Kollaborative Empfehlungssysteme im E-Commerce

Entwicklung eines Prototyps und Vergleich mit State-of-the-art Verfahren

Studien- und Masterarbeit

Mahyar Davari (1730531)

TECHNISCHE UNIVERSITÄT DARMSTADT

ise. Information Systems and Electronic Services

Knowledge Engineering

Technische Universität Darmstadt

Fachbereich Informatik

Fachgebiet Knowledge Engineering Group

Prof. Dr. Johannes Fürnkranz

Fachbereich Rechts- und Wirtschaftswissenschaften

Fachgebiet Wirtschaftsinformatik - Information Systems & Electronic Services

Prof. Dr. Alexander Benlian

Studien- und Masterarbeit zu dem Thema:

Empfehlungssysteme im E-Commerce

Entwicklung eines Prototyps und Vergleich mit State-of-the-art Verfahren

Bearbeitet von: Mahyar Davari

Matr.-Nr.: 1730531

Studiengang: M.Sc. Wirtschaftsinformatik

Eingereicht am: 30.10.2016

Förmliche Erklärung

Hiermit erkläre ich, Mahyar Davari, geboren am 09.08.1989, an Eides statt, dass ich die vorliegende Studien- und Masterarbeit ohne fremde Hilfe und nur unter Verwendung der zulässigen Mittel sowie der angegebenen Literatur angefertigt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Darmstadt, den 30.10.2016

_____

(Unterschrift)

Zusammenfassung

Empfehlungssysteme nahmen in den letzten Jahren sowohl in der Forschung als auch in der Industrie eine immer wichtiger werdende Rolle ein. Diese Arbeit beschäftigt sich mit der Funktionsweise von State-of-the-art *Recommender Systems* sowie dem Vergleich dieser mit einem von mir entwickeltem Verfahren. Kollaborative Empfehlungsalgorithmen, die auf historischen Bewertungsdaten beruhen, sind die am weitesten verbreiteten Techniken und stehen deswegen im Fokus meiner Untersuchungen. Nach einem Überblick zu der Literatur und der Vorstellung des neuen Verfahrens soll in experimentellen Untersuchungen der Frage nachgegangen werden, welche Faktoren kritisch sind für Empfehlungssysteme im E-Commerce. Der Schwerpunkt liegt dabei auf den betrachteten Verfahren und der Rangfolge bezüglich der Genauigkeit ihrer Empfehlungslisten. Weitere Faktoren stellen u.a. verschiedene Evaluationsstrategien, die Länge der Trainingshistorie sowie der Einsatz weiterer Datensätze neben dem Kaufdatensatz dar.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ALS	Alternating Least Squares
ARM	Association Rule Mining
BPRMF	Bayesian Personalized Ranking Matrix Factorization
BPR-OPT	Bayesian Personalized Ranking Optimierungskriterium
CC	Catalog Coverage
CF-Tree	Clustering-Feature-Tree
CTREE	Conditional Inference Tree
GSS	Golden Section Search
KF	Kollaborative Filter
k-NN	k nächste Nachbarn
MF	Matrixfaktorisierung
NAVSD	Neighborhood based Added Value Supervized Discretization
NBO	Next Best Offer
SGD	Statistical Gradient Descent
SVD	Singular Value Decomposition
TF-IDF	term frequency/inverse document frequency
WRMF	Weighted Regularized Matrix Factorization

1 Einleitung

Heutzutage stehen Kunden diverser Online-Shops einer Vielzahl von Produkten gegenüber. Herauszufinden, welcher Artikel am besten ihre Bedürfnisse und Wünsche befriedigt, stellt eine nahezu unmögliche Aufgabe dar, da sie nur einen kleinen Teil des gesamten Marktes untersuchen können. Daher versuchen viele Unternehmen auf ihren Seiten jene Produkte zu präsentieren, für die sie die Kaufwahrscheinlichkeit als hoch erachten. Um solche Wahrscheinlichkeiten zu berechnen oder zu ermitteln, inwieweit ein Artikel und ein Nutzer zueinander passen, wurden Empfehlungssysteme entwickelt.

Ein intelligentes Empfehlungssystem stellt ein leistungsfähiges Verkaufswerkzeug für jede E-Commerce-Plattform dar. Bereits 2006 sollen 35% der Umsätze von Amazon[1] durch Produktempfehlungen generiert worden sein [81]. Netflix[2] veranstaltete zur Verbesserung ihres Empfehlungssystems im selben Jahr einen Wettbewerb mit 1 Millionen Dollar Preisgeld, was viele Forscher dazu bewegte, sich mit diesem Thema zu befassen und neue Empfehlungsalgorithmen zu entwickeln. Der Siegeralgorithmus wurde jedoch niemals in der Praxis eingesetzt. Netflix begründete seine Entscheidung damit, dass der Gewinn an Genauigkeit nicht den technischen Aufwand rechtfertige, um ihn in eine Produktionsumgebung zu integrieren [6]. Die vorliegende Arbeit thematisiert diese selten erforschte Verbindung zwischen wissenschaftlichen Studien und industriellen Lösungen. Sie orientiert sich an der Studie von Pradel et al. [94], in der die Prognosegenauigkeit verschiedener Empfehlungsalgorithmen auf Basis von Kaufdaten eines Möbelhauses evaluiert wurde.

Die meisten Beiträge in diesem Forschungsgebiet beschäftigen sich mit Verfahren in Empfehlungssystemen. Häufig werden dabei *explizite Bewertungen* verwendet. Sie repräsentieren zwar exakte Kundenpräferenzen, haben aber den Nachteil, dass sie von vielen Kunden nicht angegeben werden. Im E-Commerce hingegen bestehen typische Datensätze aus historischen Kundentransaktionen, sodass hier der Fokus auf die Prognose von Kaufwahrscheinlichkeiten anstatt von Bewertungspunkten liegt. Studien zu solchen *spärlichen impliziten Bewertungen* wurden jedoch u.a. aufgrund des Mangels an öffentlich verfügbaren E-Commerce-Datensätzen selten durchgeführt. Pradel et al. [94] merkten bereits an:

> „*case-studies are necessary to better understand the specificities of purchase datasets and the factors that impact recommender systems for retailers.*“

Ziel dieser Arbeit ist es, mithilfe solcher Daten zu ermitteln, welche kritischen Faktoren für die Genauigkeit und Effizienz von Empfehlungssystemen existieren. Der Fokus liegt hierbei auf dem Faktor des zugrunde liegenden Verfahrens. Auf Basis verschiedener Datensätze, die mir von zwei E-Commerce-Händlern zur Verfügung gestellt wurden, vergleiche ich ausgewählte Algorithmen bezüglich ihrer Empfehlungsqualität miteinander. Dazu werden neben State-of-

[1] http://www.amazon.com.

[2] http://www.netflix.com.

the-art Methoden der von mir entwickelte *Neighborhood-based Added Value Supervized Discretization* Algorithmus untersucht. Die Analyse beschränkt sich aufgrund der verfügbaren Daten auf Verfahren der *kollaborativen Filter*. Zudem wird die Studie von Pradel et al. [94] erweitert und u.a. daraufhin untersucht, welche weiteren Datensätze neben den Kaufdaten effizient in ein solches System integriert werden können.

Die Arbeit gliedert sich in vier Teile, die jeweils einzeln betrachtet ein Themenschwerpunkt in Studien zu Empfehlungssystemen darstellen. Zunächst wird im zweiten Kapitel ihre Rolle im E-Commerce beleuchtet. Dabei verzichte ich auf methodische Gesichtspunkte und zeige insb. auf, wie und mit welchem Stellenwert sie für beteiligte Parteien, also Unternehmen und Nutzer, agieren. Kapitel 3 stellt eine Literaturübersicht von State-of-the-art Empfehlungssystemen dar. Dort werden die Funktionsweisen sowie Vor- und Nachteile der verschiedenen Arten von Systemen behandelt. In Kapitel 4 wird das bereits erwähnte NAVSD-Verfahren beschrieben. Hierbei stehen Ablauf der Methode sowie Unterschiede zu State-of-the-art Verfahren im Vordergrund. Schließlich werden in Kapitel 5 dieser Algorithmus sowie vier ausgewählte Methoden experimentell untersucht. Die Ergebnisse dieses Vergleichs sowie die Auswirkungen weiterer kritischer Faktoren werden in diesem Kapitel präsentiert und diskutiert. Ein Fazit und ein kurzer Ausblick auf offene Fragestellungen beschließen die Arbeit.

2 Empfehlungssysteme im E-Commerce

Empfehlungssysteme waren bereits ein wesentlicher Baustein erster E-Commerce Unternehmen, wie Amazon und CDNOW. Es bestand schon früh ein starkes Bedürfnis danach, Menschen durch Empfehlungen und Kundenmeinungen dabei zu unterstützen, die richtigen Produkte zu finden [20],[70],[114]. Seit ihren Anfängen hat Amazon eine Vielfalt von Empfehlungsfeatures entwickelt, die von algorithmischen Ansätzen, wie kollaborativen Filtern, bis hin zu nicht-algorithmischen Ansätzen, wie nicht-personalisierten Empfehlungen, reicht [114]. Auf der einen Seite wird durch die Nutzung eines Empfehlungssystems *kundenindividuelle Massenproduktion* (engl.: Mass Customization) dadurch angestrebt, dass jeder Nutzer auf ihn zugeschnittene Leistungen erhält. Dazu speichert Amazon sorgfältig die Transaktionen ihrer Kunden. Bezüglich der Metriken Klickrate und Konversationsrate zeigten personalisierte Empfehlungen bessere Ergebnisse als nicht-personalisierte [78]. Zudem wird Personalisierung als ein Mittel gesehen, um langfristige Beziehungen zu Kunden aufzubauen [114].
Auch heute haben Empfehlungssysteme eine hohe Bedeutung im E-Commerce, die zudem weiterhin wächst. Eine Studie des E-Commerce-Center Handels [33] ergab 2011 bei einer Befragung von 202 Online-Händlern, dass Produktempfehlungen die am häufigsten verwendete verkaufsfördernde Maßnahme darstellen. Zudem sollen mindestens drei Viertel der befragten Shop-Betreiber in Betracht ziehen, ein Empfehlungssystem anzuschaffen. Auch in der Forschung nimmt dieses Thema eine wesentliche Rolle ein. Nach der Studie von Xu et al. [131] gehört der E-Commerce- nach dem Entertainment-Bereich zum am häufigsten erforschten Anwendungsfeld von Empfehlungssystemen. Im weiteren Verlauf dieses Kapitels wird dieses Forschungsgebiet näher beleuchtet, indem insb. die Stellung von Empfehlungssystemen für Nutzer und E-Commerce erläutert wird.

2.1 Empfehlungssystem als Informationssystem

Empfehlungssysteme stellen Informationssysteme dar, die insofern in *Decision Support Systeme* kategorisiert werden können, als dass sie zum einen bei der Entscheidungsfindung verwendet werden und zum anderen dafür bestimmt sind, andere Menschen zu unterstützen, statt zu ersetzen. Gleichzeitig weisen Xiao und Benbasat [130] darauf hin, dass sich Empfehlungssysteme in bestimmten Punkten von traditionellen Decision Support Systeme unterscheiden. Letztere sind an Manager und Analysten gerichtet, die solche Systeme zur Unterstützung verschiedener Aufgaben, wie bspw. Planungsaufgaben, verwenden. Die Nutzer von Empfehlungssystemen hingegen kommen aus allen sozialen Schichten und stehen dem *Vorzugswahlproblem* (engl.: preferential choice problem) gegenüber. Während also Decision Support Systeme Prozessmodelle verwenden, bilden Empfehlungssysteme Wahlmodelle, welche die Integration von Entscheidungskriterien für die Wahl zwischen Alternativen unterstützen.
Letztere teilen zudem ähnliche Eigenschaften mit *wissensbasierten Informationssystemen*. Sie sollten ebenfalls ihren Nutzern Schlussfolgerungen oder Empfehlungen erklären, um Vertrau-

en bei diesen zu schaffen. Zwischen dem Empfehlungssystem und seinen Nutzern herrscht jedoch ein Prinzipal-Agent-Verhältnis: Der Nutzer (Prinzipal) kann sich nicht sicher sein, ob das Empfehlungssystem ausschließlich zu seinem Vorteil oder zu dem des Händlers arbeitet [130].

Des Weiteren verweist Burke [20],[21] darauf, dass Empfehlungssysteme keine *Informationsrückgewinnungssysteme* (engl.: *Information Retrieval Systems*) sind. Die einen geben für den Kunden interessante Produktempfehlungen aus, während die anderen jene Artikel ausgeben, die mit der Suche des Nutzers am besten übereinstimmen. Solche Suchmaschinen präsentieren alle passenden Produkte sortiert nach dem Maß der Übereinstimmung [21]. Hangartner [51] stellt zudem fest, dass Suchmaschinen uns dabei helfen, etwas zu finden, was wir bereits kennen, während Empfehlungssysteme uns eher dabei unterstützen, neue Produkte zu finden.

2.2 Nutzen von Empfehlungssystemen

Sowohl Nutzer als auch Anbieter profitieren von Empfehlungssystemen. Während zu Gunsten der Nutzer Transaktionskosten gesenkt und die Entscheidungsqualität verbessert wird, erhöhen solche Systeme die Umsätze für E-Commerce-Anbieter. Dieser Abschnitt geht im Detail auf die einzelnen Vorteile für beide Parteien ein.

2.2.1 Vorteile für die Nutzer

Eine wesentliche Funktion von Empfehlungssystemen besteht darin, Nutzer im Entscheidungsfindungsprozess zu unterstützen. Sie geben ihren Kunden Vertrauen (in der Literatur: *Confidence*) in den Kauf oder in eine sonstige Handlung eines Produktes. In diesem Sinne stellt ein Empfehlungssystem ein *persuasives System* dar, welches eine Überzeugungsfunktion im Entscheidungsprozess einnimmt [28].

Senkung der Transaktionskosten

Im Kontext der Transaktionskostentheorie dienen Empfehlungssysteme dem Zweck, die Transaktionskosten zu reduzieren. Solche Kosten entstehen aus dem Grund, dass die Entscheidungsfindung einer limitierten kognitiven Fähigkeit unterworfen ist, sodass nicht alle möglichen Alternativen berücksichtigt werden können [61]. Die Ursache für Transaktionskosten ist *Unsicherheit*. Diese kann während des Transaktionsprozesses in drei Formen auftreten: bezüglich des Produktes, des Prozesses und/oder der Psyche. Insgesamt betrachtet bezieht sich Unsicherheit auf jene Kosten, die aufgrund der Informationsasymmetrie mit unerwarteten Ergebnissen verbunden sind [61].

