es befinden sich unter den 992 Nutzern 502 männliche, 382 weibliche und 108 Nutzer ohne Angabe zum Geschlecht. Die Nutzer kommen aus 66 verschiedenen Ländern und ihr Durchschnittsalter liegt bei 25,3 Jahren, wobei 85 keine Angabe zu ihrem Herkunftsland und 706 keine Angabe zu ihrem Alter gemacht haben. Obwohl diese Daten in der Datenbasis vorhanden sind, werden sie vorerst nicht innerhalb des Data-Mining-Prozesses genutzt.

[3] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/index.html
[4] http://www.lastfm.de/

3.1.2 Datensammlung

Im Zuge dieser Arbeit soll die Datenbank mit weiteren Informationen zu den Künstlern gefüllt werden, um somit Profile zu erschaffen, mit denen diese spezifischer beschreiben werden können. Zu diesem Zweck werden folgende Metadaten zu den Veröffentlichungen eines Künstlern von Musicbrainz[5] und Freebase[6] gesammelt:

- Albumname,
- Musicbrainz-ID des Albums,
- Datum und Land der Veröffentlichung,
- Labelname,
- Musicbrainz-ID des Labels.

Bezüglich der Instrumente werden deren Instrumentenfamilien in die Datenbank aufgenommen und somit verschiedene Instrumente zu einem allgemeineren Begriff zusammengefasst. So werden Instrumente z.B. zu Saiten-, Blas-, Tasteninstrumenten, etc. zusammengefasst, womit sich eine Reduktion der ursprünglichen 273 Instrumente auf 90 Oberbegriffe ergibt. Dabei werden Instrumente, bei denen keine Angabe zur Instrumentenfamilie gefunden wird, unverändert übernommen.

Zur Durchführung der Datensammlung werden Anfragen mittels eines dafür entwickelten Programms an die Semantic-Web-Dienste Musicbrainz und Freebase gesendet, deren Ergebnisse in die Datenbank geschrieben werden. Hierbei werden die Informationen zu den Veröffentlichungen eines Künstlers von Musicbrainz und die Daten zu Instrumentenfamilien von Freebase bezogen. Zur Visualisierung der Änderungen in der Datenbank-Struktur ist deren Entity-Relationship-Modell in Abbildung 3.4 dargestellt. Zu sehen ist darin, dass die Spalte „instrument_family“ zu der Tabelle „instruments“ hinzugefügt wurde. Des Weiteren wurde die Tabelle „album“ erzeugt, welche zur Aufnahme der Veröffentlichungen der Künstler dient. Eine weitere Tabelle, die zu der Datenbank hinzugefügt wurde heißt „clusteringevaluations“ und hat im späteren Verlauf den Zweck die Evaluationsergebnisse aufzunehmen. Des Weiteren wurde die Tabelle „clusteringresults“ zu der Datenbank hinzugefügt, in der die Ergebnisse der Clusterings gespeichert werden.

[5] `http://musicbrainz.org`
[6] `http://www.freebase.com`

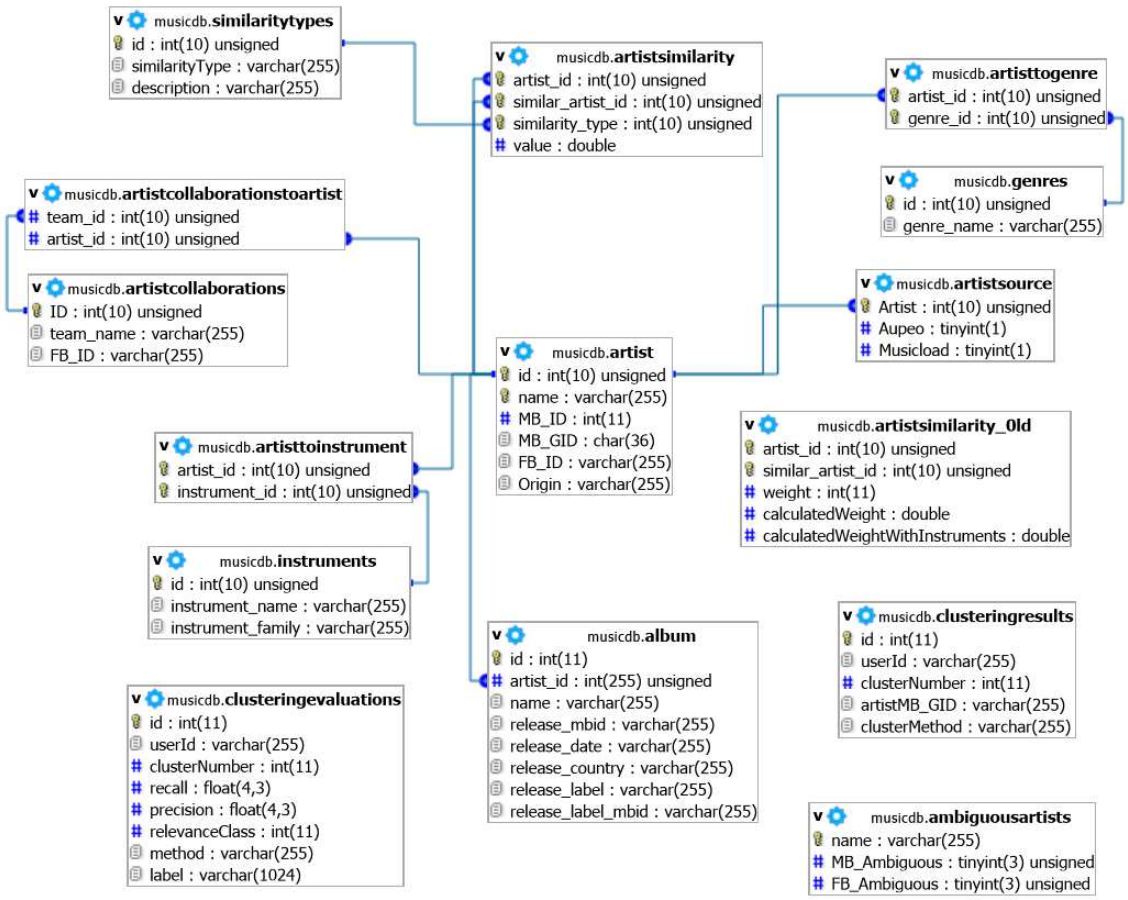


Abbildung 3.4: ERM der Datenbank nach Hinzufügen weiterer Tabellen

Musicbrainz bietet zum Abruf der Daten eine REST-Schnittstelle an, mit deren Hilfe GET-Anfragen nach einem bestimmten Schema[7] gestellt werden können. Die Ergebnisse der Anfrage werden in Form von XML zurückgeliefert und können anschließend ausgewertet werden. Da in diesem Fall die Daten über Veröffentlichungen zu einem Künstler gesucht werden ist die URL wie in Listing 3.1 aufgebaut. Dabei wird *<MB-GID>* durch die aktuelle globale Musicbrainz-ID des Künstlers ersetzt, *type* definiert das Rückgabeformat, wobei XML sich zurzeit um das einzige Format handelt, welches von dem Service angeboten wird. Die Parameter *sa-Album*, *release-events* und *labels* geben an, welche Daten zu dem spezifischen Künstler gesucht werden. In diesem Fall handelt es sich dabei um alle Alben, welche der Künstler veröffentlicht hat, die dazugehörigen Veröffentlichungstermine und die Labels unter denen die Alben erschienen sind.

Listing 3.1: URL-Schema zur Abfrage von Metadaten

```
1 http://musicbrainz.org/ws/1/artist/<MB-GID>?type=xml&inc=sa-Album
    +release-events+labels
```

In Listing 3.2 ist ein Ausschnitt des Ergebnisses dargestellt, welches die Suche mit der Musicbrainz-ID der Beatsteaks liefert. Neben den allgemeinen Eigenschaften des Künstlers ist zu sehen, dass pro Album eine Reihe von Ereignissen eingetragen sind, welche die Veröffentlichungen repräsentieren. Diese Liste der Ereignisse enthält unterschiedlich detaillierte Informationen, welche ausgewertet und in der Datenbank abgelegt werden.

Listing 3.2: Ausschnitt eines Suchergebnisses

```
1  <metadata>
2    <artist id="a9b88ebb-83aa-4ef7-b674-fabeb572c14e" type="Group">
3      <name>Beatsteaks</name>
4      <sort-name>Beatsteaks</sort-name>
5      <life-span begin="1995"/>
6      <release-list>
7        <release id="861746bb-8acc-4d0c-af87-e8cdaa0da6eb" type="
             Album␣Official">
8          <title>Launched</title>
9          <text-representation language="ENG" script="Latn"/>
10         <asin>B00004NRWP</asin>
11         <release-event-list>
12           <event date="1999" country="DE" catalog-number="6559-2"
                 barcode="8714092655926" format="CD">
13             <label id="f15a7708-66b0-4b49-a11a-837d53380d8d">
14               <name>Epitaph Europe</name>
15               <sort-name/>
16             </label>
17           </event>
18           <event date="1999" country="GB" barcode="8714092655919"
                 format="Vinyl"/>
```

[7] http://musicbrainz.org/doc/XML_Web_Service

```
19            <event date="1999" country="GB" barcode="8714092655926"
                  format="CD"/>
20            <event date="2003-01-07" country="DE"/>
21            <event date="2008-01-25" country="DE" barcode="
                 8714092655919" format="Vinyl"/>
22          </release-event-list>
23        </release>
24  ...
```

Freebase bietet ebenfalls eine Schnittstelle über die mittels GET oder POST Informationen angefordert werden können. Die Anfragen müssen dabei mittels der MQL[8] (Metaweb Query Language) formuliert werden und liefern die Ergebnisse serialisiert im JSON-Format an den Nutzer zurück. Die Anfrage, mit deren Hilfe die Instrumentenfamilien bei dem Semanic-Web-Dienst gesucht werden, ist in Listing 3.3 zu sehen. Die Parameter sind dabei *type*, wodurch in diesem Fall festgelegt wird, dass es sich bei dem Eingabetyp um ein Musikinstrument handelt und *name*, dessen Eigenschaftswert den aktuellen Instrumentennamen enthält. Der Parameter *family*, gefolgt von den eckigen Klammern legt fest, dass diese Daten als Ergebnis der Anfrage an den Nutzer zurückgeliefert werden sollen. Sie können dann mit Hilfe eines JSON-Parsers deserialisiert und in der Datenbank gespeichert werden.

Listing 3.3: URL-Schema zur Abfrage von Metadaten

```
1  http://api.freebase.com/api/service/mqlread?query={"cursor":true
       ,"query":{"type":"/music/instrument","name":"<Instrument>","
       family":[]}}
```

3.2 Datenvorverarbeitung

Die Datenvorverarbeitung befasst sich mit der Aufbereitung der gesammelten Daten, um diese für den weiteren Prozess nutzen zu können. Im ersten Schritt werden dazu alle Künstler eines Nutzers aus seiner Playlist geladen. Anschließend werden zu jedem Künstler die Semantic-Web-Metadaten, welche sich in der Datenbank befinden, geladen. Aus der Formel 3.1 ergibt sich die Menge P, welche alle Eigenschaften $p_1, \ldots, p_m$ der Künstler $a_1, \ldots, a_n$ enthält. Dabei entsprechen $P_{a_1}, \ldots, P_{a_n}$ die Mengen der Eigenschaften der einzelnen Künstler.

$$P = P_{a_1} \cup P_{a_2} \cup P_{a_3} \cup \cdots \cup P_{a_n} := \{x | (x \in P_{a_1}) \vee (x \in P_{a_2}) \vee (x \in P_{a_3}) \vee \cdots \vee (x \in P_{a_n})\} \quad (3.1)$$

Mit der Vereinigung der einzelnen Eigenschaften der Künstler zu der Menge P, erhält diese nun alle Attribute, die zur Erstellung einer Artist-Property-Matrix benötigt werden. Die

[8] http://wiki.freebase.com/wiki/MQL_Manual/Metaweb_Query_Language

	Pop	Rock	Hip Hop	Jazz
Madonna	1	0	0	0
Snoop Dogg	0	0	1	0
Metallica	0	1	0	0
Louis Armstrong	0	0	0	1
Black Eyed Peas	1	0	1	0

Tabelle 3.1: Beispiel einer Artist-Property-Matrix

Menge P wird dann mit der Menge aller Künstler $A = \{a_1, a_2, a_3, \ldots, a_n\}$ binär verknüpft, wodurch sich eine Matrix ergibt, welche die Dimension $m \times n$ besitzt. Die Erstellung der Artist-Property-Matrix bezieht sich dabei auf folgende Fallunterscheidung:

$$f(a_i, p_j) = \begin{cases} 0, & \text{wenn } p_j \notin P_{a_i}, \\ 1, & \text{wenn } p_j \in P_{a_i}. \end{cases} \tag{3.2}$$

Das Resultat einer solchen Verknüpfung ist beispielhaft in Tabelle 3.1 dargestellt. Darin enthalten sind fünf Künstler, die mit den Eigenschaften „Pop“, „Rock“, „Hip Hop“ und „Jazz“ in Verbindung gebracht werden. Solche Verknüpfungstabellen dienen dann als Grundlage für die weitere Analyse der Nutzerdaten innerhalb des Prozesses.

3.2.1 Datenreduktion

Die entstandene Tabelle kann unter Umständen sehr viele Attribute enthalten, weswegen eine Reduktion der Spaltenanzahl durchgeführt wird. Dazu wird ein Histogramm über alle Attribute erstellt und anschließend anhand deren Häufigkeiten Spalten aus der Tabelle entfernt. Es wird dabei, abhängig von der Gesamtanzahl der Künstler, eine untere Grenze für die Anzahl an Vorkommen der Eigenschaften festgelegt. Somit wird eine Eigenschaft p_j aus der Tabelle entfernt, wenn sie bei weniger als $x\%$ der Künstler vorkommt. Dies entfernt Einträge, die nur sehr selten vorkommen und somit vernachlässigt werden können. Es wurde im Verlauf der Arbeit ebenfalls mit einer oberen Schranke experimentiert, wobei jedoch festgestellt werden musste, dass diese nur in sehr wenigen Fällen Anwendung fand. Des Weiteren wäre es damit möglich gewesen unbeabsichtigt wichtige Attribute aus der Artist-Property-Matrix zu entfernen, weshalb diese obere Grenze bei der Realisierung entfernt wurde.

3.2.2 Gewichtung der Merkmale

Als Vorbereitung für den Clustering-Prozess wird in diesem Schritt eine Gewichtung der einzelnen Einträge der Artist-Property-Matrix vorgenommen. Die Einträge werden nach einem Verfahren bewertet, welches in [SM86] vorgestellt wird. Es handelt sich dabei um das TF-IDF-Maß, womit sich die Relevanz der einzelnen Einträge innerhalb der Matrix erkennen lässt. Bei dem TF-IDF-Maß handelt es sich um einen statistischen Messwert, welcher für gewöhnlich im Information Retrieval eingesetzt wird, um zu beurteilen wie wichtig ein Wort in einem Dokument innerhalb eines Dokumentenkorpus ist. Das Maß setzt sich aus der Termfrequenz eines Wortes und dessen inverser Dokumentenhäufigkeit zusammen. Die Relevanz eines Wortes steigt somit proportional mit dessen Vorkommen in einem Dokument, wird aber durch die Häufigkeit des Wortes innerhalb des gesamten Dokumentenkorpus herabgesetzt. Mit dieser Kombination bekommen Wörter, die in dem aktuellen Dokument häufig vorkommen und sonst selten sind ein höheres Gewicht als Wörter die generell häufig vorkommen.

Wie in Formel 3.3 zu sehen, besteht die Termfrequenz tf eines Wortes i im Dokument j aus dessen Anzahl der Vorkommen $n_{i,j}$, welche mittels der Gesamtanzahl aller Wörter k im Dokument j normalisiert wird, sodass die Größe eines Dokuments keinen Einfluss auf die Termfrequenz hat.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{3.3}$$

Die Formel für die inverse Dokumentenhäufigkeit idf eines Wortes i ist in Formel 3.4 zu sehen. Dabei ist $|D|$ die Gesamtanzahl aller Dokumente im Korpus und $|\{d : t_i \in d\}|$ ist die Anzahl der Dokumente, in denen der Term t_i vorkommt. Falls der Term nicht in dem Dokumentenkorpus vorkommen sollte, wird durch die Addition von 1 im Nenner eine Division durch 0 verhindert.

$$idf_i = \log \frac{|D|}{1 + |\{d : t_i \in d\}|} \tag{3.4}$$

Das endgültige Gewicht $w_{i,j}$ für ein Wort i im Dokument j wird durch die Multiplikation von tf und idf errechnet und ist in Formel 3.5 zu sehen.

$$w_{i,j} = tf_{i,j} \cdot idf_i \tag{3.5}$$

Dieser Ansatz wird auf die Gewichtung der Einträge der Artist-Property-Matrix übertragen, um somit wichtige Eigenschaften für den nachfolgenden Clustering-Prozess zu identifizieren. Wörter werden dabei durch die Attribute der Künstler ersetzt und die Dokumente durch die Künstler. Somit entspricht der Dokumentenkorpus der Gesamtheit der Künstler, die ein Nutzer gehört hat. Die binäre Verknüpfung zwischen Künstler und Eigenschaft in der Artist-Property-Matrix wird nun durch das mittels TF-IDF errechnete Gewicht ersetzt. Das Gewicht $w_{i,j}$ entspricht somit dem Wert, den der Künstler a_i für das Attribut p_j besitzt. Ein kleiner TF-IDF-Wert bedeutet dabei, dass ein bestimmtes Attribut eine geringe Wichtigkeit besitzt, währenddessen ein hoher TF-IDF-Wert eine große Wichtigkeit eines Attributs ausdrückt.

3.3 Clustering

In diesem Schritt des Data-Mining-Prozesses werden die Künstler in der Artist-Property-Matrix in einer bestimmten Anzahl an Clustern zusammengefasst.

