Erweiterung klassischer Musikempfehlungssysteme um den Kontext des Wetters

Bachelor Arbeit

Kain Kordian Gontarska
Fachbereich Mathematik und Informatik
Institut für Informatik
Freie Universität Berlin, Deutschland

29. März 2016

Betreuer:	Marko Harasic
Gutachter 1:	Marko Harasic
Gutachter 2:	Prof. Dr. Adrian Paschke

Selbstständigkeitserklärung

Hiermit bestätige ich, dass ich die vorliegende Ausarbeitung mit dem Titel *Erweiterung klassischer Musikempfehlungssysteme um den Kontext des Wetters* selbstständig und ohne unerlaubte Hilfe angefertigt habe.

Ich versichere, dass ich ausschließlich die angegebenen Quellen in Anspruch genommen habe.

Statement of Authorship

I declare that this thesis entitled *Erweiterung klassischer Musikempfehlungssysteme um den Kontext des Wetters* is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree. Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Kain Kordian Gontarska
Berlin, 29. März 2016

Abstract

Empfehlungssysteme sind eine populäre Methode zur Navigationshilfe durch große Mengen an Objekten. Typischerweise werden Informationen wie Objekteigenschaften und Benutzerpräferenzen genutzt, um passende Objekte zu empfehlen. Über die Benutzer- und Objektdimensionen hinaus lassen sich für Empfehlungen noch weitere Faktoren betrachten. Kontext im Allgemeinen ist ein Faktor dessen Betrachtung bereits zur Leistungssteigerung bisheriger Empfehlungsalgorithmen geführt hat. In der vorliegenden Arbeit wird der Einfluss des Kontexts des Wetters auf die Vorhersagegenauigkeit klassischer Musikempfehlungssysteme untersucht. Als Referenzkontext werden Wochentage gewählt. Es ergibt sich, dass die Betrachtung des Wetters für die Vorhersagegenauigkeit eines Systems irrelevanter zu sein scheint als die Betrachtung der Wochentage. Die Leistungssteigerung der klassischen Empfehlungssysteme ist nur für kurze Empfehlungslisten beobachtbar. Es konnte nicht geklärt werden, ob Wetter als Kontext tatsächlich eine geringe Rolle für die Empfehlung von Musik spielt, da der Datensatz nicht ausreichend Hörereignisse pro Benutzer enthält. Es wäre interessant zu untersuchen, ob eine Kontextgeneralisierung zu besseren Ergebnissen führe.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Sobald eine Person vor einer Wahl steht, die nicht rein intuitiv entschieden werden kann, erfordert der Prozess der Entscheidung eine kognitive, analytische Leistung. Die Person benötigt Informationen über die Objekte oder Ereignisse, welche ihr dabei helfen, eine Wahl zu treffen, die den gegebenen Anforderungen gerecht wird. Abhängig von der Anzahl der Objekte und der Menge ihrer für die Entscheidung relevanten Eigenschaften kann der Entscheidungsprozess beliebig komplex werden. Muss sich eine Person zwischen einer Banane und einem Apfel entscheiden, wird sie vermutlich schnell nach ihrer Vorliebe oder ihrem Appetit entscheiden. Stellt man dieselbe Person nun vor das Problem, sich ein neues Auto auszusuchen, wird deutlich, dass der Entscheidungsprozess aufwändiger sein kann. Vorausgesetzt die Person strebt eine möglichst optimale Lösung an, wird sie in den meisten Fällen dazu neigen, einen Experten zu konsultieren, beispielsweise einen Gebrauchtwagenhändler. Der Händler wird seine Erfahrung und sein Wissen über bereits verkaufte und empfohlene Fahrzeuge nutzen und über gezielte Fragen seine Auswahl an Fahrzeugen schrittweise filtern. So versucht der Händler, den Anforderungen des Kunden gerecht zu werden und ihm am Ende ein zufriedenstellendes Fahrzeug anzubieten.

Betrachtet man nun Fälle, in denen die Menge der Objekte und der damit verbundenen Informationen noch größer ist, wie etwa das beste Fahrzeug von allen auf dem Markt erhältlichen Fahrzeugen, oder stetig wächst, z.B. die bei einem Streamingdienst verfügbaren Filme, wird eine Person, im Versuch den Überblick zu behalten und schnell Entscheidungen fällen zu können, an die Grenzen ihrer Gedächtniskapazitäten [1] und mentalen Belastbarkeit kommen. Denkbar ist der Einsatz technischer Werkzeuge, die den Findungsprozess vereinfachen oder gänzlich übernehmen. Mittlerweile sind Informationen zu allen erdenklichen Objekten digitalisiert und in den meisten Fällen auch strukturiert, so dass sich auf diese mit Hilfe eines Computers zugreifen lässt [2]. Da die Informationen und Objekte digital vorliegen, lässt sich der Zugriff auf diese über das Implementieren von Algorithmen und Methoden vereinfachen oder beschleunigen.

Dem Benutzer werden Möglichkeiten geboten, aktiv die verfügbare Objektmenge zu reduzieren, um sich in einer übersichtlicheren Auswahl besser zurecht finden zu können.

- Filtermaske: Gibt der Benutzerin die Möglichkeit, die eigenen Präferenzen an das System zu kommunizieren.

- Suchmaschine: Ein Benutzer kann mit Hilfe von Suchmaschinen über gezielte Schlüsselwörter an gewünschte Inhalte oder Objekte gelangen.

Interagiert eine Benutzerin mit digitalen Systemen, produziert sie stets sowohl unbewusst als auch bewusst Informationen zu ihrer eigenen Person, wie zum Beispiel ihre Suchanfragen, gekaufte Produkte, Bewertungen, besuchte Seiten oder sogar ihren Standort. Stehen einem entscheidenden oder empfehlenden System nun solche Informationen zur Verfügung, so können diese eine Vorauswahl an Objekten ohne direkten Einfluss des Benutzers treffen.

Viele Anbieter nutzen bereits Empehlungssysteme, um ihren Benutzenden den Zugang zu den angebotenen Produkten oder Inhalten zu vereinfachen. *Amazon*[1] schlägt Produkte vor, für die sich andere Benutzende interessieren, *Netflix*[2] empfiehlt neue Filme oder Serien, abhängig von den bisher gesehenen Inhalten, *Spotify*[3] wertet das Hörverhalten seiner Benutzenden aus und bietet ihnen einen persönlichen Musikmix an, *Facebook*[4] empfiehlt Freunde und Freundinnen oder schlägt Artikel vor. Dies sind nur wenige aber bekannte Beispiele.

Klassische Empfehlungssysteme betrachten lediglich zwei Dimensionen, Benutzer (ihre direkten oder indirekten Bewertungen von Objekten) und Objekte (ihre Eigenschaften), nach denen ausgewertet wird [3, 4]. Der Ansatz des kollaborativen Filterns *(collaborative filtering)* benutzt das Verhalten der dem System bekannten Benutzer, um unter ihnen Gemeinsamkeiten zu finden und so einem Benutzer Objekte zu empfehlen, die ähnlichen Benutzern bereits gefallen [5, 6, 7, 8]. Empfiehlt das System Objekte, in dem es ihre Eigenschaften mit den Präferenzen des Benutzers abgleicht, spricht man vom inhaltsbasierten Filtern *(contentbased filtering)* [9, 10].

Es fällt auf, dass nicht alle einem System verfügbaren Informationen (wie Zeit und Ort der Interaktion) zur Berechnung verwendet werden. Es ist annehmbar, dass somit relevante Details, die zur erheblichen Änderung der Empfehlungen führen würden, nicht beachtet werden [11].

1.1 Motivation

Die Einbeziehung des Kontexts ist ein noch relativ neuer Ansatz auf dem Gebiet der Empfehlungssysteme. Beobachtungen wie die von Adomavicius et al. [12] zeigen, dass die Einbeziehung von Parametern wie Zeit und Ort zu verbesserten Ergebnissen führen können.

Im Folgenden werden vier denkbare Situationen vorgestellt, in denen der Kontext eine entscheidene Rolle für die Empfehlung spielen kann:

- Wenn sich der Benutzer in Tokyo befindet, möchte er keine Restaurantempfehlung für Berlin erhalten.

[1] **Amazon**: `http://www.amazon.com/`, Zugriff am: 16.02.2016
[2] **Netflix**: `https://www.netflix.com/`, Zugriff am: 16.02.2016
[3] **Spotify**: `https://www.spotify.com/`, Zugriff am: 16.02.2016
[4] **Facebook**: `https://www.facebook.com/`, Zugriff am: 16.02.2016

- Ist jemand in Begleitung einer anderen Person auf der Suche nach einem passenden Kinofilm, könnte der Geschmack der Begleitung oder die Art der Beziehung berücksichtigt werden.
- Eine Museumsempfehlung sollte das aktuelle Interesse des Benutzers widerspiegeln.
- Bei einer Empfehlung für Freizeitaktivitäten lohnt es sich, das aktuelle Wetter als Parameter einzubeziehen.

Es kann davon ausgegangen werden, dass sämtliche von einer Person getroffenen Entscheidungen stets vom Kontext abhängig sind, in dem sich die Person befindet [13]. Solche Informationen in einem Empfehlungssystem nicht zu nutzen, erscheint nicht sinnvoll [11]. Die Verfügbarkeit und Vielfalt von Kontextinformationen ist von der Anwendung abhängig. Da Anwendungen immer mehr Informationen und vor allem Informationsquellen bereitgestellt werden, ist es sinnvoll, den Wert und die Schwere unterschiedlicher Kontextarten für Empfehlungssysteme zu untersuchen. Der Kontext des Wetters ist eine noch wenig untersuchte Information. Daher wird in der vorliegenden Arbeit untersucht, ob die Beachtung des Wetters bei Empfehlungssystemen zu besseren Vorhersageergebnissen führen kann. Insbesondere wird dies für die Empfehlung von Musik untersucht.

1.2 Fragestellung

Das Hauptziel der vorliegenden Arbeit und der dazu gehörenden Implementierung ist, zu untersuchen, ob die Einbeziehung des Wetters als Kontext in einem Empfehlungssystem die Qualität der Empfehlungen im Vergleich zu klassischen Ansätzen verbessert.

Hierfür wird ein Empfehlungssystem implementiert, in dem Empfehlungen von ausgewählten Empfehlungsalgorithmen berechnet werden. Als klassische Ansätze werden *kollaboratives Filtern* und *inhaltsbasiertes Filtern* implementiert. Um den Kontext miteinzubeziehen, werden zwei unterschiedliche Ansätze implementiert, das *Vorfiltern* und das *Nachfiltern*. Dabei wird vor beziehungsweise nach dem eigentlichen Empfehlungsalgorithmus die Menge der Daten auf den in Frage kommenden Kontext reduziert oder neu gewichtet [14]. In der Evaluation wird die *Vorhersagegenauigkeit* und die *Abdeckung* der Empfehlungsalgorithmen untersucht. Dies geschieht in einem *offline Experiment* [15] mit Hilfe eines *natürlichen Datensatzes* [16] zum Musikhörverhalten realer Benutzer. Der Datensatz enthält zusätzlich Kontextinformationen wie Ort, Zeit und Wetter zu jedem Musikhörereignis.

1.3 Gliederung

Zu Beginn steht eine Einführung in die bisherigen Methoden klassischer Empfehlungssysteme so wie eine Beschreibung und Einordnung kontextsensitiver Empfeh-

lungssysteme. Es werden Methoden so wie Stärken und Schwächen einzelner Ansätze dargestellt. Im darauf folgenden Kapitel werden Empfehlungssysteme für Musik genauer betrachtet und der Ansatz, den Kontext des Wetters zu verwenden, näher diskutiert. Schließlich wird der verwendete, bereinigte und angereicherte Datensatz und die Implementierung für ein kontextsensitives Empfehlungssystem für Musik mit dem Schwerpunkt auf Wetter vorgestellt. Im Anschluss werden die Ergebnisse der Evaluation, in der die Qualität der Empfehlungen mit unterschiedlichen Messverfahren untersucht wird, dargestellt. Zum Abschluss werden die Ergebnisse diskutiert, eine Zusammenfassung der vorliegenden Arbeit und ein Ausblick auf mögliche zukünftige Arbeit gegeben.

Kapitel 2

Empfehlungssysteme

Dieses Kapitel gibt eine Übersicht zu Empfehlungssystemen, sowohl klassischen Ansätzen als auch dem kontextsensitiven Ansatz. Die Anfänge der Forschung an Empfehlungssystemen sind auf unterschiedliche Forschungsbereiche zurückzuführen, so etwa Expertensysteme [17], Information Retrieval oder Kognitionswissenschaften [3]. Empfehlungssysteme sammeln Informationen über die Präferenzen ihrer Benutzer zu einer Menge bestimmter Objekte oder Artikel (z.B. Filme, Lieder, Bücher, Programme, Restaurants, Reiseziele, Freizeitaktivitäten, E-Learning Material und Zeitungsartikel). Diese Informationen können entweder *explizit* (üblicherweise über die Bewertungen der Benutzer) oder *implizit* (für gewöhnlich beim Beobachten des Verhaltens der Benutzer, so wie gehörte Lieder, gesehene Filme, heruntergeladene Programme und besuchte Museen) sein [18, 19, 20]. Hierbei kann für die Wertung nur die Tatsache genutzt werden, dass ein gewisses Ereignis mindestens einmal eingetroffen ist (der Benutzer hat schon mal das Technikmuseum in Berlin besucht) oder Parameter wie Häufigkeit oder Dauer (der Benutzer hat das Lied nur ein mal gehört und nach 20 Sekunden abgebrochen). Zur Verbesserung der Qualität oder Verfeinerung der Empfehlungen können auch demographische Informationen (wie Alter, Herkunft oder Geschlecht) verwendet werden [3, 21]. Durch Soziale Netzwerke stehen Empfehlungssystemen nun potentiell auch Informationen zur Verfügung wie *Tweets*, *Posts*, *Follower*, *Abonnements* oder die Vernetzung zu weiteren Informationsquellen. Diese Infromationen können entweder direkt roh weiterverwendet (z.B. der sich aus Followern und Abonnements bildende Soziale Graph) oder mit Hilfe von *Regulären Ausdrücken* oder komplexeren Technologien des *Natural Language Processing* zur automatisierten Verwendung ausgewertet (wie *Tweets*, *Posts* und *Kommentare*) werden [22, 23]. Aktuell wird dazu tendiert Informationen aus dem Internet der Dinge *(Internet of Things)* [24] (wie *GPS Ortungsdaten*, *Echtzeit Gesundheitsparameter* oder *RFID Logs*) für Empfehlungssysteme zu nutzen [11]. Um dem Benutzer zufriedenstellende Vorhersagen und Empfehlungen zu liefern, nutzen Empfehlungssysteme unterschiedliche Quellen zum Erhalt einer möglichst sättigenden Menge an Informationen [25]. Die Benutzung von Empfehlungssystemen im Internet ist stetig gestiegen, was zur Benutzung in vielen, unterschiedlichen Anwendungsbereichen führte [11]. Die am häufigsten von Forschungsartikeln behandelten Anwedungsbereiche sind Filme [26] und Musik [27, 18]. Aber auch Themen wie Bücher, Dokumente, E-Learning und Fernsehen, so wie viele andere sind Gegenstand der Forschungsliteratur zu Empfehlungssystemen [11].

Möchte man das Prinzip eines Empfehlungssystems undifferenziert und kurz umschreiben, so ist es die Bestrebung eine *Brauchbarkeitsfunktion* (*utility function*) zu entwickeln.

Genauer gesagt bilden alle Objekte o einer Domäne die Objektmenge O ($o \in O$), die Benutzer u bilden zusammengefasst die Benutzermenge U ($u \in U$). Jedes Objekt o steht in einer Beziehung zu jedem Benutzer u. Diese Beziehung wird *Brauchbarkeit* oder vorhergesagte Bewertung r genannt ($r \in R$) und ist eine Abbildung der oben genannten *Brauchbarkeitsfunktion*. Daraus ergibt sich eine n (Anzahl der Objekte) mal m (Anzahl der Benutzer) Matrix, so dass für die *Brauchbarkeitsfunktion* gilt:

$$f_{Brauchbarkeit}(o_i, u_j) = r_{i,j} \mid 0 \leq i < n,\ 0 \leq j < m,\ o \in O,\ u \in U,\ r \in R \qquad (2.1)$$

In Bezug auf die Matrixdarstellung werden die Mengen O und U auch Dimensionen genannt, so spricht man von der Objektdimension und der Benutzerdimension. Initial wird die *Brauchbarkeit* lediglich für die Objekt-Benutzer-Paare gegeben, für die dem System bereits eine Interaktion bekannt ist. Die *Brauchbarkeit* stellt die Nützlichkeit eines Objektes für einen Benutzer bzw. sein Interesse für dieses dar [3]. Welche Werte und wie diese von der Bewertungsfunktion ermittelt werden, sind domäneabhängig und Teil der Designentscheidung. Oftmals sind dies *explizite* Bewertungen des Benutzers, wie die Sterne einer Produktbeurteilung bei *Amazon*[1], aber auch implizite wie *1* für "wurde schon gesehen" und *0* für "wurde noch nicht gesehen" in einem Filmempfehlungssystem eines fiktiven Videoverleihers. Die Aufgabe des Systems ist es nun, die Abbildungen der *Brauchbarkeitsfunktion* $f_{Brauchbarkeit}$ auf Objekt-Benutzer-Paare vorherzusagen und dem Benutzer die N bestbewerteten Objekte vorzuschlagen. Üblich ist es mehr als ein Objekt zu empfehlen. Ob die Objekte dem Benutzer bekannt oder unbekannt sind, ist wiederum eine Designentscheidung [25]. Hat ein Benutzer bei *Amazon* das Produkt A mit fünf Sternen und das Produkt B mit einem Stern bewertet, so bildet dies eine Wissensbasis, auf der die Bewertung für ein Prdukt C vorhergesagt werden kann.

Welche Informationen und wie diese genutzt werden, um das Abbild der *Brauchbarkeitsfunktion* vorherzusagen, macht den Unterschied zwischen den einzelnen Ansätzen und ihren spezifischen Implementierungen aus [3].

2.1 Klassische Empfehlungssysteme

Die in der Benutzung und Fachliteratur am häufgisten vorkommenden Ansätze sind das kollaborative und inhaltsbasierte Filtern. An diser Stelle wird klar gestellt, inwiefern sich die beiden Ansätze voneinander unterscheiden, welche Anforderungen an sie gestellt werden, mit welchen bekannten Problemen man konfrontiert wird und wie sich diese lösen lassen.

Die Ansätze lassen sich im Grunde durch die Art der verwendeten Informationen und auf welche Weise diese vom System verwendet werden klassifizieren. Zum einen

[1] **Amazon**: `http://www.amazon.com/`, Zugriff am: 16.02.2016

lassen sich benutzergebundene Informationen verwenden, etwa Bewertungen, Auswertung seiner Interaktion oder konkretes Wissen über den Benutzer (z.B *demografische* Angaben wie *Geschlecht, Alter* oder der *Wohnort*). Zum anderen können die Informationen nur vom Objekt abhängen, die Eigenschaften die das Objekt charakterisieren. Objekte können zusätzlich über *assoziatives Wissen* miteinander verbunden werden. Dies geschieht mit Hilfe von *social tagging* [11] oder Ontologien. Klassische Empfehlungssysteme beschränken sich für gewöhnlich auf die Objektdimension und die Benutzerdimension, so dass man von einem zwei dimensionalen Empfehlungssystem spricht. Theoretisch kann die *Brauchbarkeitsfunktion* noch weitere Parameter (wie den Kontext), die objekt- und benutzerunabhängig sind, einbeziehen.