Empfehlungssysteme sind in der Lage die Transaktionskosten zu senken, indem sie zum einen durch personalisierte Empfehlungen die Anzahl der zu durchsuchenden Produkte senken, und zum anderen Unsicherheiten bezüglich des Produktes bspw. durch Kundenbewertungen kleiner werden lassen. Sie stellen ein Mittel zur Erhöhung der Entscheidungsqualität dar [130].

Personalisierung

Personalisierung ist eine wesentliche Komponente von Empfehlungssystemen. Pommertanz et al. [93] definierten bereits Empfehlungssysteme als „*tools that provide personalized recommendations to people*“. Anstatt zwischen personalisierten und nicht-personalisierten Empfehlungen zu unterscheiden, führen Schafer et al. [114] verschiedene Ausmaße von Personalisierung auf:

1. **Generisch:** Jedem Kunden werden dieselben Produkte präsentiert. Diese stellen vollkommen nicht-personalisierte Empfehlungen dar.

2. **Demografisch:** Alle Mitglieder innerhalb einer Zielgruppe erhalten dieselben Empfehlungen.

3. **Vorübergehend:** Empfehlungen basieren auf aktuellen Kundenaktivitäten. Sie erscheinen in der Kundennavigation oder auf Produktinformationsseiten. Als Beispiel lassen sich Amazons „*Kunden, die …*“ Empfehlungen aufführen.

4. **Anhaltend:** Empfehlungen basieren auf langfristigen Kundeninteressen und benötigen von den Kunden gleichbleibende Identitäten.

Die Aufzählung ist aufsteigend nach dem Level der Personalisierung sortiert. Personalisierter muss jedoch nicht besser bedeuten. In der Tat kommen einfache, nicht-personalisierte Empfehlungen gut bei den Nutzern an [28]. Svensson et al. [124] gehen davon aus, dass in solchen Fällen, in denen ein großer Teil des Produktbestandes den Kunden gefällt und von ähnlicher Qualität ist, nicht-personalisierte Empfehlungen in Form von Bestsellerlisten gute Ergebnisse liefern können. Zudem sei Personalisierung nur in geringem Ausmaß notwendig. Auch laut Recker et al. [95] bringen personalisierte Empfehlungen einen geringeren Mehrwert mit sich, wenn Kunden im Allgemeinen ähnliche Ansichten bezüglich der Qualität der Produkte haben.

Des Weiteren ist Personalisierung sehr datengesteuert. Ob und in welchem Ausmaß Empfehlungssysteme personalisierte Empfehlungen ausgeben können, ist abhängig von den Kunden- oder Produktmodellen. Bevor das Modell nicht gebildet wurde, kann keine personalisierte Empfehlung generiert werden. Weiterhin spielt die Informationsdichte der Bewertungen eine wesentliche Rolle [29]. *Datenspärlichkeit* stellt einen Zustand dar, bei dem die Bewertungsschnittmenge zwischen Nutzern klein ist und es schwer ist, miteinander korrelierende Nutzer zu finden. Gleiches kann auch für Produkte gelten. Bevor nicht genügend Daten verfügbar sind, greifen viele kommerzielle Systeme auf weniger personalisierte Empfehlungen zurück, wie z.B. kontextbezogene, demographische und Bestseller-Ansätze [70].

Forscher empfehlen jedoch auch einige Bereiche, in denen Personalisierung eingebracht werden sollte. Knijnenburg et al. [67] sind der Ansicht, dass Empfehlungssysteme mit ihren Kunden interagieren sollten, um von verschiedenen Arten von Empfehlungen für verschiedene Typen von Nutzern zu profitieren. Harper et al. [52] haben die Auffassung, dass neben perso-

nalisierten Inhalten auch personalisierte Oberflächen berücksichtigt werden sollten. Schafer et al. [112] glauben, dass Personalisierung auf eine Gruppe von Nutzern angewandt werden sollte. Eine durchaus richtungweisende Idee ist auf Hangartner [51] zurückzuführen. Um die Bedürfnisse und Interessen eines Individuums besser zu verstehen, könne sein soziales Netzwerk als Information genutzt werden. Man solle von dem Gedanken wegkommen, „*delivering an isolating, customized experience to a person*“ und sich darauf fokussieren, „*connecting an individual with affinity communities who can provide information of value to that individual*“.

2.2.2 Vorteile der Anbieter

Empfehlungssysteme sind für E-Commerce Unternehmen ein effektives Mittel, um Umsätze zu steigern. Sie beeinflussen sowohl den *Liftfaktor* (Erhöhung des Umsatzvolumens infolge der Einführung eines Empfehlungssystems) als auch die *Konversationsrate* (Verhältnis zwischen der Anzahl der Verkäufe aufgrund der Empfehlungen und der Anzahl der Empfehlungen). Aufgrund der großen Anzahl an Produkten sind Empfehlungssysteme unverzichtbar für Online Shops. Durch sie werden Käufe diversifiziert und der *Long-Tail* Trend im E-Commerce somit weiter verstärkt [28].
Schafer et al. [114] zählen drei Wege auf, wie Umsätze im E-Commerce gesteigert werden:

1. **Umwandlung vom *Surfer* zum Käufer:** Empfehlungssysteme versuchen suchende Nutzer durch Empfehlungen interessanter Produkte zum Kaufen zu bewegen. Zudem schaffen sie dafür notwendiges Vertrauen, indem sie ihnen Signale guter Qualität liefern [114]. Ist das Verhältnis des Nutzers zum Empfehlungssystem von Vertrauen geprägt, können Empfehlungen als korrekte Antwort gesehen werden [1].

2. **Steigendes Cross-Selling:** Empfehlungssysteme sind insb. in der Lage, ergänzende oder ähnliche Produkte für Kunden vorzuschlagen. Das Platzieren von Produkten in den Warenkorb stellt eine Basis für Cross-Selling-Möglichkeiten dar, da sie aktuelle Kundeninteressen indizieren. Auch Signaleffekte, die sich aus früheren Käufen anderer Kunden ergeben, sowie Werbung, bspw. in Form von Empfehlungen, können das Cross-Selling unterstützen. Letzteres zielt insb. darauf ab, Kenntnis von unbekannten Produkten zu schaffen, um Long-Tail-Umsätze zu erhöhen [90].

3. **Kundenbindung:** Konkurrenten sind im Web nur einen Klick voneinander entfernt. Aus diesem Grund sind Kundengewinnung und –bindung essentiell im E-Commerce. Treue kann durch das Aufbauen einer wertschöpfenden Beziehung zwischen dem Nutzer und der E-Commerce-Seite verbessert werden. Das Sammeln sowie Operationalisieren der Transaktionsdaten zu personalisierten Empfehlungen schaffen Bindung und führen dazu, dass Nutzer gerne auf die Seite zurückkommen. Zudem werden die *Wechselkosten* der Nutzer erhöht. Kunden müssten eine bestimmte Zeit auf konkurrierenden Seiten investieren, um vergleichbare Empfehlungen zu erhalten [114]. Ihre Zu-

friedenheit erhöht zudem die Wahrscheinlichkeit, dass sie die Seite ihren Freunden und Verwandten weiterempfehlen. Empfehlungssysteme tragen somit dazu bei, dass eine Kundengemeinschaft um ihre Produkte entsteht [71], was ebenfalls die Bindung fördert.

Tatsächlich gehen Castagnos et al. [22] davon aus, dass jene Seiten, die keine intelligenten Werkzeuge verwenden, nicht nur ein geringeres Kaufvolumen, sondern auch weniger Verkehr erfahren. Der Grund dafür sei, dass Kunden mit einer höheren Wahrscheinlichkeit zu Seiten mit Empfehlungssystemen zurückkehren.
Neben den oben genannten Möglichkeiten identifizierten Schafer et al. [114] fünf Geschäftsziele sowie Anwendungsmodelle, die diese angehen:

1. **Hilfe für neue und sporadische Besucher:** Nicht-personalisierte Empfehlungslisten, wie Bestseller aller Produkte, Bestseller in einer Kategorie und Expertenempfehlungen, sind geeignet, um neue oder sporadische Besucher zu binden. Solche Listen benötigen wenig Input und sind für diese Nutzer effektiv, da wenig über sie bekannt ist. Auf der anderen Seite ist ein geringes Maß an Personalisierung möglich. Diese ist jedoch beschränkt auf kurzzeitige kontextuelle Informationen, wie bspw. aktuell angesehene Produkte.

2. **Glaubwürdigkeit durch Gemeinschaft:** Da Kunden oftmals der Meinung sind, dass E-Commerce Unternehmen ausschließlich darauf aus sind, Umsätze zu generieren, müssen diese Wege finden, ihre Glaubwürdigkeit zu verbessern. Kundenberichte, -kommentare und -bewertungen bieten einen Weg, glaubwürdig zu erscheinen und ein Gemeinschaftsgefühl zu entwickeln.

3. **Einladung von Kunden:** E-Commerce-Seiten, die die Interessen ihrer Kunden kennen, können diese Informationen nutzen, um sie erneut zu ihrer Seite zu locken, indem sie ihnen neue, interessante Produkte oder Angebote vorstellen.

4. **Cross-Selling:** Wie bereits beschrieben, können aus aktuellen Kundeninteressen effektive Produktempfehlungen generiert werden. Solche produktbezogenen Empfehlungen sind gut geeignet, um sie in Produktinformationsseiten zu integrieren, wie bspw. Amazons „*Kunden, die …*“ Empfehlungen.

5. **Langfristige Beziehungen:** Das Ziel der meisten E-Commerce Unternehmen sind langfristige Beziehungen. Starke Personalisierung basiert auf riesigen Mengen an Kundendaten, die es ermöglichen persistente Empfehlungen oder Prognosen zu generieren. Eine solche Art der Personalisierung wird als eine Möglichkeit angesehen, um einen Kundenwert zu bestimmen und durch die Erhöhung der Wechselkosten größere Wettbewerbsbarrieren zu bilden. Für diesen Zweck existiert eine Reihe von Algorithmen, die im Verlauf der Arbeit noch thematisiert werden.

Überzeugung

Empfehlungssysteme stellen in erster Linie Systeme dar, die ihre Kunden davon überzeugen wollen, ihren Empfehlungen nachzugehen. Neben dem Verhalten können solche *Überzeugungssysteme* Einstellungen, Ansichten und Entscheidungen der Kunden beeinflussen.
Es erscheint logisch, dass die Überzeugungskraft stark vom Vertrauen des Kunden abhängig ist. Diese nutzen schließlich Empfehlungssysteme nur dann, wenn sie ihnen vertrauen. Dabei können soziale Faktoren eine Rolle spielen. Im Grunde nehmen Nutzer Empfehlungssysteme als soziale Akteure wahr und ordnen ihnen eine Persönlichkeit zu. In ihren Augen stellen sie intelligente Personen dar, die Produkte und Kundenpräferenzen kennen [19]. Laut Komiak und Benbasat [68] identifizieren sich Nutzer mit personalisierten Systemen und entwickeln eine Art *Wir-Gefühl*, was dazu führt, dass Vertrauen und Handlungsbereitschaft steigen.
Neben den Empfehlungen hat der Prozess zur Erhebung der Bewertungen einen großen Einfluss auf die Überzeugungskraft eines Empfehlungssystems. Ein Grund dafür ist, dass dieser Erhebungsprozess Erwartungen bezüglich der Qualität der Empfehlungen erzeugt. Gretzel und Fesenmaier [48] sehen den Erhebungsvorgang als einen sozialen Prozess an. In Form eines Gesprächs beeinflusse er die Wahrnehmung des Systems und der generierten Empfehlungen. Das Stellen von mehr Fragen senke nur marginal die Freude, steigere gleichzeitig jedoch den wahrgenommenen Wert. Außerdem habe ein größerer Aufwand seitens der Nutzer zur Folge, dass höhere Erwartungen gestellt werden und nicht zufriedenstellende Empfehlungen in ungünstigerem Licht stehen.

2.3 Interfaces von Empfehlungen

Die Darstellung von Empfehlungen für die Nutzer hat neben den verschiedenen Algorithmen einen großen Einfluss auf die Effektivität von Empfehlungssystemen. Im Folgenden werden angelehnt an Schafer et al. [113] sieben verschiedene Interfaces beschrieben.

1. ***Browsing*:** Kunden ohne genaue Vorstellung über das gewünschte Produkt wenden sich im Offline-Handel an den Verkäufer, dessen Empfehlungen im Laden oftmals erst gesucht werden müssen. Empfehlungen im E-Commerce bringen hier eine Menge Vorteile gegenüber dem traditionellen Weg mit sich. Die Qualität und Quantität der Empfehlungen ist nicht vom Wissen eines Verkäufers abhängig. Zudem können die vorgestellten Artikel unmittelbar zu den Produktinformationen selbst verlinkt sein. Der Käufer erspart sich so die mühsame Suche nach weiteren Produkten. Dieses Interface hilft dabei, den *Surfer* in einen potentiellen Kunden zu verwandeln, da der Nutzer nur wenige Klicks vom Bestellen entfernt ist und eine niedrige Hürde überwinden muss.

2. **Ähnliches Produkt:** Amazons „*Kunden, die…*“ erlaubt spezifischere und individuellere Empfehlungen. Nutzer werden so auf Produkte hingewiesen, die sie eventuell vergessen haben oder gar nicht kennen. Durch diese Methode kann die Produktlinie des Kunden erweitert und im Idealfall die Zahl der verkauften Artikel erhöht werden.

3. **E-Mail:** E-Mail Empfehlungen enthalten neue Produkte bzw. spezielle Angebote, die die Aufmerksamkeit des Kunden wecken sollen. Die Konsumenten begrüßen diese Art von Empfehlungen, da sie auf einfachem Wege immer auf dem aktuellsten Stand des Produktsortiments sind. Dieses Feature erhöht den Umsatz durch eine höhere Kundenbindung und zeigt sich in einem Anstieg der Wiederbesuchsrate.