3.3.1 Clustering mittels k-Means-Algorithmus

Für das Clustering der Artist-Property-Matrix wird der k-Means-Algorithmus verwendet. Der Begriff k-Means wurde 1967 erstmals von James MacQueen in [Mac67] erwähnt, wobei die Idee des Algorithmus auf Überlegungen des polnischen Mathematikers Hugo Steinhaus zurückgeht. Der heute bekannte k-Means-Algorithmus wurde erstmals 1957 von Stuart Lloyd in [Llo82] vorgestellt, jedoch erst 1982 veröffentlicht und diente als Verfahren zur Puls-Code-Modulation. Es handelt sich bei dem k-Means-Algorithmus um den bekanntesten und einen der am häufigsten verwendeten Algorithmen zur Gruppierung von Objekten. Dabei werden die Objekte durch den Algorithmus in eine vorher festgelegte Anzahl von k Gruppen unterteilt, wobei versucht wird den durchschnittlichen Abstand zwischen den Clusterschwerpunkten sowie den restlichen Elementen der zu gruppierenden Menge zu minimieren. Ein Clusterschwerpunkt $\vec{\mu}$ von Elementen in einem Cluster ω wird in Manning et al. [MRS08] wie folgt definiert:

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x} \tag{3.6}$$

Die Definition geht davon aus, dass es sich bei den Clusterelementen $\vec{x}$ um normalisierte Vektoren in $\mathbb{R}$ handelt. Ein ideales Cluster besteht dabei aus einem Bereich um einen

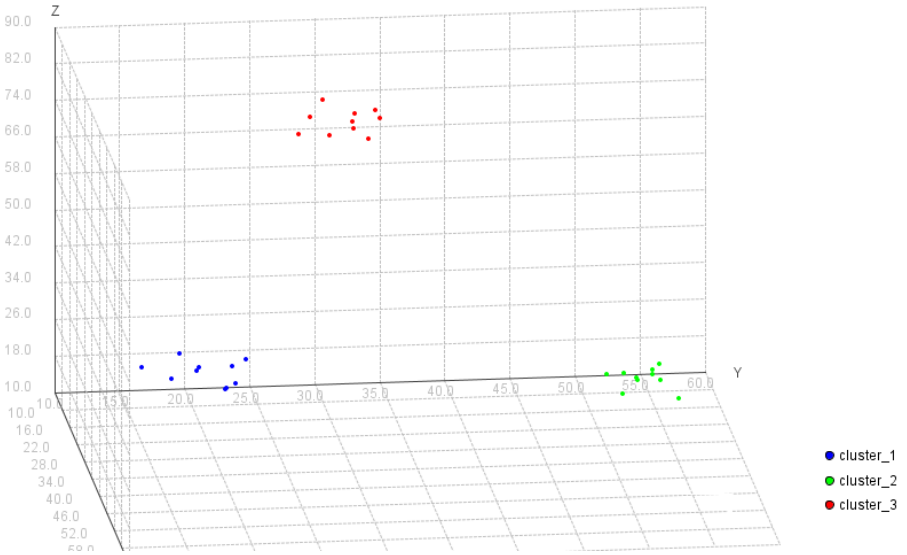


Abbildung 3.5: Beipiel für das Clustering von Punkten

Schwerpunkt herum, in dem sich die Clusterelemente befinden und es keine Überlappungen mit anderen Clustern gibt. Ein Beispiel für eine solche Clusterverteilung ist in Abbildung 3.5 dargestellt. Es werden dort drei Cluster von Punkten in einem dreidimensionalen Raum gebildet.

Die Vorgehensweise des k-Means-Algorithmus sieht wie folgt aus:

1. Die Schwerpunkte der k Cluster werden zufällig verteilt.
2. Jedes Element der Menge wird dem Cluster zugeteilt, dessen Schwerpunkt die geringste Entfernung zu ihm besitzt.
3. Eine Neuberechnung der Clusterschwerpunkte wird durchgeführt.
4. Die Schritte 2 und 3 werden solange wiederholt bis eine Abbruchbedingung erfüllt wird.

In dem Beispiel in Abbildung 3.6 werden 15 Elemente in 3 Cluster aufgeteilt. Dabei wird in 1.) die Initialisierung vorgenommen indem die drei Clusterschwerpunkte zufällig verteilt werden. In 2.) erfolgt die Zuteilung der Elemente zu den jeweiligen Clustern, wodurch die Schwerpunkte neu berechnet werden müssen. Dies ist in Schritt 3.) zu sehen und ergibt eine neue Verteilung der Elemente innerhalb der Cluster. Dieser Vorgang der Neuberechnung der Schwerpunkte und die daraus resultierende Umverteilung der Clusterelemente wiederholt sich in Schritt 4.) und endet in Schritt 5.) mit dem endgültigen Clusterergebnis, da keine weitere Veränderung an der Zuteilung der Clusterelemente herbeigeführt werden kann.

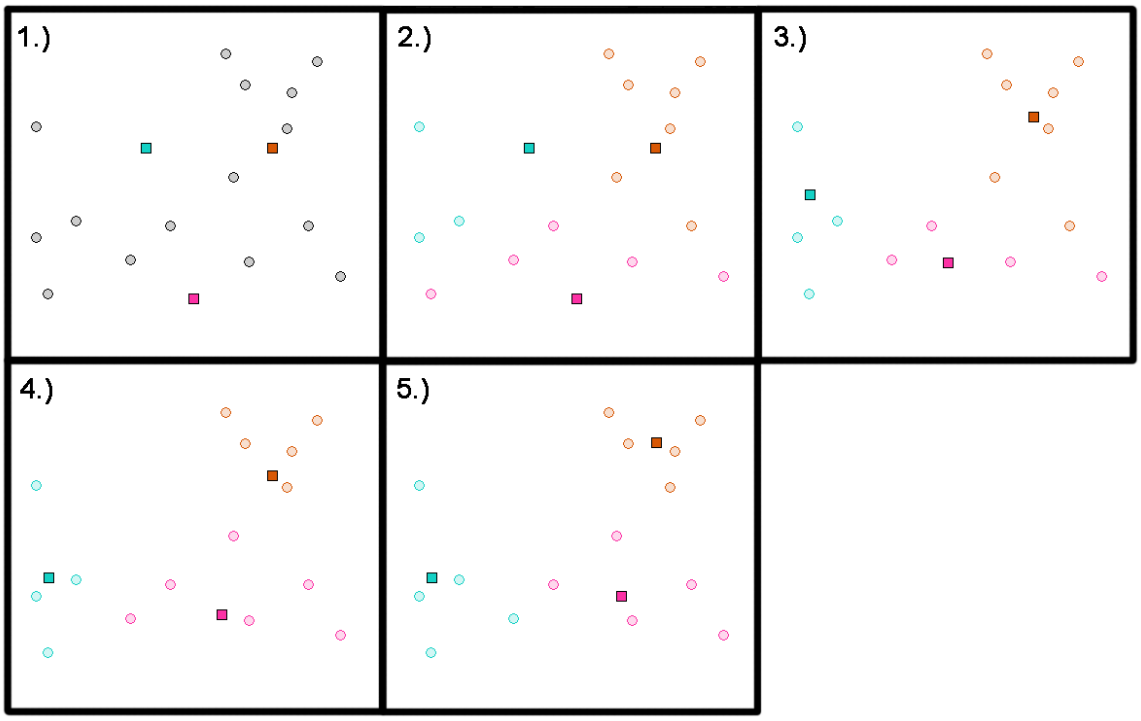


Abbildung 3.6: Beispiel für den Ablauf des k-Means-Algorithmus

Für das Clustering der Artist-Property-Matrix wird k in dem Intervall $(2 \leq k \leq 20)$ gewählt, da innerhalb des Intervalls die Anzahl der verschiedenen Musikrichtungen, die ein Nutzer hört, vermutet werden. Für jede Clusteranzahl k wird der Algorithmus zehn mal mit jeweils einer anderen Initialisierung durchgeführt. Dies soll einer Erzeugung ungünstiger Cluster entgegenwirken, da der Clusteringprozess aufgrund der zufälligen Initialisierung in einem lokalen Minimum enden kann. Auf die unterschiedlichen Ergebnisse, die dadurch entstehen können, wird im späteren Verlauf genauer eingegangen. Aus diesen zehn Versuchen wird dann das Ergebnis ausgewählt, bei dem die Verteilung der Cluster und deren Schwerpunkte am besten zu den Elementen der Menge passt. Ein Maß dafür, wie gut ein Schwerpunkt die Elemente seines Clusters repräsentiert, ist die *residual sum of squares* oder RSS (dt. Restsumme der Quadrate). Es handelt sich dabei um die Summe der Quadrate der Entfernung zwischen den Vektoren und dem Schwerpunkt des jeweiligen Clusters. Verdeutlicht wird dies in Formel 3.7, wobei $|\vec{x} - \vec{\mu}(\omega_a)|$ eine Metrik ist, durch die sich der Abstand zwischen einem Vektor und dem Schwerpunkt seines Clusters bestimmen lässt. Diese Metrik kann z.B. die Euklidische Norm, die Mahalanobis-Distanz oder eine andere Formel zur Bestimmung eines Abstands sein. In dem aktuellen Prozess zum Clustern der Artist-Property-Matrix wird die Euklidische Norm zur Berechnung des Abstands genutzt. Mittels RSS_a wird somit die Summe der Abstände aller Punkte zu den Schwerpunkten in einem Cluster ω_a bestimmt.

RSS entspricht dann der Summe aller Abstände in den Clustern $\omega_1, \ldots, \omega_k$. RSS kann somit als Zielfunktion angesehen werden, die es zu minimieren gilt.

$$RSS_a = \sum_{\vec{x} \in \omega_a} |\vec{x} - \vec{\mu}(\omega_a)|$$
$$RSS = \sum_{a=1}^{k} RSS_a \tag{3.7}$$

Bei der Iteration über die beiden Schritte (Zuteilung der Elemente zu Clustern und Neuberechnung der Schwerpunkte) des k-Means-Algorithmus werden eine oder mehrere Abbruchbedingungen benötigt. Die Abbruchbedingungen können ebenfalls variabel gewählt und somit an das entsprechende Anwendungsgebiet angepasst werden. Es stehen folgende Möglichkeiten zur Auswahl:

- Eine vorher festgelegte Anzahl an Iterationen I wird erreicht. Dadurch wird die Laufzeit des Algorithmus begrenzt, was allerdings zu nicht optimalen Ergebnissen führen muss, da gegebenenfalls nicht ausreichend Iterationen durchgeführt wurden.
- Die Zugehörigkeit zwischen den Elementen und den Clustern ändert sich zwischen den Iterationen nicht mehr. Mit Ausnahme von Fällen, in denen ein ungünstiges lokales Minimum erreicht wird, werden dadurch gute Ergebnisse erzielt, jedoch kann die Laufzeit inakzeptabel hoch werden.
- Die Schwerpunkte $\vec{\mu}(\omega_1), \ldots, \vec{\mu}(\omega_k)$ ändern sich zwischen den Iterationen nicht mehr.
- RSS fällt unter einen vorher festgelegten Grenzwert, wodurch eine Qualität des Clusterings nach der Terminierung des Algorithmus garantiert werden kann. In der Praxis muss dies aber mit einer Maximalanzahl an Iterationen kombiniert werden, um sicherzustellen dass eine Terminierung erfolgt.
- Die Abnahme von RSS fällt unter einen bestimmten Grenzwert θ. Für kleine θ bedeutet das, dass die Konvergenz von RSS fast erreicht ist, wobei auch hier eine Kombination mit einer Maximalanzahl an Iterationen benötigt wird.

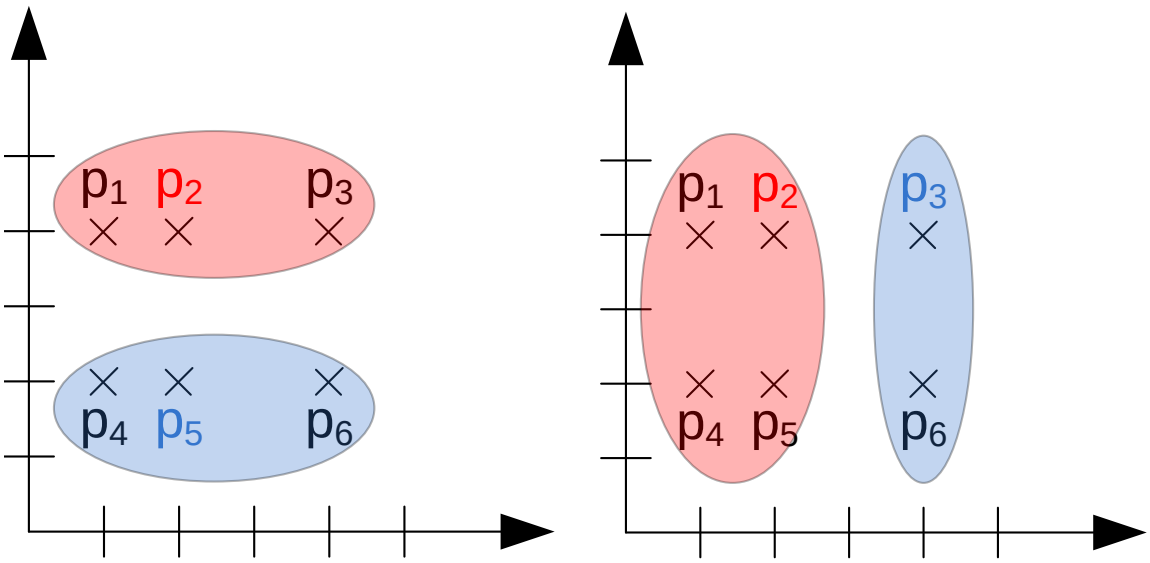
Als Abbruchbedingung in dem aktuellen Prozess wurde eine Begrenzung der Optimierungsiterationen gewählt und diese auf 100 begrenzt, um so die Laufzeit auf ein realisierbares Maß zu reduzieren.

Bei *RSS* handelt es sich um eine Funktion, welche mit jeder weiteren Optimierungsiterationen monoton fällt und sich dann einem Grenzwert annähert. Ein Problem bei der Minimierung der Funktion sind Punkte, die zu mehreren Clusterschwerpunkten den gleichen Abstand besitzen, da diese dazu führen können, dass der Algorithmus in einer Endlosschleife endet und stets einen gleichbleibenden Funktionswert besitzt. Dies kann aber durch die richtige Wahl der Abbruchbedingung verhindert werden.

Der k-Means-Algorithmus konvergiert zwar immer gegen einen Grenzwert und findet somit eine Lösung, es gibt aber keine Garantie dafür, dass stets das globale Minimum der Zielfunktion gefunden wird. Dieses Problem tritt häufig bei Mengen auf, die viele Ausreißer enthalten. Diese Ausreißer, die sich weit entfernt von anderen Elementen befinden, lassen sich dadurch schlecht zu den anderen Clustern zuordnen. Des Weiteren gibt es ein Problem, wenn solche Ausreißer als Initialisierungspunkte für Cluster gewählt werden. In diesem Fall werden in den nachfolgenden Iterationen keine anderen Elemente zu diesem Cluster zugeordnet. Man spricht dabei von einem einelementigen Cluster, wobei es höchstwahrscheinlich eine Clusterverteilung gäbe, die einen niedrigeren RSS-Wert besitzen würde. In Abbildung 3.7 ist ein Beispiel für eine unterschiedliche Verteilung der Cluster aufgrund der Initialisierung zu sehen. Dabei wurden in Abbildung 3.7(a) die Punkte p_2 und p_5 als Initialisierungspunkte gewählt, wodurch das Clustering in einem lokalen Minimum endet. Die Verteilung der Cluster enspricht dabei nicht der offensichtlichen Verteilung, wie sie in der Abbildung zu sehen ist. In Abbildung 3.7(b) wurden die Punkte p_2 und p_3 zur Initialisierung genutzt, wodurch k-Means in dem globalen Minimum konvergiert und somit die offensichtliche Verteilung der Cluster repräsentiert.

Es gibt verschiedene Methoden um eine gute Wahl der Initialisierungspunkte zu finden, wovon drei Ansätze in Manning et al. [MRS08] erwähnt werden. Für die Analyse der Nutzerdaten wird Methode 2 gewählt und somit das Clustering pro Clusteranzahl k, wie bereits erwähnt, zehn mal wiederholt.

1. Die Ausreißer werden vor der Wahl der Initialisierungspunkte entfernt.
2. Es werden unterschiedliche Initialisierungen vorgenommen und es wird das Ergebnis mit dem niedrigsten RSS-Wert gewählt.
3. Die Initialisierungspunkte werden mit Hilfe einer anderen Methode, wie z.B. hierarchisches Clustern bestimmt.



(a) k-Means konvergiert in lokalem Minimum (b) k-Means konvergiert im globalen Minimum

Abbildung 3.7: Vergleich der Ergebnisse von k-Means

3.3.2 Bestimmung der Clusteranzahl

Ein Problem des k-Means-Algorithmus ist die Bestimmung von k. Da dieser Wert ausschlaggebend für das Ergebnis des Clusteringprozesses ist muss er sinnvoll gewählt werden, sodass mit der Anzahl der Cluster die zugrunde liegenden Daten gut repräsentiert werden können. Ein einfacher Ansatz dafür wäre k in Abhängigkeit der Zielfunktion RSS zu wählen. Es wird dabei ein k gesucht, welches RSS minimiert. Sei $RSS_{min}(k)$ der minimale RSS-Wert für alle Cluster einer bestimmten Clusteranzahl k, so fällt RSS mit steigender Clusteranzahl. Da RSS für ein spezifisches k eine monoton fallende Funktion ist, erreicht $RSS_{min}(k)$ das Minimun 0, wenn $k = n$, wobei n die Anzahl der zu gruppierenden Elemente ist. Dies würde dazu führen, dass sich jedes Element in seinem eigenen Cluster befindet, wodurch kein optimales Ergebnis erreicht wird.