Die beiden Ansätze lassen sich nach der Quelle der Informationen und ihrer Nutzung wie folgt unterteilen [21]:

- ***Kollaboratives Filtern:*** Die Grundannahme hierbei ist, dass, wenn unterschiedliche Benutzer veschiedene Objekete ähnlich bewerten, sie auch einen vergleichbaren Geschmack haben. Ein Benutzer mit einem vergleichbaren Geschmack kann eine nützliche Empfehlung geben. Das System versucht demnach, ähnliche Benutzer zu ermitteln, um beispielsweise Objekte zu empfehlen, über dessen gute Qualität sich die ähnlichen Benutzer einig sind. Die Objekteigenschaften können so theoretisch komplett vernachlässigt werden.

- ***Inhaltsbasiertes Filtern:*** Hierbei sind die charakterisierenden Eigenschaften der Objekte und die Bewertung des Benutzers ausschlaggebend. Das System versucht über die Bewertungen unterschiedlicher Objekte zu bestimmen, welche Eigenschaften den Benutzer überzeugen könnten und darauf basierend, weitere ähnliche Objekte oder zumindest Objekte mit ähnlichen Merkmalen zu empfehlen.

In der Literatur werden häufig noch *semantisches Filtern* und *demografisches Filtern* als einzelne Ansätze aufgezählt. Beide lassen sich soweit abstrahieren, dass sie dem *inhaltsbasierten Filtern* bzw. *kollaborativen Filtern* untergeordnet werden können. Das Objektprofil lässt sich mit *semantischem Wissen* anreichern. Demografische Angaben lassen sich als zusätzliche Eigenschaften der Benutzerprofile sehen, die bei der Bestimmung ähnlicher Benutzer behilflich sein können [21].

2.1.1 Kollaboratives Filtern

Der Ansatz des kollaborativen Filterns basiert auf der Annahme, dass eine Person, die zum Beispiel auf der Suche nach neuer Musik ist, wahrscheinlich einen Bekannten fragen wird, von dem sie weiß, dass dieser einen ähnlichen Musikgeschmack hat. Die Vorhersage der *Brauchbarkeit* von Objekten für bestimmte Benutzer wird mit Hilfe der Bewertungen dieser Objekte von anderen Benutzern berechnet. Formeller ausgedrückt wird der Wert der *Brauchbarkeitsfunktion* $f(u, o)$ von Objekt o für Benutzer u durch die *Brauchbarkeiten* $f(u', o)$ aller Benutzer $u' \in U$ errechnet, die u *ähnlich* sind.

Das System kann eine *Benutzer-Objekt-Matrix* erzeugen, in der die bekannten Bewertungen der Benutzer enthalten sind, wie etwa die Bewertungen (z.B. zwischen einem und fünf Sternen) von Büchern. Jede Zeile dieser Matrix entspricht einer Repräsentation eines Benutzers u, also seine Präferenzen, die nun genutzt werden kann, um die ähnlichsten Benutzer zu finden, also solche deren Repräsentation mit der von u am stärksten korreliert. Nun werden die am besten bewerteteten Objekte dieser ähnlichsten Benutzer empfohlen.

Ein explizites Wissen über die zu empfehlenden Objekte ist nicht erforderlich. Auf diese Weise können subjektive Verbindungen zwischen Objekten besser angesprochen werden als bei der inhaltsbasierten Filterung. Die ersten bekannten kollaborativen Empfehlungssysteme, welche die Vorhersage von *Bewertungen* für *Benutzer-Objekt-Paare* automatisierten sind Grouplens [5], Ringo [7] und Jester [6].

Funktionsweise

Nach [28] können kollaborative Empfehlungssysteme in zwei generelle Klassen gruppiert werden *memory-based* und *model-based*. Im Folgenden werden beide Konzepte erläutert.

memory-based Diese Algorithmen sind heuristisch und generieren Bewertungsvorhersagen basierend auf allen bisher von Benutzern bewerteten Objekten. Die unbekannte Bewertung $r_{u,o}$ für den Benutzer u und das Objekt o wird für gewöhnlich über die Bewertungen einiger anderer (häufig der N ähnlichsten) Benutzer für das selbe Objekt o ermittelt:

$$r_{u,o} = \operatorname*{aggr}_{u' \in \hat{U}}(r_{u',o}) \tag{2.2}$$

Wobei $aggr(r_{u',o})$ die Aggregationsfunktion ist und $\hat{U}$ die Menge der N ähnlichsten Benutzer, welche das Objekt o bewertet haben. N kann beliebig gewählt werden und von 1 bis zur Anzahl aller Benutzer reichen. Wie die Aggregationsfunktion den Wert berechnet, ist ebenfalls eine Designentscheidung. Beispiele hierfür sind [3]:

$$r_{u,o} = \frac{1}{N} \sum_{u' \in \hat{U}} r_{u',o} \tag{2.3}$$

$$r_{u,o} = k \sum_{u' \in \hat{U}} sim(u, u') \times r_{u',o} \tag{2.4}$$

$$r_{u,o} = \bar{r}_u + k \sum_{u' \in \hat{U}} sim(u, u') \times (r_{u',o} - \bar{r}_{u'}) \tag{2.5}$$

wobei k ein normalisierender Koeffizient ist und üblicherweise als $k = \frac{1}{\sum_{u' \in \hat{U}} |sim(u,u')|}$ festgelegt wird. Die durchschnittliche Bewertung des Benutzers u $\bar{r}_u$ wird definiert

als:

$$\bar{r}_u = \frac{1}{O_u} \sum_{o \in O_u} r_{u,o} \tag{2.6}$$

wobei O_u die Menge aller vom Benutzer u bewerteten Objekte ist. Im einfachtsen Fall lässt sich die Aggregation mit einer Durchschnittsfunktion (siehe Formel 2.3) ermitteln. Der häufigste Aggregationsansatz ist jedoch die *gewichtete Summe* (siehe Formel 2.4). Das Ähnlichkeitsmaß *sim(u.u')* zwischen den Benutzern u und u' ist im Grunde die Messung des Abstandes zwischen den Benutzern und dient hier als *Gewicht*. Je *ähnlicher* sich die Benutzer sind, desto größer ist das *Gewicht* von $r_{u',o}$ für die Vorhersage der Bewertung $r_{u,o}$. Ein Hauptproblem des Ansatzes der *gewichteten Summe* (wie in Formel 2.4) ist, dass nicht berücksichtigt wird, dass unterschiedliche Benutzer die Bewertungsskala verschieden interpretieren können. Um diesem Problem entgegenzuwirken wird weitgehend die *angepasste gewichtete Summe* (siehe Formel 2.5) angewandt. Anstatt die absoluten Werte der Bewertungen zu nutzen, werden die Abweichungen von den Durchschnitssbewertungen der entsprechenden Benutzer aufsummiert.

Es gibt unterschiedliche Ansätze für die Implementierung der Ähnlichkeitsfunktion *sim(u,u')* in kollaborativen Empfehlungssystemen [3]. In den meisten dieser Ansätze ist die Ähnlichkeit von den Bewertungen der Benutzer abhängig. Die populärsten Ansätze sind der korrelationsbasierte (*correlation-based*) und der kosinusbasierte (*cosine-based*):

$$sim(x,y) = \frac{\sum_{o \in O_{xy}} (r_{x,o} - \bar{r}_x)(r_{y,o} - \bar{r}_y)}{\sqrt{\sum_{o \in O_{xy}} (r_{x,o} - \bar{r}_x)^2 \times \sum_{o \in O_{xy}} (r_{y,o} - \bar{r}_y)^2}} \tag{2.7}$$

$$sim(x,y) = \cos(x,y) = \frac{\sum_{o \in O_{xy}} r_{x,o} r_{y,o}}{\sqrt{\sum_{o \in O_x} r_{x,o}^2} \sqrt{\sum_{o \in O_y} r_{y,o}^2}} \tag{2.8}$$

Wobei O_{xy} die Schnittmenge der von den Benutzern x und y bewerteteten Objekte ist. Im korrelationsbasierten Ansatz wird der *Pearson-correlation-coefficient* (siehe Formel 2.7) zum Messen der Ähnlichkeit verwendet. Im kosinusbasierten Ansatz (siehe Formel 2.8) werden die Benutzer x und y als Vektoren in einem m-dimensionalen Raum interpretiert, wobei $m = |O_{xy}|$. Die Ähnlichkeit zwischen den beiden Vektoren kann nun als Kosinus des Winkels zwischen ihnen gesehen werden.

Unterschiedliche Empfehlungssysteme können abhängig von ihrer Domäne unterschiedliche Ansätze nutzen, um die Ähnlichkeit zwischen Benutzern und die Vorhersage der *Brauchbarkeit* von Objekten möglichst zu optimieren und so effizient wie möglich zu gestalten. Eine solche Strategie ist es, die Benutzerähnlichkeiten *sim(x,y)* bereits im Vorfeld zu berechnen und lediglich ab und an zu aktualisieren. So wird die Berechnung einer Empfehlung wesentlich effizienter und dauert kürzer [3]. Traditionell wird für das kollaborative Filtern die Ähnlichkeit der Benutzer genutzt.

Sarwar et al. [29] schlagen vor, die oben genannten Ähnlichkeitstechniken auf Objekte anzuwenden, um so Bewertungen mehr über ähnliche Objekte als über ähnliche Benutzer zu berechnen. [29] und [30] zeigen empirisch, dass ein solcher *item-based* Algorithmus *performanter* sein und vergleichbare oder bessere *Genauigkeit* bieten kann als *user-based* Algorithmen.

model-based Wenn die Menge der Daten, über die Empfehlungen generiert werden sollen, zu groß ist, als dass man noch effizient direkt auf der Benutzer-Objekt-Matrix operieren könnte, braucht man eine andere Lösung als den *memory-based* Ansatz. Eine Alternative bilden die *model-based* Algorithmen, welche die existierenden Bewertungen nutzen, um ein voraussagendes Modell zu erlernen. Mit Hilfe des Modells werden die Bewertungen für die fehlenden Benutzer-Objekt-Paare vorhergesagt [31, 28]. Hierfür gibt es unterschiedliche Ansätze, die hauptsächlich Methoden aus dem Bereich des *maschinellen Lernens* verwenden. Beispielsweise schlägt Breese [28] einen kollaborativ basierten Ansatz mit Wahrscheinlichkeitsgewichtung vor. Ein Beispiel dafür ist das *clustering* auf Grundlage des *Naive Bayes* Modells, in dem eine feste Anzahl an gleichgesinnten Benutzern aus dem Datensatz erzeugt wird, um so einfacher Vorhersagen machen zu können. Weitere erfolgreiche Ansätze sind die *latent factor models* (wie die *matrix factorization* [32]), bei denen eine feste Anzahl an Faktoren aus den Daten extrapoliert werden, welche Eigenschaften von Benutzern und Objekten charakterisieren.

Beschränkungen und Probleme

Ein großer Vorteil kollaborativer Empfehlungssysteme ist, dass sie die Objekte tatsächlich nicht inhaltlich kennen müssen. Dadurch ergibt sich die Möglichkeit, Objekte zu empfehlen, die sich von den bisherigen dem Benutzer u bekannten Objekten stark unterscheiden. Dieser Ansatz hat jedoch auch Einschränkungen, wie im Folgenden beschrieben wird.

New User Damit das System einem Benutzer Objekte empfehlen kann, muss es diesen zuerst kennen. Das System braucht also eine hinreichende Menge an Informationen über den Benutzer, insbesondere seine Bewertungen, um ihm eine gute Empfehlung geben zu können. Es wurden unterschiedliche Techniken eingeführt, um dieses Problem zu lösen. Die meisten nutzen den *hybriden* Ansatz, indem sie kollaboratives und inhaltsbasiertes Filtern verbinden [3].

New Item Ähnlich wie bei einem *neuen Benutzer*, haben kollaborative Systeme Schwierigkeiten mit neuen Objekten. Abgesehen von den Bewertungen verwendet das System zur Berechnung der Empfehlungen keine weiteren Informationen über das Objekt. Daher brauchen auch die Objekte eine ausreichende Menge an Bewertungen, um vom System bei Empfehlungen überhaupt berücksichtigt zu werden. Diesem Problem kann man ebenfalls mit *hybriden* Empfehlungssystemen entgegentreten [3]. Zusammen mit dem *new user* Problem ergibt sich das *cold-start* Problem,

das bedeutet, dass ein neu aufgesetztes System ohne gute Datengrundlage kaum oder schlecht empfehlen kann.

Sparsity Die Anzahl echter Bewertungen ist häufig sehr klein, gemessen an der Größe der gesamten *Benutzer-Objekt-Matrix*. Es kann dazu kommen, dass einige Objekte von nur einem kleinen Bruchteil der Benutzer bewertet wurden. Auch wenn diese Bewertungen hoch sind, werden diese Objekte folglich seltener empfohlen als populäre Objekte. Genauso kann ein Benutzer mit einem ungewöhnlichen Geschmack das Problem haben, dass es keine wirklich ähnlichen Benutzer im System gibt und somit schlechte Empfehlungen erhalten [33].

2.1.2 Inhaltsbasiertes Filtern

In inhaltsbasierten Emfephlungssystemen wird das Ergebnis der *Brauchbarkeitsfunktion* für ein Benutzer-Objekt-Paar $f_{Brauchbarkeit}(u, o)$ mit Hilfe der Bewertungen ermittelt, die der Benutzer anderen Objekten $o_i \in O$ gegeben hat, welche als "ähnlich" gelten. Beispielsweise versucht ein Filmempfehlungssystem aus den vom Benutzer "gemochten" Filmen die Eigenschaften zu extrahieren (wie etwa Schauspieler, Regisseur, Genre, Thema usw.) und auf diesen Informationen (*Attributen*) basierend ein Benutzerprofil zu erstellen. Es werden nun nur Filme empfohlen die eine hohe Ähnlichkeit zu diesem Benutzerprofil aufweisen.

Der inhaltsabsierte Ansatz hat seine Ursprünge in den Forschungsgebieten des *information retrieval* [34] und des *information filtering* [35]. Viele der gegenwärtigen inhaltsbasierten Empfehlungssysteme konzentrieren sich auf das Empfehlen von Objekten, die textuelle Informationen enthalten (wie Dokumente und Internetseiten) [3]. Die Verbesserung im Verhältniss zu traditionellem *information retrieval* kommt vom Gebrauch der *Benutzerprofile*, die Informationen über den Geschmack, die Präferenzen und die Bedürfnisse des jeweiligen Benutzers enthalten. Diese Informationen werden entweder *explizit* (z.B. durch Bewertungen oder Umfragen) oder *implizit* durch das Interagieren mit dem System erfasst.

Funktionsweise

Formell gesagt wird jedem Benutzer u und jedem Objekt o ein Profil zugewiesen, in dem die Gewichtungen für die im System vorkommenden *Attribute* festgehalten sind. *Attribute* sind die charakterisierenden Eigenschaften aller Objekte. Wenn sich die Profile eines Benutzers und eines Objekts ausreichend ähneln, ist das Objekt ein Kandidat für eine Empfehlung.

Objektprofil Sei *Content(o)* ein *Objektprofil*, also eine Menge von Attributen die das Objekt o charakterisieren. Für gewöhnlich wird dieses Profil erstellt, indem eine Menge an Eigenschaften aus dem Objekt extrahiert wird (der *Inhalt*) und mit diesen die Zugehörigkeit zu den Attributen berechnet wird. Da inhaltsbasierte Empfehlungssysteme hauptsächlich entwickelt wurden, um textbasierte Objekte zu emp-

fehlen, sind die Attribute meistens *keywords*, also im Text vorkommende oder diesen beschreibende Schlüsselwörter. Beispielsweise repräsentiert das *Syskill & Weber* System [9] den Inhalt von Dokumenten mit den 128 informativsten Wörtern (*keywords*). Die *Wichtigkeit* eines solchen Wortes k_i in einem Dokument d_j wird durch eine *Gewichtung* w_{ij} bestimmt. Diese *Gewichtung* kann auf unterschiedliche Weise definiert sein. Eines der bekanntesten Maße für die Gewichtung von *keywords* aus dem Bereich des *information retrieval* ist das *term frequency-inverse document frequency (TF-IDF)* Maß [34]. Somit ergibt sich für ein *Objektprofil* ein Vektor mit den Gewichten der n im System vorkommenden Attribute:

$$Content(o_i) = (w_{i1}, w_{i2}, \ldots, w_{in}) \tag{2.9}$$

Benutzerprofil Sei *ContentBasedProfile(u)* das Profil des Benutzers u. Dieses Profil besteht wiederum aus den Gewichtungen der im System vorkommenden Attribute. Diese Gewichtungen spiegeln die Präferenzen und den Geschmack des Benutzers wider. Solche Profile werden erstellt, indem der Inhalt der Objekte, mit denen der Benutzer interagiert hat, analysiert werden. Für gewöhnlich werden hierfür *keyword* Analyse Techniken aus dem Bereich des *information retrieval* angewandt [34]. So lassen sich auch die Benutzerprofile sowie die Objektprofile als Vektoren der Gewichtungen der n im System vorkommenden Attribute schreiben:

$$ContentBasedProfile(u_i) = (w_{i1}, w_{i2}, \ldots, w_{in}) \tag{2.10}$$

Die Gewichtungen w_{ij} spiegeln die Relevanz des *Attributs* k_j für den Benutzer u_i wider. Die Gewichtungen der Benutzerprofile können aus den unabhängigen Objektprofilen berechnet werden. Hierfür können unterschiedliche Techniken angewandt werden, wie etwa der *Rocchio Algorithmus* [36]. Andererseits können beispielsweise *Bayesian classifier* verwendet werden, um die Wahrscheinlichkeit zu bestimmen, dass ein Objekt vom Benutzer gemocht wird [9]. Bei Domänen mit Objekten, die viele unterschiedliche Eigenschaften haben, hat sich der *Winnow Algorithmus* als nützlich erwiesen [4].

Brauchbarkeit Für gewöhnlich wird eine *Brauchbarkeitsfunktion* in inhaltsbasierten Empfehlungssystemen wie folgt definiert [3],:

$$f_{Brauchbarkeit}(u, o) = score(ContentBasedProfile(u), Content(o)) = score(\vec{w}_u, \vec{w}_o) \tag{2.11}$$

wobei sowohl das Benutzerprofil als auch das Objektprofil gleich große Vektoren mit Gewichtungen der im System vorkommenden Attribute sind. Die *score* Funktion ist somit eine Funktion, welche die Ähnlichkeit zwischen den beiden Profilen ermittelt. Zu diesem Zweck wird häufig, wie bei kollaborativen Systemen auch schon, das *kosinus Ähnlichkeitsmaß* verwendet [3],:

$$score(\vec{w}_u, \vec{w}_o) = \cos(\vec{w}_u, \vec{w}_o) = \frac{\sum_{i=1}^{A} w_{i,u} w_{i,o}}{\sqrt{\sum_{i=1}^{A} w_{i,u}^2} \times \sqrt{\sum_{i=1}^{A} w_{i,o}^2}} \tag{2.12}$$

wobei A die Anzahl aller *Attribute* ist.