4. **Textkommentare:** Immer mehr Beliebtheit erfreut sich die Angabe von Textkommentaren, in der Käufer das Produkt bewerten. Die Kaufentscheidung wird auf diesem Wege entweder stark gefestigt oder komplett verworfen, abhängig davon, ob die angegeben Referenzen viele positive oder negative Kommentare enthalten. Je mehr positive Bewertungen ein Produkt hat, desto glaubwürdiger sind diese auch. Vergleichbar ist dieses Feature mit der Einholung von Empfehlungen von Verwandten und Bekannten.

5. **Durchschnittsbewertung:** Eine einfachere Methode stellt die Darstellung der Durchschnittsbewertung dar. Hierbei müssen die Kunden keine großartigen Texte eingeben, sondern bewerten das Produkt anhand eines Punktesystems. Der Durchschnitt aller gegebenen Punkte erscheint als Bewertung neben den Produkten.

6. **Top-N Liste:** Sobald genug Bewertungsdaten vorhanden sind, ist es möglich, jedem Kunden eine individuelle Liste der Top-N Produktvorschläge zu erstellen, die nach seinen persönlichen Vorlieben gereiht sind. Auch dieses Feature hat das Potenzial, Surfer zum Kaufen zu bringen. Zudem kann der Kunde durch diese Empfehlungen etwaige Zweifel gegenüber dem Artikel ausräumen und so seine Kaufentscheidungen festigen.

7. **Sortierte Suchergebnisse:** Ein weiteres Interface stellen sortierte Suchergebnisempfehlungen dar. Die Anzahl der Empfehlungen ist hierbei nicht limitiert. Auf Suchanfragen erhält der Nutzer Empfehlungen in Form einer Liste. Diese ist sortiert nach der Wahrscheinlichkeit, dass das empfohlene Produkt zum Kunden passt.

3 State-of-the-art Empfehlungssysteme

In den meisten Studien zu Empfehlungssystemen werden die zugrunde liegenden Verfahren thematisiert. Seit Beginn der Forschungen wurden verschiedene Empfehlungsalgorithmen vorgeschlagen und getestet. Angelehnt an Ricci et al. [99] sowie Aggarwal [3] werden folgende Kategorien von Empfehlungssystemen unterschieden:

1. **Nicht-personalisiertes Filtern:** Die Empfehlungen sind für alle Kunden gleich. Empfohlen werden z.B. die beliebtesten Produkte.

2. **Kollaboratives Filtern:** Empfehlungen basieren auf historischen Bewertungen von Kunden. Traditionell werden entweder hoch bewertete Produkte von ähnlichen Kunden oder ähnliche Produkte zu bereits hoch bewerteten Artikeln empfohlen.

3. **Inhaltsbasiertes Filtern:** Empfehlungen beruhen auf Inhalten bzw. Eigenschaften der Produkte. Es werden jene Artikel empfohlen, die inhaltlich präferierten Produkten ähneln.

4. **Wissensbasiertes Filtern:** Empfehlungen werden aufgrund von Fachwissen darüber ausgesprochen, wie bestimmte Produkteigenschaften die Befriedigung von Kundenbedürfnissen beeinflussen. Dazu werden Ähnlichkeitsfunktionen oder Wissensdatenbanken mit expliziten Regeln genutzt.

5. **Demographisches Filtern:** Empfehlungen beruhen auf soziodemographischen Daten.

6. **Hybrides Filtern:** Eine Kombination aus bereits vorgestellten Filtern werden zur Empfehlungsgenerierung genutzt.

Der Begriff *Filtern* wird in der Literatur oft synonym zu *Empfehlungssystemen* verwendet. Die ersten Beiträge aus diesem Forschungsgebiet [44] prägten dieses Wort. In den folgenden Abschnitten werden die aufgezählten Ansätze näher erläutert. Der Schwerpunkt liegt dabei auf kollaborativen Empfehlungsverfahren, die im weiteren Verlauf der Arbeit eine wesentliche Rolle spielen werden.

3.1 Nicht-personalisiertes Filtern

Nicht-personalisierte Empfehlungssysteme berücksichtigen nicht die Vorlieben der einzelnen Kunden. Sie präsentieren stattdessen allen Nutzern dieselben Empfehlungen. Dennoch sind solche Systeme allgemein bekannt, da sie auf einfachen Berechnungen basieren. Beispiele umfassen Bestsellerlisten, die bestbewerteten Produkte, aber auch die Anzeige von Überblicksstatistiken, wie etwa „98% der Kunden, die dieses Hotel bewertet haben, empfehlen es weiter“ oder „dieses Produkt wird durchschnittlich mit 3,75 Sternen bewertet, wobei sich die Bewertungen wie folgt verteilen...“ [114],[128]. Die Vor- und Nachteile nicht-personalisierter Systeme wurden bereits in Abschnitt 2.2.1 besprochen.

3.2 Kollaboratives Filtern

Kollaborative Filter (Abk.: KF) nutzen zur Empfehlungsgenerierung die gemeinschaftliche Stärke der Bewertungsdaten. Die Idee hinter solchen Verfahren ist, dass unbekannte Bewertungen prognostiziert werden können, weil bekannte Bewertungen von verschiedene Kunden und/oder Produkten oft stark miteinander korrelieren. Gegeben seien zwei Kunden Alice und Bob, die einen sehr ähnlichen Geschmack haben. Sind ihre abgegebenen Bewertungen ähnlich, wird dies vom Empfehlungsalgorithmus aufgefasst. In solchen Fällen, ist es sehr wahrscheinlich, dass Bewertungen, die nur eine von beiden Personen abgegeben hat, ebenfalls ähnlich sind. Diese Ähnlichkeitsvermutung wird genutzt, um Schlussfolgerungen über unbekannte Bewertungen zu machen. Abbildung 3.1 zeigt den schematischen Ablauf eines kollaborativen Empfehlungsprozesses. Auf Basis einer $m \times n$ Kunde-Produkt-Matrix generiert ein KF-Algorithmus entweder Empfehlungen oder Prognosen, die den Nutzern in Form einer Top-N Empfehlungsliste präsentiert werden.

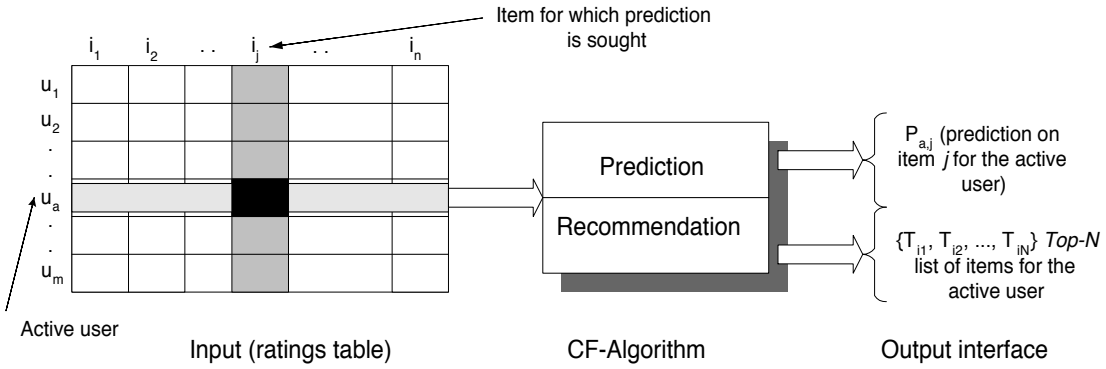

Abbildung 3.1: Kollaborativer Filterprozess (Quelle: [109], S. 288)

Es werden zwei Arten von Verfahren in kollaborativen Empfehlungssystemen unterschieden: *Speicherbasierte* und *modellbasierte*. Speicherbasierte Methoden, auch als *nachbarschaftsbasierte* Verfahren bekannt, gehören zu einer der ältesten Empfehlungsalgorithmen und prognostizieren Kundenbewertungen auf Basis von Nachbarschaften zwischen verschiedenen Kunden oder Produkten. Bei modellbasierten Methoden werden Verfahren aus dem Bereich maschinelles Lernen und Data Mining als Prognosemodell genutzt. Beispiele für solche Verfahren sind *Entscheidungsbäume*, *regelbasierende Modelle*, *Bayessche Methoden* sowie *latente Faktormodelle*. Bevor die beiden genannten Ansätze näher erläutert werden, soll beschrieben werden, mit welchen Daten kollaborative Empfehlungsalgorithmen arbeiten und welche Herausforderungen sich daraus ergeben.

3.2.1 Arten von Bewertungsdaten

Kollaborative Empfehlungssysteme arbeiten auf *expliziten* und/oder *impliziten* Bewertungsdaten. Unter expliziten Daten sind solche Bewertungen zu verstehen, die willentlich abgegeben wurden. Das typische Beispiel hierfür sind Kundenbewertungen für gekaufte oder runtergeladene Produkte. [114]. Explizite Bewertungen können wie folgt kategorisiert werden:

- **Kontinuierliche Bewertungen:** Bewertungen werden auf einer kontinuierlichen Zufriedenheitsskala spezifiziert. Ein Beispiel für ein solches Empfehlungssystem ist das System *Jester Joke* [45], in dem Bewertungen zwischen *-10* und *10* abgegeben werden können. Der Nachteil einer solchen Skala ist, dass der Nutzer einen von unendlich vielen reellen Werten wählen muss. Deswegen ist dieser Ansatz relativ selten vertreten.

- **Intervall-basierende Bewertungen:** Intervallbasierende Bewertungen werden meist von einer 5- oder 7-Punkte-Skala umfasst. Möglich sind auch 10- oder 20-Punkte-Skalen. Solche Bewertungen können bspw. numerische Werte zwischen *1* und *5*, zwischen *-2* und *2* oder zwischen *1* und *7* annehmen. Für diese Werte gilt jedoch die wichtige Annahme, dass sie abstandsgleich sind.

- **Ordinale Bewertungen:** Ordinale Bewertungen sind den intervallbasierenden ähnlich. Der Unterschied besteht darin, dass sie kategorische Werte verwenden. Beispiele für solche Werte könnten Antworten wie *Starker Widerspruch*, *Widerspruch*, *Neutral*, *Zustimmung*, *Starke Zustimmung*. Die für intervallbasierende Bewertungen geltende Annahme trifft hier nicht zu. Der Unterschied ist jedoch nur auf theoretischer Ebene, da in der Praxis den kategorischen Werten numerische zugewiesen werden.

- **Binäre Bewertungen:** Im Fall von binären Bewertungen existieren lediglich zwei Optionen: positiv oder negativ. Sie stellen Spezialfälle von intervallbasierenden und ordinalen Bewertungen dar. Als Beispiel bietet die *Pandora Internet Radio Station*[3] die Möglichkeit, dass Nutzer ein Lied entweder mit *Like* oder *Dislike* bewerten.

Implizite Daten werden aus dem Verhalten der Kunden gewonnen, ohne dass diese bewusst Präferenzen angeben und ohne dass sie wissen, dass Daten gesammelt oder zu Empfehlungszwecken genutzt werden [114]. Beispiele umfassen die Kaufhistorie oder aktuell angeklickte Produkte sowie der *Like-Button* in Facebook. An diese Art von Daten gelangen Unternehmen leichter, da es wahrscheinlicher ist, dass ein Kunde ein Produkt kauft oder klickt als dass er es bewertet. Stellt man die Transaktionen in einer Kunde-Produkt-Matrix dar, enthalten sie im Falle eines Kaufs, Klicks, etc. eine *1* oder sind ansonsten leer. Auf der einen Seite lassen sich aus impliziten Daten sehr effizient Empfehlungen generieren, wie z.B. Amazons „*Kunden, die A kauften, kauften auch B*“. Auf der anderen Seite können sie nicht unmittelbar auf alle Algo-

[3] http://www.pandora.com/restricted.

rithmen angewandt werden, die für explizite Daten bestimmt sind. Folgende Eigenschaften sind die Ursache dafür [60]:

- **Keine negativen Bewertungen:** Durch das Betrachten des Kundenverhaltens lässt sich einerseits prognostizieren, welche Produkte ein Nutzer mag und daher konsumiert. Jedoch ist schwer zu sagen, welche Artikel ihm nicht gefallen.
- **Rauschen:** Das gekaufte Produkt könnte ein Geschenk oder aus Sicht des Kunden eine Enttäuschung sein. Diese Umstände würden dazu führen, dass das Empfehlungssystem den Kauf falsch interpretiert, da es einen Kauf als gute Bewertung indiziert.
- **Vertrauen statt Präferenz:** Die numerischen Werte von impliziten Daten indizieren Vertrauen, während explizite Bewertungen Präferenzen darstellen. Der Vertrauenswert berechnet sich aus der Anzahl, wie oft der Kunde das Produkt kaufte. Ein hoher Wert zeigt jedoch nicht, wie sehr dem Kunden das Produkt gefällt. Auch umgekehrt kann es sich trotz einmaligen Kaufes um sein Lieblingsprodukt handeln.

Explizite und implizite Daten müssen sich jedoch nicht gegenseitig ausschließen. Bspw. arbeitet Amazon mit beiden Arten von Bewertungsdaten. Solche hybriden Daten können u.a. zur Lösung der im nächsten Abschnitt erläuterten Herausforderungen genutzt werden.

3.2.2 Charakteristiken und Herausforderungen

Empfehlungssysteme müssen oftmals in Umgebungen arbeiten, die eine Reihe von Herausforderungen mit sich bringen. Dennoch sollen sie im Interesse der Kunden und Unternehmen schnell genaue Empfehlungen generieren. Inwieweit kollaborative Empfehlungsverfahren dazu in der Lage sind, ist davon abhängig, wie sie mit folgenden Herausforderungen, die gleichzeitig zu ihren Charakteristiken gehören, umgehen.