In Manning et al. [MRS08] wird eine Heuristik vorgestellt, mit deren Hilfe eine Schätzung für k geliefert werden kann. Durch die zehn unterschiedlichen Initialisierungen beim Clustering wird $\widehat{RSS}_{min}(k)$ definiert, was dem minimalen RSS-Wert der zehn Clusteringergebnisse entspricht. Für die Heuristik wird nun der Verlauf von $\widehat{RSS}_{min}(k)$ mit steigendem k untersucht, wodurch sich ein Verlauf ergibt wie er beispielhaft in Abbildung 3.8 zu sehen ist. Die

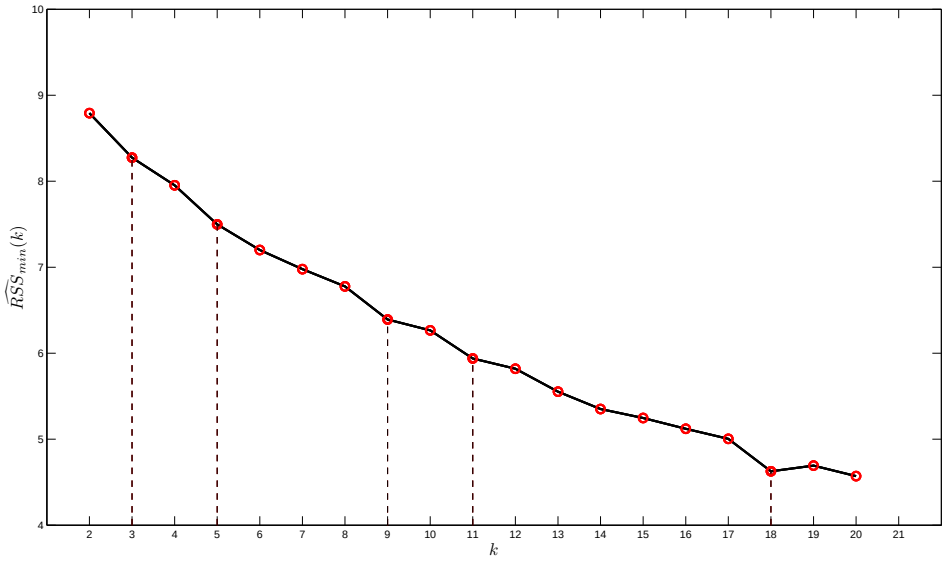


Abbildung 3.8: Beispiel für den Verlauf von $\widehat{RSS}_{min}(k)$

Differenz γ zwischen zwei Clusteranzahlen k und $k+1$ wird nun untersucht, wobei nach markanten Clusteranzahlen gesucht wird in denen γ deutlich kleiner, bzw. größer wird. Diese Clusteranzahlen werden im weiteren Verlauf als Favoriten bezeichnet, da diese als potentielle Kandidaten für die weitere Verarbeitung ausgewählt werden. Bei kleinem γ bedeutet dies, dass mit einer höheren Clusteranzahl keine sonderlich große Verbesserung herbeigeführt werden kann. Je größer γ wird, desto mehr ist dies ein Zeichen dafür, dass durch ein zusätzliches Cluster eine bessere Aufteilung der Daten möglich ist.

Die Punkte werden dabei nach einem Schema wie es in Algorithmus 3.1 zu sehen ist gewählt, wobei durch $\hat{\gamma}(a,b)$ die prozentuale Abnahme von $\widehat{RSS}_{min}$ zwischen den Clusteranzahlen a und b bestimmt wird. Von den ermittelten Favoriten werden die ersten fünf ausgewählt und für die weitere Verarbeitung genutzt. In dem Beispiel in Abbildung 3.8 sind die fünf Clusteranzahlen, welche als Favoriten gespeichert werden durch gestrichelte Linien gekennzeichnet.

Algorithm 3.1 Heuristik zur Auswahl von Clusteranzahlen aus $2 \leq k \leq 20$

```
favorits := ∅
for i = 1 to k − 2 do
  if γ̂(k, k+1) > 5% and γ̂(k+1, k+2) < 5% then
    favorits:= favorits ∪ {k+1}
  else if γ̂(k, k+1) < 5% and γ̂(k+1, k+2) > 5% then
    favorits:= favorits ∪ {k+2}
  end if
end for
```

Anzahl der Künstler i	Wahl aus den Favoriten
$i \leq 250$	1.
$250 < i \leq 300$	2.
$300 < i \leq 350$	3.
$350 < i \leq 400$	4.
$i > 400$	5.

Tabelle 3.2: Heuristik zur Wahl der Clusteranzahl

Um aus der so entstandenen Menge eine spezifische Clusteranzahl auszuwählen wird sich einer weiteren Heuristik bedient. Die Heuristik zur Auswahl der Clustering-Granularität wurde am DFKI im Projekt SocialMediaMiner empirisch entwickelt und wird in diesem Ansatz als erste Annäherung übernommen. Sie richtet sich nach der Anzahl der Künstler, die in der Artist-Property-Matrix vorhanden sind. Falls notwendig können die konkreten Werte der Heuristik in weiteren Optimierungsschritten des Algorithmus neu gelernt, bzw. an den konkreten Anwendungsfall angepasst werden, was allerdings nicht Bestandteil dieser Arbeit ist, da es nicht für notwendig erachtet wurde. Für eine grobe Granularität wird dabei der erste Wert aus den Favoriten gewählt, für eine feinere Granularität der Zweite, etc. In Tabelle 3.2 ist die Heuristik für die Wahl der Favoriten zu sehen. Sollte die Heuristik einen höheren Wert vorschreiben, als in den Favoriten gespeichert ist, so wird der letzte Wert, welcher sich in den Favoriten befindet, gewählt.

3.4 Merkmalsauswahl

In diesem Abschnitt wird beschrieben, wie die verschiedenen Merkmale innerhalb der Cluster ausgewählt werden, welche als beschreibende Labels für die Cluster fungieren sollen. Dazu werden zwei Methoden vorgestellt, mit denen Merkmale ausgewählt werden können, die besonders repräsentativ für ein bestimmtes Cluster sind. Zum Einen wird eine häufigkeitsbasierte Merkmalsauswahl über die Gewichte der einzelnen Merkmale eines Clusters vorgenommen und zum Anderen wird eine Merkmalsauswahl mittels Chi-Quadrat-Test vorgenommen. Es erfolgt anschließend eine Auswahl der wichtigsten Merkmale, welche einen bestimmten Schwellwert s überschreiten. Die Auswertung zur Wahl von s wird in Kapitel 5 vorgenommen. Es kann durchaus vorkommen, dass bei dem Clusteringprozess leere Cluster erstellt werden. Die Merkmalsauswahl würde daher für diese Cluster keine Ergebnisse liefern, weshalb die leeren Cluster vor der weiteren Verarbeitung entfernt werden.

3.4.1 Häufigkeitsbasierte Merkmalsauswahl

Bei dieser Methode werden die gewichteten Einträge der Artist-Property-Matrix pro Cluster aufsummiert. Dadurch können, aufgrund der TF-IDF-Gewichte, relevante Merkmale eines Clusters identifiziert werden, welche als Label genutzt werden können. Sei dabei $A_x = \{a_1, a_2, \ldots, a_\beta\}$ die Menge der Künstler in Cluster ω_x und $w_{i,j}$ der gewichtete Wert, den der Künstler a_i für das Attribut p_j besitzt. Es ergibt sich daraus die Berechnung des Relevanzwertes $r(p_j, \omega_x)$ für das Attribut p_j in dem Cluster ω_x, welcher formal in Formel 3.8 dargestellt ist. Das Label eines Clusters wird aus dessen relevantesten Attributen in Abhängigkeit des Top-Attributs des Clusters gewählt. Diese werden in prozentualer Abhängigkeit bestimmt, wobei sie den Schwellwert s überschreiten müssen. s wird dabei auf fünf verschiedene Stufen festgelegt, wodurch sich fünf unterschiedliche Relevanzklassen für die Labels der jeweiligen Cluster ergeben, deren Güte im Zuge der Evaluation überprüft wird.

$$r(p_j, \omega_x) = \sum_{a_i \in A_x} w_{i,j} \tag{3.8}$$

3.4.2 Merkmalsauswahl mittels Chi-Quadrat-Test

Bei der zweiten Methode, welche zur Merkmalsauswahl genutzt wird, handelt es sich um den sehr bekannten Chi-Quadrat-Test, welcher ebenfalls in Manning et al. [MRS08] beschrieben wird. Der Chi-Quadrat-Test wird in der Statistik eingesetzt, um die Unabhängigkeit zweier Ereignisse A und B zu testen. Dabei wird die Unabhängigkeit der Ereignisse durch $P(A \cap B) = P(A)P(B)$ oder gleichbedeutend mit $P(A|B) = P(A)$ und $P(B|A) = P(B)$ definiert. Die zwei Ereignisse sind in der Merkmalsauswahl das Vorkommen des Merkmals, sowie das Vorkommen der Klasse, bzw. des Clusters. Errechnet werden die Chi-Quadrat-Werte für ein Merkmal p und ein Cluster ω in dem Korpus aller Künstler D mittels der Formel 3.9. e_p und e_ω können die Werte $\{0, 1\}$ annehmen, wodurch eine Aussage darüber getroffen werden kann, ob das Merkmal p, bzw. das Cluster ω für den Korpus der Künstler D gilt oder nicht. Wenn z.B. $e_p = 1$ bedeutet es, dass nur die Künstler betrachtet werden, die das Merkmal p besitzen.

$$\chi^2(D, p, \omega) = \sum_{e_p \in \{0,1\}} \sum_{e_\omega \in \{0,1\}} \frac{(N_{e_p e_\omega} - E_{e_p e_\omega})^2}{E_{e_p e_\omega}} \tag{3.9}$$

	$e_{cluster\_1} = 1$		$e_{cluster\_1} = 0$	
$e_{Pop} = 1$	$N_{11} = 56$	$E_{11} \approx 6,2$	$N_{10} = 13.689$	$E_{10} \approx 13.738,8$
$e_{Pop} = 0$	$N_{01} = 236$	$E_{01} \approx 285,8$	$N_{00} = 630.364$	$E_{00} \approx 630.314,2$

Tabelle 3.3: Beispiel für das Merkmal „Pop“ und Cluster „*cluster_1*“

$$E_{e_p e_\omega} = N \cdot \frac{N_{e_p *}}{N} \cdot \frac{N_{* e_\omega}}{N} \tag{3.10}$$

Ein Beispiel für die Häufigkeiten ist in Tabelle 3.3 zu sehen. Dabei wird der Zusammenhang eines Merkmals „Pop“ und des Clusters „cluster_1“ veranschaulicht. $N_{e_p e_\omega}$ entspricht den gemessenen Häufigkeiten in Abhängigkeit von Merkmal und Cluster. Somit ist N_{11} in dem Beispiel die Anzahl an Künstlern, in denen das Merkmal „Pop“ vorkommt und die ebenfalls in Cluster „cluster_1“ liegen. Die erwarteten Häufigkeiten von Merkmal und Cluster werden mittels $E_{e_p e_\omega}$ angegeben. Angenommen, dass Merkmal und Cluster unabhängig sind, würde beispielsweise mit E_{11} die erwartete Häufigkeit der Künstler angegeben werden, in denen das Merkmal p vorkommt und die in dem Cluster ω liegen. Die Berechnung von $E_{e_p e_\omega}$ erfolgt über die Formel 3.10, wobei $N_{e_p *}$ der Summe der Häufigkeiten des Merkmals unabhängig von e_ω entspricht, d.h. N_{1*} ergibt sich in dem Beispiel aus $56 + 13.689$. Äquivalent dazu ergibt sich mit $N_{* e_\omega}$, die Summe der Häufigkeiten, welche von e_p unabhängig sind.

Die Formel für den Chi-Quadrat-Test, welche in dem Prozess zur Nutzeranalyse verwendet wurde, ist in der Gleichung 3.11 zu sehen. Es handelt sich dabei um eine arithmetisch vereinfachte Form, welche äquivalent zu der Formel 3.9 ist. Der Vorteil besteht darin, dass dadurch für die Berechnung der Chi-Quadrat-Werte lediglich die realen Häufigkeiten $N_{e_p e_\omega}$ benötigt werden und nicht die Erwarteten $E_{e_p e_\omega}$ errechnet werden müssen.

$$\chi^2(D, p, \omega) = \frac{(N_{11} + N_{10} + N_{01} + N_{00}) \times (N_{11} N_{00} - N_{10} N_{01})^2}{(N_{11} + N_{01}) \times (N_{11} + N_{10}) \times (N_{10} + N_{00}) \times (N_{01} + N_{00})} \tag{3.11}$$

4 Implementierung

In diesem Kapitel wird die Implementierung, die im Zuge der Arbeit entstanden ist, näher beschrieben und auf deren Besonderheiten eingegangen. Es werden die einzelnen Klassen vorgestellt und die verwendeten Funktionen erläutert. Des Weiteren befasst sich ein Abschnitt des Kapitels mit der Einbindung der Data-Mining-Bibliothek *Rapidminer* in die Applikation. Weiterhin wird die zugrunde liegende Struktur der Datenbank beschrieben und auf die darin enthaltenen Tabellen eingegangen.

4.1 Allgemeine Beschreibung

Für die Extraktion der Nutzerprofile wurde eine Konsolenapplikation mit Java[1] (Version 6 Update 24) entwickelt. Die Implementierung erfolgte nach dem Paradigma der Objektorientierten Programmierung, wodurch ein modularer Aufbau erreicht wurde, durch den einzelne Komponenten leicht durch andere ersetzt werden könnten. Zur Anwendungsentwicklung wurde die Entwicklungsumgebung Eclipse SDK[2] verwendet, welche in Version 3.6.1 vorliegt. Zusätzlich wurden die externe Bibliotheken MySQL Connector/J[3] (Version 5.1.13), sowie Rapidminer 5.0[4] verwendet. Das System, auf dem die Anwendung entwickelt und getestet wurde, lässt sich wie folgt spezifizieren:

- Betriebssystem: Windows 7 (64-bit),
- Prozessor: Intel Core 2 Quad Q6600 @ 2.40GHz,
- Arbeitsspeicher: 4GB.

4.2 Klassenübersicht

Eine Übersicht über die entstandenen Klassen ist in Abbildung 4.1 zu sehen. Die Abbildung zeigt die zehn Klassen des ProfileExtractors, sowie die darin enthaltenen Methoden. Im

[1] `http://www.java.com/`
[2] `http://www.eclipse.org/`
[3] `http://dev.mysql.com/`
[4] `http://rapid-i.com/`

folgenden werden diese Klassen näher beschrieben und auf ihre Funktion innerhalb des Projekts eingegangen.

4.2.1 ProfileExtraktor

Dies ist die Hauptklasse der Anwendung, welche die *main()*-Funktion enthält. Sie enthält die Hauptschleife, in der die Extraktion der Benutzerprofile gesteuert wird.

4.2.2 User

Die Klasse *User* modelliert die Nutzer, deren Playlisten verarbeitet werden sollen. Sie dient als Container für die relevanten Daten eines Nutzers, die aus der Datenbank extrahiert werden können. Die wichtigste Eigenschaft dabei ist eine Liste der Künstler, die der Nutzer gehört hat. In dem Konstruktor können Daten, wie z.B. Alter, Geschlecht, Herkunft, etc. übergeben werden. Diese Daten werden allerdings nicht zur Erstellung der Benutzerprofile herangezogen. Die restlichen Methoden der Klasse dienen zum Lesen und Schreiben der Klassenvariablen.

4.2.3 Artist

Mit dieser Klasse werden die Künstler modelliert, welche in der *musicdb*-Datenbank enthalten sind. Die Klasse dient als Container für den Namen des Künstlers, seine MusicBrainz-ID, eine Liste der gespielten Instrumente und Genres, die der Künstler bedient, sowie eine Liste der Veröffentlichungen. Wie bei der Klasse *User* werden dem Konstruktor diese Eigenschaften übergeben und die übrigen Methoden der Klasse dienen zum Lesen und Setzen der Klassenvariablen.

4.2.4 Release

Die Klasse *Release* stellt eine Veröffentlichung eines Künstlers dar. Sie dient als Container für Informationen, wie z.B. Namen, Label, Veröffentlichungsdatum, etc. Identifiziert wird die Veröffentlichung über ihre MusicBrainz-ID, welche bei der Erstellung eines Release-Objekts angegeben werden muss. Die übrigen Methoden der Klasse dienen auch hier dem Lesen und Schreiben der Klassenvariablen.