Beispielsweise ist ein inhaltsbasiertes Empfehlungssystem in der Lage, einem Benutzer u, der viele *Artikel* zum Thema Bioinformatik liest, passende Artikel zu empfehlen. Dies erklärt sich durch die Tatsache, dass solche Artikel höchstwahrscheinlich mehr fachspezifische *Begriffe* (wie etwa "Genom" oder "Sequenzierung") enthalten als Artikel zu anderen Themen. Dementsprechend wird das Benutzerprofil des Benutzers u hohe Werte für die Gewichtung w_{ui} dieser speziellen Begriffe k_i aufweisen. Schließlich wird das System mit der *Brauchbarkeitsfunktion* denjenigen Objekten eine bessere Bewertung geben, die ebenfalls hohe Gewichtungen für Bioinformatik bezogene Begriffe haben.

Neben den traditionellen heuristischen Ansätzen, die hauptsächlich auf den Methoden des *information retrieval* basieren, werden auch andere Techniken wie der *Bayesian classifier* [37] und diverse Methoden aus dem Bereich des *maschinellen Lernens* wie das *clustering*, *Entscheidungsbäume* und *künstliche neuronale Netze* verwendet [9].

Beschränkungen und Probleme

Auch inhaltsbasierte Empfehlungssysteme haben methodenbedingte Einschränkungen, welche im Folgenden näher erläutert werden.

Limited Content Analysis Inhaltsbasierte Techniken sind durch die Attribute, die von den zu empfehlenden Objekten explizit ausgehen, eingeschränkt. Um also eine ausreichende Menge an Attributen zu erhalten, müssen entweder die Objekte in einer Form sein, die sich automatisiert *parsen* lässt (wie etwa ein Text) oder die Attribute müssen manuell hinzugefügt werden. Während Techniken aus dem Bereich des *information retrieval* bei Objekten wie Dokumenten gut zum Extrahieren von Attributen funktionieren, gibt es Domänen, deren Objekte eine automatisierte Ableitung von Attributen problematisch machen. So lassen sich automatisierte Methoden zum Extrahieren von Attributen für multimediale Daten, wie Bilder, Audiostreams oder Videostreams, nur schwer anwenden. Ein weiteres Problem der *Limited Content Analysis* ist die Tatsache, dass, wenn zwei unterschiedliche Objekte zufällig das gleiche Profil haben, sie für das System nicht zu unterscheiden sind. Beispielsweise können zwei Bücher oder wissenschaftliche Artikel, die sehr ähnliche Begriffe in ähnlicher Menge verwenden, für das System als nahezu identisch gelten. Die Qualität eines solchen Textes spielt hierbei keine Rolle. Einem Benutzer, dessen Profil zu den Objekten passt, werden beide Objekte empfohlen, obwohl eines der Bücher möglicherweise sehr schlecht geschrieben und uninteressant ist.

Overspecialization Wenn ein Empfehlungssystem dem Benutzer nur Objekte empfehlen kann, deren Profile eine sehr hohe Ähnlichkeit zu seinem Profil aufweisen,

besteht die Gefahr, dass nur sehr homogene Empfehlungen gemacht werden. Beispielsweise wird einer Person, die hauptsächlich Rockmusik hört, eher kein Jazzlied empfohlen, obwohl ihr dieses vielleicht gefallen könnte. Andererseits sollten die Empfehlungen auch nicht zu unterschiedlich ausfallen, um die Zufriedenheit des Benutzers zu sichern. Idealerweise wird das System dem Nutzer eine ausgewogene Menge an Objekten empfehlen, die zum einen mit hoher Wahrscheinlichkeit seinen Geschmack treffen und zum anderen aus Bereichen kommen, die der Benutzer eventuell nicht selbst entdeckt hätte.

New User Wie bei kollaborativen Empfehlungssystemen haben auch inhaltsbasierte Ansätze *cold-start* Probleme. Diese beschränken sich allerdings auf neue Benutzer. Ein Benutzer muss mit einer ausreichenden Menge an Objekten interagiert haben, um zufriedenstellende Empfehlungen erhalten zu können. Das Empfehlungssystem ist nicht in der Lage, treffende Empfehlungen für Benutzerprofile mit wenig aussagenden Gewichtungen zu ermitteln.

2.1.3 Anforderungen und Schwierigkeiten

Beide Ansätze, sowohl das *kollaborative* als auch das *inhaltsbasierte* Filtern, haben ihre Stärken und Schwächen. Für das *cold-start* Problem sind beide anfällig, allerdings in unterschiedlichen Ausprägungen. So besteht beim inhaltsbasierten Ansatz lediglich das *new-user* Problem, welches sich beispielsweise mit Hilfe demografischer Daten zum neuen Benutzer lösen lässt, indem diese Informationen als *a priori* Wissen interpretiert und zur Klassifizierung in einem *cluster-basierten* Model benutzt werden [21]. Ein *new-item* Problem besteht nicht, da die Eigenschaften des neuen Objekts ausreichen, um dieses in die bestehende Menge an Objekten einzuordnen. Um das *cold-start* Problem zu lösen, werden häufig beide Ansätze mit weiteren Methoden vermischt. So entstehen *hybride* Empfehlungssysteme, in denen versucht wird, die Stärken aus unterschiedlichen Ansätzen zu vereinen [3].

Der Ansatz des kollaborativen Filterns hat den starken Vorteil, dass zwischen einzelnen Objekten subjektive Verbindungen erzeugt werden können, welche nicht durch den schieren Vergleich der Eigenschaften möglich gewesen wären. Die Objekte benötigen jedoch hinreichend viele Bewertungen, um populär zu werden und so eine gewisse Relevanz für das System zu erlangen. Nur so ist es möglich, dass Objekte überhaupt Teil der besten N Empfehlungen werden. Dieses Problem wird auch als das *long tail* Problem bezeichnet [38]. Ein weiteres Problem beider Systeme ist, dass ein System dazu tendiert, sich auf die dominanten Vorlieben ihrer Benutzer zu konditionieren und nur noch sehr monotone Empfehlungen zu geben. Dieses Problem der *overspecialization* [21] zieht noch weitere Bedenken mit sich. Denkbar ist ein Szenario, in dem ein Empfehlungssystem die einzige Schnittstelle zwischen Benutzer und Objekten ist. So wird der Benutzer nur Zugang zu den Objekten haben können, welche vom System als relevant angesehen werden. Eine solche "filter-bubble"könnte also dazu führen, dass einem Benutzer der Zugang zu potentiell wichtigen Informationen verwehrt bleibt [39]. Trotz des Problems der *overspecialization* haben inhalts-

basierte Empfehlungssysteme häufig die Eigenart, die Popularität oder Qualität von Objekten nicht zu erkennen. Dies kann sich positiv auf das Nutzererlebnis auswirken: Der potentiell positiv wahrgenommene Effekt wird *serendipity* genannt und bedeutet, dass der Benutzer mit tendentiell unbekannten oder unpopulären Objekten "überrascht" werden kann, die er ansonsten höchstwahrscheinlich nicht selbst gefunden hätte.
Insgesamt ist es wichtig, dass sowohl zu unterschiedliche Objekte als auch zu ähnliche Objekte aus der Menge der Empfehlungen herausgefiltert werden. Beispielsweise möchte ein Benutzer üblicherweise thematisch verschiedene Zeitungsartikel empfohlen bekommen. Dieses Problem wird als *accuracy-diversity dilemma* bezeichnet [40].

Benutzer ändern im Laufe der Zeit ihre Präferenzen, das System muss schnell genug auf solche Änderungen reagieren, um den Benutzer zufriedenzustellen. In der Regel reagieren Systeme nur sehr träge auf Präferenzänderung [3]. Andererseits möchte man als Benutzer nicht, dass Kurzzeitinteressen mit Langzeitinteressen inferieren, wie z.B dass ein plötzliches Interesse an Weihnachtsliedern im Winter, dazu führt, dass man diese noch im darauf folgenden Sommer empfohlen bekommt. Solche Probleme werden in der Litaratur als *plasticity vs. stability* zusammengefasst [41]. Diesem Problem lässt sich beispielsweise mit Hilfe von Kontextinformationen entgegenwirken. So ließen sich alle Empfehlungen ausschließlich im Kontext der Jahreszeiten tätigen. Dadurch hätten Interaktionen mit dem System im Winter keinen Enfluss auf Empfehlungen im Sommer. Die Verwendung von Kontextinformationen für Empfehlungen wird im nächsten Abschnitt näher erläutert.

Kapitel 3

Kontext

Die meisten existierenden Empfehlungssysteme verwenden zur Berechnung von Empfehlungen lediglich die Benutzerdimnesion und die Objektdimension. Obendrein werden in den meisten Ansätzen relevante Objekte empfohlen, ohne zusätzliche kontextuelle Informationen, wie etwa Zeit, Ort oder die Begleitung anderer Personen, hinzuzunehmen [14]. Dass der Kontext bei Entscheidungsprozessen eine wichtige Rolle spielt, zeigt bereits die Verhaltensforschung [13]. Von einem Empfehlungssystem könnte nun also gefordert werden, dass dieses nicht stets die gleichen Empfehlungen gibt, sondern die Entscheidungen vom aktuellen Kontext abhängig macht. So können die Empfehlungen an einen Benutzer bei unveränderter Datenlage kontextabhängig variieren. Die Einbeziehung von Kontextinformationen kann nicht nur die Qualität der Empfehlungen potentiell steigern, vielmehr ermöglicht sie erst einige neue Anwendungen. Beispielsweise spielt die *Zeit* in vielen Szenarien eine wichtige Rolle. Jede Person tendiert im Allgemeinen dazu, zu unterschiedlichen Tageszeiten unterschiedlichen Tätigkeiten nachzugehen. Dies wirkt sich für gewöhnlich auf die Interessen und Bedürfnisse der Personen aus. Morgens wird eher anregende Musik gehört, wohingegen abends Musik zur Entspannung gehört wird [42]. In Anwendungen für mobile Endgeräte spielt der *Ort* oftmals eine wichtige Rolle. Ermöglicht der Benutzer dem Gerät den Zugriff auf die Standortdaten, können diese verwendet werden um etwa ein Restaurant in der näheren Umgebung zu empfehlen, anstatt blind eines in einem anderen Land zu empfehlen.

In diesem Kapitel soll darauf eingegangen werden, was unter dem Konzept des Kontexts verstanden wird, wie solche Informationen beschafft werden können und wie diese Informationen in Empfehlungssystemen verwendet werden können. Es werden insbesondere drei Ansätze, kontextuelle Informationen in den Empfehlungsprozess mit einfliessen zu lassen, näher erläutert: das kontextuelle Vorfiltern (*pre-filtering*), Nachfiltern (*post-filtering*) sowie das kontextuelle Modellieren.

3.1 Kontext: Begriffsklärung

Was genau ist gemeint, wenn von *Kontext* gesprochen wird? Welche Arten von Kontext gibt es? Wie lässt sich dieser erfassen, modellieren und formalisieren? Welcher Kontext ist wann relevant? Dieser Abschnitt soll genau diese Fragen thematisieren und beantworten.

3.1.1 Definition

Der *Kontext* ist ein facettenreiches Konzept, an dessen Erforschung viele wissenschaftliche Fachrichtungen interessiert sind. Unter ihnen sind die Informatik (insbesondere Künstliche Intelligenz), Kognitionswissenschaften, Linguistik, Psychologie und die Philosophie [14]. Da das Konzept des Kontexts von so vielen unterschiedlichen Fachrichtungen untersucht wird und diese dazu tendieren, ihre eigene spezifische Sicht auf die Bedeutung dieses Begriffs zu haben, ist es schwierig eine einheitliche Definition zu finden. Eine einfache und allgemeine Definition des Begriffs aus einem Wörterbuch wie "Bedingungen und Umstände die eine Sache beeinflussen" [43] gibt zwar eine grobe Idee davon, was gemeint ist, wird unserer Anforderung allerdings nicht gerecht, da der Begriff nicht ausreichend abgegrenzt wird. Bazire und Brézillon haben 150 verschiedene Definitionen des Begriffs Kontext vorgestellt und untersucht [44]. Sie sind zu folgender Beobachtung gekommen:

> *"[...]in jeder Fachrichtung ist es schwierig, eine relevante und zufriedenstellende Definition zu finden. Ist Kontext die Umgebung eines gegebenen Objekts? Ist es die Menge an Elementen die irgendeinen Einfluss auf das Objekt haben? Ist es möglich, Kontext a priori zu definieren oder müssen die Effekte a posteriori beobachtet werden? Ist er etwas Statisches oder etwas Dynamisches? [...] Welcher Kontext ist für unsere Forschung relevant? Der Kontext der Person? Der Kontext der Aufgabe? Der Kontext der Interaktion? Der Kontext der Situation? Wann beginnt ein Kontext und wann endet er? Was sind die echten Zusammenhänge zwischen Kontext und Kognition?"*

Es wird deutlich, dass sich die Absicht eine einheitliche hinreichende Definition zu finden als äußerst schwierig darstellt. Allerdings geben die oben genannten Fragen wichtige Erkenntnisse darüber, welche Abgrenzungen essentiell sind, um so etwas wie einen Kontext zu formalisieren und in einem automatisierten System zu verwenden. Dourish klassifiziert den Kontext in zwei unterschiedliche Konzepte [45]: die gegenständliche Ansicht (*representational view*) und die interaktionelle Ansicht (*interactional view*).

In der *gegenständlichen Ansicht* wird der Kontext als eine *Menge vordefinierter Attribute* gesehen, die identifizierbar und a priori bekannt sind. Sie sind gewissermaßen statisch, da sie im Laufe der Zeit keiner signifikanten Veränderung unterliegen. Diese können daher problemlos in einem kontextsensitiven System verwendet werden. Im Kontrast dazu nimmt die *interaktionelle Ansicht* an, dass das Verhalten einer Person zwar von einem gewissen zu Grunde liegendem Kontext induziert wird, dieser aber nicht zwangsweise beobachtbar ist. Vielmehr wird eine *bidirektionale* Beziehung zwischen Kontext und Aktion gesehen. Der Kontext beeinflusst eine Tätigkeit, die Tätigkeit wiederum beeinflusst oder kreiert sogar erst einen Kontext. Einfach gesagt ist dieser Ansicht nach der Kontext ein sich dynamisch ergebender Umstand, dessen Merkmale sich weder vorhersagen noch zur Vorhersage ordentlich verwenden lassen.

Es stellt sich also die viel bedeutendere Frage, inwieweit sich Kontext formalisieren lässt, um für konkrete Anwendungen in Empfehlungssystemen verwendet werden zu können.

3.1.2 Kontext in Empfehlungssystemen

Wie bereits erwähnt lässt sich der Kontext als endliche Menge von Attributen sehen, die relativ statisch sind und dementsprechend kaum einer Veränderung unterliegen. Tatsächlich folgen die meisten Ansätze kontextsensitiver Empfehlungssysteme dieser *gegenständlichen Ansicht* [45]. Entscheidend ist, dass jeder eine Benutzer-System-Interaktion umgebender Faktor auf die ein oder andere Weise als Kontext interpretiert werden kann. Intuitiv lassen sich, zumindest scheinbar, unabhängige Faktoren als Kontext voneinander abgrenzen und bennen. Darunter fallen bereits genannte Umstände wie *Ort*, *Zeit*, *Stimmung*, *Wetter* oder *Intention* des Benutzers. Eine beobachtbare Verhalten ändernde Intention wäre beispielsweise die Absicht eines Benutzers, ausnahmsweise ein Produkt für einen Bekannten zu kaufen. Da dies nicht dem üblichen Kaufverhalten entspräche, sollte diese Interaktion nicht für zukünftige Empfehlungen berücksichtig werden.

Eine Möglichkeit Kontext zu klassifizieren, ist zu unterscheiden, worauf sich dieser bezieht oder aus welchem Umstand heraus er ermittelt wird. Es wird danach unterschieden, ob er aus dem die Interaktion umgebenden Umfeld stammt oder ausdrücklich den Benutzer betrifft. Umfeldorientierte Informationen ergeben sich aus Kontexten wie Zeit, Ort oder Wetter. Benutzerbezogene Informationen sind zum Beispiel Stimmung, Intention, akutes Interesse oder demografische Angaben (wie etwa das Alter, welches sich für gewöhnlich mit der Zeit unidirektional ändert). Wie auch immer der Kontext klassifiziert wird, er muss in eine Form gebracht werden, die für ein Empfehlungssystem verwendbar ist.

3.1.3 Kontextrepräsentation

Um den Kontext formal zu repräsentieren, wird eine Menge von Variablen definiert. Diese Variablen modellieren die relevanten Informationen, die von einem Kontext ausgehen. Damit diese Informationen logisch verarbeitet werden können, werden Strukturen benötigt, in die die jeweiligen Kontexte eingebettet werden.
Eine solche Struktur kann von den Eigenschaften der Informationen zum Zeitpunkt der Erfassung abgeleitet werden. Zeitstempel und Temperatur sind beispielsweise bereits in numerischer Form dargestellt und darüber hinaus ergeben sie ohne weitere Interpretation eine kontinuierliche Struktur. Die kontinuierliche Struktur der Zeit kann beispielsweise in einem *zeitsensitiven* Empfehlungssystem verwendet werden. In einem *kollaborativen* Empfehlungssystem könnte zusätzlich mit einer *Vergessensfunktion* die Gewichtung der Bewertung beeinflusst werden. Je länger die letzte Interaktion mit einem Objekt her ist, desto geringer wird die Bewertung gewichtet [46].

Lässt sich ein Kontext aufgrund seiner Komplexität oder Abstraktheit nicht ausdrücklich skalieren oder in eine numerisch kontinuierliche Form bringen, wie etwa die Stimmung oder Intention eines Benutzers, dann lässt sich dieser Kontext zumindest in Kategorien aufteilen [14]. Die Stimmung ließe sich etwa in "glücklich", "traurig", "wütend", "neutral" oder "gelangweilt" aufteilen. Aber auch eindeutiger Kontext, wie ein *Zeitstempel* oder ein *Standort*, lassen sich kategorisieren. Der Ort könnte also beispielsweise mit den Kategorien ("zu Hause", "bei der Arbeit", "bei Oma" oder "in der Universität") benannt sein. Stattdessen könnte man den Ort auch lediglich als Stadtnamen repräsentieren. Es wird ersichtlich, dass sich jeder Kontext beliebig kategorisieren lässt.

In vielen Fällen ist es möglich und praktikabel, den Kontext zu hierarchisieren, das heißt in mehreren Ebenen zu kategorisieren. Dies dient im Allgemeinen dazu, einen feingranularen Kontext in einem System repräsentieren zu können. So ließe sich beispielsweise der Ort in folgender Form hierarchisieren: (Land → Region → Stadt → Bezirk → Straße → Hausnummer). Anstatt aus jeder Adresse eine gänzlich eigene Kategorie zu machen, wird der Kontext also aufgeteilt, um übersichtlicher und anwendbarer zu sein. Mit Hilfe einer solchen Hierarchisierung ließe sich auch ein *Baum* implementieren, der dazu genutzt werden könnte, einen speziellen Kontext zu konstruieren. Dabei werden gewisse Kontexte erst erreichbar, wenn ein vorhergehender zunächst unabhängiger Kontext ausgewählt wurde.