Spärlichkeit der Daten

Viele Empfehlungssysteme arbeiten mit riesigen Mengen an Produkten. Dies führt in den meisten Fällen dazu, dass die Kunde-Produkt-Matrix extrem spärlich besetzt ist, was wiederrum die Qualität der Prognosen bzw. der Empfehlungen beeinträchtigt. Spärlichkeit tritt in unterschiedlichen Formen auf. Zum einen ist das *Kaltstartproblem* zu nennen. Es beinhaltet die Schwierigkeit, aufgrund mangelnder Bewertungen für neue Kunden oder neue Produkte ähnliche Objekte zu finden [2],[133]. Neue Artikel werden so lange nicht empfohlen, bis sie nicht eine gewisse Anzahl an Bewertungen aufweisen. Selbiges gilt für neue Kunden.
Ein weiteres Problem, welches durch die Spärlichkeit der Bewertungsdaten entsteht, betrifft die *Coverage* (engl.: *Abdeckung*). Sie definiert den Anteil der empfohlenen Produkte am gesamten Produktbestand. Das Problem der reduzierten Coverage entsteht dann, wenn die Zahl der Kundenbewertungen im Vergleich zur Produktanzahl klein ist. Dadurch ist das Empfehlungssystem nicht in der Lage, für spärlich bewertete Artikel Empfehlungen zu generieren.

Das Phänomen der *Nachbar-Transitivität* betrifft das Problem, dass Kunden mit ähnlichen Präferenzen nicht als solche identifiziert werden können, da sie kein gemeinsam bewertetes Produkt aufweisen. Dies führt in Empfehlungssystemen mit paarweisen Kundenvergleichen dazu, dass die Qualität der Empfehlungen sinkt.
Es wurden bereits mehrere Ansätze entwickelt, um das Problem der Spärlichkeit zu mindern. So entfernen Methoden des *dimensionsreduzierenden Ansatzes*, wie z.B. die *Singularwertzerlegung* [14], nicht repräsentative oder insignifikante Kunden oder Produkte, um die Anzahl der Dimensionen der Kunde-Produkt-Matrix unmittelbar zu reduzieren. Dies birgt jedoch die Gefahr, dass nutzvolle Informationen verloren gehen und so die Empfehlungsqualität sinkt [78], [107]. Hybride Empfehlungsverfahren, wie bspw. der *content-boosted* KF-Algorithmus [83], sind ebenfalls hilfreich, dieses Problem anzugehen. Mithilfe zusätzlicher Inhaltsinformationen, die typische Eingabedaten eines inhaltsbasierten Empfehlungssystems darstellen, werden Prognosen für neue Kunden oder Produkte berechnet.

Skalierbarkeit

Empfehlungssysteme müssen unverzüglich auf Online-Anfragen reagieren und für alle Nutzer, ungeachtet ihrer Bewertungs- oder Transaktionshistorie, Empfehlungen produzieren [78]. Jedoch weisen traditionelle KF-Methoden bei immer größer werdender Zahl von Kunden und Produkten Skalierungsprobleme auf. Auch für diese Herausforderung existiert eine Reihe von Lösungen. Hier ist zum einen die inkrementelle Singularwertzerlegung [110] zu nennen, welches nach Hinzufügen zusätzlicher Bewertungen ohne Wiederberechnung des dimensionsreduzierten Modelles neue Prognosen berechnet [30]. So ist das System nicht gezwungen, nach jeder Aktualisierung des Datenbestandes den gesamten Algorithmus neu zu berechnen. Dennoch ist festzuhalten, dass die teure Matrixfaktorisierung durchgeführt werden muss.
Speicherbasierte Ansätze, wie die produktbasierte KF-Methode, können ebenfalls eine gute Skalierbarkeit erreichen. Anstatt alle möglichen Produktkombinationen zu betrachten, werden nur Ähnlichkeiten zwischen von einem Kunden gemeinsam bewerteten Produktpaaren berechnet [78]. *Bayessche KF-Verfahren* gehen das Skalierungsproblem an, indem sie Prognosen auf Basis bekannter Bewertungen berechnen [85]. *Clustermethoden* hingegen suchen zur Empfehlungsgenerierung nicht innerhalb der gesamten Datenbank nach ähnlichen Kunden, sondern innerhalb kleiner Cluster [24]. Jedoch geht hier die erhöhte Skalierbarkeit auf Kosten der Prognosegenauigkeit.

Gray Sheep

Unter *Gray sheep* sind jene Nutzer zu verstehen, deren Präferenzen weder beständig übereinstimmend noch nicht-übereinstimmend mit Bewertungen anderer Kundengruppen sind und somit vom kollaborativen Empfehlungsverfahren nicht profitieren. Im Gegensatz dazu gehören zu den *Black sheeps* solche Kunden, deren einzigartige Geschmäcker Empfehlungen nahezu unmöglich machen. Zur Lösung des *gray sheep* Problems entwickelten Claypool et al. [25]

einen hybriden Ansatz aus inhaltsbasierten und kollaborativen Verfahren, der einen gewichteten Durchschnitt beider Prognosen berechnet. Die Gewichte werden dabei kundenweise festgelegt, um für jeden Nutzer eine optimale Mischung zu generieren.

Shilling Attacks

In manchen Fällen können alle Besucher Bewertungen abgeben unabhängig davon, ob das Produkt beschaffen wurde oder nicht. So werden eigene Artikel weiterempfohlen, während Konkurrenzprodukte schlechte Bewertungen erhalten. Dieses Phänomen wird in der Literatur als *Shilling Attacks* bezeichnet [98]. Bell und Koren [13] schlugen eine Lösung des Problems für speicherbasierte Verfahren vor. Dazu entfernten sie globale Effekte während der Datennormalisierung und nutzten zur Nachbarschaftssuche ihre Residuen. Letztlich erlangten sie durch Experimente mit Netflix-Datensätzen verbesserte Ergebnisse.

3.2.3 Speicherbasierte Ansätze

Die Grundidee bei speicherbasierten oder nachbarschaftsbasierten Methoden ist die, dass Ähnlichkeiten zwischen Zeilen und/oder Spalten der Kunde-Produkt-Matrix berechnet werden. Jede Zeile stellt die Bewertungshistorie eines Nutzers dar, jede Spalte die eines Produktes. Das Konzept der Nachbarschaft impliziert, dass die ähnlichsten Kunden oder Produkte gefunden werden müssen. Es stehen somit zwei unterschiedliche Ansätze zur Auswahl:

1. **Kundenbasierte Modelle:** Ähnliche Kunden weisen vergleichbare Bewertungen für dieselben Produkte auf. Wenn Alice und Bob in der Vergangenheit Filme ähnlich bewerteten, kann die bekannte Bewertung von Alice für den Film *Terminator* genutzt werden, um Bobs fehlende Meinung zu diesem Film zu prognostizieren.

2. **Produktbasierte Modelle:** Kunden bewerten ähnliche Produkte auf vergleichbare Art und Weise. Bobs Bewertungen für ähnliche Science Fiction Filme wie *Alien* und *Predator* können genutzt werden, um seine Bewertung für *Terminator* zu prognostizieren.

Angelehnt an die Studie von Xia et al. [129] werden im Folgenden beide Modelle sowie gemeinsame Komponenten speicherbasierter Verfahren näher erläutert.

Kundenbasierte Modelle

Die kundenbasierte KF-Methode wurde das erste Mal im Artikelempfehler *GroupLens* [69] verwendet. Der Musikempfehler *Ringo* [115] sowie das Video-Empfehlungssystem *BellCore* [56] machen ebenfalls von dieser Methode Gebrauch. Bei diesem Ansatz werden Nachbarschaften auf Basis ähnlicher Kunden definiert. Um die Nachbarschaft des Zielkunden ermitteln zu können, werden Ähnlichkeiten zu allen anderen Nutzern berechnet. Eine solche Ähnlichkeitsberechnung ist nicht einfach, da die Kunden mit unterschiedlichen Skalen bewerten. Ein

Nutzer neigt dazu, viele Produkte zu mögen, während einem anderen weniger Artikel gefallen. Zudem kann das Problem der Nachbar-Transitivität auftreten, bei dem möglicherweise ähnliche Kunden unterschiedliche Produkte bewerteten und somit keine bis wenige *Schnittprodukte* vorhanden sind. Diese Aspekte müssen von den Mechanismen identifiziert und berücksichtigt werden.

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

Tabelle 3.1: Beispiel für eine Bewertungsmatrix

Kundenbasierte Nachbarschaftsmethoden prognostizieren die Bewertung $\hat{r}_{ui}$ des Nutzers u für das Produkt i, indem sie die Bewertungen für i von jenen Kunden nutzen, die u am ähnlichsten sind. Diese oder ein Teil dieser bilden die Nachbarschaft von u. Nach Berechnung der Ähnlichkeitsgewichte w_{uv} zwischen Nutzer u und allen anderen Nutzern $v \neq u$ werden die k *nächsten Nachbarn* $\mathcal{N}_i(u)$ von u bestimmt. Diese bestehen aus den k Kunden mit der höchsten Ähnlichkeit w_{uv}. Zusätzlich werden zur Prognose nur jene Nutzer herangezogen, die das Produkt i bewerteten. $\hat{r}_{ui}$ kann daraufhin aus der Durchschnittsbewertung der Nachbarschaft von u berechnet werden:

$$\hat{r}_{ui} = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in \mathcal{N}_i(u)} r_{vi} \qquad (3.1)$$

Diese Formel berücksichtigt jedoch nicht, dass die Nachbarn unterschiedliche Ähnlichkeitslevel haben können. Betrachten wir als Beispiel die Tabelle 3.1. Dazu wird angenommen, dass die zwei nächsten Nachbarn von Eric Lucy und Diane sind. Es erscheint unangebracht, beide Bewertungen für den Film *Titanic* gleich zu behandeln, da der Geschmack von Lucy dem von Eric deutlich näher ist. Es gibt zwei Möglichkeiten, dieses Problem zu umgehen: Zum einen können die Nachbarn unterschiedlich gewichtet werden. Ergibt die Summe der Gewichte dabei nicht 1, folgt eine Prognose außerhalb der erlaubten Werte. Aus diesem Grund stellt der zweite Lösungsansatz, die Normalisierung der Gewichte, die bessere Variante dar:

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \qquad (3.2)$$

w_{uv} kann durch w_{uv}^{α} ersetzt werden, wobei $\alpha > 0$ ein Verstärkungsfaktor darstellt [16].
Um mit Formel 3.2 Erics Bewertung für den Film *Titanic* zu prognostizieren, wird beispielhaft eine Ähnlichkeit von 0,75 zu Lucy sowie von 0,15 zu Diane angenommen. Die prognostizierte Bewertung beträgt

$$\hat{r}_{Eric,Titanic} = \frac{0{,}75\times5 + 0{,}15\times3}{0{,}75 + 0{,}15} \cong 4{,}67$$

und liegt näher an Lucys als an Dianes Bewertung. Formel 3.2 hat den Nachteil, dass zwei oder mehr Kunden ein Produkt unterschiedlich bewerten, obwohl sie dasselbe Maß an Wertschätzung für dieses aufbringen. Ein Nutzer gibt die höchste Bewertung nur für wenige, herausragende Produkte, während ein anderer diese Bewertung vielen Produkten gibt, die ihm gefallen. Zur Lösung dieses Problems werden die Bewertungen r_{vi} der Nachbarn v normalisiert [97]:

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{v\in\mathcal{N}_i(u)} w_{uv} h(r_{vi})}{\sum_{v\in\mathcal{N}_i(u)} |w_{uv}|}\right) \qquad (3.3)$$

$h(r_{vi})$ stellt die normalisierte Form der Nachbarbewertung dar. Weiterhin muss die prognostizierte Bewertung wieder in die ursprüngliche Skala umgewandelt werden. Dafür ist die Funktion h^{-1} zuständig. Die unterschiedlichen Ansätze der Normalisierung werden in einem späteren Abschnitt besprochen.

Produktbasierte Modelle

Die produktbasierte Methode bietet sich bei Unternehmen mit großer Kundenbasis an, da kundenbasierte Modelle in solchen Fällen Skalierungsprobleme aufweisen. Erstere wurde das erste Mal von Sarwar et al. [109] und Karypis [64] beschrieben. Heute gehört sie zu den am weitesten verbreiteten Verfahren kollaborativer Empfehlungssysteme. Um Kundenbewertungen zu prognostizieren, werden statt Ähnlichkeiten zwischen Bewertungsverhalten von Kunden Ähnlichkeiten zwischen Bewertungsmustern von Produkten berechnet. Gefallen bzw. missfallen zwei Artikel denselben Nutzern, sind sie sich ähnlich. Man erwartet somit, dass ein Kunde dieselben Präferenzen für ähnliche Produkte hat. Betrachten wir dazu erneut das Beispiel aus Tabelle 3.1. Anstatt andere Personen mit gleichem Geschmack zu befragen, entscheidet sich Eric, den Film *Titanic* unter Berücksichtigung jener Filme zu bewerten, die er bereits gesehen hat. Er bemerkt, dass Leute, die diesen Film bewerteten, ähnliche Meinungen zu den Filmen *Forrest Gump* und *Wall-E* haben. Da ihm beide gefallen, folgert er, dass er *Titanic* ebenfalls mögen wird.

Zur Formalisierung dieses Ansatzes wird angenommen, dass die Menge $\mathcal{N}_u(i)$ jene Produkte beinhaltet, die vom Nutzer u bewertet wurden und dem Artikel i am ähnlichsten sind. $\hat{r}_{ui}$ kann als gewichteter Durchschnitt aus den Bewertungen von u für die Artikel $j \in \mathcal{N}_u(i)$ berechnet werden:

$$\hat{r}_{ui} = \frac{\sum_{j\in\mathcal{N}_u(i)} w_{ij} r_{uj}}{\sum_{j\in\mathcal{N}_u(i)} |w_{ij}|} \qquad (3.4)$$

Basiert die Prognose erneut auf den zwei nächsten Nachbarn und weisen *Forrest Gump* und *Wall-E* mit Bewertungen von 5 und 4 Ähnlichkeitswerte von 0,85 und 0,75 zu *Titanic* auf,

ergibt sich Eric diese beiden Filme mit 5 und 4 bewertet hat, ergibt sich für Eric folgende Prognose:

$$\hat{r}_{Eric,Titanic} = \frac{0{,}85\times5 + 0{,}75\times4}{0{,}85 + 0{,}75} \cong 4{,}53$$

Auch hier können individuelle Bewertungsskalen durch die Normalisierung der Bewertungen mit der Funktion h berücksichtigt werden:

$$\hat{r}_{ui} = h^{-1}\left(\frac{\sum_{j\in\mathcal{N}_u(i)} w_{ij} h(r_{uj})}{\sum_{j\in\mathcal{N}_u(i)} |w_{ij}|}\right) \tag{3.5}$$

Gleichung 3.4 weist einen weiteren Schwachpunkt auf: Im Falle von impliziten Bewertungen funktioniert das Mitteln nicht. Für den Fall, dass alle Ähnlichkeitsgewichte positiv sind und Kunde u Produkt j gekauft hat, sodass $r_{uj} = 1$ gilt, lautet das Ergebnis der Gleichung immer 1. Um dieses Problem zu umgehen, werden für den Kunden u durch simples Aggregieren der Ähnlichkeiten w_{ij} zu Produkten j aus der Kaufhistorie I_u Pseudo-Prognosen $\tilde{p}_{ui}$ berechnet. Hierbei erwies sich das Summieren gegenüber dem Durchschnitt und dem Maximum als geeignetere Aggregierungsfunktion [31]:

$$\tilde{p}_{ui} = \sum_{j\in I_u} w_{ij} \tag{3.6}$$

Die Werte für $\tilde{p}_{ui}$ sind nicht dazu geeignet, um ein exaktes Kundenverhalten zu prognostizieren. Bei impliziten Daten steht die Berechnung von Wahrscheinlichkeiten üblicherweise nicht im Vordergrund. Die Prognosewerte sollen vielmehr dazu genutzt werden, um für jeden Nutzer eine Rangfolge unter den kandidierenden Artikeln zu bestimmen. Letztlich bietet diese Berechnung für die produktbasierte KF-Methode eine gute Basis, um Empfehlungen auf Basis historischer Kaufdaten zu generieren [31],[64],[78].