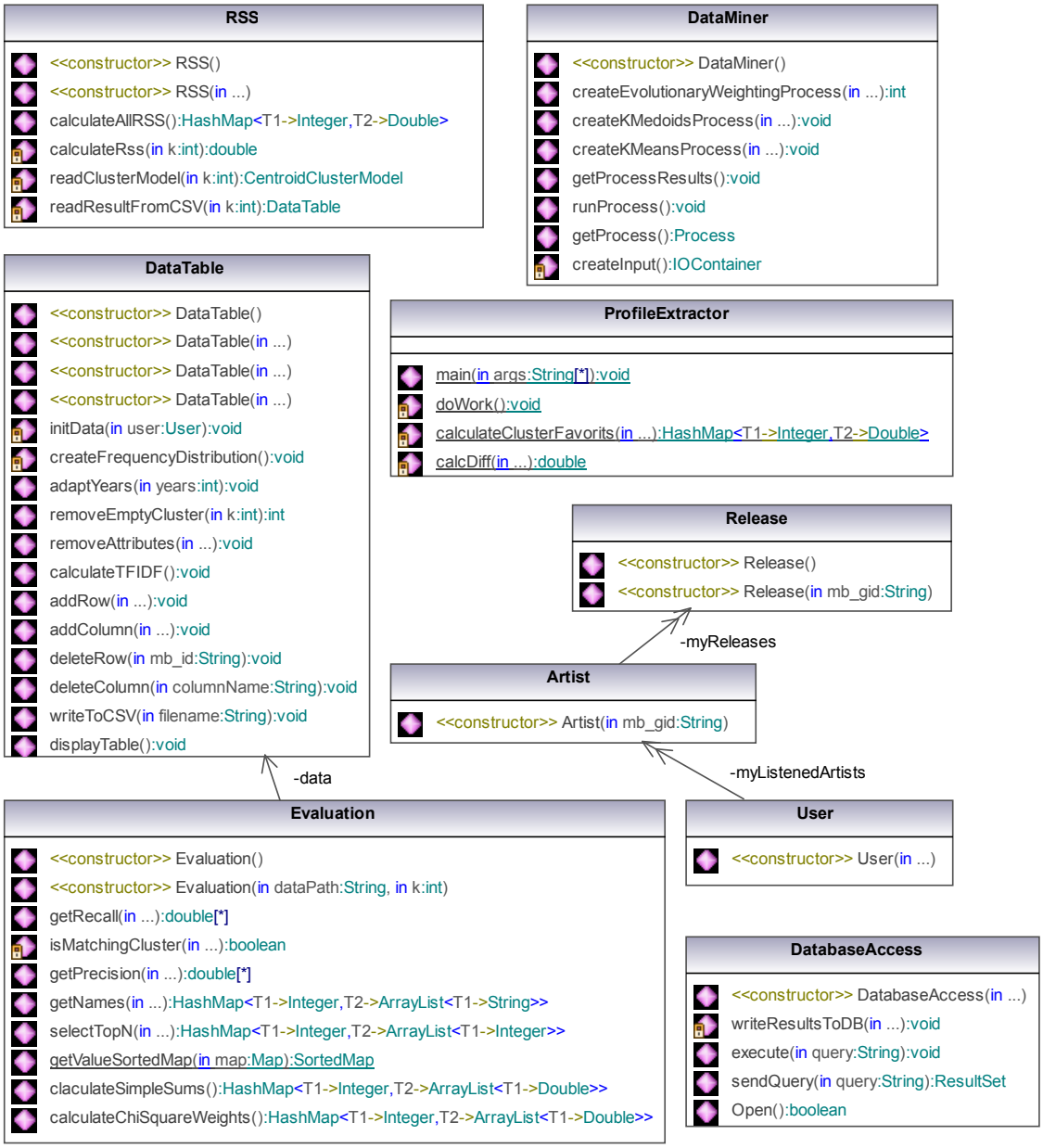


Abbildung 4.1: Klassendiagramm des ProfileExtractors

4.2.5 DataTable

Mit der Klasse *DataTable* ist die Erstellung einer Artist-Property-Matrix (s. Kapitel 3) möglich, d.h. die Verknüpfung zwischen Künstlern und den Attributen, die sie besitzen. Die Klasse ist abgeleitet von *AbstractTableModel* und besitzt zwei Listen, welche die Namen der Spalten, sowie der Zeilen enthalten. Die Speicherung der Daten wird durch eine *ArrayList* realisiert, welche wiederum eine *ArrayList* enthält, in der Werte vom Typ *Double* gespeichert werden können. Dadurch spannt sich ein zweidimensionales Feld auf, in dem die Verknüpfungen von Künstlern und deren Attributen dargestellt werden können. Es existieren verschiedene Konstruktoren für die Klasse, die verschiedene Anforderungen erfüllen. Die wichtigsten dabei sind zum Einen die Erstellung einer *DataTable* aus einem Objekt vom Typ *User* und zum Anderen die Erstellung einer *DataTable* aus einer CSV-Datei, die durch einen Dateipfad angegeben wird. Zusätzlich werden Funktionen zur Verfügung gestellt, die es erlauben die Häufigkeitsverteilung der Attribute der Matrix zu bestimmen oder mit denen es möglich ist, die Matrix als CSV-Datei zu exportieren.

4.2.6 DatabaseAccess

Diese Klasse dient dem Zugriff auf eine MySQL-Datenbank. Zu diesem Zweck wird die Bibliothek MySQL Connector/J eingebunden. Der Konstruktor der Klasse nimmt den Namen des Datenbanktreibers, den Namen der Datenbank, mit der eine Verbindung hergestellt werden soll, den Benutzernamen und das Passwort entgegen. Die Klasse stellt Funktionen zur Verfügung, mit denen es möglich ist eine Verbindung zu der Datenbank aufzubauen und zu beenden. Mittels zwei Methoden können zum Einen Befehle und zum Anderen Abfragen, die ein Ergebnis liefern, an die Datenbank gesendet werden.

4.2.7 DataMiner

In dieser Klasse wird die RapidMiner-Bibliothek eingebunden. In dem Konstruktor wird die notwendige Initialisierung der Bibliothek vorgenommen. In der Klasse sind Methoden enthalten, die zur Erstellung eines Prozesses dienen. Der Prozess kann dabei z.B. zuvor mittels Rapidminer-GUI erstellt worden sein. Anhand des Dateipfades zur Prozessdatei ist es dann möglich den Prozess in die Applikation zu laden. Eine Methode zur Ausführung des Prozesses ist ebenfalls Bestandteil der Klasse.

4.2.8 RSS

Auch in dieser Klasse wird die RapidMiner-Bibliothek verwendet. Dadurch können die Ergebnisse des Data-Mining-Prozesses geladen und ausgewertet werden. Dem Konstruktor werden Teile der Dateipfade zu Clustermodellen, sowie Ergebnistabellen übergeben, die durch das vorhergehende Clusteringverfahren entstanden sind. Die Klasse enthält Methoden, mit denen es möglich ist diese Clustermodelle zu lesen und mittels der Daten aus den Artist-Property-Matrizen die RSS-Werte zu bestimmen.

4.2.9 Evaluation

Durch diese Klasse wird die Funktionalität zum Evaluieren der Ergebnisse bereitgestellt. Des Weiteren lassen sich mit der Klasse die Labels für die einzelnen Cluster bestimmen. Der Konstruktor nimmt einen Dateipfad zu einer CSV-Datei entgegen, die die Ergebnisse des Clusteringprozesses enthält, sowie die Anzahl der Cluster k. Die Klasse besitzt eine Memberklasse *ValueComparer*, die das Interface *Comparator* implementiert. Die Memberklasse erfüllt den Zweck zwei Terme, die im Datentyp *Map* enthalten sind, zu vergleichen. Somit ist es möglich eine *Map* absteigend, nach der Relevanz ihrer Einträge, zu sortieren und dadurch die Auswahl der beschreibenden Terme für ein Cluster zu bestimmen.

4.3 Methodenübersicht

In diesem Abschnitt werden die wichtigsten Methoden der einzelnen Klassen näher beschrieben. Dazu wird auf die wichtigsten Übergabeparameter eingegangen, sowie die interne Verarbeitung geschildert.

4.3.1 ProfileExtractor:doWork()

Bei *doWork()* handelt es sich um die Methode, in der die Extraktion der Profile, sowie die Evaluation gesteuert wird. Ein Überblick über die wichtigsten Elemente des Ablaufs der Methode ist in Abbildung 4.2 in Form eines Flussdiagramms dargestellt. Die Methode stellt zu Beginn eine Verbindung zu den beiden Datenbanken *lastfm* und *musicdb* her, um den Zugriff auf die benötigten Daten zu erhalten. Sobald die Verbindung geöffnet ist, werden alle Nutzer mit dem Befehl

```
SELECT userID FROM lastfm.userprofiles
```

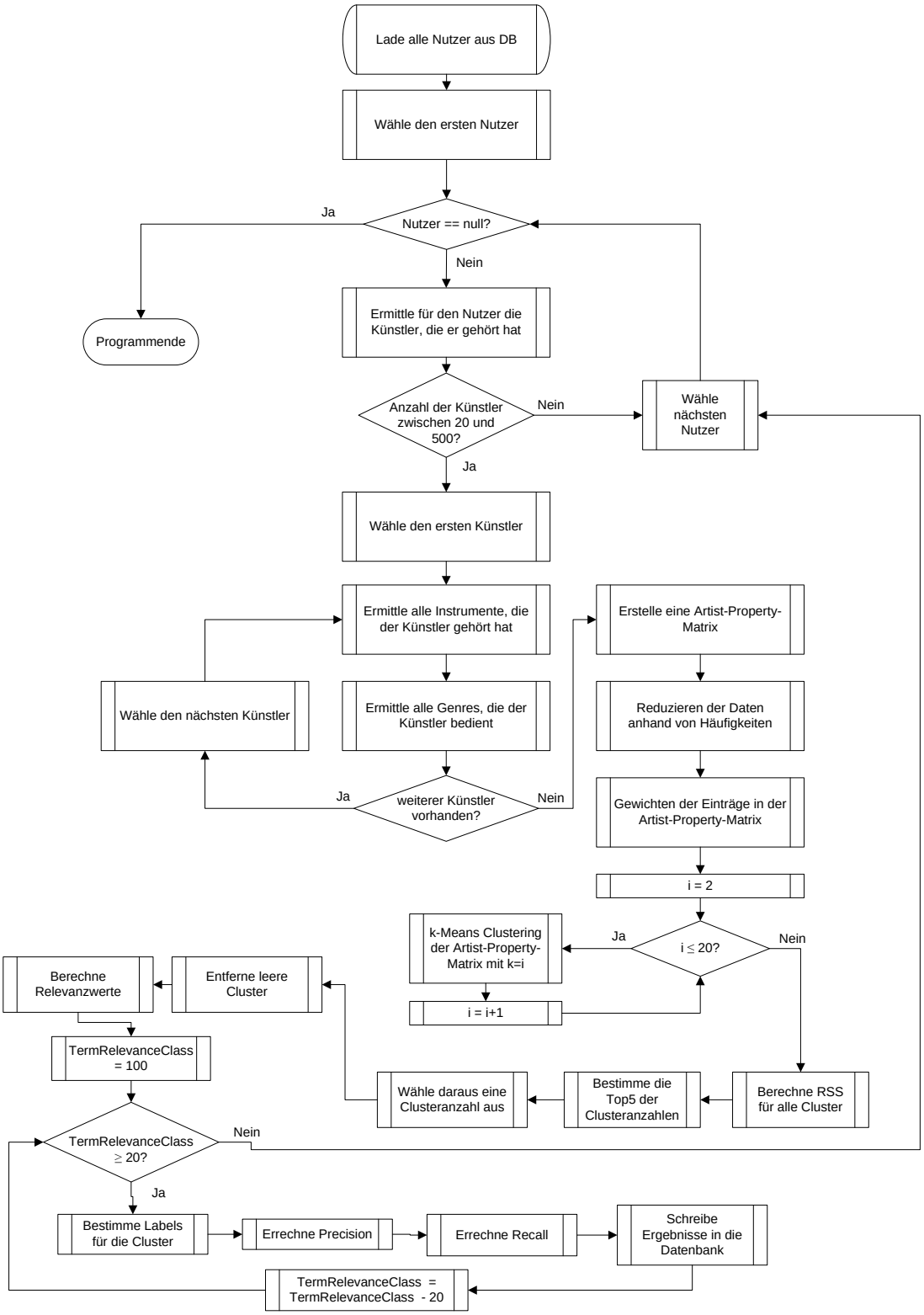


Abbildung 4.2: Flussdiagramm der Methode *doWork()*

aus der Datenbank geladen und in einer Liste von *User*-Objekten gespeichert. Der Hauptrahmen der Methode besteht nun in der Iteration durch die Liste der Nutzer. Für einen Nutzer werden dann mittels dem SQL-Befehl

```
SELECT DISTINCT musicBrainzArtistID FROM lastfm.playlists
WHERE userID = '<userID>' AND musicBrainzArtistID != ""
```

alle Künstler aus der *lastfm*-Datenbank geladen, die von ihm gehört wurden. Es folgt eine Iteration durch die Liste der Künstler, wobei zu jedem Künstler die vorhandenen Daten aus der *musicdb*-Datenbank geladen werden. Die Instrumente, die von einem Nutzer gespielt werden, lassen sich durch

```
SELECT DISTINCT a.instrument_family
FROM musicdb.instruments AS a, musicdb.artistToInstrument AS b
WHERE b.artist_ID = '<artistID>'
AND a.ID = b.instrument_ID
```

ermitteln und werden dem aktuellen Objekt eines Künstlers hinzugefügt. Um die Genres eines Künstlers zu ermitteln wird folgender SQL-Befehl erstellt und an die Datenbank gesendet:

```
SELECT DISTINCT a.genre_name
FROM musicdb.genres AS a, musicdb.artisttogenre AS b
WHERE b.artist_ID = '<artistID>'
AND a.ID = b.genre_ID
```

Da nach der Iteration durch die Künstler alle benötigten Daten vorhanden sind, kann anschließend eine Artist-Property-Matrix erstellt werden. Der Aufbau einer solchen Matrix wurde bereits in Kapitel 3 ausführlich beschrieben. Dazu wird mit Hilfe des *User*-Objekts ein Objekt der Klasse *DataTable* erstellt. Im folgenden Schritt erfolgt die Vorverarbeitung der Daten. Dazu wird eine Filterung nach Häufigkeiten der Attribute vorgenommen. Anschließend werden die verbleibenden Einträge der Matrix mittels TD-IDF-Maß, das im Kapitel 3 näher beschrieben wurde, gewichtet. Diese Matrix wird dann mittels k-Means geclustert, wobei das Ergebnis des Clusterings, sowie das Clustermodell auf der Festplatte gespeichert werden.

Von diesen Ergebnissen werden dann mittels der Klasse *RSS* die RSS-Werte der Cluster bestimmt und in einer *HashMap* gespeichert. Mittels dieser *HashMap* und der Methode *calculateClusterFavorits* werden dann Favoriten von Clusteranzahlen bestimmt, die für die weitere Verarbeitung genutzt werden. Aus diesen Favoriten wird anschließend mittels

Heuristik (s. Kapitel 3) ein Kandidat gewählt, mit dem die Verarbeitung fortgesetzt wird. Mit Hilfe der Mehtode *removeEmptyCluster* wird innerhalb der Daten nach leeren Clustern gesucht und falls vorhanden eine Neuzuteilung der Cluster-Nummern vorgenommen, sowie der Parameter k neu bestimmt.

Anschließend wird mit der Extraktion der Labels begonnen. Dazu wird ein Objekt der Klasse *Evaluation* erstellt mit dessen Hilfe die Relevanzwerte der Terme bestimmt werden können. Diese werden ebenfalls in einer *HashMap* gespeichert und absteigend nach ihrer Relevanz sortiert. In einer Schleife wird dann der Relevanzschwellwert schrittweise herabgesetzt. Alle Attribute, die über dem entsprechenden Schwellwert liegen, werden dann zu dem Label hinzugefügt.

Zum Abschluss der Methode werden von den entsprechenden Labeln in den unterschiedlichen Relevanzklassen die Messgrößen Recall und Precision errechnet (s. Kapitel 5) und in *double*-Arrays gespeichert. Die gewonnenen Daten werden dann unter Angabe von Nutzer-ID, Cluster-Nummer, Relevanzklasse, Recall, Precision, Labelnamen und einem *String*, der die Methode beschreibt (bspw. „kMeans,chiSquare“) in die *musicdb*-Datenbank eingefügt.

4.3.2 DataTable:initData(User actualUser)

In dieser Methode der Klasse *DataTable* wird die Initialisierung der Artist-Property-Matrix vorgenommen. Dazu werden mittels der Befehle

```
SELECT genre_name FROM musicdb.genres,
SELECT DISTINCT instrument_family FROM instruments
```

die in der *musicdb*-Datenbank vorhandenen Genre-Namen und Instrumenten-Familien abgefragt. Gespeichert werden die Ergebnisse, welche mit einem Präfix für die jeweilige Kategorie versehen werden, in einer *ArrayList*. Diese Liste fungiert als Kopfzeile in der Artist-Property-Matrix und enthält die Spaltennamen. Im nachfolgenden Schritt wird für jeden gehörten Künstler eine Zeile mit Nullen zu der Matrix hinzugefügt. Abschließend werden die Spaltennamen alphabetisch sortiert.

4.3.3 DataTable:DataTable(User actualUser)

In diesem Konstruktor der Klasse wird am Anfang das Element „Artist_MB_ID“ zu der Liste der Spaltennamen hinzugefügt. Diese Spalte ist somit für die MusicBrainz-IDs der

Künstler vorgesehen. Danach wird die Methode *initData* aufgerufen. Anschließend wird durch alle Künstler iteriert, die der Nutzer gehört hat. Es werden dann die Eigenschaften der Künstler abgerufen und an den entsprechenden Stellen in der Matrix durch eine „1“ markiert. Abschließend wird mit der Methode *createFrequencyDistribution* die Häufigkeitsverteilung der Attribute bestimmt.

4.3.4 DataTable:DataTable(String path)

Bei diesem Konstruktor wird ein Dateipfad zu einer CSV-Datei übergeben, welche eine Artist-Property-Matrix enthält. Somit können gespeicherte Daten in das Programm geladen werden. Es erfolgt zu Beginn das Einlesen der Datei mittels *FileReader* und *BufferedReader*, die im Paket „java.io“ enthalten sind. Als erstes wird dabei die Kopfzeile ausgelesen und separat gespeichert. Anschließend wird die Spalte bestimmt in der sich die MusicBrainz-IDs der Künstler befinden und deren Index an die erste Stelle in der Liste verlegt, um eine gleichbleibende Struktur der Matrix zu gewährleisten. Im Folgenden wird die Datei zeilenweise ausgelesen und die einzelnen Elemente in die Artist-Property-Matrix eingefügt. Am Ende des Konstruktors wird ebenfalls die Häufigkeitsverteilung der Attribute mit der Methode *createFrequencyDistribution* bestimmt.

4.3.5 DataTable:removeAttributes(float lowerBorder)

Mit dieser Methode können Attribute entfernt werden, die eine bestimmte Anzahl an Vorkommen unterschreiten. Dabei wird die untere Grenze als Prozentangabe übergeben. Zu Beginn wird mittels der Prozentangabe und der Anzahl der Künstler die absolute Grenze bestimmt. Es erfolgt anschließend eine Iteration über die Spalten der Artist-Property-Matrix. Es wird dann der entsprechende Wert der Häufigkeitsverteilung mit der Grenze verglichen und entschieden, ob die Spalte gelöscht werden muss.

4.3.6 DataTable:calculateTFIDF()

In dieser Methode werden die Einträge der Artist-Property-Matrix mittels TF-IDF-Maß gewichtet. Dazu wird anfangs in zwei verschachtelten Schleifen die Anzahl der Vorkommen der einzelnen Attribute pro Künstler bestimmt. Es folgen wiederum zwei verschachtelte *for*-Schleifen in denen mittels der Formel 3.5, die in Kapitel 3 vorgestellt wird, die Gewichte für die einzelnen Einträge der Artist-Property-Matrix errechnet werden.

4.3.7 RSS:calculateRss(int k)

Diese Methode errechnet den RSS-Wert für eine Clusteranzahl k. Es werden dazu die Ergebnisse des Clusteringprozesses und das entstandene Clustermodell gelesen. Es werden anschließend die Zentroide der einzelnen Cluster ausgelesen und in einer *ArrayList* gespeichert. Es folgen zwei verschachtelte *for*-Schleifen über alle Elemente der Artist-Property-Matrix, in denen zuerst die Differenz zwischen dem aktuellen Eintrag der Matrix, sowie dem Zentroid des Clusters, in dem der Künstler liegt, bestimmt wird. Diese Differenz wird quadriert und zu einer *double*-Variable addiert, in der der RSS-Wert des aktuellen Clusters gespeichert wird.