Prinzipiell lässt sich der Kontext in einem System, das mehrere Kontexte zulässt oder Kontexte auf mehrere hierarchische Ebenen zerlegt, in einer Variable zusammenfassen und als *k-Tupel* darstellen. Wobei k die Anzahl aller Kontexte bzw. Kontextebenen im System ist.

$$c = (c_1, c_2, \ldots, c_k) \tag{3.1}$$

$$c' = (zuHause, mitPartner, Sommer, abends, Regen) \tag{3.2}$$

Kontextabstraktion

Sollen kontextuelle Informationen von einem Empfehlungssystem verarbeitet werden, werden diese in der Regel in kategorisierter Form benötigt. Numerisch kontinuierliche Kontextinformationen, wie etwa Zeitstempel oder die Temperatur, können daher in Intervalle unterteilt werden. Ein Zeitstempel wird als Kontextinformation erst dann interessant, wenn er in von Menschen akzeptierte Intervalle unterteilt wird. Es folgen Beispiele für die Unterteilung von Kontextinformationen. Ein Tag kann in seine Tageszeiten unterteilt werden (siehe Abbildung 3.1), ein Jahr in seine Jahreszeiten (siehe Abbildung 3.2) oder die gemessene Temperatur in *empfundene Wärme* (siehe Abbildung 3.3).

Erst so können wiederkehrende Verhaltensweisen von Menschen als Information genutzt werden, um die Qualität der Empfehlungen zu optimieren.

Kontext	Morgen	Mittag	Abend	Nacht
Zeit	0:00 - 6:00	6:00 - 12:00	12:00 - 18:00	18:00 - 0:00

Abbildung 3.1: Kategorisierung der Kontextinformation *Zeit* in Tageszeiten

Kontext	Winter	Frühling	Sommer	Herbst
Monate	Jan. - März	April - Juni	Juli - Sep.	Okt. - Dez.

Abbildung 3.2: Kategorisierung der Kontextinformation *Zeit* in Jahreszeiten

Kontext	sehr kalt	kalt	neutral	warm	heiß
Temperatur	$<$ -5°C	-5°C - 10°C	10°C - 20°C	20°C - 30°C	$>$ 30°C

Abbildung 3.3: Kategorisierung der Kontextinformation *Temperatur* in empfundene Wärme

Kontextunschärfe Kategorisierung erfordert eine strikte Unterteilung in Intervalle [14]. Solche Grenzen sind jedoch zu abstrakt, häufig zu grob und bilden die Realität nicht richtig ab. Die Grenzen können verschwimmen, Für einen Benutzer werden der 4. April und der 20. Juni für gewöhnlich wenig gemeinsam haben, obwohl beide in den Kontext des Frühlings fallen werden. Der 30. März hingegen wird dem 4. April viel ähnlicher sein, was den allgemeinen Kontext angeht. Diese beiden Tage fielen jedoch in unterschiedliche Kategorien. Neben dem Verschwimmen der Grenzen können diese auch subjektiv wahrgenommen werden. Was der Sommer für einen Deutschen ist, ist für einen Neuseeländer der Winter. Diese Beobachtung und die damit zusammenhängenden Probleme werden Kontextuntschärfe genannt (*fuzzy context*). Die im Beispiel für die Unterteilung der Temperatur angegebenen Werte sind lediglich eine Schätzung, die Zuweisungen werden nicht jeder Person gerecht, da das Wärme-/Kälteempfinden sehr subjektiv ist. Als Lösungsansatz schlagen Lee und Park et al. [47] vor, die Kontexte stets zu gewichten, falls eine Unterteilung nicht eindeutig ist. Der Kontext c für eine Interaktion mit dem System wäre demnach nicht mehr lediglich eine Menge von Begriffen c_i (siehe Beispiel 3.2), sondern vielmehr eine Menge von Tupeln der Form (c_i, w_i), wobei c_i jeweils jeden im System vorkommenden Kontext darstellt und w_i das Gewicht. Das Gewicht i gibt an, wie sehr der Kontext i gerade zutrifft. So könnte der erste Frühlingstag als 50 % Winter der Kategorie und 50 % der Kategorie Frühling zugehörigen gelten. Er könnte wie in Beispiel 3.3 dargestellt werden.

$$c'' = \Big((Winter, 0.5), (Frühling, 0.5), (Sommer, 0), (Herbst, 0)\Big) \tag{3.3}$$

3.1.4 Beschaffen kontextueller Informationen

Orientiert man sich beim Konzept des Kontexts nach der *gegenständlichen Ansicht*, impliziert dies laut Adomavicius et al. [12], dass der Kontext, der vom System verwendet werden soll, im Vorfeld identifiziert und beschafft werden muss. Das heißt bevor eine Empfehlung gemacht wird, muss feststehen, welcher Kontext überhaupt verwendet wird. Darüberhinaus müssen die einzelnen Kontextinformationen bereits im Datensatz, der als Wissensgrundlage für das System gilt, vorhanden sein. Welcher Kontext für ein konkretes Empfehlungssystem betrachtet wird, ist eine Designentscheidung und muss von einem *Domänenexperten* ermittelt werden. Häufig stellt der *Domänenexperte* zunächst eine breite Auswahl an unterschiedlichen Kontexten auf. Als nächstes wird der Datensatz mit entsprechenden Kontextinformationen angereichert, falls diese überhaupt ermittelbar sind. Nachdem die Kontextinformationen in den Datensatz integriert wurden, werden statistische Verfahren angewandt, um empirisch zu evaluieren, ob sich die ausgewählten kontextuellen Attribute signifikant auf die Entscheidungen der Benutzer auswirkten. Es kann also betrachtet werden, ob beispielsweise die Vorhersagegenauigkeit eines Filmempfehlungssystems unter Betrachtung der Wochentage steigt.

Kontextuelle Informationen können auf drei unterschiedliche Arten gewonnen werden: explizit, implizit sowie durch Inferenz [14].

Explizit Die *explizite* Gewinnung erfolgt, indem Benutzer direkt nach Informationen gefragt werden. Dies kann in Form einer Umfrage oder von Dialogen passieren. Beispielsweise könnte ein Musikempfehlungssystem nach der aktuellen *Stimmung* fragen, wenn gerade Musik gehört wird. Ein Internetdienst könnte auch spezifische Fragen an den Benutzer stellen, bevor es den Zugang zu bestimmten Funktionalitäten freigibt.

Implizit Kontextinformationen können auch *implizit* gewonnen werden. Zum einen lassen sich *temporale* Informationen über die Interaktion mit dem System in Form eines Zeitstempels erfassen. Weitere Informationen wie *Ort*, *Druck* oder *Temperatur* lassen sich mit Hilfe von Sensoren messen. Stehen dem System entsprechende Informationen zur Verfügung, lassen sich über zusätzliche Wissensquellen (wie etwa Datenbanken, Internetservices oder Soziale Netzwerke) weitere Informationen ableiten. So lassen sich Informationen zum Wetter an einem bestimmten Ort zu einer bestimmten Zeit über einen Internetservice ermitteln, wenn dem System *Zeitstempel* und *Standort* zur Verfügung stehen. Insbesondere muss mit dem Benutzer, in der Absicht Kontextinformationen zu erhalten, nicht interagiert werden.

Inferenz Der Kontext lässt sich mit Hilfe von *data mining* oder satistischen Methoden inferieren. Lässt sich ein Kontext nicht direkt aus den Daten lesen, existiert immer noch die Möglichkeit einen gewissen Kontext aus den Daten zu schlussfolgern. So lässt sich beispielsweise anhand der Auswahl an Kanälen, die eine Person in einem

Benutzer	Objekt	Bewertung
Alice	Asia-Imbiss	5
Alice	Charlie's Pizza	4
Alice	Eisdiele	3
Bob	Eisdiele	3
Bob	Asia Imbiss	4

Tabelle 3.1: Beispiel - Restaurant Bewertungen

Haushalt mit demografisch unterschiedlichen Bewohnern trifft, mit einer gewissen Wahrscheinlichkeit schlussfolgern um welche Person es sich vermutlich handelt. Um solche Inferenzen möglich zu machen, muss ein Vorhersagemodell mit entsprechend repräsentativen Daten trainiert werden. Der Erfolg solcher Inferenzen ist insbesondere von eben diesem Klassifzierer (Vorhersagemodell) abhängig.

3.2 Methoden in kontextsensitiven Empfehlungssystemen

Klassische Empfehlungssysteme operieren in einem zweidimensionalen Raum, die Dimensionen sind *Benutzer* und *Objekte*. Die Datengrundlage, aus der diese Matrix erzeugt wird, besteht aus Tupeln der Form (*Benutzer,Objekt,Bewertung*). Damit wirddargestellt, welche Bewertung ein Benutzer einem Objekt (implizit oder explizit) gibt. Eine beispielhafte Datengrundlage befindet sich in Tabelle 3.1.

Eine *Brauchbarkeitsfunktion* ($f_{Brauchbarkeit} : U \times O \rightarrow R$, im Folgenden f_B), die auf einer solchen 2D-Matrix ($Benutzer \times Objekte$) operiert, bekommt jeweils einen Benutzer ($u_i \in U$) und ein Objekt ($o_j \in O$) als Parameter und bildet auf eine Bewertung ($r_{ij} \in R$) ab. Möchte ein Benutzer nun eine Liste von Empfehlungen erhalten, wird das Empfehlungssystem die Objekte zurückgeben, die die höchste vorhergesagten Bewertungen durch die *Brauchbarkeitsfunktion* erhalten haben.

Soll der Kontext nun als zusätzliche Information betrachtet werden, kann dieser als zusätzlicher Parameter für die *Brauchbarkeitsfunktion* verstanden werden. Die zweidimensionale Definition wird also um den Kontext erweitert ($f_B : U \times O \times C \rightarrow R$). Wie bei klassischen Empfehlungssystemen bestimmt die *Brauchbarkeitsfunktion*, welche Bewertung für ein Benutzer-Objekt-Paar vorhergesagt wird. Zusätzlich wird hier unterschieden, in welchem Kontext diese Bewertung gilt. Dies beruht auf der Annahme, dass sich Benutzer in verschiedenen Kontexten unterschiedlich entscheiden. Beispielsweise wird sich ein Benutzer (*Bob*) im *Winter* weniger für eine *Eisdiele* interessieren als im *Sommer*. Daher sollten die vorhergesagten Bewertungen erwartungsgemäß unterschiedlich ausfallen.

$$f_B(Bob, Eisdiele, Sommer) > f_B(Bob, Eisdiele, Winter)$$

Durch die zusätzlichen Parameter (es können mehrere unterschiedliche Kontexttypen gleichzeitig betrachtet werden) in der *Brauchbarkeitsfunktion* ergibt sich ein

Benutzer	Objekt	Kontext (Jahreszeit)	Bewertung
Alice	Asia-Imbiss	Sommer	5
Alice	Asia-Imbiss	Winter	5
Alice	Charlie's Pizza	Herbst	4
Alice	Eisdiele	Frühling	3
Bob	Eisdiele	Sommer	5
Bob	Eisdiele	Winter	1
Bob	Asia Imbiss	Frühling	4

Tabelle 3.2: Beispiel - Restaurant Bewertungen mit Kontext

multidimensionaler Raum, auf dem das System arbeitet. Damit das System einem Benutzer eine Menge von Objekten empfehlen kann, braucht es ein Benutzerprofil. Das Benutzerprofil wird aus den in das System eingegebenen Informationen ermittelt. Für ein kontextsensitives System besteht die Datengrundlage nun aus Tupeln der Form (*Benutzer,Objekt,Kontext,Bewertung*). Es wird also dargestellt, wie ein Benutzer das Objekt in einem bestimmten Kontext bewertet (implizit oder explizit). Wie eine solche Datengrundlage aussehen kann, wird in Tabelle 3.2 vorgestellt.

Die Matrix des Empfehlungssystems, in dem die ermittelten Bewertungen gespeichert sind, ist nun multidimensional. Um eine Empfehlungsliste zu generieren, benötigt das System über die Identität des Benutzers hinaus noch den Kontext, in dem empfohlen werden soll, so dass jede Empfehlung stets kontextgebunden ist. Adomavicius et al. [14] unterscheiden drei unterschiedliche Ansätze, wie der Kontext in einem Empfehlungssystem eingesetzt werden kann.

- ***Pre-Filtering / Vorfiltern:*** Die Kontextualisierung der Empfehlungssystem*eingabe*. Die Datengrundlage wird noch vor der Ermittlung der vorhergesagten Bewertungen auf die Tupel reduziert, welche den gewünschten Kontext enthalten. Nach der Vorauswahl kann ein klassisches 2D-Empfehlungssystem zur Vorhersage der Bewertungen und Ausgabe der Empfehlungen genutzt werden.

- ***Post-Filtering / Nachfiltern:*** Die Kontextualisierung der Empfehlungssystem*ausgabe*. Eingangs werden kontextuelle Informationen ignoriert. Die Liste der Empfehlungen wird von einem klassischen 2D-Empfehlungssystem auf Grundlage *aller* Daten generiert. Erst die Liste der Empfehlungen wird an den gewünschten Kontext angepasst.

- ***Contextual Modelling / kontextuelles Modellieren:*** Die Kontextualisierung der Empfehlungs*funktion*. Die kontextuellen Informationen fließen direkt in den Empfehlungsalgorithmus ein.

3.2.1 Pre-Filtering / Vorfiltern

Der Ansatz des kontextuellen Vorfilterns nutzt kontextuelle Informationen um im Vorfeld eine Auswahl aus der allgemeinen Datengrundlage auszuwählen. Zum Erstellen der Benutzer-Objekt-Matrix und dem Ermitteln der vorhergesagten Bewertungen werden lediglich Informationen verwendet, die dem gefragten Kontext zuzuordnen sind. Möchte eine Benutzerin eine Empfehlung für einen Film, den sie mit ihrem Beziehungspartner am Samstagabend im Kino gucken will (wie in Formel 3.4 beispielhaft dargestellt), werden zum Ermitteln der empfohlenen Filme nur die (*Benutzer,Objekt,Kontext,Bewertung*)-Tupel aus der Datengrundlage verwendet, die dem Kontext entsprechen. Alle anderen in der Datengrundlage vorkommenden Informationen werden ignoriert.

$$c = (Beziehungspartner, Kino, Samstagabend) \tag{3.4}$$

Der Vorteil dieses Ansatzes ist, dass für das Empfehlen, also die Ermittlung der Bewertungen und die Ausgabe der Empfehlungen, ein klassisches gut erforschtes Empfehlungssystem wie das *kollaborative Filtern* oder das *inhaltsbasierte Filtern* eingesetzt werden kann. Ein so exakter Kontext wie im gegebenen Beispiel kann jedoch zu schmal sein. Einige der Kontexte könnten irrelevant sein, da das Verhalten der Benutzerin unabhängig von diesem spezifischen Kontext ist. Beispielsweise könnte die Filmwahl einer Benutzerin, unabhängig von *Ort*, *Wochentag* und *Tageszeit* sein, wenn gegeben ist, dass dies mit dem *Beziehungspartner* passiert. Eine so spezifische Filterung der Datengrundlage wäre demnach unnötig. Alternativ könnte das Filmwahlverhalten der selben Benutzerin am *Samstag* identisch mit dem Verhalten am *Freitagabend* oder *Sonntag* sein. Der Kontext *Samstag* ließe sich also generalisieren. Statt *Samstag* könnte der Kontext nun *Wochenende* heißen und mehr Daten aus der Datengrundlage zulassen. Ein anderes sich aus diesem Ansatz ergebendes Problem ist, dass der Kontext so schmal oder unpopulär (wie etwa Sportaktivitätsempfehlungen bei Regen) sein kann, dass die aus dem Vorfiltern resultierende Datengrundlage zu klein ist, um ausgereifte qualitiativ hochwertige Empfehlungen zu geben. Dieses Problem wird in der Literatur als Seltenheit (*sparsity*) bezeichnet [14]. Beide Probleme können nach Adomavicius et al. [12] mit Hilfe der Kontexgeneralisierung angegangen werden. Die Idee basiert darauf, dass sich ähnliche Kontexte generalisieren lassen, falls der Kontext zu spezifisch und die Datengrundlage zu klein ist. So kann eine Kontexthierarchie erzeugt werden, in der nach Belieben weiter nach oben gegangen werden kann, falls der Kontext immer noch zu spezifisch ist. Für die Begleitung, den Ort und die Zeit könnten die Generalisierungshierarchien wie folgt aussehen.

$$Begleitung: Beziehungspartner \rightarrow Freund \rightarrow Bekannter \rightarrow Irgendwer \tag{3.5}$$

$$Ort: Kino \rightarrow Irgendwo \tag{3.6}$$

$$Zeit: Samstagabend \rightarrow Samstag \rightarrow Wochenende \rightarrow Irgendwann \tag{3.7}$$

Der zuvor angegebene Kontext könnte nun also beispielsweise folgende Formen haben:

$$c' = (Beziehungspartner, Irgendwo, Irgendwann) \tag{3.8}$$
$$c'' = (Bekannter, Kino, Wochenende) \tag{3.9}$$
$$c''' = (Irgendwer, Kino, Samstagabend) \tag{3.10}$$

Für welche Generalisierung man sich entscheidet wird nun zum wichtigen Problem. Zum einen besteht die Möglichkeit, gewisse Kontexte statisch zu generalisieren, wie etwa Samstag und Sonntag zu *Wochenende*, bzw. die restlichen Wochentage als *Arbeitswoche* zusammenzufassen. Oder es wird empirisch durch die Evaluation der Vorhersageleistung aller Kontextgeneralisierungen entschieden, wie im gegebenen Fall generalisiert werden soll.

3.2.2 Post-Filtering / Nachfiltern

Die Kontextinformationen werden beim Nachftilern vor der Erstellung der Empfehlungsliste ignoriert. Die Empfehlungsliste wird mit einem klassischen 2D-Empfehlungssystem auf der gesamten Benutzer-Objekt-Matrix generiert. Erst nachdem die Empfehlungsliste vom klassischen System ausgegeben wurde, wird der spezifische gewünschte Kontext benutzt, um die Empfehlungen zu filtern oder die Reihenfolge durch Anpassung und Gewichtung zu ändern.
Das System analysiert die Benutzerpräferenzen für den gewünschten Kontext und benutzt dann diese Erkenntnisse, um im gewünschten Kontext irrelevante oder unwichtige Objekte herauszufiltern, bzw. die Bewertungen anzupassen und somit die Reihenfolge der Empfehlungen zu ändern.

So wie viele andere Ansätze im Bereich der Empfehlungssysteme lässt sich auch das Nachfiltern in *heuristische* und *modellbasierte* Ansätze klassifizieren. Der *heuristische* Nachfilterungsansatz fokussiert sich darauf, gemeinsame Eigenschaften (*Attribute*) von Objekten für einen Benutzer in einem speziellen Kontext zu ermitteln. Die Objekte in der Empfehlungsliste werden dann entweder danach *herausgefiltert*, ob sie eine vorher festgelegte Mindestanzahl dieser im Kontext relevanten Attribute besitzen, oder die Reihenfolge wird zusätzlich davon bestimmt, wieviele dieser relevanten Attribute die Objekte besitzen. In *modellbasierten* Ansätzen können Vorhersagemodelle gebaut werden, welche die Wahrscheinlichkeit ermitteln, mit welcher ein Benutzer sich für ein Objekt im gegebenen Kontext entscheiden würde. Sie nutzen dann diese ermittelte Wahrscheinlichkeit, um entweder Objekte aus der Empfehlungsliste *herauszufiltern*, die unter einem bestimmten Schwellenwert liegen oder wieder die Reihenfolge zu ändern, indem die vorhergesagte Bewertung zusätzlich mit der ermittelten Wahrscheinlichkeit der Akzeptanz gewichtet wird.