Kundenbasierte und produktbasierte Methode im Vergleich

Bei der Wahl zwischen kunden- und produktbasierter Methode sollten folgende Unterschiede berücksichtigt werden:

- **Genauigkeit:** Die Genauigkeit der nachbarschaftsbasierten Empfehlungsalgorithmen ist meistens vom Verhältnis zwischen der Anzahl Kunden und Produkte abhängig. Generell gilt, dass eine kleine, zuverlässige Nachbarschaft einer großen vorzuziehen ist. Aus diesem Grund generiert die produktbasierte KF-Methode genauere Empfehlungen, wenn die Zahl der Nutzer $|\mathcal{U}|$ größer ist als die der Artikel $|\mathcal{I}|$, wie bspw. bei Amazon [37],[109]. Auf der anderen Seite profitieren Empfehlungssysteme mit verhältnismäßig vielen Produkten von der kundenbasierten KF-Methode. Als Beispiel hierfür sind Empfehler für Forschungsartikel zu nennen, die mit Tausenden von Nutzern und Hunderttausenden von Artikeln arbeiten [54].

- **Effizienz:** Tabelle 3.2 verdeutlicht, dass auch die Speicher- und die Berechnungseffizienz von oben genanntem Verhältnis abhängig ist. Die Funktionen $p = \max_u |\mathcal{I}_u|$ bzw. $q = \max_i |\mathcal{U}_i|$ stellen die maximale Anzahl der Bewertungen pro Kunde bzw. pro Produkt dar. Falls $|\mathcal{U}|$ größer ist als $|\mathcal{I}|$, benötigt die produktbasierte Methode für den Trainingsprozess weniger Ressourcen. Für den Anwendungsprozess ist die Komplexität identisch. Sie ist für beide Verfahren von $|\mathcal{I}|$ und der maximalen Zahl Nachbarn k abhängig. In der Praxis werden oftmals nur die k größten Ähnlichkeitsgewichte gespeichert, die größer als 0 sind, sodass die Berechnung der Ähnlichkeitsmatrix nicht annähernd so speicherintensiv ist, wie sie als Worst-Case in Tabelle 3.2 dargestellt ist.

Methode	**Speicher**	**Zeit**	
		Training	**Online**
Kundenbasiert	$O(\|\mathcal{U}\|^2)$	$O(\|\mathcal{U}\|^2 p)$	$O(\|\mathcal{I}\|k)$
Produktbasiert	$O(\|\mathcal{I}\|^2)$	$O(\|\mathcal{I}\|^2 q)$	$O(\|\mathcal{I}\|k)$

Tabelle 3.2: Speicher- und Zeitkomplexität der nachbarschaftsbasierten Methoden

- **Stabilität:** Bei der Wahl des Verfahrens sollte weiterhin berücksichtigt werden, in welcher Häufigkeit und in welchem Ausmaß sich Kunden- und Produktbestand ändern. Erweist sich die Liste der Artikel als verhältnismäßig statisch, ist die produktbasierte Methode vorzuziehen. In solchen Fällen müssen Ähnlichkeitsgewichte nur selten erneut berechnet werden, um neuen Kunden Produkte empfehlen zu können. Dagegen stellt in Umgebungen mit ständigen Veränderungen des Produktbestandes das kundenbasierte Verfahren die geeignetere Methode dar.

- **Erklärbarkeit:** Die produktbasierte Methode hat den Vorteil, dass ihre Empfehlungen einfacher zu begründen sind. Dem Nutzer wird zur Erklärung der Empfehlungen eine Liste mit Nachbarprodukten und ihren Ähnlichkeitsgewichten präsentiert. Zudem kann es Kunden ermöglicht werden, die Liste der Nachbarn und/oder ihre Gewichte zu modifizieren und so interaktiv am Empfehlungsprozess teilzunehmen. Das kundenbasierte Verfahren ist hingegen weniger geeignet für solche Erklärungen. Das Präsentieren ähnlicher Kunden erscheint nicht sinnvoll, da der betroffene Nutzer diese nicht kennt.

- **Serendipität:** Bei der produktbasierten Methode wird eine Produktbewertung auf Basis von Bewertungen ähnlicher Artikel prognostiziert. Z.B. werden Nutzern jene Filme empfohlen, die bezüglich dem Genre, den Schauspielern oder den Regisseuren bereits bewerteten gleichen. Diese Empfehlungen helfen den Kunden jedoch nicht dabei, neuartige, interessante Produkte zu entdecken. Auf dem kundenbasierten Verfahren beruhende Empfehlungssysteme sind eher in der Lage solche *zufälligen* Entdeckungen für die Nutzer zu generieren. Dies gilt insb. dann, wenn Empfehlungen aus einer kleinen Nachbarschaft entstehen. Betrachten wir als Beispiel einen Kunden *A*, der ausschließ-

lich Comedy-Filme schaut und dem Kunden *B* bezüglich der Bewertungen für diese Art von Filmen sehr ähnlich ist. *B* schaut hingegen sehr gerne einen Film aus einem anderen Genre. Dieser Film kann *A* aufgrund seiner Ähnlichkeit zu B empfohlen werden.

Komponenten der nachbarschaftsbasierten Methoden

Bei der Implementierung des nachbarschaftsbasierten Verfahrens können folgende Faktoren berücksichtigt werden: die Normalisierung der Bewertungen, die Berechnung der Ähnlichkeitsgewichte und das Auswahlverfahren der Nachbarn. Alle drei Komponenten haben sowohl für die kunden- als auch für die produktbasierte Methode einen wesentlichen Einfluss auf die Genauigkeit und Effizienz. Die nächsten Abschnitte setzen sich mit diesen Ansätzen samt ihrer Vor- und Nachteile auseinander.

Normalisierung der Bewertungen

Jeder Nutzer hat eine eigene Bewertungsskala. Die einen geben eher höhere Produktbewertungen ab, die anderen eher niedrigere. Um diese individuellen Skalen auf eine allgemeingültige zu bringen, sollten die Bewertungen normalisiert werden. Zwei der am weitesten verbreiteten Normalisierungsverfahren sind das *Durchschnittszentrieren* sowie der *Z-Score*.

Die Idee des Durchschnittszentrierens [16],[97] besteht darin, die Bewertung eines Kunden im Verhältnis zum Durschnitt aller seiner Bewertungen zu betrachten. So wird bei der Prognose berücksichtigt, ob der Nutzer eine eher hoch oder niedrig bewertende Person ist. Auch Produkte können unterschiedliche Bewertungsskalen aufweisen. Um dieser Tatsache Rechnung zu tragen, wird die Bewertung eines Produktes in Verhältnis zu seiner Durchschnittsbewertung gesetzt. Somit ergeben sich für kunden- und produktbasierte Ansätze folgende Normalisierungsfunktionen:

$$h(r_{ui}) = r_{ui} - \bar{r}_u \qquad (3.7) \qquad h(r_{ui}) = r_{ui} - \bar{r}_i \qquad (3.8)$$

Die Prognose der Bewertung berechnet sich für beide Methoden wie folgt:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \qquad (3.9) \qquad \hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij}(r_{uj} - \bar{r}_j)}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|} \qquad (3.10)$$

Die Z-Score-Normalisierung berücksichtigt nicht nur die unterschiedlichen Durchschnittsbewertungen, sondern auch die Streuung [53]. Nehmen wir als Beispiel zwei Kunden *A* und *B* mit einer Durchschnittsbewertung von 3. Während *A* Bewertungen zwischen *1* und *5* vergibt, bewertet *B* alle Artikel mit *3*. Bewerten A und B dasselbe Produkt mit 5, spiegelt die Bewertung von B eine größere Begeisterung für dieses wieder. Auch hier unterscheiden sich die kunden- und produktbasierte Methode vom Ansatz her nicht.

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_u}{\sigma_u} \qquad (3.11) \qquad h(r_{ui}) = \frac{r_{ui} - \bar{r}_i}{\sigma_i} \qquad (3.12)$$

Die durchschnittszentrierte Normalisierung der Bewertung r_{ui} wird durch die Standardabweichung σ_u bzw. σ_i der Bewertungen des Nutzers u bzw. des Artikels i dividiert. Letztlich ergeben sich folgende Prognoseformeln für beide Verfahren:

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv}(r_{vi} - \bar{r}_v)/\sigma_v}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \quad (3.13) \qquad \hat{r}_{ui} = \bar{r}_i + \sigma_i \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij}(r_{uj} - \bar{r}_j)/\sigma_j}{\sum_{j \in \mathcal{N}_u(i)} |w_{ij}|} \quad (3.14)$$

Die Berücksichtigung der Streuung ist jedoch nur in solchen Fällen sinnvoll, in denen die Bewertungsskala eine große Spanne diskreter Werte aufweist oder kontinuierlich ist. Während erste Studien zeigten, dass das Durchschnittszentrieren und der Z-Score vergleichbare Ergebnisse liefern [53], kamen Howe und Forbes [59] in einem aktuelleren Forschungsbeitrag zum Ergebnis, dass Prognosen mehr von der Z-Score-Normalisierung profitieren.
Die Normalisierung von Bewertungen kann jedoch auch unerwünschte Effekte mit sich bringen. Nehmen wir als Beispiel einen Kunden, der Produkte hoch bewertet, wenn er sie kauft. Beide Normalisierungsverfahren charakterisieren diesen Nutzer als *einfach zu überzeugen*. Zudem wäre jede Bewertung unter der höchsten als negativ zu betrachten. In der Tat ist es durchaus möglich, dass der Kunde *schwer zu überzeugen* ist und Artikel je nach Zufriedenheitsgrad gut bewertet. Falls Bewertungen eines Nutzers oder eines Produktes durchgängig dieselben Werte aufweisen, beträgt die Standardabweichung 0, was zu undefinierten Prognosewerten führt. Dennoch erwies sich die Normalisierung in Fällen nicht zu spärlicher Bewertungsdaten durchgehend als ein Faktor, der Prognosen verbessert [53],[59].

Berechnung des Ähnlichkeitsgewichtes

Während der Trainingsphase berechnen nachbarschaftsbasierte Methoden Ähnlichkeiten zwischen Kunden oder Produkten. Linden et al. [78] beschreiben diesen Prozess in ihrem Beitrag zur produktbasierten Methode von Amazon. Nehmen wir an, dass $\mathcal{I}_u$ die Menge aller Produkte sei, die von Nutzer u bewertet wurden, und $\mathcal{U}_i$ die Menge aller Kunden, die Produkt i bewerteten. Dieses Berechnungsverfahren, welches wir im weiteren Verlauf der Arbeit als *Standardtrainingsverfahren* bezeichnen, ermittelt aus Laufzeitgründen lediglich Produktkombinationen, die mindestens von einem Kunden bewertet wurden.

Algorithmus 3.1 Standardtrainingsverfahren

Input: Trainingsbasisdatensatz $B\ \langle Kunde, Produkt \rangle$
Output: Ähnlichkeitsgewicht w_{ij} zwischen Produkt i und j

```
1: function WeightComputation (B)
2:     For i ∈ B
3:         For u ∈ U_i
4:             For j ∈ I_u
5:                 n_ij ← n_ij + 1
6:                 Add j to S_i
7:         For j ∈ S_i
8:             Compute w_ij
```

Im Falle impliziter Daten wird neben den vorher bekannten Werten n_i und n_j, die Anzahl der Kunden n_{ij} ermittelt, die sowohl Produkt *i* als auch Produkt *j* kauften. Auf Basis dieser Zahlen sowie der Menge $\mathcal{S}_i$, die alle *i* ähnelnden Artikel *j* enthält, werden letztlich die Ähnlichkeitsgewichte w_{ij} berechnet. Diese spielen in nachbarschaftsbasierten Verfahren eine wesentliche Rolle. Sie ermöglichen die Auswahl der ähnlichsten Nachbarn und messen den Einfluss dieser für die Prognose. Die Wahl des Ähnlichkeitsmaßes stellt einen kritischen Faktor speicherbasierter Empfehlungssysteme dar, da sie einen signifikanten Einfluss auf Genauigkeit und Effizienz haben. Zwei der am weitesten verbreiteten Maße sind die *Kosinus-* und die *Pearson-Ähnlichkeit*. Bei der Berechnung des Kosinusmaßes [9] wird der Kunde *u* als Vektor $x_u \in \mathbb{R}^{|\mathcal{I}|}$ dargestellt, für dessen Einträge $x_{ui} = r_{ui}$ gilt, falls *u* Produkt i bewertete, und $x_{ui} = 0$, falls nicht. Die Ähnlichkeit zwischen den Nutzern *u* und *v* bzw. zwischen den Artikeln *i* und *j* berechnet sich wie folgt:

$$sim_{Kosinus}(u,v) = cos(x_u, x_v) = \frac{\sum_{i \in \mathcal{I}_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in \mathcal{I}_u} r_{ui}^2 \sum_{j \in \mathcal{I}_v} r_{vj}^2}} \tag{3.15}$$

$$sim_{Kosinus}(i,j) = cos(y_i, y_j) = \frac{\sum_{u \in \mathcal{U}_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \in \mathcal{U}_i} r_{ui}^2 \sum_{v \in \mathcal{U}_j} r_{vj}^2}} \tag{3.16}$$

$\mathcal{I}_{uv}$ beinhaltet jene Produkte, die sowohl von *u* als auch von *v* bewertet wurden, $\mathcal{U}_{ij}$ jene Kunden, die sowohl *i* als auch *j* bewerteten. Dieses Ähnlichkeitsmaß hat jedoch den Nachteil, dass es unterschiedliche Bewertungsdurchschnitte und -streuungen von *u* und *v* nicht berücksichtigt. Die bekannte Pearson-Korrelation behebt diesen Nachteil. Ihre kundenbasierte und produktbasierte Ähnlichkeitsformeln lauten wie folgt [31],[109]:

$$sim_{Pearson}(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{vi} - \bar{r}_v)^2}} \tag{3.17}$$

$$sim_{Pearson}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_j)^2}} \tag{3.18}$$

Hierbei werden im Gegensatz zum Z-Score lediglich Standardabweichungen von Bewertungen für Schnittprodukte $\mathcal{I}_{uv}$ bzw. für Schnittkunden $\mathcal{U}_{ij}$ berechnet.