4.3.8 DataMiner:createKMeansProcess(int k, String process, String data)

Diese Methode dient dem Laden eines Rapidminer-Prozesses, dessen Stelle im Dateisystem durch die Zeichenkette „process“ spezifiziert wird. Dieser Prozess kann, wie bereits erwähnt, durch andere Hilfsmittel (bspw. Rapidminer-GUI) erstellt worden sein. Die Erstellung eines Prozess-Objektes erfolgt mittels *myProcess = new Process(new File(process))*, wobei alle Operatoren, deren Verbindungen und Einstellungen aus dem geladenen Prozess übernommen werden. Um nun Einstellungen an diesem Prozess vornehmen zu können, müssen die entsprechenden Operatoren aus dem Prozess extrahiert werden. Dies wird durch die Funktion *Process:getRootOperator().getAllInnerOperators()* realisiert, wobei alle Operatoren in einer *List⟨Operator⟩* gespeichert werden. Da die Typen der Operatoren allerdings unbekannt sind, wird durch einen Vergleich von *Operator:getClass()* und der Klasse des entsprechenden Operators (z.B. *KMeans.class*) der Typ bestimmt.

Nun können die Parameter der Operatoren entsprechend den Anforderungen des Hauptprogramms verändert werden. So lässt sich beispielsweise der Parameter k für den KMeans-Operator durch *Operator:setParameter(KMeans.PARAMETER_K, String.valueOf(k))* ändern. Im Allgemeinen nimmt die *setParameter*-Methode Parameter in Form von „key“ und „value“ an, wobei der „key“ abhängig von dem jeweiligen Operator ist. Äquivalent dazu werden im weiteren Verlauf der Methode die Parameter für die Operatoren *CSVExampleSetWriter*, *CSVDataReader* und *ClusterModelWriter* angepasst.

4.3.9 Evaluation:calculateChiSquareWeights()

Hierbei werden die Ergebnisse für den Chi-Quadrat-Test errechnet, welche für die Merkmalsauswahl benötigt werden. Gespeichert werden sie in einer *HashMap*, wobei der „key“ eine Cluster-Nummer enthält und der dazugehörige „value“ eine *ArrayList* mit den Ergebnissen für die einzelnen Attribute. Die Berechnung der Chi-Quadrat-Werte erfolgt mittels drei verschachtelter *for*-Schleifen. Die Erste iteriert über die Cluster, die Zweite und Dritte über alle Werte der Artist-Property-Matrix. Dabei werden die Werte $N_{00},\ N_{01},\ N_{10}$ und N_{11}, welche in Kapitel 3.4.2 beschrieben werden, bestimmt. Anschließend wird der Chi-Quadrat-Wert mittels der Formel 3.11 errechnet und in eine *ArrayList* eingefügt.

4.3.10 Evaluation:claculateSimpleSums()

Ähnlich der Methode *calculateChiSquareWeights()* werden bei dieser Methode die Relevanzwerte errechnet, die für die Merkmalsauswahl erforderlich sind. Hierbei erfolgt allerdings eine häufigkeitsbasierte Merkmalsauswahl anhand der TF-IDF-Gewichte der einzelnen Attribute. Gespeichert wird das Ergebnis nach dem gleichen Schema wie bei dem Chi-Quadrat-Test in einer *HashMap*. Auch hierbei erfolgt die Berechnung innerhalb drei verschachtelter *for*-Schleifen. Es werden dabei die TF-IDF-Gewichte eines Attributs für alle Künstler aufsummiert, die in dem aktuellen Cluster liegen. Die Relevanzwerte der einzelnen Attribute werden ebenfalls in einer *ArrayList* gespeichert und anschließend in die *HashMap* eingefügt.

4.3.11 Evaluation:selectTopN(HashMap⟨Integer, ArrayList⟨Double⟩⟩results, int threshold)

Mittels dieser Methode können Merkmale, abhängig von dem aktuellen Grenzwert für die Relevanz, als Labels für die jeweiligen Cluster ausgewählt werden. In der *HashMap* „results“ befinden sich die errechneten Relevanzwerte, wobei der Schlüssel eine Cluster-Nummer enthält und die dazu gehörende *ArrayList* die Werte für die einzelnen Attribute.

Zu Beginn der Methode iteriert eine *for*-Schleife über die einzelnen Cluster in der *HashMap*. Es werden anschließend die Relevanzwerte der Terme absteigend nach ihrer Größe sortiert. Der konkrete Grenzwert für die Auswahl der beschreibenden Terme wird anhand des „threshold“-Parameters, welcher eine Prozentangabe enthält, und dem Wert des relevantesten Attributs bestimmt. Somit ergibt sich bspw. bei einer Grenze von 60% und einem Relevanzwert des Top-Attributs von 150 ein Grenzwert von 90 für die Auswahl anderer Attribute. Es folgt eine

Iteration über die einzelnen Attribute des Clusters und der Vergleich mit dem Grenzwert. Sollte der Relevanzwert eines Attributs größer gleich dem aktuellen Grenzwert sein, so wird dessen Index in eine *ArrayList* aufgenommen. Andernfalls wird die Iteration beendet und die Liste der Indices unter Angabe der aktuellen Cluster-Nummer in einer *HashMap* gespeichert.

4.3.12 Evaluation:getRecall(HashMap⟨Integer, ArrayList⟨Integer⟩⟩topN)

Diese Methode errechnet den Recall der Labels in ihren Clustern (s. Kapitel 5). Die Funktion bekommt eine *HashMap* übergeben, welche die Cluster-Nummern und deren Labels enthält. Anfangs wird in einer *for*-Scheife pro Cluster die Anzahl der Künstler bestimmt, die sich darin befinden. Es folgt eine Bestimmung der Künstler, welche die entsprechenden Cluster und deren Labels matchen. Anschließend wird anhand der Formel 5.2 für jedes Cluster der Recall errechnet.

4.3.13 Evaluation:getPrecision(HashMap⟨Integer, ArrayList⟨Integer⟩⟩topN)

Mittels dieser Methode wird die Precision der Labels in ihren Clustern bestimmt (s. Kapitel 5). Auch diese Funktion bekommt die Daten, wie Cluster-Nummer und Label, mittels einer *HashMap* übergeben. Zu Beginn der Methode wird in zwei verschachtelten *for*-Schleifen die Anzahl der Künstler bestimmt, welche mittels der Cluster-Labels gefunden werden. Anschließend wird die Anzahl der Künstler bestimmt, die in dem entsprechenden Cluster liegen. Mittels diesen beiden Messwerten wird dann die Precision (s. Formel 5.1) für jedes Cluster errechnet.

5 Evaluation

In diesem Kapitel wird die Evaluation des Verfahrens zur Erstellung der Kontext-sensitiven Benutzerprofile beschrieben. Dazu wird anfangs auf den Evaluationsvorgang eingegangen und einige Grundlagen, die dafür benötigt werden, näher erläutert. Im Anschluss daran erfolgt die Vorstellung und die Auswertung der Ergebnisse, die mit diesem Verfahren erreicht werden konnten.

5.1 Vorgehen

Die Evaluation der ermittelten Nutzerprofile erfolgt anhand der gefundenen Labels in ihren Clustern. Diese werden dahingehend überprüft, wie gut es mit ihnen möglich ist die einzelnen Cluster voneinander zu trennen und wie spezifisch die Labels für die jeweiligen Cluster sind. Dies wird mittels der Messwerte Precision, Recall und F-Maß bestimmt, welche im Folgenden näher beschrieben werden. Es werden zu jedem Cluster jeweils fünf Labels erstellt, welche durch unterschiedliche Relevanzklassen entstehen. Diese werden mit Hilfe eines Schwellwerts ermittelt, der in Abhängigkeit des Top-Attributs entsteht. Es wird dazu eine prozentuale Grenze von 100%, 80%, 60%, 40% und 20% dessen Relevanzwerts festgelegt, wodurch mit sinkendem Schwellwert mehr Attribute zu dem Label hinzugefügt werden. Ein Beispiel für die Schwellwerte und die Verteilung der Attribute in den jeweiligen Klassen ist in Abbildung 5.1 zu sehen.

Die Erstellung der Labels erfolgt zum Einen auf Grundlage der Relevanzwerte, die durch die häufigkeitsbasierte Merkmalsauswahl errechnet werden und zum Anderen auf Basis des Chi-Quadrat-Tests. Somit kann die Qualität der Labels für diese beiden Verfahren zur Merkmalsauswahl verglichen werden. Von jedem Label, das auf diese Weise entstanden ist, werden anschließend Precision und Recall errechnet und in der *musicdb*-Datenbank gespeichert. Aus diesen beiden Messwerten wird dann das F-Maß errechnet, was ebenfalls

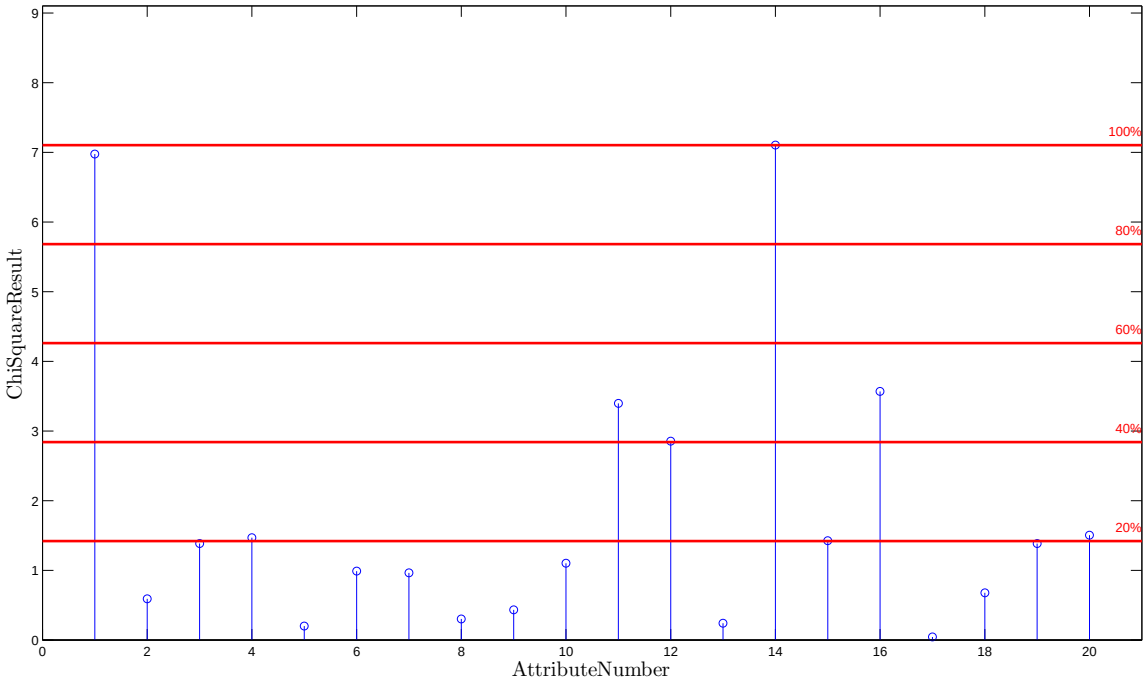


Abbildung 5.1: Beispiel für die Relevanzklassen und die Verteilung der Attribute

im folgenden Abschnitt erläutert wird.

Errechnet werden diese Werte pro Cluster, welche dann mittels dem arithmetischen Mittel pro Benutzer und der jeweiligen Relevanzklasse zusammengefasst werden. Der Gesamtdurchschnitt ergibt sich dann aus den einzelnen Werten der Benutzer, wodurch gezeigt werden kann wie groß Precision, Recall und F-Maß im Durchschnitt pro Benutzer sind. Anschließend erfolgt die Auswertung der Ergebnisse, welche mit dem in dieser Arbeit beschriebenen Verfahren erreicht werden konnten. Dazu werden die Messwerte bezüglich der unterschiedlichen Relevanzklassen verglichen. Zusätzlich dazu werden für die Messwerte Recall und F-Maß Lorenz-Kurven der unterschiedlichen Relevanzklassen erstellt, wodurch gezeigt werden kann für wie viele Benutzer sinnvolle Cluster erstellt werden können. Die theoretischen Grundlagen der Lorenz-Kurve werden ebenfalls im nächsten Abschnitt vorgestellt. Es wird hierbei auf die Precision verzichtet, da mit dem F-Maß bereits gezeigt werden kann, wie spezifisch die Labels für die entsprechenden Nutzer sind. Eine separate Betrachtung des Recalls soll dazu dienen, zu zeigen ob für eine gewisse Anzahl an Nutzern viele Künstler mit den Labels gefunden werden können. Abschließend werden beispielhaft verschiedene Labels vorgestellt, welche bei den Nutzern extrahiert werden konnten. Somit kann mit dem exemplarischen Benutzerprofil die Diversität in den präferierten Musikrichtungen veranschaulicht werden.

5.2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen behandelt, welche für die Evaluation nötig sind. Dazu werden im Folgenden die Messwerte Precision, Recall und das F-Maß beschrieben, da diese zur Ermittlung der Güte der Ergebnisse genutzt werden. Es handelt sich dabei um drei weit verbreitete Messwerte im Bereich des Information Retrieval. Des Weiteren wird auf die theoretischen Grundlagen der Lorenz-Kurve eingegangen, da diese ebenfalls für die Evaluation genutzt wird.

5.2.1 Precision

Mittels Precision kann die Genauigkeit der extrahierten Labels überprüft werden. Somit kann festgestellt werden, wie viel Prozent der gefundenen Künstler korrekte Ergebnisse sind. Die Precision ist in diesem Fall definiert, als das Verhältnis der Anzahl von Künstlern, die in dem entsprechenden Cluster liegen und das Label matchen (sum^1) und der Anzahl aller Künstler, die das Label matchen (sum^2). Formal ist dieses Verhältnis in der Formel 5.1 dargestellt. Der Wertebereich der Precision liegt im Intervall $[0, 1]$, wobei versucht wird den Wert der Precision zu maximieren.

$$precision = \frac{sum^1}{sum^2} \tag{5.1}$$

5.2.2 Recall

Ein weiteres Maß, das in diesem Zusammenhang genannt werden muss, ist der Recall. Mittels diesem Messwert wird die Vollständigkeit der gefundenen Ergebnisse überprüft. Somit kann das Verhältnis zwischen den Künstlern, die in dem Cluster mit dem Label ermittelt werden, und denen die durch das Label nicht gefunden werden, aber jedoch in dem Cluster liegen, bestimmt werden. Der Recall ist definiert als das Verhältnis zwischen der Anzahl an Künstlern, die in dem entsprechenden Cluster liegen und das Label matchen (sum^1) und der Anzahl aller Künstler in dem Cluster (sum^3). Dargestellt wird dies in Formel 5.2. Auch hierbei liegt der Wertebereich in dem Intervall $[0, 1]$ und es gilt den Wert des Recalls zu maximieren.

$$recall = \frac{sum^1}{sum^3} \tag{5.2}$$

Sinnvoll ist dabei jedoch nur die Betrachtung beider Messwerte, da sonst keine vollständige Auskunft über das Ergebnis getroffen werden kann. Bei alleiniger Betrachtung des Recalls, könnte dieser maximiert werden, was durch ein Label geschehen könnte, mit dem viele Künstler in einem Cluster gefunden werden können. Dies kann jedoch nur auf Kosten der Precision geschehen.

Andererseits ist die alleinige Betrachtung der Precision ebenso wenig sinnvoll, da diese auf Kosten des Recalls maximiert werden kann. Dies geschieht beispielsweise bei einem Label, mit dem nur Künstler gefunden werden, die in dem entsprechenden Cluster liegen. Dadurch bleibt jedoch die Vollständigkeit des Ergebnisses unbeachtet, was dazu führen würde, dass der Recall nur einen sehr geringen Wert erreichen würde. Daher wird bei vielen Evaluierungen der Recall-Precision-Graph verwendet, bei dem auf der x-Achse die Precision und auf der y-Achse der Recall abgetragen wird. Dies soll dabei helfen die beiden Größen miteinander in Beziehung zu setzen.

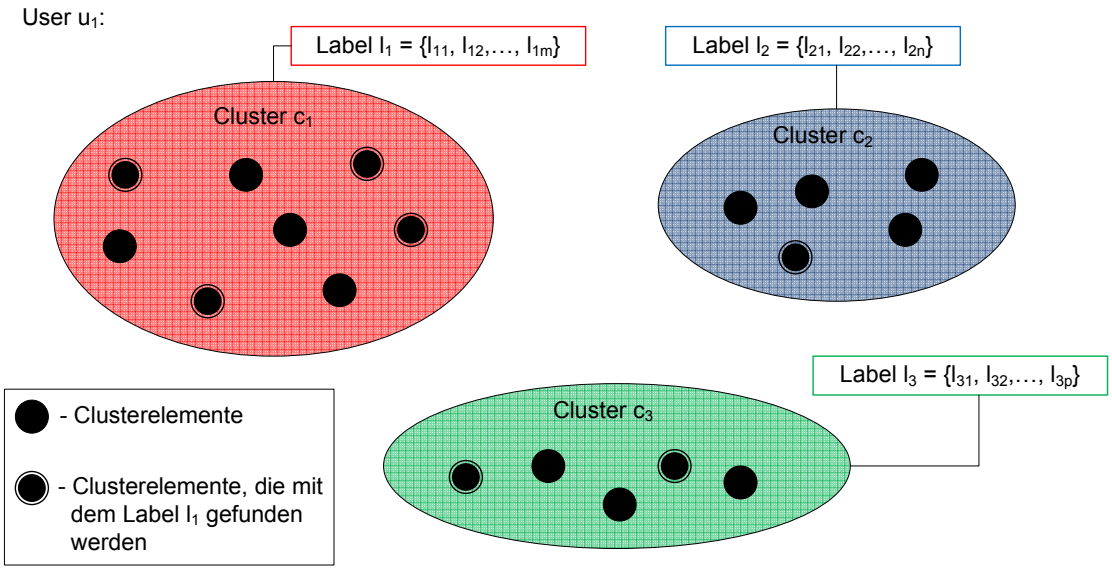


Abbildung 5.2: Beispielhafte Verteilung der Elemente, die mit einem Label gefunden werden

5.2.3 F-Maß

Eine weitere Möglichkeit Precision und Recall zu kombinieren liegt in der Anwendung des F-Maßes. Die beiden Messwerte werden dabei mittels des gewichteten harmonischen Mittels zusammengefasst, was in Formel 5.3 zu sehen ist. Da sich Precision und Recall gegenseitig beeinflussen, können mittels dieser Kombination der beiden Messwerte die besten Ergebnisse bestimmt werden.