Wie beim *Vorfiltern* ist ein großer Vorteil des *Nachfilterns*, dass für den Empfehlungsprozess selbst bereits gut erforschte 2D-Empfehlungssysteme verwendet werden können.

3.2.3 Contextual Modelling / Kontextuelles Modellieren

Die Besonderheit dieses Ansatzes ist, dass im Gegensatz zu den beiden oben genannten Ansätzen, die Kontexinformationen direkt in den Prozess der Empfehlung mit einbezogen wird und einen Parameter zur Vorhersage der Bewertungen bildet. Dementsprechend wird tatsächlich auf einer multiddimensionalen Matrix gearbeitet. Auch im kontextuellen Modellieren lassen sich die Ansätze in *heuristische* und *modellbasierte* klassifizieren. Klassische 2D-Ansätze wie das kollaborative Filtern lassen sich auf den multidimensionalen Raum ausdehnen. Es werden geeignete Abstands- bzw. Ähnlichkeitsfunktionen gebraucht, die den Abstand zwischen Punkten im multidimensionalen Raum messen können oder Vektoren vergleichen können. Beispielsweise können Bewertungen aus anderen Kontexten in den Empfehlungsprozess gewichtet mit einbezogen statt ignoriert zu werden. Für die modellbasierten Empfehlungssysteme können ebenfalls einige bereits bewährte 2D-Systeme in den multidimensionalen Raum erweitert werden.

Kapitel 4

Konzept

Im folgenden Kapitel wird die Herangehensweise an die Fragestellung vorgestellt. Das Ziel war, zu untersuchen, ob und wie sich der Kontext des Wetters auf die Qualität der Empfehlungen in Musikempfehlungssystemen auswirkt. Es wird erläutert, welche Kontexttypen untersucht wurden und wie der Versuchsaufbau aufgestellt wurde, also welche Ansätze gewählt wurden um die Fragestellung zu untersuchen. Des Weiteren wird näher darauf eingegangen, welche Designentscheidungen für die einzelnen Empfehlungsalgorithmen getroffen wurden.

Die empfohlenen Objekte der Musikempfehlungssysteme in der vorliegenden Arbeit sind *Artists*. Andere mögliche zu empfehlende Objekte in Musikempfehlungssystemen sind Lieder oder Genres. Es wurde entschieden, dass Musikgenres einen zu groben Raum für Empfehlungen bilden und Lieder zu feingranular für die geplante Evaluation sind. Der Datensatz gibt zu wenige differenzierte Informationen zum Musikhörverhalten der einzelnen Benutzer, als dass die gewünschte Qualität für Empfehlungen zu erreichen wäre. Eine Besonderheit der Domäne *Musikempfehlung* ist die Tatsache, dass Objekte, mit denen Benutzer bereits interagiert haben, tendenziell wiederkehren. Diese Information, der Hörzähler eines Benutzers für einen Artist, lässt sich zur Inferenz einer impliziten Bewertung nutzen, falls eine explizite Bewertung aus den Daten nicht hervorgeht [48].

Es werden zwei klassische Empfehlungsalgorithmen entwickelt, die als Basisalgorithmen für die zu implementierenden kontextsensitiven Empfehlungssysteme dienen. Die beiden klassischen Empfehlungssysteme sind konkrete Anwendungen kollaborativer und inhaltsbasierter Ansätze. Die kontextsenstiven Algorithmen verfolgen jeweils die Ansätze des Vor- und Nachfilterns. Am Ende werden also die Leistungen von sechs Algorithmen untereinander verglichen: *Kollaboratives Filtern*, *inhaltsbasiertes Filtern*, *kollaboratives Vorfiltern*, *inhaltsbasiertes Vorfiltern*, *kollaboratives Nachfiltern* und *inhaltsbasiertes Nachfiltern*.
Eine nähere Ausführung der verwendeten Algorithmen und die konkreten Entscheidungen für die Implementierung dieser befinden sich im Kapitel 4.2.

Die Fragestellung fokussiert sich auf die Rolle des Wetters bei dem Hörverhalten von Benutzern. Es wird untersucht, ob sich die Vorhersagegenauigkeit eines klassischen Musikempfehlungssystems verbessert, wenn man Wetter als zusätzlichen Kontextparameter aufnimmt.

4.1 Kontext

Eine sehr allgemein gehaltene Definition des Begriffs Wetter gibt der Duden [1]:

> *"Zustand der Atmosphäre zu einem bestimmten Zeitpunkt, an einem bestimmten Ort, der in Gestalt von Sonnenschein, Regen, Wind, Wärme, Kälte, Bewölkung o.Ä. in Erscheinung tritt."*

Es gibt demnach keine einheitliche Form, in der Wetter ausgedrückt wird, vielmehr können akute Zustände der Toposphäre an einem bestimmten Ort lediglich durch meteorologische Elemente wie Strahlung, Luftdruck, Lufttemperatur, Luftfeuchtigkeit und Wind, oder die daraus ableitbaren Elemente Bewölkung, Niederschlag, Sichtweite etc. beschrieben werden. Ein konkretes Beispiel für die Verwendung des Wetters als Kontext in Empfehlungssystemen ist, dass ein Musikempfehlungssystem für einen regnerischen Tag andere Artists empfiehlt als an einem sonnigen Tag.

Zusätzlich ist es wichtig zu erkennen ob die Ergebnisse, tatsächlich durch das Wetter bedingt sind und nicht nur an dem für die Empfehlung verwendeten Ansatz liegen. Zu diesem Zweck wird zusätzlich ein Referenzkontext in den gleichen Ansätzen untersucht, der unabhängig vom Wetter und bereits gut erforscht ist. Die Wahl für den Referenzkontext fiel auf die Kategorisierung der Zeit zu Wochentagen (siehe Abbildung 4.1), da der verwendete Datensatz diese Informationen bereits enthielt und der positive Einfluss des Kontexts der Wochentage auf die Qualität der Empfehlungen bereits nachgewiesen wurde [42]. Im Folgenden werden die Aspekte des Wetters vorgestellt, die als Kontexttyp verwendet wurden und wie sich diese Kontexttypen unterteilen.

Wetterbeschreibung Durch die Anreicherung des Datensatzes mit Wetterdaten, kamen Informationen in Form einer textuellen Beschreibung über das Wetter hinzu. Ursprünglich waren dies 48 unterschiedliche Beschreibungen. Diese wurden in die fünf folgenden Kategorien generalisiert:

$$Schnee, Regen, Sonnig/Klar, Nebel, Bewölkt$$

Temperatur Die gefühlte Temperatur (in ° Celsius) war in den angereicherten Daten als kontinuierlicher Wert angegeben. Diese Werte wurden in 5 Kategorien abstrahiert (siehe Abbildung 3.3).

Wolkenbedeckung Die Wolkenbedeckung wurde mit einem prozentualen Wert angegeben. Die prozentualen Werte wurden in vier Kategorien abstrahiert (siehe Abbildung 4.2).

[1] `http://www.duden.de/node/645896/revisions/1370437/view`, Zugriff am: 24.03.2016

Kontext	Arbeitswoche	Wochenende
Wochentage	Mo - Fr	Sa - So

Abbildung 4.1: Kategoriesierung der Kontextinformation *Wochentag*

Kontext	Klar	Teilweise Bewölkt	Überwiegend Bewölkt	Bewölkt
proz. Wert	0 - 25 %	25 - 50 %	50 - 75 %	75 - 100 %

Abbildung 4.2: Kategoriesierung der Kontextinformation *Wolkenbedeckung*

4.2 Algorithmen

Musikabspielprogramme oder Streamingdienste fordern den Benutzer selten dazu auf, eine explizite Bewertung vorzunehmen. Als Resultat gibt es kaum Datensätze mit expliziten Bewertungen der Objekte (häufig Artists oder Lieder). Um anspruchsvolle Empfehlungen geben zu können, muss ein Musikempfehlungssystem also die Bewertung eines Objekts implizit annehmen. Die häufigste Form, in der Musikdatensätze von Streamingdiensten oder von Musikabspielprogrammen vorliegen, sind Hörereignisse. Ein Hörereignis kann verkürzt beispielsweise so aussehen:

$$event = (User, Artist, Zeitstempel) \tag{4.1}$$

Die genaue Form und Beschaffenheit des Datensatzes für die vorliegende Arbeit wird in Kapitel 5.1 näher erläutert.

Aus diesen Hörereignissen wird zunächst der Hörzähler (*listening count*), also wie häufig ein Benutzer einen bestimmten Artist gehört hat, für jedes Benutzer-Artist-Paar ermittelt. Mit Hilfe des Hörzählers wird für jedes Benutzer-Artist-Paar eine implizite Bewertung geschätzt. Bei der Schätzung der Bewertung hat sich das Logarithmieren des Zählers bewährt. Hu et al. haben gezeigt, dass der Logarithmus des Zählers gut die Tendenzen der expliziten Bewertungen widerspiegelt [49]. Daher wurde in der vorliegenden Arbeit entschieden, diesem Verfahren nachzugehen und zusätzlich den Zähler über den größten Zähler des Benutzers zu normieren. Die Normierung ist für die Bestimmung der vorhergesagten Bewertungen im kollaborativen Filtern notwendig, da in der *angepassten gewichteten Summe* die konkreten Bewertungen der ähnlichsten Benutzer genutzt werden, um die vorhergesagten Bewertungen für die zu empfehlenden Artists zu berechnen. Die Formel zur Berechnung der Bewertung wird in Formel 4.2 dargestellt,

$$r_{ij} = \frac{ln\big(1 + freq(u_i, o_j)\big)}{ln(1 + maxfreq(u_i))}, \quad maxfreq(u_i) = maximum\{freq(u_i, o) | o \in O_i\} \tag{4.2}$$

wobei *freq* die Funktion ist, die angibt, wie häufig der Benutzer u_i den Artist o_j gehört hat.

4.2.1 Kollaboratives Filtern

Das erste klassische Empfehlungssystem, welches als Basis für die kontextsensitiven Empfehlungssysteme gewählt wurde, ist das kollaborative Filtern (siehe Kapitel 2.1.1). Es wurde entschieden, den *user-based* Ansatz zu wählen, also die ähnlichsten Benutzer zu suchen, anhand derer die Bewertungen für die Benutzer-Artist-Paare vorhergesagt werden, da dies die traditionelle und am weitesten verbreitete Methode ist [3]. Weiter wurde die *angepasste gewichtete Summe* (siehe Formel 2.5) als *Brauchbarkeitsfunktion* gewählt und die Funktion zur Ermittlung des *Pearson-korrelations Koeffizienten* (siehe Formel 2.7) als Ähnlichkeitsfunktion, da beide bevorzugte Methoden im Implementieren des *kollaborativen Filterns* in Musikempfehlungssystemen sind [48]. Damit die *angepasste gewichtete Summe* eine Bewertung für einen Artist vorhersagen kann, ist es erforderlich fest zu setzen, wieviele ähnlichste Benutzer vom Empfehlungssystem beachtet werden sollen. Dies ist ein Parameter, der von der Domäne und von der Datengrundlage abhängig ist. Herlocker et al. gibt an, dass sich die Menge von 50 ähnlichsten beachteten Benutzern bewährt hat. Besser ist es jedoch, den idealen Wert experimentell zu ermitteln.

4.2.2 Inhaltsbasiertes Filtern

Das andere klassische Empfehlungssystem basiert auf dem Ansatz des *inhaltsbasierten Filterns* (siehe Kapitel 2.1.2). Als Eigenschaften der Artists dienen von der *Last.fm Community* frei vergebene *Social Tags*. Auf Basis dieser *Tags* werden die Profile der Artists erzeugt. Ein Artistprofil ist ein Vektor der die Gewichte aller im System beachteten *Tags* für den Artists selbst enthält. Wieviele und welche Tags als Attribute für die Artistprofile zugelassen werden, ist eine Designentscheidung und hängt von der Domäne und der Art der für die Attribute benutzten Eigenschaften ab. Dieser Parameter wird vorzugsweise experimentell ermittelt. Die Gewichte der einzelnen *Tags* für die jeweiligen Artists wurden mit dem *TF-IDF-Maß* ermittelt (siehe Formeln 4.3 - 4.5) [50],

$$TF - IDF(t_k, d_l) = TF(t_k, d_l) \times IDF(t_k) \tag{4.3}$$

$$TF(t_k, d_l) = \frac{freq_{kl}}{maxfreq_l} \tag{4.4}$$

$$IDF(t_k) = ln\Big(\frac{N}{n_k}\Big) \tag{4.5}$$

wobei t_k der Term bzw. *Tag* ist und d_l das Dokument bzw. der *Artists* für die das Gewicht ermittelt werden soll. TF (Formel 4.4) teilt das Vorkommen des gefragten Terms durch das höchste Vorkommen eines Terms im Dokument. Im Fall des zu implementierenden Musikempfehlungssystems ist es jeweils die Anzahl der *Tags*, die einem Artist gegeben wurde. Als nächstes wird für den Benutzer, dem Artists empfohlen werden sollen, ebenfalls ein Profil angelegt. Dabei erhält der Benutzer all die Attribute, die ihm von den Profilen der von ihm bewerteten Artists zugeordnet werden. Die Gewichte der einzelnen *Tags* für den Benutzer ergeben sich aus den

normalisierten Summen der Gewichte der gehörten Artists jeweils mit der Bewertung des Benutzer-Artist-Paares relativiert (siehe Formel 4.6) [50],

$$w_{i,k} = weight(u_i, t_k) = \frac{1}{|O_i|} \sum_{o \in O_i} r_{i,o}\, w_{o,k} \tag{4.6}$$

wobei O_i die Menge aller vom Benutzer u_i bewerteten Artists ist. Zum Erzeugen der Empfehlungslisten werden nun die dem Benutzer ähnlichsten Artists ermittelt und ausgegeben. Die Ähnlichkeiten zwischen dem Benutzerprofil und den Artistprofilen werden mit dem Kosinus-Ähnlichkeitsmaß ermittelt (siehe Formel 2.12). Die Ähnlichkeit gilt als vorhergesagte Bewertung für den Artist.

4.2.3 Vorfiltern

Wie bereits im Kapitel 3.2.1 vorgestellt basiert der Ansatz des Vorfilterns darauf, dass der Datensatz im Vorfeld kontextualisiert wird. Er wird auf die Hörereignisse reduziert, die in dem gefragten Kontext stattgefunden haben. Daraufhin wird mit einem klassischen zweidimensionalen Algorithmus empfohlen. Dieser Ansatz soll wie von der Literatur vorgeschlagen ungeändert übernommen werden. Es ist nicht erforderlich, die Vorfiltermethode an den für die Empfehlung verwendeten Algorithmus anzupassen.

4.2.4 Nachfiltern

Um wie im Kapitel 3.2.2 vorgestellt die von einem Empfehlungssystem generierten Empfehlungen zu kontextualiseren, muss entschieden werden, welche empfohlenen Artists für den gewünschten Kontext relevant sind. Ein von Adomavicius et al. [14] vorgeschlagenes Verfahren ist die Ermittlung von Objekteigenschaften, die für einen Benutzer im gewünschten Kontext relevant sind. Welche Eigenschaften dies sein können, ist von den Informationen abhängig, die das Empfehlungssystem und die Datengrundlage bieten. Erst die so ermittelten für den gewünschten Kontext relevanten Artists werden dem Benutzer empfohlen. Für das Empfehlungssystem der vorliegenden Arbeit wurde die Relevanzermittlung nach dem zugrundeliegenden Algorithmus unterschieden. Beide Verfahren werden im Folgenden erläutert.

Kollaboratives Filtern Die Eigenschaften, die einem Artist zugerodnet werden, sind lediglich die ähnlichsten Benutzer. Ob ein Artist in einem gewissen Kontext für einen Benutzer relevant ist, ist also davon abhängig, ob mindestens einer der ähnlichsten Benutzer diesen Artist im gewünschten Kontext gehört hat. Alle Artists, die nicht von mindestens einem ähnlichsten Benutzer im gewünschten Kontext gehört wurden, werden aus der Menge der Empfehlungen entfernt.

Inhaltsbasiertes Filtern Die Entscheidung, ob ein Artist empfohlen wird, ist wieder davon abhängig, ob gewisse Eigenschaften den Artist für einen Benutzer in einem

gewissen Kontext relevant machen. Die betrachteten Eigenschaften sind in diesem Ansatz die *Last.fm Tags*. Es wird also zunächst für den Benutzer ermittelt, welche *Tags* im gewünschten Kontext relevant sind. Dies ist die vereinigte Menge der *Tags* aller Artists, die vom Benutzer im gefragten Kontext gehört wurden. Ob ein Artist aus der initialen Empfehlungsliste relevant ist, wird danach entschieden, ob er eine Mindestanzahl von für den Kontext relevanten Tags besitzt. Ist dies nicht der Fall, wird er aus der Empfehlungsliste entfernt.

Kapitel 5

Implementierung und Evaluation

Dieses Kapitel stellt den Datensatz vor, wie dieser gewonnen und bereichert wurde, so wie welche Schwierigkeiten sich dabei ergaben. Weiter wird kurz erläutert, welche Technologien verwendet wurden, um das Konzept umzusetzen und eine Evaluation möglich zu machen. Es wird eine Einführung in die Evaluation von Empfehlungssystemen gegeben, welche Messverfahren für das Experiment der vorliegenden Arbeit verwendet wurden und welche Ergebnisse das Experiment gebracht hat. Abgeschlossen wird das Kapitel mit einer Diskussion der Ergebnisse.

5.1 Datensatz

Es gestaltet sich schwierig, an frei verfügbare natürliche Datensätze für Musik zu kommen. Die *Privacy Policy* bekannter Streamingdienste erlaubt es nicht, Teilmengen der Datensätze zu veröffentlichen. Dadurch wären personalisierte Daten der Benutzer öffentlich verfügbar, was zur Verletzung der Privatsphäre führen würde.

Da Musikempfehlungssysteme zu den wenigen besser erforschten Domänen gehören, existieren bereits einige veröffentlichte Datensätze zum Musikhörverhalten. Darunter fallen die bekannteren Datensätze: *Million Song Dataset*[1], *Last.fm Dataset - 360K users*[2] und *Last.fm Dataset - 1K users*[3], wobei alle Musikhörereignisse von *Last.fm*[4] stammen. Leider erfüllen die meisten öffentlich verfügbaren Datensätze nicht die Anforderungen an die Bedürfnisse der vorliegenden Arbeit, welche im Folgenden vorgestellt werden.