Die Unterschiede der Bewertungsskalen aus Produktperspektive sind oftmals nicht sehr ausgeprägt. Kunden sind bezüglich der Bewertungen individueller als Produkte. Deswegen erscheint es für die produktbasierte Pearson-Korrelation angebrachter, Bewertungen zu vergleichen, die durch die Durchschnittsbewertung der Nutzer zentriert wurden. Die angepasste Kosinus-Ähnlichkeit [109] berücksichtigt diesen Umstand. Sie stellt eine Modifikation der Pearson-Korrelation dar und berechnet sich wie folgt:

$$sim_{AngepKosinus}(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_u)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{uj} - \bar{r}_u)^2}} \quad (3.19)$$

Tatsächlich zeigte eine Studie, dass dieses Ähnlichkeitsmaß der produktbasierten Pearson-Korrelation bezüglich der Prognose von Bewertungen überlegen sein kann [109].

Selektion der Nachbarschaft

Die Anzahl der nächsten Nachbarn sowie das Verfahren, nach dem diese ausgewählt werden, haben ebenfalls einen wesentlichen Einfluss auf die Empfehlungsqualität. Um den Speicherbedarf zu reduzieren und den Empfehlungsprozess schneller zu gestalten, kann v.a. in Empfehlungssystemen, die mit großen Mengen an Daten arbeiten, die Anzahl der Nachbarn durch einen der folgenden Pre-Filter-Prozesse reduziert werden:

- **Top-N:** Für jeden Kunden bzw. für jedes Produkt werden lediglich die *N* nächsten Nachbarn samt ihrer Ähnlichkeitsgewichte gespeichert. Die Wahl von *N* muss jedoch sorgfältig geschehen, da ihre Größe Auswirkungen auf die Genauigkeit und Effizienz haben kann. Wird *N* zu groß gewählt, führt dies sowohl zu einem erhöhten Speicheraufwand, als auch zu einem langsameren Anwendungsprozess. Dagegen kann ein zu klein gewähltes *N* eine reduzierte Coverage zur Folge haben.

- **Schwellenwert:** Eine flexiblere Variante als das Wählen einer fixen Nachbarschaftsgröße bietet das Entfernen von jenen Nachbarn, die unter einem gegebenen Schwellenwert w_{min} liegen. So werden nur einflussreiche Nachbarn gespeichert. Die Wahl des optimalen Wertes für w_{min} erweist sich jedoch als sehr schwierig.

- **Negative Werte:** Das Speichern von ausschließlich positiven Ähnlichkeiten stellt eine weitere Variante dar. Positive Ähnlichkeitsgewichte indizieren, dass zwei Kunden oder Produkte zu einer gemeinsamen Gruppe gehören (z.B. Science Fiction Fans) und sind grundsätzlich zuverlässiger als negative Ähnlichkeiten. Diese deuten eher die Zugehörigkeit einer anderen Gruppe an.

Eine Kombination aus mehreren Varianten ist ebenfalls denkbar. Als Beispiel ist das Entfernen negativer Ähnlichkeiten in Kombination mit den Top-N Nachbarn zu nennen.
Grundsätzlich ist der Pre-Filter-Prozess jedoch von der Auswahl jener Nachbarn zu unterscheiden, die in die Prognose einfließen. Dieser erfolgt üblicherweise nach dem *k-nächste-Nachbarn-Verfahren* (Abk.: k-NN) [27]. Das k-NN-Verfahren gehört zu den am weitesten verbreiteten Methoden in kollaborativen Empfehlungssystemen und wird in nahezu jeder Studie dieses Forschungsgebietes beschrieben [2],[107],[109]. Es berücksichtigt zur Prognose lediglich die *k* Nachbarn mit den höchsten Ähnlichkeitsgewichten. Der Parameter *k* stellt einen kritischen Faktor für die Genauigkeit und Effizienz dar. Die Suche nach seiner optimalen Größe stellt die größte Herausforderung dieses Verfahrens dar. Wird er zu klein gewählt, erhöht

sich das Risiko, dass die Prognose sensitiv auf das Rauschen der Daten reagiert. Wird k zu groß gewählt, kann die Nachbarschaft zu viele Objekte aus anderen Gruppen enthalten. Somit ist die Funktion der Prognosegenauigkeit in Abhängigkeit von *k* für üblich konkav.
Das k-NN-Verfahren bringt einige Vorteile mit sich. Es ist einfach zu implementieren und generiert dennoch genaue Empfehlungen. Zudem eignet es sich für Verbesserungen von kollaborativen Methoden. Da es zur Klasse der *Lazy Learner* gehört und somit kein Modell erlernen muss, kann sich das System schnell den Änderungen der Bewertungsmatrix anpassen. Dies geht jedoch auf Kosten der Wiederberechnung der Ähnlichkeitsmatrix. Aus diesem Anlass entwickelten Amatriain et al. [7] ein Modell, welches einen reduzierten Satz von *Experten* zur Selektion der Nachbarn nutzen.

Mögliche Erweiterungen

Für speicherbasierte Methoden existieren Erweiterungen, um die in Abschnitt 3.2.2 beschriebenen Herausforderungen und Charakteristiken von kollaborativen Empfehlungssystemen anzugehen. Dieser Abschnitt gibt einen kurzen Überblick über verschiedene Verfahren und ihre Vorgehensweisen.

Default Voting

Zur Berechnung von Ähnlichkeiten werden Schnittbewertungen zwischen verschiedenen Kunden oder Produkten betrachtet. Diese Menge kann jedoch so klein sein, dass keine verlässlichen Ähnlichkeiten berechnet werden können. *Default Voting* geht dieses Problem an, indem sie künstlich eine höhere Bewertungsdichte erzeugt. Chee et al. [24] nehmen die Durchschnittsbewertung einer kleinen Gruppe, um die Kundenhistorie zu ergänzen. Breese et al. [16] verwenden neutrale oder eher negative Präferenzen, um fehlende Bewertungen zu ergänzen, und berechnen aus den resultierenden Bewertungsdaten die Ähnlichkeiten.

Inverse User Frequency

Die Idee hinter *Inverse User Frequency* [106] ist die, dass allgemein bekannte Produkte zur Ähnlichkeitsberechnung nicht so geeignet sind wie seltener gekaufte Artikel. Um diesem Umstand Rechnung zu tragen, wird die *inverse Häufigkeit* berechnet. Sie errechnet sich aus dem Logarithmus der Division aller n durch jene n_i Kunden, die Produkt i bewertet haben:

$$f_i = log\left(\frac{n}{n_i}\right) \tag{3.20}$$

Falls i von allen Kunden bewertet wurde, gilt $f_i = 0$. Um die nachbarschaftsbasierten Methoden wie üblich anzuwenden, werden die ursprünglichen Bewertungen mit dem Faktor f_i zu *transformierten Bewertungen* multipliziert [16].

Case Amplification

Case Amplification beinhaltet eine Umwandlung der in die Prognose einfließenden Ähnlichkeitsgewichte. Dabei werden durch folgende Formel hohe Gewichte weiter gestärkt und niedrige abgeschwächt [16]:

$$w'_{ij} = w_{ij} \times |w_{ij}|^{\rho-1} \tag{3.21}$$

ρ stellt einen Verstärkungsfaktor dar, für den $\rho \geq 1$ gilt und $\rho = 2{,}5$ einen üblichen Wert darstellt [77]. Dieses Verfahren reduziert das Rauschen in den Daten. Es begünstigt größere Werte, indem es kleine Gewichte so reduziert, dass diese letztlich vernachlässigbar sind.

Imputation verstärkte KF Algorithmen

Je spärlicher Bewertungsdaten sind, desto ungenauer werden kollaborative Empfehlungssysteme, die auf speicherbasierten Methoden beruhen. Su et al. [121],[122] entwickelten ein imputationsbasiertes KF-Verfahren (IBKF), welches zunächst zur Füllung der fehlenden Einträge eine Imputationstechnik nutzt. Sie führten umfangreiche Versuche mit unterschiedlichen Standard-Imputationstechniken, wie Durchschnittsimputation, lineare Regressionsimputation, Prognose-Durchschnittsanpassung [79] und mehrfache Bayessche Imputation [102], sowie mit Klassifikatoren aus dem Bereich maschinellen Lernens [122], wie Naïve Bayes, Support Vector Machines, neuronale Netze und Entscheidungsbäume durch. Letztlich kamen sie zum Ergebnis, dass ihr vorgeschlagener multipler IBKF-Algorithmus *IBCF-NBM* am effektivsten ist. Dieses IBKF-Verfahren stellt eine Mischung aus dem naïven Bayes Klassifikator und der Durchschnittssimputation dar. Ersteres Verfahren wird für dichtere Daten verwendet, während letzteres für ziemlich spärliche Daten zum Einsatz kommt. Ihre vorgeschlagene Methode liefert zudem ohne Hinzunahme externer Inhaltsdaten bessere Ergebnisse als der *content-boosted* KF-Algorithmus, der ein repräsentatives Verfahren hybrider Empfehlungssysteme darstellt.

3.2.4 Modellbasierte Ansätze

Nachbarschaftsbasierte Verfahren gehören zu den *instanzenbasierten Lernern* (auch *Lazy Learner* genannt), in denen kein erlerntes Modell, sondern die Trainingsinstanzen selbst Wissen darstellen. Modellbasierte Methoden hingegen erstellen mit Hilfe von überwachten oder unüberwachten maschinellen Lernverfahren im Voraus ein Datenmodell. Sie haben gegenüber den speicherbasierten Ansätzen folgende Vorteile:

1. **Speichereffizienz:** Während nachbarschaftsbasierte Methoden eine Speicherkomplexität von $O(m^2)$ bzw. $O(n^2)$ haben, ist die Größe der erlernten Modelle in modellbasierten Verfahren deutlich kleiner als die ursprüngliche Bewertungsmatrix. Ihre Speicheranforderungen sind daher oftmals niedriger.

2. **Trainingsgeschwindigkeit:** Modellbasierte Methoden sind für üblich in der Trainingsphase deutlich schneller als speicherbasierte, deren Zeitkomplexität ebenfalls im quadratischen Bereich liegt.

3. **Überanpassung:** Überanpassung ist ein Problem, das in vielen maschinellen Lernverfahren auftritt. Dabei werden Prognosen zu sehr durch zufällige Komponenten der Daten beeinflusst. Modellbasierte Ansätze mindern dieses Problem mit Hilfe von Regularisierungsfaktoren.

Obwohl nachbarschaftsbasierte Ansätze zu den ersten und bekanntesten Verfahren der kollaborativen Filter zählen, gehören sie nicht immer zu den genausten. Insb. Ansätze der *Matrixfaktorisierung* erwiesen sich in manchen Vergleichsstudien als Verfahren, die bezüglich der Empfehlungsqualität mit traditionellen KF-Methoden mehr als mithalten können [74]. Sie gehören inzwischen zu State-of-the-art und werden im weiteren Verlauf dieses Abschnitts sehr ausführlich behandelt. Weitere modellbasierte Ansätze sind *Bayes Klassifizierer*, *Clusterverfahren*, *regelbasierte Methoden*, *Entscheidungsbäume*, *Regressionsmodelle*, *Support Vector Machines* und *neuronale Netze*. Eine Übersicht über alle Verfahren würde den Rahmen dieser Arbeit sprengen, sodass lediglich die ersten drei Modelle vorgestellt werden.

Bayessche Kollaborativer Filter

Der einfache Bayesche KF-Algorithmus basiert auf dem *naïven Bayes* Ansatz. Er geht von der Annahme aus, dass die Features einer gegebenen Klasse unabhängig voneinander sind. Nachdem die Wahrscheinlichkeiten für die einzelnen Klassen bei gegebenen Features berechnet wurde, wird die Klasse mit der höchsten Wahrscheinlichkeit als Prognose ausgegeben [85]. Bei unvollständigen Daten wird die Berechnung mit den vorhandenen Daten o wie folgt durchgeführt:

$$class = \arg \max_{j \epsilon classSet} p(class_j) \prod_o P(X_o = x_o \mid class_j) \qquad (3.22)$$

Zudem wird der *Laplace-Schätzer* verwendet, um die Wahrscheinlichkeitsberechnung zu glätten und eine bedingte Wahrscheinlichkeit von 0 zu verhindern:

$$P(X_i = x_i \mid Y = y) = \frac{\#(X_i = x_i,\ Y = y) + 1}{\#(Y = y) + |X_i|} \qquad (3.23)$$

$|X_i|$ stellt hierbei die Größe des Klassesatzes $\{X_i\}$ dar. Der einfache Bayessche KF-Algorithmus kann auf binäre Daten wie in [84] und [85] angewandt werden. Ihre einfache Anwendbarkeit geht jedoch auf Kosten der Skalierbarkeit und des Verlustes von mehrklassigen Informationen. Es kann jedoch auch auf mehrwertigen Daten wie in [120] arbeiten. Su und Khoshgoftaar kamen in dieser Studie zum Ergebnis, dass der einfache Bayessche KF-Algorithmus gegenüber dem Pearson-Korrelation basierenden KF-Verfahren Vorteile bezüglich der Skalierbarkeit, jedoch Nachteile bezüglich der Genauigkeit aufweist. Aus diesem Grund wurden *Bayessche Netz*

Algorithmen entwickelt. In Verbindung mit einer *erweiterten logistischen Regressio*n, welche auf dem Gradientenverfahren [47] beruht, soll sie besser mit unvollständigen Daten umgehen. Sowohl in Kombination mit dem *tree augmented naïve Bayes* [38], als auch mit dem naïve Bayes lieferte diese Methode genauere Empfehlungen als der einfache Bayessche und der Pearson-Korrelation basierende KF-Algorithmus [120].