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.3}$$

Eine beispielhafte Verteilung der Künstler, die der Nutzer u_1 gehört hat ist in Abbildung 5.2 zu sehen. Dabei werden die Künstler auf drei unterschiedliche Cluster c_1, c_2, c_3 verteilt, für die die Label l_1, l_2, l_3 extrahiert wurden. Ebenfalls zu sehen sind die Elemente, die mit Hilfe des Labels l_1 gefunden werden. Anhand dieses Beispiels wird nun die Berechnung der Messgrößen Precision, Recall und F-Maß vorgenommen. Die dazu benötigten Werte können mittels der Abbildung bestimmt werden. Die Anzahl der korrekten Treffer (sum^1), d.h. die Elemente, die das Label l_1 matchen und in Cluster c_1 liegen beträgt vier. Die Anzahl aller Elemente, die mit dem Label l_1 gefunden werden (sum^2), beträgt sieben. Die Gesamtanzahl der Elemente, die in dem Cluster c_1 liegen (sum^3), beträgt acht. Mittels dieser drei Werte lassen sich nun Precision und Recall errechnen, wie es in den Formeln 5.4a und 5.4b zu sehen ist. In Formel 5.4c ist dargestellt, wie Precision und Recall anschließend durch das F-Maß zusammengefasst werden können.

$$precision = \frac{sum^1}{sum^2} = \frac{4}{7} \approx 0,57 \tag{5.4a}$$

$$recall = \frac{sum^1}{sum^3} = \frac{4}{8} = 0,5 \tag{5.4b}$$

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} = 2 \cdot \frac{0,57 \cdot 0,5}{0,57 + 0,5} \approx 0,53 \tag{5.4c}$$

5.2.4 Lorenz-Kurve

Bei der Lorenz-Kurve handelt es sich um eine grafische Darstellung einer statistischen Verteilung. Sie wurde 1905 von dem US-amerikanischen Statistiker und Ökonom Max Otto Lorenz entwickelt [Wik11b]. Mit ihr lässt sich die relative Konzentrationsverteilung von

Merkmalen darstellen, d.h. mit der Konzentrationsbewertung lässt sich charakterisieren, wie sich die Summe der Merkmalswerte innerhalb der Grundgesamtheit verteilt. Dies wird häufig bei der Darstellung der Einkommensverteilung in einem Land genutzt.

Für die formale Beschreibung wird davon ausgegangen, dass für n Nutzer die Merkmalswerte (z.B. Recall, F-Maß, etc.) $x_1, x_2, \ldots, x_n$ vorliegen. Diese Merkmalswerte werden zu Beginn der Größe nach aufsteigend sortiert und ergeben die geordnete Liste $L = \{x_{<1>}, x_{<2>}, \ldots, x_{<n>}\}$. In diesem Fall werden die Merkmalswerte allerdings auf eine Nachkommastelle gerundet und somit in die Wertebereiche $[0;0,1], (0,1;0,2], \ldots, (0,9;1]$ zusammengefasst. Die Merkmalssumme p_n ergibt sich aus der Summe der einzelnen Merkmalswerte und ist in Formel 5.5 formal dargestellt. Um den trivialen Fall auszuschließen, bei dem alle Merkmalswerte gleich Null sind, wird definiert, dass $p_n > 0$ sein muss.

$$p_n = \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_{<i>} \ (p_n > 0) \tag{5.5}$$

Um nun eine Analyse der Konzentration durchzuführen, muss festgestellt werden, wie sich die Merkmalssumme innerhalb der Gesamtheit der n Nutzer verteilt. Dazu wird für jedes $(j = 1, 2, \ldots, n)$ die Summe p_j der ersten j Merkmalswerte der geordneten Liste L errechnet. In Formel 5.6 ist dies formal dargestellt.

$$p_j = \sum_{i=1}^{j} x_{<i>} \tag{5.6}$$

Daraus ergeben sich n Teilsummen, welche die kummulierten Merkmalswerte der Nutzer enthalten. Für die Darstellung ist nun das Verhältnis v_j von Interesse, welches sich aus der jeweiligen Teilsumme p_j und der Gesamtsumme p_n zusammensetzt, was in Formel 5.7 zu sehen ist. Dieses Verhältnis liegt im Intervall $[0, 1]$, wobei der Wert 1 für $j = n$ angenommen wird. In diesem Fall werden allerdings die diskreten Werte $0,1; 0,2; \ldots; 1$ verwendet, wodurch sich zehn Teilsummen ergeben.

$$v_j = \frac{p_j}{p_n} \tag{5.7}$$

Aus den Verhältnissen v_j ergibt sich somit die erste Dimension, die zur Darstellung der

Lorenz-Kurve benötigt wird. Als zweite Dimension wird nun das Verhältnis u_j der ersten j Nutzer an der Gesamtheit aller n Nutzer errechnet, welches ebenfalls im Intervall $[0, 1]$ liegt. Durch das Verhältnis u_j ergibt sich eine gleichmäßige Zerlegung des Intervalls. Diese beiden Verhältnisse können nun in einem Koordinatensystem, in Form der Lorenz-Kurve, dargestellt werden, wobei sie sich im Einheitsquadrat der 1. Quadranten befindet. Bei einer Gleichverteilung der Merkmalswerte würde sich somit eine Gerade ergeben, die durch den Koordinatenursprung und dem Punkt $(1, 1)$ verläuft. Allgemein besteht eine Lorenz-Kurve, die sich auf n Nutzer bezieht, also aus einem monoton wachsenden Polygonzug, der sich aus n Teilstrecken zusammensetzt. In diesem Fall bestehen die Lorenz-Kurven allerdings lediglich aus zehn solcher Teilstrecken, da wie bereits erwähnt, die Messwerte auf eine Nachkommastelle gerundet werden.

Im Zuge der Evaluation kann dementsprechend mittels der Lorenz-Kurve gezeigt werden, welche Verteilung sich bezüglich der Nutzer und den errechneten Messgrößen ergibt. Dazu werden die Messgrößen Precision, Recall und F-Maß für die einzelnen Cluster errechnet. Die einzelnen Werte werden anschließend pro Benutzer durch das arithmetische Mittel zusammengefasst. Die Berechnung der Verhältnisse v_j ergibt sich dann abhängig von dem Relevanzschwellwert (RSW) und dem verwendeten Verfahren.

5.3 Ergebnisse

In diesem Abschnitt werden die Ergebnisse vorgestellt, die mit dem entwickelten Verfahren erreicht werden konnten. Dazu wird in Kapitel 5.3.1 die Vorstellung der Ergebnisse vorgenommen, welche mittels dem häufigkeitsbasierten Ansatz zur Merkmalsauswahl erreicht werden. In Kapitel 5.3.2 folgen dann die Ergebnisse der extrahierten Labels, welche mittels Chi-Quadrat-Test gefunden werden konnten. Mit diesem Verfahren und den zugrunde liegenden Daten konnten insgesamt 3.093 Cluster für 441 Nutzer gefunden werden. Dies ergibt eine durchschnittliche Clusteranzahl von 7,01 Cluster pro Benutzer.

5.3.1 Häufigkeitsbasierte Merkmalsauswahl

In diesem Kapitel werden die Ergebnisse vorgestellt, die bei der Evaluation der Labels erreicht wurden, welche mit Hilfe der häufigkeitsbasierten Merkmalsauswahl extrahiert wurden.

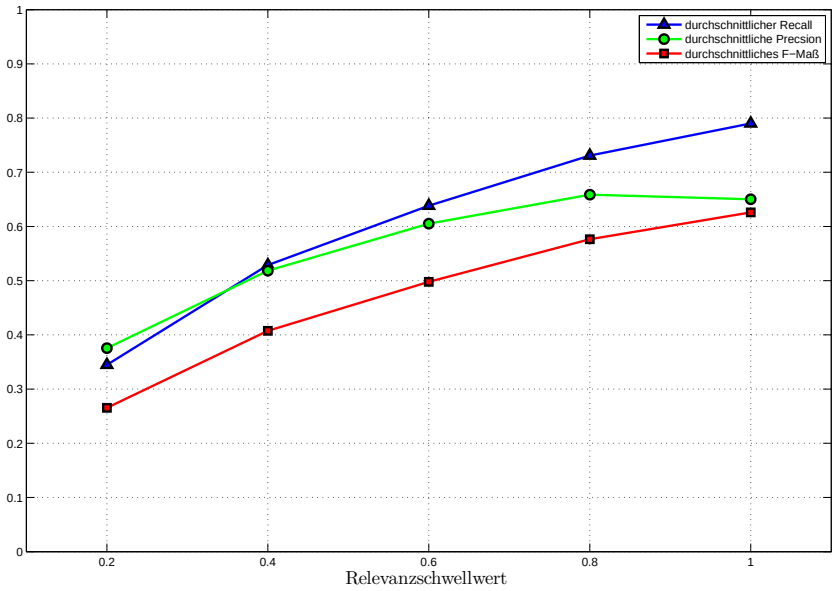


Abbildung 5.3: Ergebnisse für die Labels der unterschiedlichen Relevanzklassen

Dazu werden in Abbildung 5.3 die Messwerte Precision, Recall und F-Maß dargestellt. Auf der Abszissenachse wird dabei der Relevanzschwellwert von 20% bis 100% abgetragen. Auf der Ordinatenachse wird der jeweilige Messwert abgetragen, der durchschnittlich für jeden Benutzer erreicht werden konnte.

Des Weiteren wird gezeigt, wie sich die ermittelten Messwerte auf die einzelnen Benutzer verteilen. Dazu werden in Abbildung 5.4 die Lorenz-Kurven der Recall-Werte für die unterschiedlichen Relevanzklassen dargestellt. Dabei werden auf der Abzissenachse die entsprechenden Verhältnisse v_j der Recall-Werte abgetragen und auf der Ordinatenachse die Verhältnisse u_j der Benutzer.

In Abbildung 5.5 sind die Lorenz-Kurven dargestellt, die auf Grundlage der F-Maß-Werte erzeugt wurden. Auch hierbei werden die Verhältnisse v_j und u_j in dem Koordinatensystem abgetragen.

Abschließend werden nun Labels vorgestellt, die für die Nutzer extrahiert werden konnten. Es werden dabei Nutzer ausgewählt, bei denen das F-Maß einen durchschnittlichen Mindestwert von $0,6$ erreicht hat. Es erfolgt dann eine zufällige Auswahl von fünf Nutzern, deren Labels mit einem Relevanzschwellwert von 100% in der Tabelle 5.1 dargestellt werden. Die

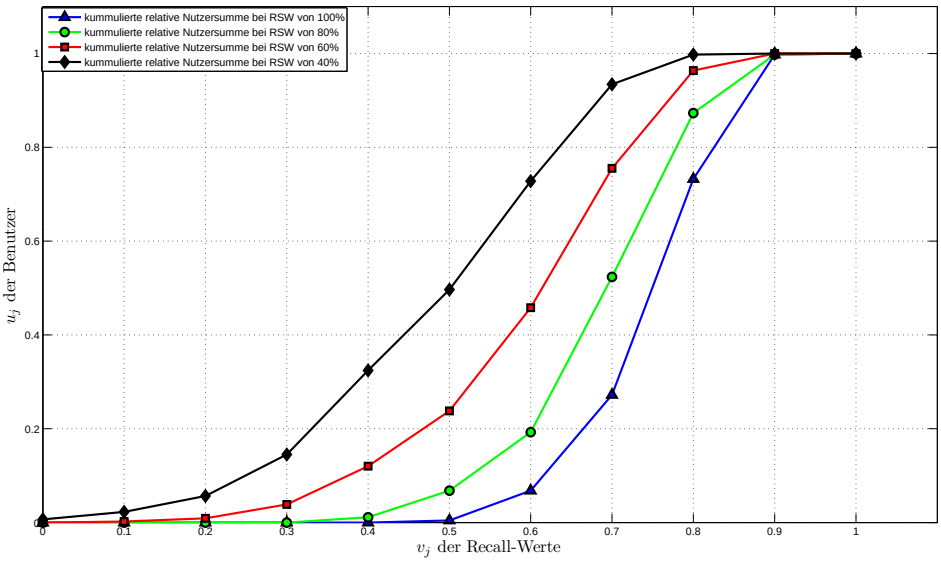


Abbildung 5.4: Lorenz-Kurven der Recall-Werte

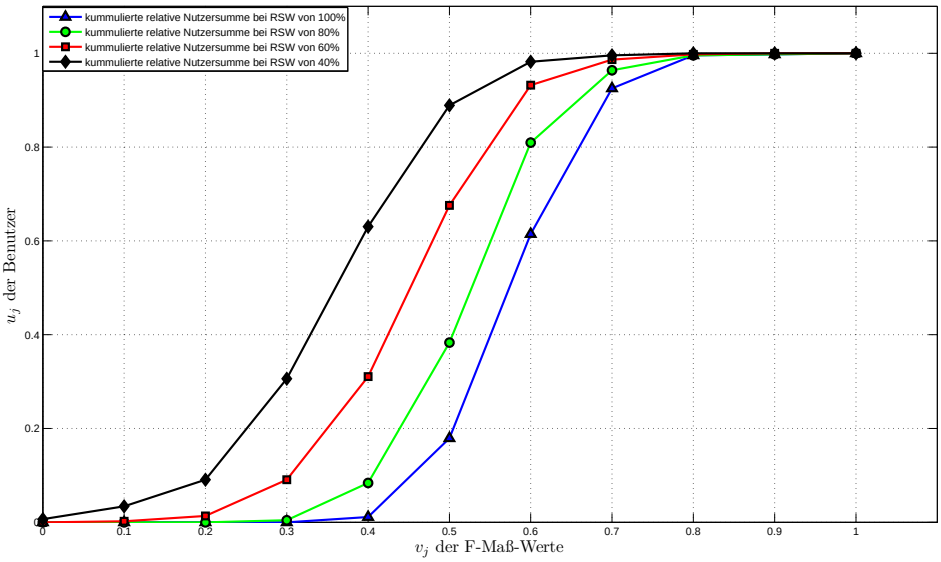


Abbildung 5.5: Lorenz-Kurven der F-Maß-Werte

Nutzer-ID	Labels
232	{electronic music} {hip hop} {indie rock}
402	{indie rock} {smooth jazz} {salsa music} {pop music}
511	{electronic music} {pop music} {string instrument} {indie rock} {indie pop} {film score} {garage rock} {hard rock} {dream pop} {power pop} {britpop} {dance-punk}
740	{rock music} {speed metal} {punk rock} {hard rock} {grindcore} {melodic death metal} {trash metal} {death metal} {progressive metal}
865	{rock music} {string instrument} {punk rock} {indie rock} {heavy metal} {electronic music}

Tabelle 5.1: Extrahierte Labels von fünf zufällig gewählten Nutzern

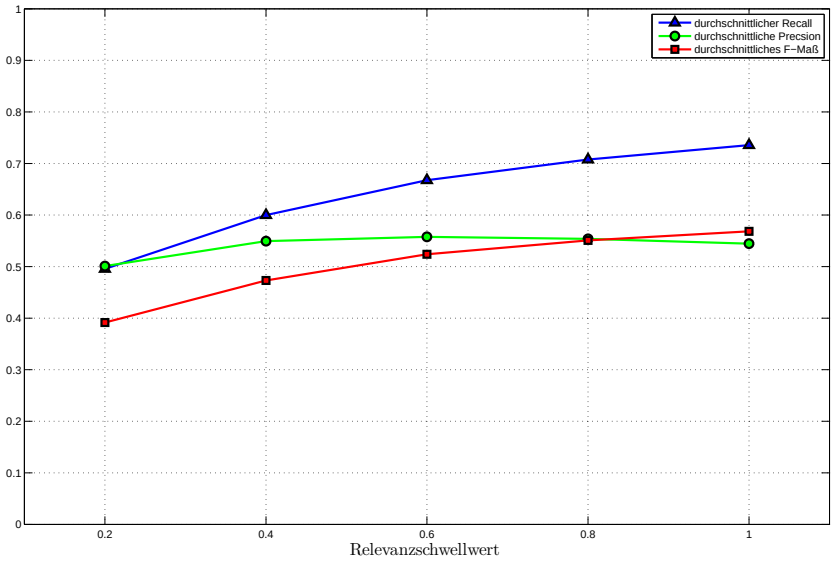


Abbildung 5.6: Ergebnisse für die Labels der unterschiedlichen Relevanzklassen

unterschiedlichen Cluster werden dabei durch die geschweiften Klammern abgegrenzt.

5.3.2 Merkmalsauswahl mittels Chi-Quadrat-Test

Hier werden die Ergebnisse vorgestellt, welche mit den Labels erreicht werden konnten, deren Merkmale durch den Chi-Quadrat-Test ausgewählt wurden. Die Darstellungsform entspricht dabei der selben, wie sie bei der häufigkeitsbasierten Merkmalsauswahl verwendet wird. Die Ergebnisse von Precision, Recall und F-Maß sind in Abbildung 5.6 zu sehen.