Anforderung an den Datensatz: Der Datensatz soll aus einzelnen Musikhörereignissen bestehen und mindestens die Metainformationen (bzw. Kontextinformationen) *Zeitstempel* und *Standort* enthalten. Beide Informationen sind erforderlich, um den Datensatz mit Wetterinformationen über einen *Online Wetterdienst* anzureichern. Ein Musikhörerignis soll also möglichst folgende Form haben:

[1] `http://labrosa.ee.columbia.edu/millionsong/`
Zugriff am: 23.03.2016

[2] `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-360K.html`
Zugriff am: 23.03.2016

[3] `http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html`
Zugriff am: 23.03.2016

[4] **Last.fm:** `http://www.last.fm/`, Zugriff am: 23.03.2016

$$event = (Benutzer, Artist, Zeitstempel, Standort) \tag{5.1}$$

Ürsprünglich war geplant die *Last.fm API*[1] zu nutzen, um einen Datensatz mit Musikhörereignissen für Deutschland zu gewinnen. Jeder Benutzer von *Last.fm*[2] kann mit der *Scrobbler Software*[4] sämtliche Hörereignisse von bekannten Musikabspielprogrammen wie *Apple Itunes*[5] oder *Winamp*[6], so wie Streamingdiensten wie *Spotify*[7], *Deezer*[8] oder *Apple Music*[9] mitschneiden lassen. *Last.fm*[2] nutzt unter anderem diese Informationen für ihr Empfehlungssystem. Zusätzlich bot die *Last.fm API*[1] bis August 2015 API-Benutzern Zugriff auf diese Daten. Alle über die *Last.fm API*[1] erhaltenen Daten durften veröffentlicht und für den privaten Gebrauch (wie etwa eigene Applikationen) verwendet werden. Die Hörereignisse hatten abgesehen vom Herkunftsland des Benutzers keine Standort Daten. Der Standort sollte über die besuchten Veranstaltungen (*Events*) approximiert werden. Da *Last.fm*[2] im August 2015 unerwartet den kompletten Service zur unausgereiften Beta-Version eines neueren Systems *portierte*, wurden die meisten Funktionalitäten der Entwickler API bis heute vorerst stillgelegt[10]. Es war also unmöglich, an die Daten im Rahmen der Bearbeitungszeit dieser Arbeit zu kommen.

Nach längerer Recherche wurden zwei Datensätze gefunden, welche Hörereignisse von *Spotify* veröffentlichen, die mit Hilfe des Mikroblogdienstes *Twitter*[11] gesammelt wurden[23, 51]. Insbesondere der Datensatz von Hauger et al. (*MMTD*[12]) erfüllte die Anforderungen der vorliegenden Arbeit. Die Musikhörereignisse sind Benutzern zugeordnet und enthalten Metainformationen wie *Zeitstempel*, *Standort* und zusätzliche Informationen zu den Artists, welche verwendet werden konnten, um den Datensatz mit Wetterdaten und *Last.fm* Social Tags anzureichern.

5.1.1 Anreicherung

Um den Datensatz mit Wetterdaten anzureichern, musste ein Online Wetterdienst per API angesprochen werden. Als Dienst wurde hierbei die API *World Weather Online*[13] für historische Wetterdaten genutzt. Die Benutzung der API setzte voraus, dass die Zeitstempel im ISO 8601[14] Format angegeben sind. Hierfür wurden die

[1] **Last.fm API**: `http://www.last.fm/api`, Zugriff am: 23.03.2016
[2] **Last.fm:** `http://www.last.fm/`, Zugriff am: 23.03.2016
[4] **Last.fm Scrobbler**: `http://www.last.fm/download`, Zugriff am: 23.03.2016
[5] **Apple Itunes**: `https://www.apple.com/de/itunes/`, Zugriff am: 23.03.2016
[6] **Winamp**: `http://www.winamp.com/`, Zugriff am: 23.03.2016
[7] **Spotify**: `https://www.spotify.com/`, Zugriff am: 23.03.2016
[8] **Deezer**: `https://www.deezer.com/`, Zugriff am: 23.03.2016
[9] **Apple Music**: `http://www.apple.com/de/music/`, Zugriff am: 23.03.2016
[10] `https://getsatisfaction.com/lastfm/topics/api-known-issues`, Zugriff am 23.03.2016
[11] **Twitter:** `https://twitter.com/`, Zugriff am: 23.03.2016
[12] **Million Musical Tweet Dataset:** `http://www.cp.jku.at/datasets/MMTD/`, Zugriff am: 23.03.2016
[13] `http://www.worldweatheronline.com/`, Zugriff am: 23.03.16
[14] **ISO 8601:** internationaler Standard für die Formatierung zur Darstellung von Zeitangaben

vom ursprünglichen Datensatz verfügbaren Zeitstempel angepasst und mit der Zeitzonen Information erweitert (welche sich über die *Standortinformationen* ermitteln ließ). Die API von *World Weather Online* gibt historische Wetterdaten in höchstens stündlicher Granularität zurück und maximal gesammelte Wetterdaten von einem Tag pro API-Aufruf. Um die Anzahl der API-Aufrufe möglichst zu minimieren, wurden die Musikhörereignisse jeweils in Tage und Stunden zum Standort gruppiert. Alle Zeitstempel der Hörereignisse wurden zur näheren vollen Stunde gerundet.

Ausserdem wurde der Datensatz um *Tags* über die *Last.fm API* bereichert. Diese *Tags* (im Datensatz mit lfm_genre gekennzeichnet) dienen dem *inhaltsbasierten Ansatz* als Attribute der Artists und bilden somit die Eigenschaften der zu empfehlenden Objekte des Empfehlungssystems.

5.1.2 Bereinigung

Der Datensatz von Hauger et al.[23] enthielt ursprünglich 1.000.000 Hörereignisse. Aufgrund starker Verteilung der Hörereignisse/Benutzer ergibt sich eine enorme *sparsity* und viele Benutzer hatten eine nicht ausreichende Anzahl an Hörereignissen, um vertretbare implizite Bewertungen zu ermitteln. Ein Empfehlungssystem, welches nicht ausreichend Informationen über einen Benutzer besitzt, kann keine qualitativ hochwertigen Empfehlungen für diesen Benutzer generieren [11]. Daher wurden zunächst alle Benutzer aus dem Empfehlungssystem gefiltert, auf die weniger als 20 Hörereignisse fallen und Ausreißer mit mehr als 3000 Hörereignissen. Als nächstes wurden zum Erzielen einer geringeren *sparsity* also einer höheren Benutzer zu Artist-Dichte nach dem Vorbild von Herrera et al. [42] alle Artists aus dem Datensatz entfernt, die eine unterdurchschnittliche Popularität hatten und alle Benutzer entfernt, deren Menge bewerteter Artists unter den Median fielen. Die Begründung dieser Bereinigung liegt darin, dass die Hörereignisse ihren Ursprung im Streamingdienst *Spotify* haben. Spotify Benutzer tendieren dazu Radioplaylists und Playlists anderer Benutzer zu hören. Das führt dazu, dass einzelne Artists nicht im gewohnten Hörverhalten der Benutzer wieder auftauchen.

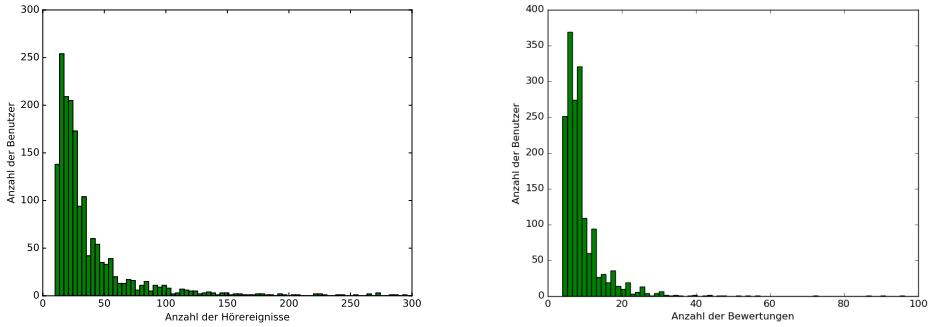
5.1.3 Eigenschaften

Einen Überblick zu den Werten des bereinigten Datensatzes gibt Tabelle 5.1. Es

	Benutzer	**Hörereignisse**	**Bewertungen**	**Artists**
Anzahl	1694	74441	15903	3104
∅ / **Benutzer**	-	44	9	-

Tabelle 5.1: Überblick Datensatz

ergeben sich durchschnittlich 9 bewertete Artists und 44 Hörereignisse pro Benutzer. Die Verteilungen der Hörereignisse pro Benutzer können in der Abbildung 5.1a und die Verteilung der Bewertungen pro Benutzer in der Abbildung 5.1b eingesehen werden.



(a) Verteilung der Hörereignisse (b) Verteilung der Bewertungen

Abbildung 5.1: Anzahl der Hörereignisse bzw. Bewertungen pro Anzahl der Benutzer

Sämtliche Hörereignisse des Datensatzes sind in einer der Arbeit in digitaler Form beiliegenden *TSV*-Datei (Tab-Seperated Values) gespeichert, deren *Header* folgende Form hat:

```
user_id artist_id datetime artist_name lfm_genre temperature
  \ cloudcover description weekday
```

Das Empfehlungssystem betrachtet vier verschiedene Kontexttypen aus dem Datensatz, wovon drei dem Kontext *Wetter* zugeordnet sind und einer die *Wochentage* (Tabelle 5.2) darstellt. Die Wetter-Kontexttypen sind: *Wetterbeschreibung* (Tabelle 5.3), *Temperatur* (Tabelle 5.4) und *Wolkenbedeckung* (Tabelle 5.5).

Kontext	Arbeitstag	Wochenende
Anzahl	53082	21359

Tabelle 5.2: Verteilung der Hörereignisse, Kontext: Wochentag

Kontext	Sonnig/Klar	Regen	Bewölkt	Nebel	Schnee
Anzahl	30529	21346	16819	4735	1012

Tabelle 5.3: Verteilung der Hörereignisse, Kontext: Wetterbeschreibung

Kontext	Warm	Neutral	Kalt	Heiß	Sehr Kalt
Anzahl	20145	19754	18603	12810	3129

Tabelle 5.4: Verteilung der Hörereignisse, Kontext: gefühlte Temperatur

Kontext	Klar	Teilweise Bewölkt	Bewölkt	Überwiegend Bewölkt
Anzahl	34648	17649	11734	10410

Tabelle 5.5: Verteilung der Hörereignisse, Kontext: Wolkenbedeckung

5.2 Implementierung

Für die Implementierung aller für die vorliegende Arbeit notwendigen Elemente wurde die Programmiersprache *Python*[15] in der Version 3.4 verwendet. Die Wahl fiel auf diese Programmiersprache, da der Entwicklerin viele Bibliotheken, Extensions und Frameworks zur Verfügung stehen, die insbesondere für wissenschaftliche Arbeit von großem Nutzen sein können. Darüber hinaus erlaubt die Syntax der Programmiersprache ein angenehmes Arbeiten und schnelles Erreichen von Zielen. Es wurde jeweils ein *Crawler* für das Gewinnen der für die Experimente erforderlichen Wetterinformationen und der für das inhaltsbasierte Filtern erforderlichen *Last.fm Tags* implementiert. Die Daten wurden mit einmaligen *Python-Skripten* angereichert und in die endgültige bereinigte Form gebracht. Die im Kapitel 4 konzeptuell vorgestellten Verfahren wurden für die Experimente als Prototypen umgesetzt. Ein Evaluationsframework zur Ermittlung der Leistungen der Algorithmen wurde für den Versuchsaufbau mit Schnittstellen zu den Algorithmen ebenfalls in *Python* implementiert. Für das Plotten der Evaluationsergebnisse in Diagrammen wurde die Python Bibliothek *matplotlib* [52] verwendet.

5.3 Evaluation

Seitdem die Forschung im Bereich der Empfehlungssysteme begann, wurde die Evaluation der Vorhersagen und Empfehlungen immer wichtiger. Es wurden immer mehr unterschiedliche Ansätze entwickelt, die unterschiedliche Stärken aufweisen. Für gewöhnlich muss sich eine Entwicklerin für einen von vielen Ansätzen für ihr Empfehlungssystem entscheiden. Hierbei könnte die Entwicklerin an unterschiedlichen Eigenschaften der Ansätze interessiert sein, wie etwa der *Vorhersagegenauigkeit*, Abdeckung (*coverage*) des Objektkatalogs, Neuerscheinungen (*novelty*) oder der Serendipität (*serendipity*) [15]. Um solche Eigenschaften in Abhängigkeit des ausgewählten Ansatzes, der Domäne und der verfügbaren Datengrundlage untersuchen zu können, sind einheitliche Evaluationsmetriken erforderlich. Darüber hinaus muss für den Versuchsaufbau geklärt sein, welche Datengrundlage zur Verfügung steht. Ist diese *natürlich* oder *synthetisch*? Findet die Evaluation lediglich *offline* mit einem Datensatz statt, oder lassen sich zusätzliche Rückmeldungen echter Benutzer berücksichtigen? Wie wird insgesamt evaluiert, also wie lassen sich die Messverfahren auf den Raum aller Benutzer anwenden, um eine möglichst zuverlässige Aussage machen zu können? Diese Fragen werden im Folgenden jeweils in Unterpunkten beantwortet [11].

[15] `https://www.python.org/`, Zugriff am: 27.03.16

5.3.1 Auswahl des Datensatzes und der Evaluationsart

Bevor ein Experiment ablaufen kann, muss geklärt sein, auf welcher Datengrundlage überhaupt evaluiert werden kann. Reicht es aus, für die gewünschten Metriken *offline* auf einem bestehenden Datensatz zu evaluieren oder sollten Tests mit echten Benutzern durchgeführt werden? Ist es angemessen Benutzerverhalten zu synthetisieren, falls nicht genügend natürliche Daten verfügbar sind?

Natürlicher vs. synthetischer Datensatz Häufig sind Forschende mit dem Problem konfrontiert, dass ein verfügbarer natürlicher Datensatz nicht genau die Anforderungen für die zu erforschende Domäne erfüllt. Sie stehen also vor der Wahl, diesen nicht gut passenden Datensatz zu nehmen oder einen Datensatz zu synthetisieren, der ihren Anforderungen gerecht wird. Es lassen sich Objekte mit gewünschten Eigenschaften und Benutzer mit speziellen Präferenzen modellieren. Mit einem Generator lassen sich unterschiedliche Verteilungen von Eigenschaften und Präferenzen erreichen. Simulierte Daten eignen sich sehr gut, um offensichtliche Fehler eines Empfehlungssystems zu ermitteln, sind jedoch auf keinen Fall genau genug, um das Verhalten und die Eigenschaften natürlicher Benutzer und Objekte zu modellieren [16]. Darüber hinaus können synthetische Datensätze die Evaluation stark verzerren. Es kann unfair sein, *andere* Algorithmen mit dem zu vergleichen, für den der Datensatz simuliert wurde, da der Datensatz zu gut an die Anforderungen angepasst sein könnte [16]. Es lässt sich also sagen, dass synthetische Datensätze idealerweise höchstens dann verwendet werden sollten, wenn noch nicht ausreichend natürliche Daten gesammelt wurden.

Online vs. Offline Evaluation Evaluationen können mit *offline* Analysen oder experimentellen Methoden mit echten Benutzern durchgeführt werden. Die Forschung um die Evaluation der Algorithmen fokussiert sich hauptsächlich auf die *offline* Analyse der Vorhersagegenauigkeit [16]. Die generierten Empfehlungen können mit einer Auswahl an Messverfahren untersucht und ausgewertet werden [15]. Ein großer Vorteil ist hierbei, dass die Evaluationen selbst unabhängig von echten Benutzern durchgeführt werden können. Evaluationen können schnell auf großen Datenmengen mit unterschiedlichen Algorithmen ausgeführt werden. Ein solches Experiment erfordert lediglich das Ausführen der Algorithmen auf den Datenmengen in einer vereinheitlichten Form. Für das Durchführen eines *offline* Experiments eignet sich die Methode der *k-fold cross-validation*, die im nächsten Unterkapitel vorgestellt wird. Das Nichteinbeziehen echter Benutzer bei der Evaluation hat zwei große Nachteile. Zum einen schränkt die natürliche *sparsity* der Bewertungen in natürlichen Datensätzen die Menge der Objekte ein, die evaluiert werden können . Zum anderen sind diese Experimente auf eine objektive Auswertung der Vorhersagegenauigkeiten beschränkt. In einer offline Analyse lässt sich nicht feststellen, ob ein Benutzer das Empfehlungssystem aufgrund seiner Vorhersagegenauigkeit verwenden würde, oder ob es dafür subjektivere Gründe geben kann, wie das äußere Erscheinungsbild der Benutzerschnittstelle (*User Interface*). Ein alternativer Ansatz ist es, Experimente

mit echten Benutzern durchzuführen. Dazu kann das System von der Benutzerin explizite Rückmeldung fordern (wie etwa "Hat Ihnen diese Empfehlung gefallen?") oder die Benutzerin kann dazu aufgefordert werden, eine Umfrage auszufüllen [16].

5.3.2 K-Fold Cross-Validation

Eine für die Evaluation von Empfehlungssystemen übliche Methode der Validierung ist die *k-fold cross-validation* [11]. *Cross-validation* ist eine Technik zur Validierung von Modellen und wird hauptsächlich für die Evaluation von prädiktiven Verfahren wie den Empfehlungssystemen genutzt. Das Ziel ist die Ermittlung der Vorhersagegenauigkeit eines Systems. In einem Durchlauf der *cross-validation* wird dem System ein Trainigsdatensatz, die Wissensbasis, gegeben, daraufhin werden die vom System generierten Vorhersagen gegen den Testdatensatz, dem System bis dahin unbekannte Daten, geprüft [53]. Für die Durchführung der *k-fold cross-validation* in Empfehlungssysteme, werden zunächst alle Benutzer in k gleichgroße Mengen unterteilt. Dann werden die Daten aus dem Datensatz den Benutzern in den Mengen zugeordnet. So entstehen k Partitionen des ursprünlgichen Datensatzes.
Im Anschluss gibt es k Validierungsdurchläufe. Jede Partition ist einmal der Testdatensatz, die restlichen $k-1$ Partitionen bilden dabei jeweils den Trainingsdatensatz. Nach allen Validierungsdurchläufen wird der Schnitt der Experimentergebnisse berechnet und als Ergebnis ausgegeben. Der Vorteil dieser Methode ist, dass alle Daten sowohl zum Trainieren als auch zum Validieren verwendet werden und jede Information genau einmal zum Validieren verwendet wird [53].

5.3.3 Metriken

Ergebnisse zu Evaluationsmetriken können dem Entwickler Informationen darüber geben, ob ein System gewünschte Eigenschaften aufweist. Die in Empfehlungssystemen am häufigsten gewünschte Eigenschaft ist das korrekte Vorhersagen von Objekten. In einem *Offlineexperiment* lässt sich nicht erkennen, ob ein empfohlenes Objekt, für welches keine Bewertung vom Benutzer im Datensatz vorhanden ist, dem Benutzer gefallen wird. Daher wird häufig so evaluiert, dass die Bewertungen für bereits bekannte Objekte vom System neu berechnet werden und analysiert wird, ob die bereits bekannten Objekte in der Empfehlungsliste enthalten sind [15]. Für die Ermittlung der Genauigkeit (*accuracy*) eines Systems werden oft die Maße *Precision* und *Recall* aus dem Bereich des *Information Retrieval* verwendet sowie ihr harmonisches Mittel, das $F_1 - Measure$ [16].

Wenn dem System die Bewertungen eines Benutzers (also die tatsächlich benutzten Objekte) und die Empfehlungen bekannt sind, gibt es für die Objekte in beiden Mängen vier Zustandsmöglichkeiten, die in Tabelle 5.6 dargestellt sind [15].

Precision ist das Verhältnis zwischen den relevanten Empfehlungen (*true-positives*) und allen empfohlenen Objekten (Formel 5.2). Mit *Precision* wird die Wahrscheinlichkeit repräsentiert, dass ein empfohlenes Objekt relevant ist. *Recall* stellt das Verhältnis zwischen relevanten Empfehlungen (*true-positives*) und allen bekannten

	Empfohlen	Nicht Empfohlen
Benutzt	True-Positive (tp)	False-Negative (fn)
Nicht Benutzt	False-Positive (fp)	True-Negative (tn)

Tabelle 5.6: Klassifizierung möglicher Ausgänge einer Objektempfehlung für einen Benutzer

Objekten (Formel 5.3) dar. Mit *Recall* wird die Wahrscheinlichkeit dargestellt, dass ein bekanntes Objekt empfohlen wird.