Cluster Kollaborativer Filter

Ein Cluster stellt eine Ansammlung von Datenobjekten dar. Datenobjekte desselben Clusters ähneln einander, während Objekte verschiedener Cluster unähnlich zueinander sind [50]. Das Ähnlichkeitsmaß zwischen Objekten wird durch Metriken, wie die *Minkowski-Distanz* und die *Pearson-Korrelation* berechnet. Die weit verbreitete Minkowski-Distanz ist definiert als

$$d(X,Y) = \sqrt[q]{\sum_{i=1}^{n} |x_i - y_i|^q}, \qquad (3.24)$$

wobei n die Anzahl der Dimensionen, x_i, y_i die Werte der i-ten Dimension der Objekte X und Y darstellen und $q \in \mathbb{N}$ gilt. Für $q = 1$ bildet d die *Mannhattan-Distanz*, für $q = 2$ die *Euklidische Distanz* [50].

Clustermethoden können in drei Kategorien eingeteilt werden: in *partitionierende*, *dichtebasierte* und *hierarchische* Verfahren [123]. Eine häufig genutzte Form des partitionierenden Ansatzes stellt das *k-means-Verfahren* von MacQueen [82] dar. Dieses Verfahren bildet aus einer Menge ähnlicher Objekte k Gruppen. Dabei werden die ersten k Elemente als Clusterzentren gewählt und jedes neue Element nacheinander jenem Cluster zugeordnet, das die geringste Varianzerhöhung erfährt. Das k-means-Verfahren ist einfach zu implementieren und effizient, weist jedoch erhebliche Nachteile auf. Zum einen sind die Lösungen stark von den gewählten Startobjekten und der im Voraus definierten Größe k abhängig. Dadurch dass jedes Objekt einem Cluster zugeordnet wird, kann dieses Verfahren zudem keine Ausreißer erkennen. Abhilfe kann hier ein dichtebasierter Ansatz wie der *DBSCAN-Algorithmus* [35] schaffen. Dieser ignoriert Rauschobjekte und unterscheidet drei Arten von Objekten. Zum einen sind hier *Kernobjekte* zu nennen, dessen Dichte den vorgegebenen Schwellenwert *minPts* überschreitet. Ihre Dichte ergibt sich aus der Anzahl Objekte in der Nachbarschaft mit dem Radius ε. *Dichte-erreichbare Objekte* sind keine Kernobjekte, haben jedoch welche in ihrer Nachbarschaft. Sie stellen somit das Randgebiet des Clusters dar. Objekte, die weder dicht sind noch in der Nachbarschaft eines dichten Objektes liegen, sind *Rauschobjekte*. Letztlich bilden durch dieselben Kernobjekte miteinander verbundene Objekte ein Cluster.

Hierarchische Methoden, wie *BIRCH* [134], führen eine hierarchische Zerlegung des Datensatzes nach bestimmten Kriterien durch. Dabei wird ein *Clustering-Feature-Tree* (Abk.: CF-Tree) verwendet, um multidimensionale Objekte inkrementell und dynamisch zu Clustern zusammenzufassen. Der BIRCH-Algorithmus durchläuft zwei Phasen: Zunächst werden alle

Daten durchsucht, um einen CF-Tree aufzubauen. Zu diesem Zeitpunkt befinden sich alle Clusterinformationen im Hauptspeicher. In der zweiten Phase wird die eigentliche Clusteranalyse auf Basis des CF-Trees durchgeführt.

In den meisten Fällen stellt das Clustern ein Zwischenschritt des Empfehlungsprozesses dar. Mithilfe der Cluster können darauf aufbauend weitere Analysen und Prozesse durchgeführt werden. Sarwar et al. [111] sowie O'Connor und Herlocker [26] nutzten sie, um speicherbasierte Methoden innerhalb der Cluster anzuwenden.

Das k-means-Verfahren mit $k = 2$ entspricht der *RecTree-Methode* von Chee et al. [24]. Dabei werden die ursprünglichen Bewertungsdaten durch rekursive Splits in zwei Sub-Cluster geteilt. Der resultierende *RecTree* ähnelt einem nicht balancierten Binärbaum. Seine Blätterknoten enthalten eine Ähnlichkeitsmatrix, während seine inneren Knoten *Bewertungszentroide* ihrer Sub-Bäume bilden. Die Prognose wird innerhalb jenes Blattes berechnet, in das der aktive Kunde gehört. Das Modell von Ungar und Foster [127] clustert Kunden und Produkte separat mit Varianten des k-means-Verfahrens und *Gibbs Sampling* [43]. Nutzer werden hierbei auf Basis ihrer Produktbewertungen und Artikel auf Basis der bewertenden Kunden geclustert. Jeder Kunde wird je nach Grad der Zugehörigkeit einer Klasse zugeordnet. Dieser ergibt sich aus der Ähnlichkeit des Kunden zum Durchschnitt der Klasse.

Insgesamt sind Clustermodelle besser zu skalieren als typische KF-Methoden. Sie erstellen Prognosen innerhalb einer Ansammlung von Datenobjekten, die deutlich kleiner sind als die gesamte Kunden- oder Produktbasis [24],[87],[107],[132]. Die komplexe und teure Clusterbildung läuft zudem offline. Ihr wesentlicher Nachteil liegt in der geringen Empfehlungsqualität. Diese kann durch die Erzeugung vieler feiner granularer Gruppen verbessert werden. Die Online-Klassifizierung in diese Segmente wäre dann jedoch ähnlich teuer wie das Finden ähnlicher Objekte bei speicherbasierten Methoden [78].

Regelbasierte Kollaborative Filter

Das Verfahren der Assoziationsregeln [4] wurde zunächst im Kontext der Auffindung von Mustern in Supermarktdaten entwickelt. Es wird üblicherweise auf binäre, manchmal aber auch auf kategorische und numerische Daten angewendet. Für die folgende Beschreibung wird von impliziten Bewertungen ausgegangen, welche in Supermarkttransaktionen häufig in Form von Käufen vorliegen. Dabei betrachten wir die Transaktionen $\mathcal{T} = \{T_1 \dots T_m\}$ sowie die Produkte $\mathcal{I} = \{I_1 \dots I_n\}$. Diesem Verfahren liegt die Idee zugrunde, dass stark miteinander korrelierende Artikel gesucht werden. Dazu werden die Maße *Support* und *Konfidenz* verwendet, welche die Assoziationsstärke zwischen Produkten quantifizieren. Der Support eines Produktsets $X \subseteq \mathcal{I}$ definiert den Anteil der X enthaltenden Transaktionen in $\mathcal{T}$ an der Gesamtmenge. Ein Produktset wird als *frequent* bezeichnet, wenn sein Support mindestens den vordefinierten Minimumwert s übersteigt. Eine *Assoziationsregel* hat die Form $X \rightarrow Y$ (z.B. $Milch, Windeln \rightarrow Bier$). Der linke Teil der Regel wird als *Body*, der rechte als *Head* bezeichnet. Vom Produktsetsupport zu unterscheiden ist der Support einer Assoziationsregel. Letzte-

rer definiert den Teil der Transaktionsmenge $\mathcal{T}$, der sowohl X als auch Y enthält. Die *Konfidenz* einer Regel definiert den Anteil der Y an der X enthaltenden Transaktionsmenge. Sie berechnet somit die (bedingte) Wahrscheinlichkeit mit der Kunden, die X bereits beschafften, auch Y kaufen. Auch für dieses Maß wird vorab ein minimaler Schwellenwert c definiert.
Das Ziel der Assoziationsregelanalyse ist es, jene Regeln zu finden, für die $Support \geq s$ und $Konfidenz \geq c$ gelten. Der *brute-force* Ansatz listet alle möglichen Assoziationsregeln auf, errechnet Support und Konfidenz und entfernt letztlich alle Regeln, die nicht beide Bedingungen erfüllen. Da dieser Ansatz zu rechenintensiv ist, wird eine Vorgehensweise mit zwei Schritten präferiert: Zunächst werden alle Produktsets ermittelt, welche die Supportbedingung erfüllen (engl.: *Frequent Itemset Generation*). Für diese werden dann unter der Konfidenzbedingung Regeln generiert (engl.: *Rule Generation*). Es gibt verschiedene Möglichkeiten, die Generierung frequenter Produktsets zu optimieren. Die Diskussion solcher Algorithmen würde den Rahmen dieser Arbeit sprengen, sodass für eine ausführliche Diskussion auf [4] verwiesen wird.
Vielmehr soll der Bezug zu kollaborativen Filtern aufgezeigt werden. Auch hier werden lediglich Assoziationsregeln mit Mindestsupport s und Mindestkonfidenz c berücksichtigt, jedoch mit dem Unterschied, dass diese genau ein Produkt im Head enthalten. Beide Parameter haben Einfluss auf die Genauigkeit und Effizienz von Empfehlungssystemen. Der resultierende Regelsatz stellt schließlich das Modell dar, welches zur Empfehlungsgenerierung verwendet wird. Dazu werden für jeden Kunden die *korrespondierenden Regeln* ermittelt. Darunter sind jene Regeln zu verstehen, deren Bodyprodukte Teil der Kundenhistorie sind. Diese werden dann für jeden Nutzer üblicherweise absteigend nach der Konfidenz sortiert, um letztlich die Top-N Produkte zu empfehlen. Dieser Ansatz stellt eine vereinfachte Form des Verfahrens in [109] dar.

Latente Variablenmodelle

Nachbarschaftsbasierte Ansätze haben den Vorteil, dass sie leicht nachvollziehbar sind. Allerdings ist die große Bewertungsmatrix meist sehr spärlich besetzt. Im Zuge des bereits erwähnten Netflix-Wettbewerbs haben latente Variablenmodelle an Popularität gewonnen und werden mittlerweile als State-of-the-art in Empfehlungssystemen berücksichtigt. Hierbei sind insb. Methoden der *Matrixfaktorisierung* (Abk.: MF) zu erwähnen, die im weiteren Verlauf der Arbeit eine wesentliche Rolle spielen. Andere Verfahren, wie *neuronale Netze* [104] oder *probabilistic Latent Semantic Analysis* [57] gehören ebenfalls zu dieser Familie, werden jedoch nicht näher erläutert.
Modelle dieser Art nehmen eine *Dimensionsreduktion* vor, indem sie die Bewertungsmatrix auf einen Raum niedrigerer Dimension abbilden. Merkmalausprägungen in diesem Raum werden dann verwendet, um Artikeleigenschaften und Nutzerpräferenzen abzugleichen.
Die Gemeinsamkeit aller KF-Methoden ist die Betrachtung von Korrelationen zwischen Objekten der Bewertungsmatrix. Während speicherbasierte KF-Methoden kundenweise oder pro-

duktweise Korrelationen verwenden, bieten MF-Verfahren einen Weg, alle Zeilen- und Spaltenkorrelationen gemeinsam zu berechnen und der gesamten Datenmatrix anzunähern. Dies ist der Grund dafür, warum latente Variablenmodelle so beliebt sind und zu den am weitesten verbreiteten Verfahren gehören. Angelehnt an die Studie von Koren et al. [74] beschäftige ich mich in den nächsten Abschnitten ausführlich mit den MF-Komponenten.

Grundprinzipien der Matrixfaktorisierung

Im Basismodell der Matrixfaktorisierung wird die $m \times n$ Bewertungsmatrix R näherungsweise in eine $m \times k$ Matrix P und eine $n \times k$ Matrix Q faktorisiert:

$$R \approx Q^T P \tag{3.25}$$

Jede Spalte von P bzw. Q wird als *latenter Vektor* oder *latente Komponente*, jede Zeile als *latenter Faktor* bezeichnet. Der *Kundenfaktorvektor* p_u in P enthält k Einträge, welche die Affinität des Kunden u zu den k *latenten Konzepten* der Bewertungsmatrix ausdrücken. Genauso beschreibt der *Produktfaktorvektor* q_i in Q die Affinität des Artikels i zu den k Konzepten. Anders formuliert repräsentiert ein Produktfaktor die Eigenschaften eines Produktes und der Kundenfaktor das Interesse an diesen Eigenschaften. Abbildung 3.2 illustriert ein Beispiel für ein Empfehlungssystem eines Filmanbieters. Die Filmgenres *History* und *Romance* stellen Konzepte dar. Produktfaktoren geben an, wie viel History und Romance in den Filmen steckt, Kundenfaktoren verdeutlichen hingegen, von welchem Ausmaß das Interesse an den einzelnen Genres geprägt ist.

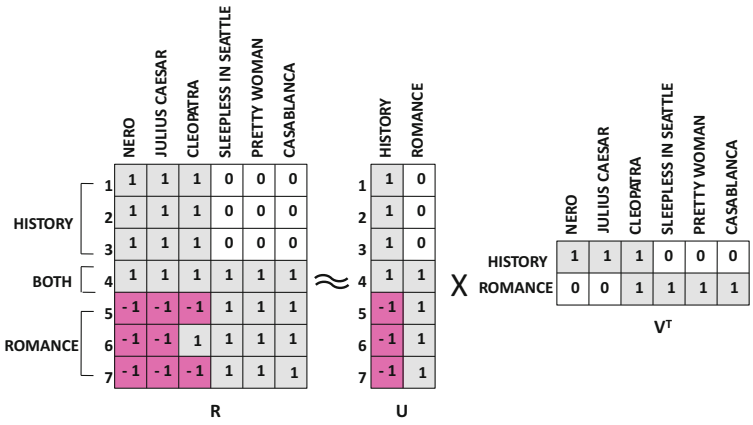

Abbildung 3.2: Beispiel einer zweidimensionalen Matrixfaktorisierung (Quelle: [3], S. 95)

Aus der Gleichung 3.25 folgt, dass jede Bewertung r_{ui} in R näherungsweise durch das Skalarprodukt des Kundenfaktors p_u und des Produktfaktors q_i berechnet wird:

$$\hat{r}_{ui} = q_i^T p_u \tag{3.26}$$

Ein latenter Faktor stellt einen Vektor dar, der positive und negative Werte enthält und oftmals nicht semantisch interpretierbar ist wie in Abbildung 3.1. Dennoch werden dominierende Korrelationsmuster der Bewertungsdaten aufgezeigt.