Es wird auch hierbei gezeigt, wie sich die verschiedenen Messwerte auf die Benutzer ver-

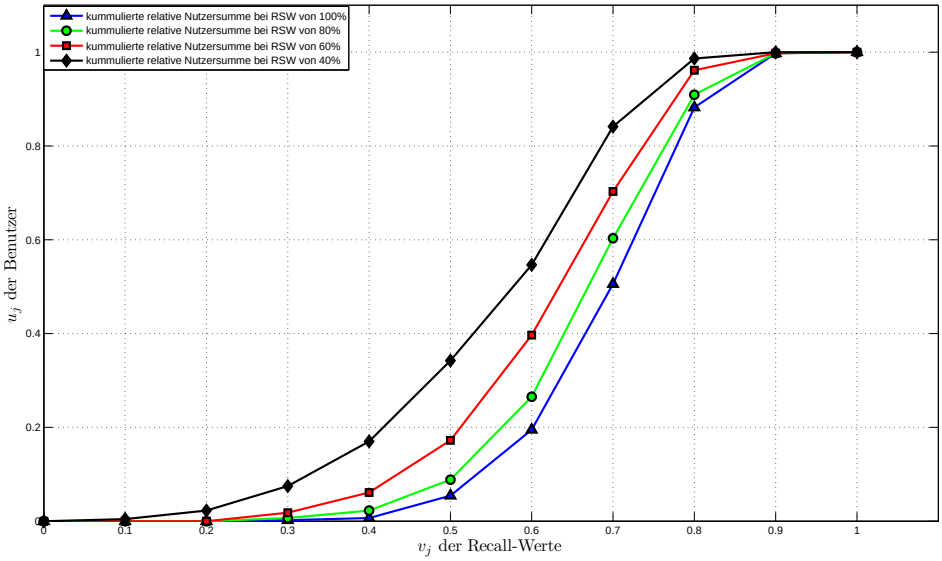


Abbildung 5.7: Lorenz-Kurven für die Recall-Werte

teilen. Dazu werden zum Einen die Lorenz-Kurven der Recall-Werte für diesen Ansatz in der Abbildung 5.7 dargestellt. Zum Anderen sind die Ergebnisse, die auf Grundlage der F-Maß-Werte errechnet werden in Abbildung 5.8 zu sehen.

Abschließend werden auch hier einige Labels vorgestellt, welche mit diesem Ansatz der Merkmalsauswahl ermittelt werden konnten. Dazu wurden ebenfalls zufällig fünf Benutzer ausgewählt, deren Labels mit einem Relevanzschwellwert von 100% in der Tabelle 5.2 zu sehen sind. Auch hierbei dienen die geschweiften Klammern zur Abgrenzung der verschiedenen Cluster.

5.4 Auswertung

In diesem Abschnitt erfolgt die Auswertung der vorgestellten Ergebnisse. Dazu wird im ersten Abschnitt auf die Messwerte Precision, Recall und F-Maß eingegangen. Im Anschluss daran werden die Lorenz-Kurven, sowie die zufällig gewählten Labels ausgewertet.

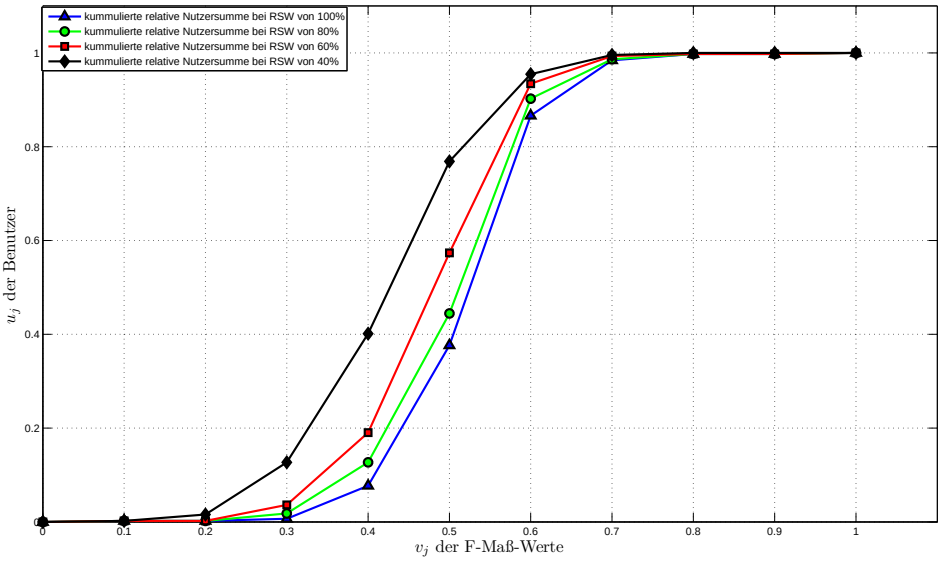


Abbildung 5.8: Lorenz-Kurven für die F-Maß-Werte

Nutzer-ID	Labels
101	{turntablism, electronic instruments} {post-punk revival} {noise pop, rock music, shoegazing, surf rock} {hip hop} {funk-rock} {electronic dance music, folk rock, indie folk} {dance-pop, disco, electroclash, glam rock} {dance music} {country, country rock} {breakbeat} {alternative hip hop}
352	{wind instrument} {alternative rock} {ambient music, gothic rock, indie pop} {ambient music, gothic rock} {indie pop} {ambient music, electronic music}
464	{rhythm and blues} {indie rock} {reggae} {indie rock} {indie pop} {pop music}
608	{folk rock} {art rock, experimental rock} {power pop} {post-rock} {film score} {post-punk} {string instrument} {trip hop}
864	{hyphy} {alternative rock, funk} {percussion} {jazz} {country-rap} {hip hop} {east coast hip hop} {electronic instruments}

Tabelle 5.2: Extrahierte Labels von fünf zufällig gewählten Nutzern

5.4.1 Häufigkeitsbasierte Merkmalsauswahl

In Abbildung 5.3 ist zu sehen, dass mit steigendem Relevanzschwellwert die Qualität der extrahierten Labels zunimmt. So wird bei Labels mit einem Relevanzschwellwert von 20% für das F-Maß lediglich ein durchschnittlicher Wert von ca. 0,28 pro Benutzer erreicht. Bei einem Relevanzschwellwert von 100% hingegen kann ein F-Maß von ca. 0,62 erreicht werden. Dies bedeutet, dass die Labels, weche lediglich die relevantesten Attribute enthalten, die zugrunde liegenden Cluster spezifischer beschreiben können. Der Recall erreicht in diesem Fall sogar einen Wert von ca. 0,79, was bedeutet, dass ca. 80% der Künstler in den Clustern durch die Labels gefunden werden können. Wobei die Precision bei ca. 0,65 liegt und bedeutet, dass 65% der gefundenen Künstler korrekte Treffer sind. Aus der Betrachtung von Precision und Recall, sowie dem F-Maß lässt sich also erkennen, dass die besten Ergebnisse für die häufigkeitsbasierte Merkmalsauswahl erreicht werden können, wenn lediglich die relevantesten Merkmale zu den Labels hinzugefügt werden.

Anhand der Lorenz-Kurven in den Abbildungen 5.4 und 5.5 lässt sich ebenfalls erkennen, dass die besten Ergebnisse mit den Labels erreicht werden können, die die relevantesten Merkmale enthalten. Betrachtet man dabei den Recall, so kann ein Wert von über 0,6 bei 93% der Nutzer erreicht werden, wohingegen dies mit einem Relevanzschwellwert von 40% lediglich bei 27% der Nutzer erreicht werden kann. Betrachtet man nun die Kombination der beiden Messwerte Precision und Recall, so kommt man zu einem ähnlichen Ergebnis. Der Unterschied besteht darin, dass ein F-Maß von 0,6 oder höher bei weniger Nutzern erreicht werden kann. So liegen nun, bei einem Relevanzschwellwert von 100%, 39% der Nutzer über dieser Grenze. Allerdings kann ebenfalls festgestellt werden, dass für 75% der Nutzer mit diesem Ansatz ein F-Maß zwischen 0,5 und 0,7 erreicht werden kann. Mit Herabsetzen des Relvanzschwellwertes auf 40% sinkt dieser Wert auf lediglich noch 11%.

Bei Betrachtung der Labels, welche aus den Playlisten der zufällig ausgewählten Nutzer extrahiert werden konnten, lässt sich erkennen, dass es mit ihnen möglich ist, unterschiedliche Musikrichtungen zu identifizieren. So lassen sich beispielsweise bei Nutzer 232 drei grundverschiedene Musikrichtungen erkennen. Ebenso lässt sich der Musikgeschmack von Nutzer 402 sehr gut anhand der Labels beschreiben. Allerdings werden auch Cluster gefunden, bei denen sich die zugrunde liegenden Musikrichtungen nicht sonderlich voneinander unterscheiden. Dieser Fall ist bei Nutzer 511 zu sehen, bei dem zu vier verschiedene Clustern die Labels „dream pop“, „power pop“, „brit pop“ und „pop music“ annotiert wurden. Dies bedeutet ent-

weder, dass eine sehr feine Unterteilung unterschiedlicher Künstler vorgenommen wurde oder aber, dass ähnliche Künstler in verschiedene und somit überflüssige Cluster eingeteilt wurden.

Abschließend kann festgestellt werden, dass mittels der häufigkeitsbasierten Merkmalsauswahl gute Ergebnisse erzielt werden können, wenn der Relevanzschwellwert bei 100% festgelegt wird. Mit den so entstandenen Labels lassen sich viele Künstler innerhalb der entsprechenden Cluster finden. Des Weiteren liefert das Verfahren für einen Großteil der Benutzer gute bis befriedigende Ergebnisse und bei Betrachtung der Labels lässt sich die anfänglich aufgestellte These bestätigen, dass die unterschiedlichen präferierten Musikrichtungen der Nutzer identifiziert werden können. Des Weiteren wurde eine hohe Diversität innerhalb der Benutzerprofile festgestellt, wodurch gezeigt werden kann, dass es Nutzer gibt, die mehrere unterschiedliche Musikrichtungen bevorzugen.

5.4.2 Merkmalsauswahl mittels Chi-Quadrat-Test

In Abbildung 5.6 ist auch bei diesem Ansatz zu sehen, dass mit steigender Relevanzklasse die Qualität der Ergebnisse zunimmt. Das Maximum erreicht dabei der durchschnittliche Recall mit ca. 0,74 bei einem Relevanzschwellwert von 100%, wobei das F-Maß einen Wert von ca. 0,57 erreicht. Diese Ergebnisse sind etwas schlechter als bei der häufigkeitsbasierten Merkmalsauswahl, wohingegen jedoch die Qualität mit abnehmender Relevanzklasse bei diesem Ansatz nicht so stark fällt. Bei einem Relevanzschwellwert von 20% kann noch ein durchschnittlicher Recall von ca. 0,5 und ein F-Maß von 0,39 erreicht werden, womit beide Werte über denen des häufigkeitsbasierten Ansatzes liegen. Die Precision hingegen erreicht in diesem Ansatz nahezu durchgehend schlechtere Ergebnisse und hat ihr Maximum bei ca. 0,56.

Bei Betrachtung der Lorenz-Kurven in den Abbildungen 5.7 und 5.8 lässt sich ebenfalls erkennen, dass die Unterschiede zwischen den einzelnen Relevanzklassen nicht so stark sind wie bei dem anderen Ansatz. Bei einem Relevanzschwellwert von 100% lassen sich hierbei jedoch für nur 81% der Nutzer Recall-Werte von über 0,6 erreichen, wobei es bei 40% immerhin noch für 45% der Nutzer der Fall ist. Betrachtet man nun das F-Maß, so kann festgestellt werden, dass ein Wert von über 0,6 lediglich für 13% der Nutzer erreicht werden kann, was 26% weniger sind, als bei dem anderen Ansatz. Der Großtteil der Nutzer (79%) befindet sich bei diesem Verfahren in dem Intervall von 0,4 bis 0,6. Im Intervall von 0,5 bis 0,7 befinden sich hierbei 61% der Nutzer, was einer Abnahme von 14% im Vergleich zu dem

Label	Precision, Recall, F-Maß
rhythm and blues	0,909; 1,0; 0,952
indie rock	0,806; 0,746; 0,775
reggae	0,714; 1,0; 0,833
indie rock	0,048; 0,016; 0,024
indie pop	0,929; 1,0; 0,936
pop music	0,684; 0,867; 0,765

Tabelle 5.3: Labels und Messwerte des Nutzers 464

häufigkeitsbasierten Ansatz entspricht.

Die erste Auffälligkeit bei den Labels, welche mit Hilfe dieses Ansatzes gefunden wurden, ist, dass es pro Cluster mehrere Attribute gibt, welche den selben Relevanzwert besitzen. Aus diesem Grund gibt es nun Cluster, welche durch mehrere Attribute beschrieben werden. Bei näherer Betrachtung fällt auf, dass z.B. bei Nutzer 352 zwei Cluster existieren, die sich anhand der Labels überschneiden würden ({ambient music, gothic rock, indie pop}, {ambient music, gothic rock}). Des Weiteren existieren Cluster, welche identische Label besitzen, wie es z.B. bei dem Nutzer 464 der Fall ist, bei dem zwei Cluster mit dem Label „indie rock“ existieren. Bei der Betrachtung der entsprechenden Messwerte für die Cluster des Nutzers 464 (Tabelle 5.3) sieht man jedoch eindeutig, dass sich eines der Cluster merklich in der Qualität von dem anderen unterscheidet. Dies liegt daran, dass ein Label nur für ein Cluster spezifisch sein kann. Mit dem Label, welches die schlechteren Messwerte besitzt werden dann größtenteils die Künstler des anderen Clusters gefunden, was zu der geringen Qualität des Labels führt. Es könnte sich bei diesem Cluster evtl. um ein Cluster handeln, in dem sich Ausreißer befinden, was dazu führt dass kein eindeutiges Label bestimmt werden kann, das die Künstler eindeutig repräsentiert. Positiv aufgefallen ist, dass die Kombination der verschiedenen Musik-Genres als beschreibende Labels durchaus als sinnvoll erachtet werden kann, wie z.B. die Kombination von „turntableism“ als Genre und dem verwendeten Instrument „electronic instruments“.

Abschließend kann festgestellt werden, dass die Qualität der extrahierten Labels mit diesem Ansatz etwas schlechter ist als mit der häufigkeitsbasierten Merkmalsauswahl. Die Qualitätsunterschiede innerhalb der verschiedenen Relevanzklassen sind hierbei geringer, wodurch bessere Ergebnisse in den unteren Relevanzklassen erzielt werden können. Die besten Ergebnisse werden aber auch hier mit den Labels erreicht, welche lediglich die relevantesten Attribute enthalten. Für den Großteil der Nutzer lassen sich somit befriedigende Ergebnisse erzielen. Die geringere Qualität der Labels lässt sich evtl. damit erklären, dass es bereits

bei einer hohen Relevanzklasse zu einer Kombination mehrerer Attribute zu einem Label kommt. Dadurch wird es schwieriger passende Künstler für diese Labels zu finden, was in geringeren Precision- und Recall-Werten resultiert.

6 Fazit & Ausblick

In dieser Masterarbeit wurde ein Ansatz vorgestellt, mit dem es möglich ist, Musik-Präferenzen von Nutzern zu analysieren und in Form von Benutzerprofilen zu extrahieren. Dabei werden insbesondere die unterschiedlichen Musikrichtungen der Nutzer berücksichtigt. Anfänglich wurde das Umfeld beschrieben, in das die Arbeit einzuordnen ist. Dazu wurde der Bereich der Empfehlungssysteme näher erläutert und auf dessen unterschiedliche Ausprägungen eingegangen. Es wurden die Ansätze inhaltsbasierter, kollaborativer und hybrider Empfehlungssysteme beschrieben und deren Vor- und Nachteile dargestellt. Des Weiteren wurde in dem Umfeld das Semantic-Web vorgestellt und dessen zugrunde liegenden Konzepte und Technologien charakterisiert. Es wurde der Grundgedanke vermittelt, mit dem es möglich sein soll, Informationen mit maschinenlesbaren Bedeutungen anzureichern. Als wichtige Technologien für die Umsetzung dieses Konzepts wurden RDF, RDFS, SPARQL, sowie OWL identifiziert und ebenfalls beschrieben. Abschließend wurde das Prinzip der Linked-Data vorgestellt, mit dem es möglich ist, beliebige Daten miteinander in Beziehung zu setzen. Aus der Linked-Data-Cloud wurden dann die zwei Semantic-Web-Dienste MusicBrainz und Freebase näher beschrieben, da diese für die Realisierung des Ansatzes genutzt wurden. Anschließend wurden verwandte Arbeiten aus dem Bereich der Empfehlungssysteme, die Semantic-Web-Daten nutzen, näher erläutert. Es wurde dazu ein Ansatz von Alexandre Passant beschrieben, in dem Messgrößen entwickelt wurden, mit denen es möglich ist, die semantische Distanz zwischen verschiedenen Einträgen der Linked-Data-Cloud zu bestimmen. Er entwickelte dazu verschiedene Varianten, die genutzt werden können, um die direkte und die indirekte Entfernung zwischen den Einträgen zu errechnen. Auf Basis dieser Entfernungen wurden dann Empfehlungen für z.B. Nutzer von Musikangeboten generiert. Ein weiterer Ansatz, der in diesem Zusammenhang erwähnt wurde, war das Verfahren von Wang und Kong, in dem Semantic-Web-Metadaten genutzt wurden, um Ähnlichkeiten zwischen Nutzern zu bestimmen. Dazu wurden diverse Ähnlichkeitsmaße vorgestellt, in denen unter Anderem Film-Genres von Semantic-Web-Diensten genutzt wurden, um zu bestimmen, ob zwei Nutzer den selben Filmgeschmack besitzen. Anschließend wurde mit den ermittelten Daten ein Clustering der Nutzer vorgenommen. Anhand der Cluster konnten dann ähnliche Nutzer

identifiziert werden, welche die Basis für die Empfehlungen darstellten. Der dritte Ansatz der hier vorgestellt wurde, wurde von Baumann el al. veröffentlicht. In diesem Ansatz ging es darum, Ähnlichkeiten zwischen Musik-Künstlern mit Hilfe von Semantic-Web-Metadaten zu bestimmen. Zu den einzelnen Künstlern wurden verschiedene Merkmale gesammelt, auf deren Basis mittels Kosinus-Maß, die Ähnlichkeit zwischen den Künstlern bestimmt werden konnte. Diese Ähnlichkeiten bildeten dann die Grundlage für die Erzeugung von Empfehlungen.