Eine der größten Herausforderungen beim Benutzen von *Precision* und *Recall* zum Vergleichen unterschiedlicher Algorithmen besteht darin, dass beide Metriken zusammen betrachtet werden müssen, um die Leistungen vollständig untersuchen zu können. Es ist zu beobachten, dass beide Werte invers zusammenhängend sind. Bei länger werdender Liste der Empfehlungen ist zu erwarten, dass der *Recall-Wert* steigt und der *Precision-Wert* sinkt [15]. Um den Vergleich zwischen Systemen zu vereinfachen, ist es hilfreich eine Metrik einzuführen, die *Precision* und *Recall* in einen Wert vereint. Der häufig dafür verwendete Ansatz ist das $F_1 - Measure$ (Formel 5.4). Das $F_1 - Measure$ stellt ein harmonisches Mittel der beiden anderen Metriken dar. Die Angabe eines F_1 Wertes ist stets auf die Menge der empfohlenen Objekte bezogen, da sich der Wert bei unterschiedlich langen Empfehlungslisten ändern kann.

$$Precision = \frac{\#tp}{\#tp + \#fp} \tag{5.2}$$

$$Recall = \frac{\#tp}{\#tp + \#fn} \tag{5.3}$$

$$F_1 - Measure = 2\frac{Precision\, Recall}{Precision + Recall} \tag{5.4}$$

Abgesehen von der Genauigkeit, können von einem Empfehlungssystem noch viele weitere Eigenschaften gefordert werden. Shani et al. [15] führen diese mit Vorschlägen für Metriken genauer aus. Eine dieser Eigenschaften ist die vom System erzielte Abdeckung (*Coverage*), also wieviele der im System vorkommenden Objekte überhaupt empfohlen werden können. Dafür wird für gewöhnlich einfach das Verhältnis zwischen der Vereinigung aller vom System empfohlenen Objekte und allen im System vorkommenden Objekten angegeben.

Bei der *offline-Analyse* der Vorhersagegenauigkeit eines Empfehlungssystems ist es wichtig, dass beim Validieren lediglich die bekannten Objekte des Testbenutzers beachtet werden, die sich bereits im Trainigsdatensatz befinden. Objekte, die sich nicht im Trainingsdatensatz befinden, können in den für die vorliegende Arbeit implementierten Ansätzen nicht empfohlen werden und würden somit die Leistungswerte der Analysen verfälschen.

5.3.4 Evaluation kontextsensitiver Empfehlungssysteme

Die Besonderheit bei der Evaluation eines kontextsensitiven Systems besteht darin, dass die Empfehlungslisten stets abhängig vom Kontext sind. Um eine Vorhersagegenauigkeitsanalyse durchzuführen, muss also gegen den gefragten Kontext evaluiert werden. Es ergibt sich somit, dass nicht mehr einmal pro Benutzer evaluiert wird, sondern für jeden Benutzer so viele Evaluationen stattfinden wie Kontexte vom System beachtet werden. Am Ende wird ein Schnitt für jeden Kontexttyp berechnet. Wenn mehr als ein Kontexttyp untersucht wird, werden für jeden Kontexttyp die eigenen *Precision*, *Recall* und *F-Measure* Werte festgehalten. Zur Ermittlung der Abdeckung für einen Kontexttyp werden alle Empfehlungen aus allen evaluierten Kontexten vereint und durch die Anzahl aller Objekte geteilt.

5.4 Experimente

Im Rahmen der Leistungsanalysen wurde das Evaluationsframework für drei Versuchsreihen verwendet. Zunächst wurden optimale Parameter für das kollaborative und inhaltsbasierte Filtern ermittelt. Da sich für das kollaborative Filtern ein abweichender optimaler Parameter für die Anzahl der betrachteten ähnlichsten Benutzer ergab, wurde für das Nachftilern mit dem inhaltsbasierten Filtern ebenfalls eine eigene Versuchsreihe gestartet. Es sollte ermittelt werden, wieviele *relevante Attribute* (siehe Kapitel 4.2.4) vom System berücksichtigt werden sollen, um die Ergebnisliste effektiv zu kntextualisieren. Zum Schluss wurden jeweils Versuche für die beiden Ansätze auf den kontextsensitiven Empfehlungsalgorithmen ausgeführt. Die Ergebnisse wurden untereinander und mit den jeweiligen Basisalgorithmen abgeglichen.

Die Leistung der Algorithmen wurde anhand der Maße $F_1 - Measures$ und $Coverage$ gemessen (siehe Kapitel 5.3.3). Um die Leistung der Algorithmen besser analysieren zu können, wurde mit unterschiedlich langen Empfehlungslisten gemessen. Die Längen der Empfehlungslisten liegen zwischen 5 und 50 empfohlenen Artists. Auf diese Weise lässt sich erkennen, wie sich die Leistung der Algorithmen in Abhängigkeit von der Anzahl der besten empfohlenen Artists entwickelt. Es lassen sich also Hinweise darüber finden, wann und wieviele relevante Artists empfohlen werden.

5.4.1 Parameterermittlungen

In vielen Systemen gibt es frei wählbare Parameter, wie die Anzahl der beachteten ähnlichsten Benutzer, die sich auf die Leistung der Algorithmen auswirken können. Im Folgenden wird die Ermittlung dreier optimaler Parameter für die jeweiligen Ansätze vorgestellt.

Kollaboratives Filtern Der Ansatz des kollaborativen Filterns generiert vorhergesagte Bewertungen basierend auf den Bewertungen der N ähnlichsten Benutzer

(siehe Unterkapitel 2.1.1). Ein guter Parameter für diese Variable wird von Herlocker et al. [16] mit der Zahl **50** angegeben. Da die Leistungsergebnisse in der vorliegenden Implementierung des kollaborativen Filterns sehr schwache Ergebnisse brachte, wurde entschieden, den Parameter experimentell zu ermitteln. Dafür wurde das kollaborative Empfehlungssystem jeweils mit $N \in [5, 10, 15, 20, 25]$ ermittelt und mit dem $F_1 - Measure$ ausgewertet. Die Ergebnisse sind in der Abbildung 5.2 einzusehen. Basierend auf den Ergebnissen wurde für das weitere Verwenden des kollaborativen Filterns als Basisalgorithmus für die kontextsensitiven Empfehlungssysteme die Anzahl der beachteten ähnlichsten Benutzer auf **10** gesetzt.

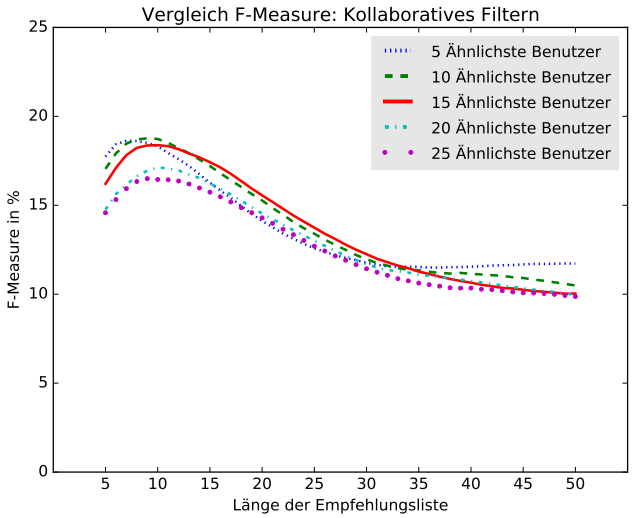


Abbildung 5.2: Leistungsvergleich kollaborativer Filter mit unterschiedlicher Anzahl betrachteter ähnlichster Benutzer.

Inhaltsbasiertes Filtern Die Eigenschaften der Artists im inhaltsbasierten Ansatz der vorliegenden Arbeit setzen sich aus den von *Last.fm* gewonnenen *Socialtags* zusammen (siehe Unterkapitel 5.1.1). Da diese *Socialtags* bei Last.fm frei vergeben werden können, wurde entschieden empirisch zu erheben, wieviele populärste *Socialtags* pro Artist als Attribute für das System zugelassen werden sollen. Dazu wurde für jedes *Socialtag* die Frequenz ermittelt, also wieviele Artists das Socialtag teilen. Danach wurde die Popularität der Tags ermittelt. In den Versuchsreihen wurde evaluiert, wie sich die Leistung des inhaltsbasierten Ansatzes ändert, wenn sich die Anzahl K der betrachteten populärsten Atribute ändert. Dafür wurde das Experiment für jeweils $K \in [10, 25, 50, 100, alle]$ wiederholt und mit dem $F_1 - Measure$ ausgewertet. Die Ergebnisse sind in der Abbildung 5.3 einzusehen. Es gibt kaum Artists deren Anzahl an Attributen die Zahl 100 überschreitet, daher liegen die Graphen für *100* und *alle* übereinander. Basierend auf den Ergebnissen wurde für das weitere Verwenden des inhaltsbasierten Filterns als Basisalgorithmus für die kontextsensiti-

ven Empfehlungssysteme die Anzahl der beachteten populärsten Attribute auf **100** gesetzt.

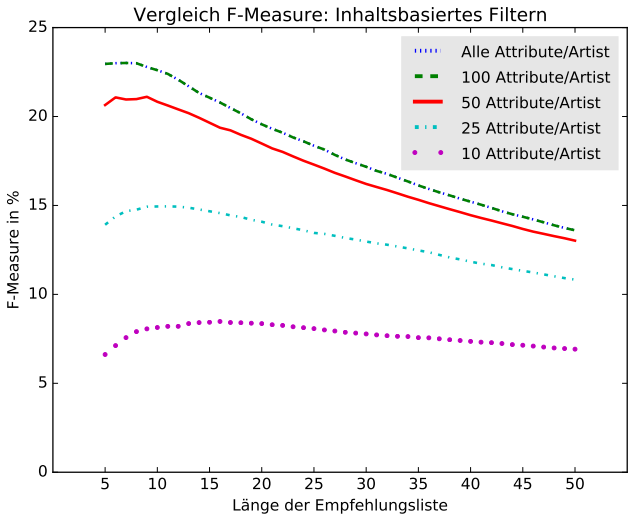


Abbildung 5.3: Leistungsvergleich inhaltsbasierter Filter mit unterschiedlicher Anzahl zugelassener Attribute pro Artist.

Nachfiltern mit inhaltsbasiertem Basisalgorithmus Wie in Kapitel 3.2.2 erläutert, muss für das Nachfiltern festgesetzt werden, wieviele für den gefragten Kontext relevante Eigenschaften ein empfohlenes Objekt mindestens haben muss. Im Falle des inhaltsbasierten Filterns sind dies die zum Artist zugehörigen *Last.fm Tags*. Die Leistung des Algorithmus wurde für unterschiedliche Werte des Parameters K, also die Mindestanzahl der im Kontext relevanten *Tags*, gemessen. Die Validierung wurde dafür jeweils mit $K \in [1, 2, 3, 5, 10, 20]$ durchgeführt und mit dem $F_1 - Measure$ gemessen. Um über die unterschiedlichen im System vorkommenden Kontexttypen ein einheitliches Maß zu erhalten, wurden die $F_1 - Measure$ Werte zunächst für alle vier Kontexttypen ermittelt und im Anschluss für jede Länge der Empfehlungsliste N gemittelt. Die Ergebnisse der Evaluation sind in Tabelle 5.4 einzusehen. Es wurde entschieden K auf den Wert **10** für die Evaluation festzustetzen. Die Entscheidung wird im Kapitel 5.5 Diskussion näher erläutert.

5.4.2 Vergleich: kontextsensitive Empfehlungssysteme - Vorfiltern

Die Experimente für die kontextsensitiven Empfehlungssysteme mit dem Ansatz des Vorfilterns (siehe Kapitel 3.2.1) wurden jeweils auf den beiden Basisalgorithmen mit den zuvor empirisch ermittelten Parametern ausgeführt. Dazu wurden in beiden Fällen den Basisalgorithmen jeweils nur die Teildatensätze zum Trainieren zur

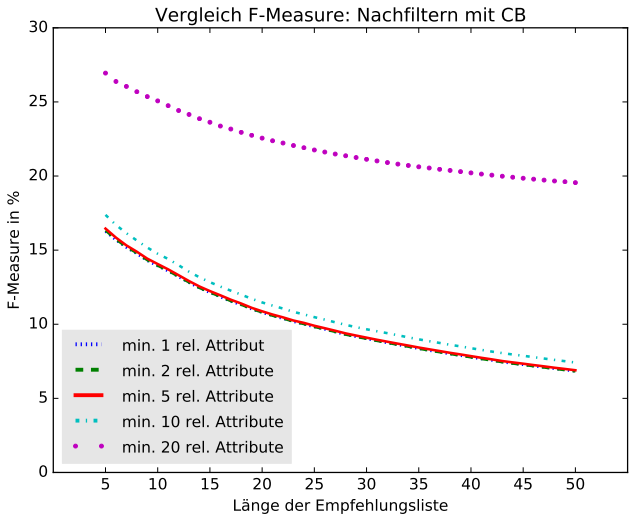
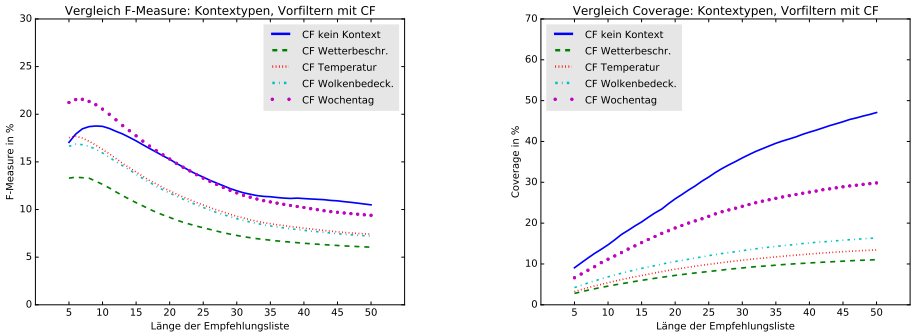


Abbildung 5.4: Leistungsvergleich kontextuelles Nachftilern mit inhaltsbasiertem Filtern, nach Anzahl erforderlicher relevanter Attribute für kontextualisierte Empfehlung.

Verfügung gestellt, dessen Hörereignisse im gewünschten Kontext stattgefunden haben. Beim Ermitteln der $F_1 - Measure$ wurden lediglich die im Kontext bekannten Artists der Benutzer betrachtet, die bereits im kontextualisierten Trainingsdatensatz vorkamen.

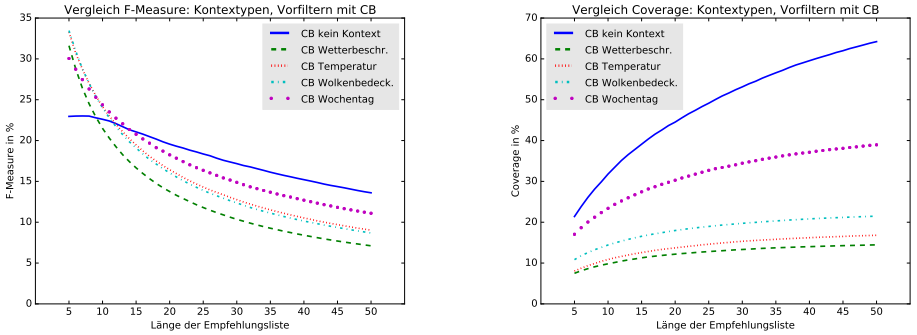
Kollaboratives Filtern In Abbildung 5.5a werden die Ergebnisse der $F_1 - Measure$ Messungen für die jeweiligen im System vorkommenden Kontexttypen für kollaboratives Filtern vorgestellt. Die Abbildung 5.5b zeigt die Ergebnisse der *Coverage* Messungen der einzelnen Kontexttypen. In beiden Abbildungen ist der Basisalgorithmus als Referenz mit angegeben (durchgezogene Linie). Der Begriff *kollaboratives Filtern* ist in beiden Abbildungen mit *CF* abgekürzt.



(a) Vergleich F-Measure (b) Vergleich Coverage

Abbildung 5.5: Vergleich der Leistung des kollaborativen Filterns mit **vorgefilterten** Kontexten

Inhaltsbasiertes Filtern Die Abbildungen 5.6a und 5.6b stellen jeweils die Ergebnisse für die $F_1 - Measure$ und $Coverage$ Messungen für inhaltsbasiertes Filtern dar. In beiden Diagrammen wird der Basisalgorithmus als Referenz (durchgezogene Linie) mit angegeben. Der Begriff *inhaltsbasiertes Filtern* wurde mit den Buchstaben *CB* abgekürzt.



(a) Vergleich F-Measure (b) Vergleich Coverage

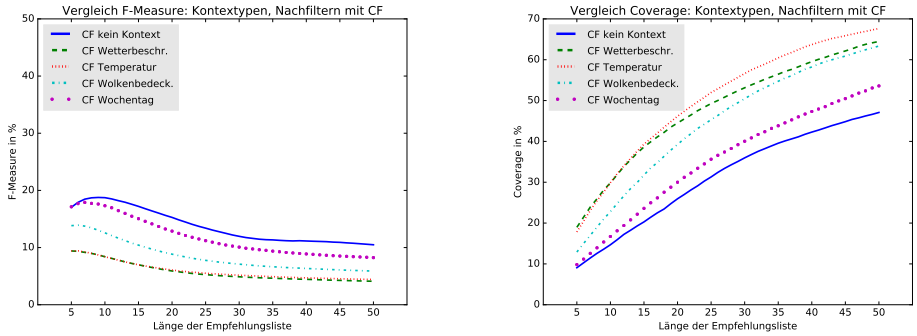
Abbildung 5.6: Vergleich der Leistung des inhaltsbasierten Filterns mit **vorgefilterten** Kontexten

5.4.3 Vergleich: kontextsensitive Empfehlungssysteme - Nachfiltern

Für die Evaluation der kontextsensitiven Empfehlungssysteme mit dem Ansatz des Nachfilterns (siehe Kapitel 3.2.2) wurden zuerst für jeden Benutzer alle vorhergesagten Bewertungen durch die jeweiligen Basisalgorithmen berechnet. Die Menge aller vorhergesagten Bewertunge wurden dann für jeden Kontexttyp und jeden Kontext mit der jeweiligen Methode kontextualisiert, also auf die im Kontext relevanten

Artists reduziert. Die kontextualisierten Artists wurden anschließend wieder absteigend nach der Bewertung sortiert und wie gewohnt für die Listenlängen der N besten Empfehlungen zwischen 5 und 50 validiert.

Kollaboratives Filtern Die Abbildungen 5.7a und 5.7b stellen jeweils die Ergebnisse für die $F_1 - Measure$ und $Coverage$ Messungen dar. In beiden Diagrammen wird der Basisalgorithmus als Referenz (durchgezogene Linie) mit angegeben. Der Begriff *kollaboratives Filtern* wurde mit den Buchstaben *CF* abgekürzt.