Im weiteren Verlauf der Arbeit wurde dann die Durchführung und die dafür notwendigen theoretischen Grundlagen erläutert. Anfangs wurde dafür ein grober Überblick über das Verfahren gegeben. Anschließend daran wurden die für den Data-Mining-Prozess notwendigen Daten näher beschrieben. Zum Einen wurden dabei die bereits vorhandenen Daten erwähnt und zum Anderen wurde die Sammlung der neuen Daten, welche zum Clustering benötigt wurden, beschrieben. Dabei wurde auf die Anfragen und die gelieferten Ergebnisse der Semantic-Web-Dienste eingegangen. Im Anschluss an die Datensammlung wurde die Datenvorverarbeitung näher betrachtet. Der Hauptpunkt dieses Abschnittes war die Beschreibung der Verknüpfung von Künstlern und deren Eigenschaften, was zur Erstellung der Artist-Property-Matrix führte. Anschließend wurde beschrieben wie die vorhandenen Eigenschaften aufgrund der Anzahl ihrer Vorkommen entfernt wurden. Es wurden dabei Merkmale entfernt, die zu selten vorkamen und somit nur einen geringen Informationsgehalt besaßen. Im weiteren Verlauf wurde die Gewichtung der Merkmale mittels TF-IDF-Maß erläutert. Dazu wurden die theoretischen Grundlagen behandelt und beschrieben, wie sich die jeweiligen Gewichte errechnen. Es folgte der Abschnitt, der sich mit dem Clustering der Künstler befasste. In ihm wurde der Clusteringalgorithmus k-Means näher betrachtet und auf dessen Vor- und Nachteile eingegangen. Daran anschließend wurde ein Verfahren vorgestellt, mit dem es möglich ist Clusteranzahlen zu schätzen, die die zugrunde liegende Menge besonders gut repräsentieren. In dem darauf folgenden Abschnitt wurden zwei Möglichkeiten betrachtet, mit denen es möglich ist relevante Merkmale der Cluster zu bestimmen. Zum Einen wurde ein häufigkeitsbasierter Ansatz und zum Anderen die Merkmalsauswahl auf Basis des Chi-Quadrat-Tests vorgestellt.

Es folgte ein Kapitel, in dem die Implementierung, welche im Zuge dieser Masterarbeit entstanden ist, näher betrachtet wurde. In dem Kapitel wurde die verwendete Soft- und Hardware erwähnt und ein Überblick über die entstandenen Klassen gegeben. Des Weiteren wurden die Methoden und deren Funktionalitäten beschrieben, sowie auf die verwendeten Datentypen eingegangen. Zusätzlich wurde ein detailierter Überblick über den Ablauf des

Prozesses in Form eines Flussdiagramms gegeben.

In dem vorletzten Kapitel der Arbeit wurde die Evaluation der erreichten Ergebnisse behandelt. Dazu wurde anfänglich das Vorgehen skizziert, um ein Verständnis für den Ablauf der Evaluation zu schaffen. Es wurde auf die Entstehung unterschiedlicher Labels aufgrund der Relevanzklassen eingegangen und die Verwendung unterschiedlicher Messwerte und Diagramme erläutert. Die theoretischen Grundlagen, die für das Verständnis der Evaluation notwendig waren, wurden in dem folgenden Abschnitt dargelegt. Darin wurden die verwendeten Messwerte Precision, Recall und F-Maß definiert und auf deren Aussagefähigkeit eingegangen. Des Weiteren wurde in diesem Abschnitt die Lorenz-Kurve, welche zur Veranschaulichung der Konzentrationsverteilung der Messwerte genutzt wurde, vorgestellt. Es wurde dazu formal auf die Berechnung einer solchen Kurve eingegangen und die daraus resultierende Darstellung beschrieben. Es folgte ein Abschnitt, in dem die Ergebnisse der Evaluation vorgestellt wurden. Dies geschah zum Einen für die häufigkeitsbasierte Merkmalsauswahl und zum Anderen für die Merkmalsauswahl mittels Chi-Quadrat-Test. Dabei wurden zu jedem der Verfahren die Messwerte Precision, Recall und F-Maß vorgestellt, sowie die Lorenz-Kurven für den Recall und das F-Maß visualisiert. Abschließend wurden die extrahierten Labels von jeweils fünf zufällig gewählten Nutzern vorgestellt. Dadurch sollte gezeigt werden, ob es mit den extrahierten Labels möglich ist, unterschiedliche Musikrichtungen der Nutzer zu identifizieren. Abschließend wurden in dem Kapitel Evaluation die Ergebnisse für die zwei unterschiedlichen Ansätze zur Merkmalsauswahl ausgewertet. Dazu wurde eine Einschätzung von Precision, Recall und F-Maß gegeben, sowie die Auswertung der Lorenz-Kurven vorgenommen. Des Weiteren wurde die Qualität der stichprobenartig gewählten Labels eingeschätzt und auf deren Besonderheiten eingegangen. Ein Vergleich der beiden vorgestellten Verfahren bildete den Abschluss des Kapitels.

Abschließend kann gesagt werden, dass mit diesem Verfahren gezeigt werden konnte, dass Semantic-Web-Metadaten ein großes Potential für zukünftige Empfehlungssysteme bieten. Das Ziel der Arbeit war die Entwicklung eines Verfahrens, mit dem es möglich ist, die unterschiedlichen bevorzugten Musikrichtungen von Benutzern zu identifizieren und zu extrahieren. Dazu sollten Meta-Daten der Künstler aus dem Semantic-Web genutzt werden, wodurch das Clustering dieser Künstler ermöglicht werden sollte. Durch die Evaluation der Ergebnisse, welche mit diesem Verfahren erreicht wurden, stellte sich heraus, dass es möglich ist, die unterschiedlichen Musikrichtungen, die ein Nutzer bevorzugt, darzustellen. Es stellte sich dabei ebenfalls heraus, dass es eine hohe Diversität innerhalb der bevorzugten Musi-

krichtungen gibt, was eine wichtige Erkenntnis für zukünftige Musikempfehlungen darstellt. Des Weiteren konnte gezeigt werden, dass mit Hilfe dieses Ansatzes Labels extrahiert werden können, die spezifisch für diese Musikrichtungen sind. Mit Hilfe dieser Benutzerprofile könnte es somit möglich sein bei inhaltsbasierten Empfehlungssystemen einer Überspezialisierung entgegen zu wirken. Durch die Evaluation konnte allerdings ebenfalls gezeigt werden, dass das Verfahren erfordert, dass Nutzer unterschiedliche Musikrichtungen hören, um gute Ergebnisse erzielen zu können.

Um mittels der Benutzerprofile Musikempfehlungen zu generieren, sind diverse Szenarien denkbar, welche sich teilweise bereits in der Entwicklung befinden. Da die Labels bei einem Relevanzschwellwert von 100% meistens aus dem Genre bestehen, mit dem die darin enthaltenen Künstler assoziiert werden, könnten die Labels direkt genutzt werden, um z.B. persönliche Empfehlungen von Internet-Radio-Stationen zu geben, die auf Grundlage der entsprechenden Genre ausgewählt werden. Eine weitere Möglichkeit für die Integration dieser Benutzerprofile in Empfehlungssysteme könnte sich durch den Ansatz von Baumann et al. ergeben, der in [BSS10] vorgestellt wurde. Dabei werden nicht die Labels für die Auswahl von Empfehlungen genutzt, sondern es werden besonders repräsentative Künstler aus den Clustern gewählt. Auf deren Basis werden dann ähnliche Künstler ermittelt, welche dann als Empfehlungen für die Nutzer dienen. Dabei können sowohl die unterschiedlichen Musikrichtungen berücksichtigt, als auch ein großer Anteil unbekannter Künstler empfohlen werden.

Literaturverzeichnis

[Arm01] Armstrong, J.S.: *Principles of Forecasting - A Handbook for Researchers and Practitioners*. Kluwer Academic, 2001

[AT05] Adomavicius, Gediminas ; Tuzhilin, Alexander: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. In: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 17 (2005), Nr. 6, S. 734–749

[BC92] Belkin, Nicholas J. ; Croft, W. B.: Information filtering and information retrieval: two sides of the same coin? In: *Commun. ACM* 35 (1992), December, S. 29–38. – ISSN 0001–0782

[BHC98] Basu, Chumki ; Hirsh, Haym ; Cohen, William: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In: *In Proceedings of the Fifteenth National Conference on Artificial Intelligence*, AAAI Press, 1998, S. 714–720

[BHK98] Breese, John S. ; Heckerman, David ; Kadie, Carl: Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: *Proc. 14th Conf. Uncertainty in Artificial Intelligence*, Morgan Kaufmann, 1998, S. 43–52

[BL09] Berners-Lee, Tim: *Linked Data*. `http://www.w3.org/DesignIssues/LinkedData`, Juni 2009. – [Online; accessed 17-Mai-2011]

[BLHL01] Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: The Semantic Web. In: *Scientific American* 284 (2001), Mai, Nr. 5, 34-43. `http://www.scientificamerican.com/article.cfm?id=the-semantic-web`. – ISSN 0036–8733

[BP00] Billsus, D. ; Pazzani, M.: User Modeling for Adaptive News Access. In: *User Modeling and User-Adapted Interaction* Bd. 10, 2000, S. 147–180

[BS97] Balabanović, Marko ; Shoham, Yoav: Fab: content-based, collaborative recommendation. In: *Commun. ACM* 40 (1997), March, S. 66–72. – ISSN 0001–0782

[BS01] Bradley, Keith ; Smyth, Barry: *Improving Recommendation Diversity.* 2001

[BSS10] Baumann, Stephan ; Schirru, Rafael ; Streit, Bernhard: Towards a Storytelling Approach for Novel Artist Recommendations. In: *Proceedings of the 8th International Workshop on Adaptive Multimedia Retrieval, AMR'2010* Bd. 6817 of LNCS, 2010, S. 1–15

[Bur00] Burke, Robin: Knowledge-based Recommender Systems. In: *Encyclopedia of Library and Information Systems* Bd. 69, 2000

[BYRN99] Baeza-Yates, Ricardo A. ; Ribeiro-Neto, Berthier: *Modern Information Retrieval.* Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1999. – ISBN 020139829X

[CGM+99] Claypool, M. ; Gokhale, A. ; Miranda, T. ; Murnikov, P. ; Netes, D. ; Sartin, M.: Combining Content-Based and Collaborative Filters in an Online Newspaper. In: *ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation*, 1999

[CJ10] Cyganiak, Richard ; Jentzsch, Anja: *The Linking Open Data cloud diagram.* `http://richard.cyganiak.de/2007/10/lod/`, September 2010. – [Online; accessed 17-Mai-2011]

[Gru93] Gruber, Thomas R.: A translation approach to portable ontology specifications. In: *Knowledge Acquisition* 5 (1993), Nr. 2, S. 199–220. `http://dx.doi.org/http://dx.doi.org/10.1006/knac.1993.1008`. – DOI http://dx.doi.org/10.1006/knac.1993.1008. – ISSN 1042–8143

[HSRF95] Hill, Will ; Stead, Larry ; Rosenstein, Mark ; Furnas, George: Recommending and evaluating choices in a virtual community of use. In: *Proceedings of the SIGCHI conference on Human factors in computing systems.* New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995. – ISBN 0–201–84705–1, S. 194–201

[KM01] Koivunen, Marja-Riitta ; Miller, Eric: *W3C Semantic Web Activity.* `http://www.w3.org/2001/12/semweb-fin/w3csw`, Dezember 2001. – [Online; accessed 17-Mai-2011]

[KMM+97] Konstan, Joseph A. ; Miller, Bradley N. ; Maltz, David ; Herlocker, Jonathan L. ; Gordon, Lee R. ; Riedl, John: GroupLens: Applying Collaborative Filtering to Usenet News. In: *Commun. ACM* 40 (1997), March, S. 77–87. – ISSN 0001–0782

[LKM92] Lilien, G.L. ; Kotler, P. ; Moorthy, K.S.: *Marketing Models*. Prentice Hall, 1992

[Llo82] Lloyd, S.P.: Least squares quantization in PCM. In: *IEEE Transactions on Information Theory* 28 (1982), S. 129–137

[LSY03] Linden, Greg ; Smith, Brent ; York, Jeremy: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. In: *IEEE Internet Computing* 7 (2003), S. 76–80. – ISSN 1089–7801

[Mac67] MacQueen, J.: *Some methods for classification and analysis of multivariate observations*. Proc. 5th Berkeley Symp. Math. Stat. Probab., Univ. Calif. 1965/66, 1, 281-297 (1967)., 1967

[MH04] McGuinness, Deborah L. ; Harmelen, Frank van: *OWL Web Ontology Language*. `http://www.w3.org/TR/2004/REC-owl-features-20040210/`, Februar 2004. – [Online; accessed 17-Mai-2011]

[MRS08] Manning, Christopher D. ; Raghavan, Prabhakar ; Schütze, Hinrich: *Introduction to Information Retrieval*. Cambridge University Press, 2008

[MS03] Murthi, B. P. S. ; Sarkar, Sumit: The Role of the Management Sciences in Research on Personalization. In: *Manage. Sci.* 49 (2003), October, S. 1344–1362. – ISSN 0025–1909

[Not05] Notholt, Jochen: *Die Standards des Semantic Web (Teil 2)*. `http://www.jurpc.de/aufsatz/20050065.htm`, Mai 2005. – [Online; accessed 17-Mai-2011]

[OWL09] OWL Working Group: *OWL 2 Web Ontology Language*. `http://www.w3.org/TR/owl2-overview/`, October 2009. – [Online; accessed 08-Juli-2011]

[Pas10] Passant, Alexandre: Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations. In: *Proceedings of the AAAI Spring Symposium "Linked Data Meets Artificial Intelligence"*, 2010

[Paz99] Pazzani, Michael J.: A Framework for Collaborative, Content-Based and Demographic Filtering. In: *Artif. Intell. Rev.* 13 (1999), December, S. 393–408. – ISSN 0269–2821

[PBMW97] Pazzani, Michael ; Billsus, Daniel ; Michalski, S. ; Wnek, Janusz: Learning and Revising User Profiles: The Identification of Interesting Web Sites. In: *Machine Learning*, 1997, S. 313–331

[Pow81] Powell, M.J.D.: *Approximation Theory and Methods*. Cambridge University Press, 1981

[PR08] Passant, Alexandre ; Raimond, Yves: Combining Social Music and Semantic Web for Music-Related Recommender Systems. In: *Social Data on the Web Workshop of 7th International Semantic Web Conference*. Karlsruhe, Deutschland, Oktober 2008

[PS08] Prud'hommeaux, Eric ; Seaborne, Andy: *SPARQL Query Language for RDF*. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`, Januar 2008. – [Online; accessed 17-Mai-2011]

[Ric79] Rich, Elaine: User modeling via stereotypes. In: *Cognitive Science* 3 (1979), Nr. 4, S. 329–354

[RIS+94] Resnick, Paul ; Iacovou, Neophytos ; Suchak, Mitesh ; Bergstrom, Peter ; Riedl, John: GroupLens: an open architecture for collaborative filtering of netnews. In: *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. New York, NY, USA : ACM, 1994 (CSCW '94). – ISBN 0–89791–689–1, S. 175–186

[Roc71] Rocchio, J.J.: Relevance Feedback in Information Retrieval. In: Salton, G. (Hrsg.): *SMART Retrieval System-Experiments in Automatic Document Processing*. Prentice Hall, 1971, Kapitel 14

[Sal88] Salton, Gerald: *Automatic text processing*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1988. – ISBN 0–2:1–1227–8

[SKKR00] Sarwar, Badrul M. ; Karypis, George ; Konstan, Joseph A. ; Riedl, John T.: Application of Dimensionality Reduction in Recommender System – A Case Study. In: *IN ACM WEBKDD WORKSHOP*, 2000

[SM86] Salton, Gerard ; McGill, Michael J.: *Introduction to Modern Information Retrieval*. New York, NY, USA : McGraw-Hill, Inc., 1986. – ISBN 0070544840

[SM93] Sheth, B. ; Maes, P.: Evolving agents for personalized information filtering. In: *Proc. Ninth IEEE Conf. Artificial Intelligence for Applications*, 1993, S. 345–352

[SM95] Shardanand, Upendra ; Maes, Pattie: Social information filtering: algorithms for automating "word of mouth". In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995 (CHI '95). – ISBN 0–201–84705–1, S. 210–217

[SN99] Soboroff, Ian ; Nicholas, Charles: Combining Content and Collaboration in Text Filtering. In: *In Proceedings of the IJCAI'99 Workshop on Machine Learning for Information Filtering*, 1999, S. 86–91

[TC00] Tran, T. ; Cohen, R.: Hybrid Recommender Systems for Electronic Commerce. In: *Knowledge-based Electronic Markets, Papers from the AAAI Workshop*, AAAI Press, 2000

[Wik11a] Wikipedia: *Freebase — Wikipedia, The Free Encyclopedia*. `http://en.wikipedia.org/w/index.php?title=Freebase&oldid=428166189`, 2011. – [Online; accessed 17-May-2011]

[Wik11b] Wikipedia: *Lorenz-Kurve — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Lorenz-Kurve&oldid=91279411`. Version: 2011. – [Online; Stand 19. Juli 2011]

[Wik11c] Wikipedia: *MusicBrainz — Wikipedia, Die freie Enzyklopädie*. `http://en.wikipedia.org/w/index.php?title=Freebase&oldid=428166189`. `http://de.wikipedia.org/w/index.php?title=MusicBrainz&oldid=88001525`. Version: 2011. – [Online; Stand 17. Mai 2011]

[Wik11d] Wikipedia: *Semantisches Web — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=Semantisches_Web&oldid=88074007`, 2011. – [Online; Stand 14. Mai 2011]

[Wik11e] Wikipedia: *SPARQL — Wikipedia, Die freie Enzyklopädie*. `http://de.wikipedia.org/w/index.php?title=SPARQL&oldid=88722228`, 2011. – [Online; Stand 14. Mai 2011]

[WK07] Wang, Rui-Qin ; Kong, Fan-Sheng: Semantic-Enhanced Personalized Recommender System. In: *2007 International Conference on Machine Learning and Cybernetics* (2007), Nr. August, 4069–4074. `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4370858`

[ZCM02] Zhang, Yi ; Callan, Jamie ; Minka, Thomas: Novelty and redundancy detection in adaptive filtering. In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA : ACM, 2002 (SIGIR '02). – ISBN 1–58113–561–0, S. 81–88

[ZH09] Zhang, Mi ; Hurley, Neil: Novel Item Recommendation by User Profile Partitioning. In: *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*. Washington, DC, USA : IEEE Computer Society, 2009 (WI-IAT '09). – ISBN 978–0–7695–3801–3, 508–515