(a) Vergleich F-Measure (b) Vergleich Coverage

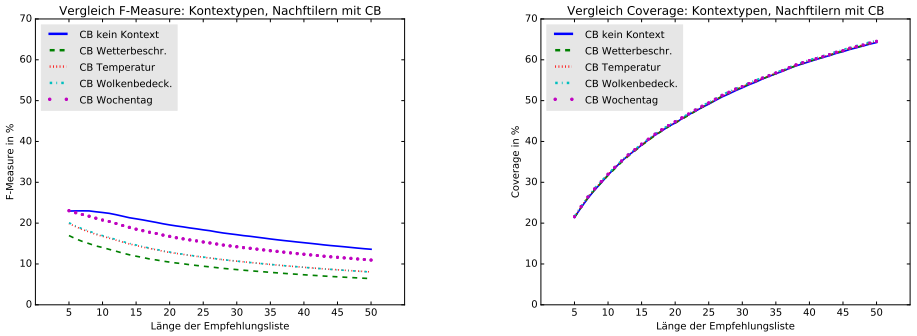
Abbildung 5.7: Vergleich der Leistung des inhaltsbasierten Filterns mit **nachgefilterten** Kontexten

Inhaltsbasiertes Filtern In Abbildung 5.8a werden die Ergebnisse der $F_1 - Measure$ Messungen für die jeweiligen im System vorkommenden Kontexttypen vorgestellt. Die Abbildung 5.8b zeigt die Ergebnisse der $Coverage$ Messungen der einzelnen Kontexttypen. In beiden Abbildungen ist der Basisalgorithmus als Referenz mit angegeben (durchgezogene Linie). Der Begriff *inhaltsbasiertes Filtern* ist in beiden Abbildungen mit *CB* abgekürzt.

5.5 Diskussion

Im Folgenden werden zunächst die Ergebnisse der Versuchsdurchläufe zur Ermittlung der optimalen Parameter für die einzelnen Datensätze näher betrachtet. Im zeiten Teil wird auf die Beantwortung der Fragestellung, also ob die Betrachtung des Wetters als Kontext die Leistung klassischer Empfehlungssysteme verbessern kann, eingegangen.

Es überrascht, dass der optimale Parameter für die Anzahl der betrachteten ähnlichsten Benutzer von 10 für das kollaborative Filtern so weit vom von Herlocker et al. [16] vorgeschlagenen Idealwert von 50 abweicht. Es ist anzunehmen, dass dies durch die Natur des Datensatzes bedingt ist. Die Menge der dem System bekannten



(a) Vergleich F-Measure

(b) Vergleich Coverage

Abbildung 5.8: Vergleich der Leistung des kollaborativen Filterns mit **nachgefilterten** Kontexten

Benutzer ist mit 1694, im Vergleich zur Anzahl der Artists mit 3104, so klein, dass es schwierig ist viele ähnliche Benutzer zu finden da ein Benutzer im Durchschnitt lediglich 9 Artists kennt. Ein Resultat daraus ist, dass relativ unähnliche Benutzer in der Menge der ähnlichsten Benutzer landen können, die wenige gemeinsame Artists mit den anderen ähnlichsten Benutzern haben. Eine denkbare Konsequenz dessen wäre, dass die Vorhersage von Bewertungen für einzelne Artists nur von diesen unähnlichen Benutzern abhängt. Wenn diese Artists nun hohe Bewertungen von den unähnlichen Benutzern erhalten haben, landen sie fälschlicherweise vorne in den Top-Empfehlungen und verdrängen wertvollere Empfehlungen, die zu *true-positives* führen könnten.

Dass die Vorhersagegenauigkeit des inhaltsbasierten Filterns mit wachsender Anzahl der zugelassenen Attribute pro Artist steigt, lässt sich damit erklären, dass das Benutzerprofil am ehesten den Profilen der ihm bekannten Artists entspricht. Das Profil des Benutzers enthält die Attribute, die von den ihm bekannten Artists stammen. Betrachtet man beim Suchen der ähnlichsten Artists auch die seltensten und unpoplärsten Attribute der Artists, steigt die Wahrschienlichkeit, dass durch genau diese Attribute die Artists empfohlen werden, von denen die Attribute stammen.

Trotz der Tendenz, dass die Vorhersagegenauigkeit mit der Anzahl der geforderten für den Kontext relevanten Eigenschaften (siehe Kapitel 3.2.2) im nachgefilterten inaltsbasierten Ansatz steigt, wurde entschieden den Parameter auf 10 zu setzen. Die mit der strengeren Anforderung, der Anzahl der geforderten Eigenschaften, steigende Vorhersagegenauigkeit hängt damit zusammen, dass ähnlich wie beim Festlegen des Parameters für das reine inhaltsbasierte Filtern, die Anforderung an die empfohlenen Artists soweit zugespitzt werden kann, dass hauptsächlich die Artists empfohlen werden, die vom Benutzer bereits vorher gekannt wurden. Es handelt sich bei den empfohlenen Artists bei ausreichend hoher Anforderung um genau die Artists, dessen Attribute als die für den Kontext zu erfüllenden relvanten Attribute gewählt wurden. Adomavicius et al. [14] schlagen vor, den Parameter auf die Zahl 2 festzusetzen. Da

sich die Leistung bis zum Parameter 10 zwar merklich aber im Vergleich zu 20 nicht stark erhöht, wurde 10 als Parameter gewählt. Erst ab diesem Parameter scheint der oben beschriebene Effekt der "Selbstempfehlung" einzusetzen.

Die Frage ob das Einbeziehen des Wetters als Kontextinformation in kontextsensitiven Empfehlungssystemen die klassischen Empfehlungssysteme, die als Basis dienen, in der Leistung übersteigt, muss differenziert betrachtet werden. Insgesamt lässt sich sagen, dass Wetter als Kontext im Vorfilter Ansatz teilweise zu besseren Vorhersagegenauigkeiten führt. Dagegen wirkt sich im Ansatz des Nachfilterns jegliches Bearbeiten der vom Basisalgorithmus empfohlenen Artists negativ auf die Vorhersagegenauigkeit aus. Der Referenzkontext des Wochentages führt bei *allen* Versuchsreihen zu besseren Leistungen als die Betrachtung der einzelnen Wetter-Kontexttypen. Beim Ansatz des Vorfilterns ergibt sich, dass sich die Vorhersagegenauigkeit der Algorithmen größtenteils bei kurzen Empfehlungslisten verbessert. Mit dem kollaborativen Filtern als Basisalgorithmus sind die Leistungen der Algorithmen, welche die Wetterkontexte betrachten, durchgehend schlechter als die des Basisalgorithmus. Der einzige Kontext, der hierbei den Basisalgorithmus verbessert, sind die Wochentage. Das inhaltsbasierte Filtern lässt sich für kurze Empfehlungslisten mit allen Kontexttypen verbessern. Die Vorhersagegenauigkeit sinkt jedoch bei länger werdender Empfehlungsliste unter die Leistung des Basisalgorithmus. Die im Kontext als relevant erkannten Artists bekommen demnach sehr hohe Bewertungen und tendieren dazu unter den besten empfohlenen Artists zu sein. Artists, die im Kontext vorkommen, aber geringere initiale Bewertungen vom Benutzer bekommen haben, erhalten dementsprechend auch niedrigere vorhergesagte Bewertungen und werden erst sehr spät oder gar nicht empfohlen. Ein Nachteil des Vorfilter Ansatzes ist, dass das System eine geringere Abdeckung (*coverage*) erreicht (siehe Abbildungen 5.5b und 5.6a). Der Abdeckungswert korreliert mit den Größen der für das Trainig verwendeten Datensätze. Je stärker der Datensatz partitioniert wurde, desto geringer ist die Coverage, da jeweils nur Teilmengen aus den reduzierten Datensätzen empfohlen werden können. Dies deutet darauf hin, dass es trotz der reduzierten Datensätze dazu kommt, dass es ein Popularitätsgefälle zwischen den Artists gibt.

Im Nachfilter Ansatz sind die Vorhersagegenauigkeiten aller kontextsensitiven Algorithmen durchgehend unter den Leistungen der beiden Basisalgorithmen. Dagegen sind die Werte der Abdeckungsmessungen durchgehend höher. Dies könnte darin begründet sein, dass durch die Entfernung hoch bewerteter Artists sonst vom System nicht empfohlene Artists empfohlen werden können und es zu einer höheren *Diversity* kommt, also dass auch Objekte aus dem *long-tail* empfohlen werden.

Abschließend lässt sich sagen, dass der Wetterkontext allein kaum zu einer Verbesserung der Vorhersagegenauigkeit führt. Die beobachteten Effekte könnten allerdings durch die Natur des - trotz der Bereinigung immer noch sehr *sparsen* - Datensatzes begründet sein. Zum einen kennen die Benutzer im Durchschnitt nur wenige Artists und zum anderen sind die betrachteten Kontextinformationen für Wetter im Datensatz nicht gut verteilt.

Kapitel 6

Zusammenfassung

Die vorliegende Arbeit gibt einen Einstieg und Überblick in den Forschungsbereich der Empfehlungssysteme. Klassische Empfehlungssysteme wurden in Funktionsweise, Stärken und Schwächen unterschieden. Hierbei wurde insbesondere auf die Ansätze des kollaborativen und des inhaltsbasierten Filterns eingegangen. Kontext kann als weiterer Faktor für Empfehlungssysteme betrachtet werden. Es wird darauf eingegangen, wie Kontext für Empfehlungssysteme definiert, gewonnen und formalisiert werden kann. Die kontexteinbindenden Ansätze für Empfehlungssysteme wurden unterschieden und vorgestellt. Im Rahmen der vorliegenden Arbeit wurde ein Konzept ertstellt, in dem zwei klassische Empfehlungsalgorithmen mit den kontextsenstiven Methoden des Vorfilterns und Nachftilerns erweitert wurden. Designentscheidungen und die Art der Kontexte wurden näher geschildert. Das Konzept soll die Untersuchung des Einflusses von Kontexinformationen auf die Vorhersagegenauigkeit von klassichen Empfehlungssystemen ermöglichen. Als zu untersuchende Kontexte wurden unterschiedliche Aspekte des Wetters und die Wochentage ausgewählt. Für die Evaluation wurde ein natürlicher Datensatz mit Musikhörereignissen bereinigt und mit Kontextinformationen angereichert. Des Weiteren wurde eine Einführung in die Evaluation von Empfehlungssystemen dargestellt, insbesondere wird auf die für die Experimente verwendeten Methoden und Messverfahren eingegangen. In einer Reihe von Versuchen wurden zum einen optimale Parameter für die Empfehlungssysteme gesucht. Zum anderen wurde der Effekt der Einbeziehung des Wetters auf die Vorhersagegenauigkeit klassischer Empfehlungssysteme untersucht. Die Auswertung der Experimente hat ergeben, dass die Betrachtung kompletter Kontexttypen nur mit dem Ansatz des Vorfilterns zur Leistungssteigerung der klassischen Empfehlungssysteme führt. Die Kehrseite dieses Ansatzes ist die Einschränkung des von Empfehlungssystemen abgedeckten Raums der Objekte (*Coverage*). Alle Versuchsreihen ergeben, dass die Einbeziehung der Wochentage als Kontext durchgehend zu besseren Vorhersagen der Empfehlungssysteme führt als die Einbeziehung des Wetters.

6.1 Ausblick

Die Evaluation des vorgestellten Konzepts hat gezeigt, dass es Hinweise für die Steigerung der Vorhersagegenauigkeit unter Verwendung des Wetters als Kontextinformation gibt. Es bleibt zu untersuchen, ob die von Adomavicius et al. [14]

vorgeschlagene Methode der Kontextgeneralisierung zur vollständigen, also über alle Längen der Empfehlungslisten, Steigerung der Vorhersagegenauigkeit führt. Zu diesem Zweck müssten die Kontexte einzeln untersucht werden und nur diese Kontexte beim Empfehlungsvorgang betrachtet werden, die sich als relevant erwiesen haben. So ließe sich ein Basisalgorithmus kontextuell optimieren. Da der Kontext des Wetters ein interessanter Kontext zu sein scheint, würde es sich lohnen, die Wetterinformationen direkt in den Empfehlungsvorgang einfließen zu lassen, also ein Verfahren für das kontextuelle Modellieren zu finden. Darüber hinaus ließen sich die unterschiedlichen kontextsensitiven Ansätze gleichzeitig verwenden und ein Hybridsystem erzeugen, das die Stärken der einzelnen Ansätze verbindet. Es wäre außerdem interessant, in einer *Onlineevaluation* zu ermitteln, ob die *false-positives*, also die empfohlenen aber dem Benutzer unbekannten Artists, auf eine positive Resonanz stoßen würden. Die rapide abnehmende Vorhersagegenauigkeit der vorfilternden kontextsensitiven Empfehlungssysteme lässt annehmen, dass sich vor allem unter den Top-Empfehlungen ähnliche Artists befinden. Um eine hochwertigere *Offlineevaluation* durchführen zu können, wäre ein größerer, aber vor allem dichterer Datensatz erforderlich. Zusätzlich wären genauere Informationen bezüglich der Wohnorte der Benutzer interessant, da so kulturelle Einflüsse auf die Benutzer zusätzlich beachtet werden könnten.

Es wurde gezeigt, dass die Anreicherung eines Datensatzes mit Wetterinformationen möglich ist, sobald ein *Zeitstempel* und ein *Standort* zur Verfügung stehen. Es könnten Auswirkungen des Wetters auf die Empfehlungsqualität von Systemen in weiteren Domänen untersucht werden. Beispielhafte naheliegende Domänen in der die Wirkung des Wetters als Kontext untersucht werden könnte, sind Freizeitaktivitäten oder Reisen. Weiter ließe sich eine Wetterprognose in Empfehlungssystemen verwenden, wenn das Wetter mögliche Empfehlungen eindeutig einschränkt. Ein Beispiel dafür wäre ein Kinofilmempfehlungssystem, welches das Programm eines in der Nähe liegenden Freiluftkinos nur dann berücksichtigt, wenn das Wetter gut ist.

Literaturverzeichnis

[1] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, pages 81–97, 1956.

[2] K. G. Coffman and A. M. Odlyzko. The size and growth rate of the internet. *First Monday*, 3, 1998.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[4] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *ARTIFICIAL INTELLIGENCE REVIEW*, 13:393–408, 1999.

[5] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *COMMUNICATIONS OF THE ACM*, 40(3):77–87, 1997.

[6] Ken Goldberg, Theresa Roeder, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4:133–151, 2001.

[7] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[8] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, January 2009.

[9] Michael Pazzani, Daniel Billsus, S. Michalski, and Janusz Wnek. Learning and revising user profiles: The identification of interesting web sites. In *Machine Learning*, pages 313–331, 1997.

[10] Ken Lang. Newsweeder: Learning to filter netnews. In Armand Prieditis and Stuart J. Russell, editors, *ICML*, pages 331–339. Morgan Kaufmann, 1995.

[11] J. Bobadilla, F. Ortega, A. Hernando, and A. GutiéRrez. Recommender systems survey. *Know.-Based Syst.*, 46:109–132, July 2013.

[12] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[13] J. Payne, J. Bettman, and J. Johnson. The adaptive decision maker. *Cambridge University Press*, 1993.

[14] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.

[15] Guy Shani and Asela Gunawardana. Evaluating recommender systems. Technical Report MSR-TR-2009-159, November 2009.

[16] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM TRANSACTIONS ON INFORMATION SYSTEMS*, 22:5–53, 2004.

[17] N. Tintarev and J. Masthoff. *Designing and Evaluating Explanations for Recommender Systems*, page 479. 2011.

[18] Seok Kee Lee, Yoon Ho Cho, and Soung Hie Kim. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences*, 180(11):2142 – 2155, 2010.

[19] Edward Rolando Núñez-Valdéz, Juan Manuel Cueva Lovelle, Oscar Sanjuán Martínez, Vicente García-Díaz, Patricia Ordoñez de Pablos, and Carlos Enrique Montenegro Marín. Implicit feedback techniques on recommender systems applied to electronic books. *Computers in Human Behavior*, 28(4):1186 – 1193, 2012.

[20] Douglas Oard and Jinmook Kim. Implicit feedback for recommender systems. In *in Proceedings of the AAAI Workshop on Recommender Systems*, pages 81–83, 1998.

[21] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, November 2002.

[22] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[23] David Hauger, Markus Schedl, Andrej Kosir, and Marko Tkalcic. The million musical tweet dataset - what we can learn from microblogs. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, November 4-8 2013. `http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/85_Paper.pdf`.

[24] Friedemann Mattern and Christian Floerkemeier. *From the Internet of Computers to the Internet of Things*, volume 6462 of *LNCS*, pages 242–259. Springer, 2010.

[25] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11):10059 – 10072, 2012.

[26] Pinata Winoto and Tiffany Y. Tang. The role of user mood in movie recommendations. *Expert Syst. Appl.*, 37(8):6086–6092, August 2010.

[27] Shulong Tan, Jiajun Bu, Chun Chen, Bin Xu, Can Wang, and Xiaofei He. Using rich social media information for music recommendation via hypergraph model.

[28] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithm for collaborative filtering. In *Proceedings of the 14 th Conference on Uncertainty in Artificial Intelligence*, pages 43–52, 1998.

[29] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[30] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems*, 22:143–177, 2004.

[31] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[32] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[33] Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40:66–72, 1997.

[34] Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[35] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, December 1992.

[36] J. J. Rocchio. *Relevance Feedback in Information Retrieval*. Prentice Hall, Englewood, Cliffs, New Jersey, 1971.

[37] Raymond J. Mooney, Paul N. Bennett, and Loriene Roy. Book recommending using text categorization with extracted information. In *IN RECOMMENDER SYSTEMS. PAPERS FROM 1998 WORKSHOP*, pages 49–54. AAAI Press, 1998.

[38] Yoon J. Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 11–18, New York, NY, USA, 2008. ACM.

[39] Eli Pariser. *The filter bubble: what the Internet is hiding from you.* Penguin Press, 2011.

[40] Jian-Guo Liu, Kerui Shi, and Qiang Guo. Solving the accuracy-diversity dilemma via directed random walks. *CoRR*, abs/1201.6278, 2012.

[41] Robin Burke. The adaptive web. chapter Hybrid Web Recommender Systems, pages 377–408. Springer-Verlag, Berlin, Heidelberg, 2007.

[42] P. Herrera, Z. Resa, and M. Sordo. Rocking around the clock eight days a week: an exploration of temporal patterns of music listening. In *1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*, Barcelona, 26/09/2010 2010.

[43] N. Webster. Webster's new twentieth century dictionary of the english language. Technical report, 1980.

[44] Mary Bazire and Patrick Brézillon. Understanding context before using it. In *In Modeling and Using Context*, 2005.

[45] Paul Dourish. What we talk about when we talk about context. *Personal Ubiquitous Computing*, 8(1):19–30, 2004.

[46] Pedro G. Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, February 2014.

[47] Dongjoo Lee and Sung Eun Park. Exploiting contextual information from event logs for personalized recommendation. In *In Computer and Information Science, Studies in Computational Intelligence*, pages 121–139. Springer, 2010.

[48] Marius Kaminskas and Francesco Ricci. Contextual music information retrieval and recommendation: State of the art and challenges. *Computer Science Review*, 6(2-3):89–119, 2012.

[49] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *In IEEE International Conference on Data Mining (ICDM 2008*, pages 263–272, 2008.

[50] Pasquale Lops, Marco Gemmis, and Giovanni Semeraro. *Recommender Systems Handbook*, chapter Content-based Recommender Systems: State of the Art and Trends, pages 73–105. Springer US, Boston, MA, 2011.

[51] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. #nowplaying music dataset: Extracting listening behavior from twitter. In *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*, WISMM '14, pages 21–26, New York, NY, USA, 2014. ACM.

[52] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[53] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *JOURNAL OF MACHINE LEARNING RESEARCH*, 5:1089–1105, 2003.