Die Anzahl der Dimensionen k stellt bei dimensionsreduzierenden Ansätzen einen kritischen Faktor dar. Dieser Wert sollte groß genug sein, um alle Muster in der Bewertungsmatrix aufzufassen, jedoch klein genug, um Überanpassung zu verhindern [108].
Die Schwierigkeit der Matrixfaktorisierung liegt in der Abbildung der Produkte und der Kunden auf ihre Faktorvektoren p^* und q^*. Diese werden erlernt, indem der regulierte, quadrierte Fehler in der Menge der bekannten Bewertungen minimiert wird:

$$\min_{q^*,p^*} \sum_{(u,i)\in\varkappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \qquad (3.27)$$

$\varkappa$ stellt die Menge der (u,i)-Paare dar, für die r_{ui} bekannt ist. Damit das Empfehlungssystem nicht zu sehr an vorab bekannte Bewertungen angepasst wird, sondern auch zukünftige, unbekannte prognostiziert, sollte Überanpassung an Trainingsdaten vermieden werden. Dazu werden die erlernten Parameter durch einen Term regularisiert, der zur Bestrafung großer Koeffizienten einen Bias erzeugt. Die nicht-negative Konstante λ kontrolliert den Grad der Regularisierung.

Lernverfahren

Es gibt zwei Methoden, um die Gleichung 3.27 zu minimieren: das *stochastische Gradientenabstiegsverfahren* (engl.: Statistical Gradiend Descent, Abk.: SGD) sowie die *Methode der alternierenden kleinsten Quadrate* (engl.: Alternating Least Squares, Abk.: ALS). Das SGD-Verfahren, welches von Funk vorgeschlagen [39] und mit großem Erfolg im Netflix-Wettbewerb eingesetzt wurde [73], prognostiziert für jeden Fall im Trainingsdatensatz die Bewertung r_{ui} und bestimmt den Fehler e_{ui}:

$$e_{ui} := r_{ui} - q_i^T p_u \qquad (3.28)$$

Danach modifiziert es die Parameter q_i und p_u mithilfe dieser Fehler um die Größenordnung γ in die entgegengesetzte Richtung des Gradienten:

$$q_i = q_i + \gamma(e_{ui} p_u - \lambda q_i) \qquad (3.29)$$

$$p_u = p_u + \gamma(e_{ui} q_i - \lambda p_u) \qquad (3.30)$$

Der Parameter γ kann als *Lernrate* gesehen werden. Sie gibt an, in welchen Schritten sich das Verfahren dem Minimum nähert. Meist wird ein kleiner konstanter Wert wie $\gamma = 0{,}005$ verwendet. Um lokale Minima zu vermeiden und den Annäherungsprozess schneller zu gestalten, kann γ nach jeder Iteration [11] oder je nach Faktorvektor [125] angepasst werden. Die optimalen Werte für die Parameter λ und γ werden üblicherweise durch Kreuzvalidierung bestimmt.
Das ALS-Verfahren deckt sich in weiten Teilen mit dem SGD-Algorithmus. Jedoch werden nicht beide Matrizen P und Q in einer Iteration modifiziert. Stattdessen werden Kunden- und Produktfaktoren abwechselnd aktualisiert, indem eine der beiden als fix angenommen wird.

Die Gleichung 3.27 stellt in diesem Fall ein *quadratisches Optimierungsproblem* mit exakt einem Minimum dar.

Das ALS- Verfahren hat gegenüber der SGD-Methode den Vorteil, dass es sich aufgrund der unabhängigen Berechnung der Faktorvektoren parallelisieren lässt. Dies gab Forschern Anlass zu einer massiven Parallelisierung dieses Verfahrens [135]. Zudem eignet sich eine gewichtete Variante des ALS-Verfahrens besser für spärliche, implizite Bewertungsdaten. In einem solchen Verfahren werden vorhandene Bewertungen höher gewichtet als fehlende. Da das SGD-Verfahren über jeden einzelnen Trainingsfall und somit auch über die Nullwerte iteriert, weist es eine hohe Komplexität auf. Das ALS-Verfahren kann solche Fälle effizienter lösen, wie die Studie von Hue et al. [60] zeigt. Sie entwickelten speziell für implizite Daten ein Verfahren, welches jedoch Schwächen bezüglich der Effizienz bei großen Mengen an expliziten Bewertungsdaten aufweist. Für diese hat der SGD-Algorithmus kürzere Laufzeiten.

Hinzufügen von Biases

Mithilfe der Gleichung 3.26 sollen jene Interaktionen zwischen Kunden und Produkten erfasst werden, die für unterschiedliche Bewertungen verantwortlich sind. Ein Teil dieser Unterschiede ist auf kunden- und produktspezifische Faktoren zurückzuführen, die als *Biases* bekannt sind. Sie umfasst bspw. die Fälle, in denen zwei Nutzer die Bewertungsskala verschieden interpretieren oder in denen manche Artikel generell besser als alle anderen bewertet werden, obwohl sie objektiv betrachtet nicht besser sind. Daher erscheint es nicht sinnvoll, die gesamte Bewertung r_{ui} durch $q_i^T p_u$ anzunähern. Vielmehr muss der Teil individueller Kunden- und Produktbiases identifiziert werden, um lediglich den *reinen* Teil der Beziehungen zu modellieren. Der Bias b definiert sich als

$$b_{ui} = \mu + b_i + b_u \tag{3.31}$$

mit μ als globaler Durchschnittsbewertung, b_i als Abweichung des Mittelwertes von Produkt i vom Mittelwert aller Produkte sowie mit b_u als Abweichung des Mittelwertes des Kunden u vom Mittelwert aller Kunden. Als Beispiel für erstere Bias ist ein Film zu nennen, der im Schnitt 0,5 Sterne besser bewertet wird als jeder andere. Letztere hingegen kann ein kritischer Nutzer sein, der immer 0,4 Sterne weniger vergibt als der durchschnittliche Kunde. Formal betrachtet erweitern Biases die Gleichung 3.26 wie folgt:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u. \tag{3.32}$$

Eine Bewertung besteht somit aus vier Komponenten: dem globalen Durchschnitt, der Produkt- und dem Kundenbias sowie der Kunde-Produkt-Korrelation. Das Modell minimiert nun den Fehler über die angepasste Funktion:

$$\min_{q^*,p^*} \sum_{(u,i)\in\varkappa} (r_{ui} - \mu - b_i - b_u + q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_i^2 + b_u^2) \tag{3.33}$$

Die Biases werden ähnlich zu den Faktorvektoren in die entgegengesetzte Richtung des Gradienten modifiziert:

$$b_i = b_i + \gamma(e_{ui} - \lambda b_i) \tag{3.34}$$

$$b_u = b_u + \gamma(e_{ui} - \lambda b_u) \tag{3.35}$$

Singularwertzerlegung

Die Singularwertzerlegung (engl.: Singular Value Decomposition, Abk.: SVD) stellt eine Form der Matrixfaktorisierung dar, bei der die Matrizen U und V orthogonal zueinander stehen. Die Bewertungsmatrix R wird wie folgt faktorisiert:

$$R \approx V_k^T \Sigma_k U_k \tag{3.36}$$

U_k, Σ_k und V_k sind Matrizen der Größe $m \times k$, $k \times k$ und $n \times k$. Die Matrizen U_k und V_k bestehen aus den k größten Eigenvektoren aus RR^T und R^TR, während die Diagonalmatrix Σ_k die nichtnegativen Quadratwurzeln der k größten Eigenwerte von einer der beiden Matrizen enthält. Die Matrix U_k mit den größten Eigenvektoren aus R^TR stellt die reduzierte Darstellung dar, die für die Dimensionsreduzierung des Zeilenraums benötigt wird. Aus Gleichung 3.36 ist ersichtlich, weshalb der SVD-Algorithmus grundsätzlich als ein Verfahren der Matrixfaktorisierung kategorisiert wird. Zwar erfolgt die Faktorisierung mithilfe von drei anstatt zwei Matrizen, jedoch können entweder die Kundenfaktoren U_k oder die Produktfaktoren V_k mit der Diagonalmatrix Σ_k zusammengeführt werden, sodass sich folgende Formeln ergeben:

$$P = U_k \Sigma_k \tag{3.37}$$

$$Q = V_k \tag{3.38}$$

Nun ist die Faktorisierung der Bewertungsmatrix R analog zu Gleichung 3.25 definiert. Der einzige Unterschied ist letztlich das Orthogonalitätskriterium, was bedeutet, dass dieselbe Zielfunktion über einen kleineren Lösungsraum optimiert wird.

Lösungen für implizite Bewertungsdaten

In Abschnitt 3.2.2 sind Probleme beleuchtet worden, die mit Eigenschaften von impliziten Bewertungsdaten verbunden sind. Einer dieser Herausforderungen betrifft den Umgang mit fehlenden Bewertungen. Diese werden für explizite Daten für üblich ignoriert. Für implizite Bewertungen hingegen hätte das Ignorieren unbekannter Meinungen, die zumeist negative Präferenzen implizieren, einen höheren Prognosefehler zur Folge. Dieses Phänomen ist ein Beispiel für Überanpassung mangels negativer Bewertungen. Das Ersetzen der fehlenden Einträge durch Nullwerte stellt eine Lösungsalternative dar, die jedoch Laufzeitprobleme verursachen würde. Ein weiteres Mittel, um die Genauigkeit zu verbessern, bietet ein *Ensemble-Ansatz*[4]. Dazu werden die fehlenden Werte der Bewertungsmatrix mehrere Male mit jeweils unterschiedlichen Samples von Nullen ersetzt. Die unterschiedlich ausfallenden Prognosen werden daraufhin gemittelt, um letztlich ein finales Ergebnis zu erlangen. Das Nutzen von

[4] Darunter ist im Allgemeinen die Aggregation der Prognosen mehrerer Modelle zu einer finalen Prognose zu verstehen.

Stichproben unterschiedlicher Größe bietet zudem einen Weg, negative Bewertungen anders als positive zu gewichten.

Eine weitere Lösung beinhaltet die Einführung von Gewichten in die Zielfunktion. Nullwerte ersetzen dabei die fehlenden Einträge. Diese werden jedoch geringer gewichtet, um zu vermeiden, dass sie eine zu dominante Rolle während des Optimierungsprozesses einnehmen. Die folgende Heuristik wird verwendet, um das Gewicht c_{ui} des Eintrages (u, i) zu berechnen [60]:

$$c_{ui} = 1 + \alpha r_{ui} \tag{3.39}$$

Ein modifiziertes SGD-Verfahren kann zwar mit solchen gewichteten Einträgen arbeiten, seine Laufzeit ist jedoch aufgrund der komplett gefüllten Bewertungsmatrix zu hoch. Das gewichtete ALS-Verfahren von Hu et al. [60] geht hingegen effizienter mit den vielen Nullwerten um. Sie integrierten das für implizite Daten konzipierte *Weighted Regularized Matrix Factorization* (Abk.: WRMF) Modell in ein Empfehlungssystem für TV-Sendungen. Da implizite Bewertungen keine exakten Präferenzen wiedergeben, führten sie unterschiedliche Konfidenzlevel c_{ui} ein, welche die entsprechenden Präferenzen schätzen (Formel 3.39). Solche Konfidenzwerte ergeben sich aus der Bewertung r_{ui} sowie einem konstanten Gewicht α, das bei jedem (u, i)-Eintrag aufaddiert wird.

Rendle et al. [96] gehen mit ihrem *Bayesian Personalized Ranking Matrix Factorization* (Abk.: BPRMF) Modell das Problem der fehlenden negativen Bewertungen an, indem sie Produktpaare als Trainingsdaten verwenden. Während in vorherigen Modellen einzelne Produktbewertungen betrachtet werden, wird in diesem Modell bezüglich der korrekten Rangfolge der Produktpaare optimiert.

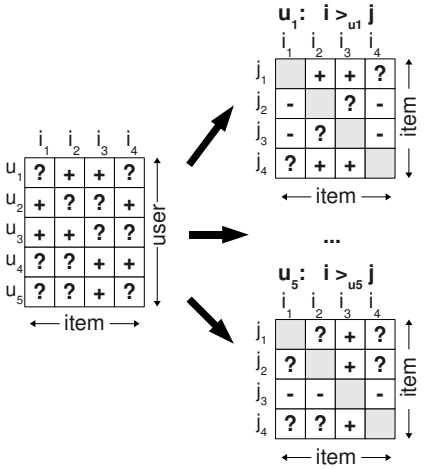

Abbildung 3.3: Modifikation der Trainingsdaten im BPRMF-Algorithmus (Quelle: [96], S. 3)

Abbildung 3.3 illustriert, wie der BPRMF-Algorithmus seine Trainingsdaten erzeugt: Für jeden Kunden u wird verglichen, ob für die Produkte i und j $i >_u j$ gilt. Diese Ungleichung gilt nur dann, wenn Kunde u für das Produkt i eine positive Bewertung und für j keine Bewertung aufweist (gekennzeichnet durch ein +). Es kann jedoch der umgekehrte Fall eintreten, sodass

die Trainingsdaten auch negative Bewertungen enthalten können (gekennzeichnet durch ein -). Sind beide Produktbewertungen bekannt oder unbekannt, kann keine Aussage über die Rangfolge gemacht werden, sodass der Eintrag leer bleibt (gekennzeichnet durch ein *?*). Ein weiterer Vorteil ist, dass die Rangfolge von Produktpaaren, die aus zwei unbekannten Produktbewertungen bestehen, exakt jene Paare darstellen, die in Zukunft rangiert werden müssen. Ihr vorgeschlagenes *LEARNBPR-Verfahren* stellt ein auf dem SGD-Verfahren beruhender Algorithmus dar, der das *BPR-OPT-Kriterium* optimiert. Das hierbei verwendete stellt bei großen Datenmengen gegenüber kompletten Zyklen einen weiteren Vorteil dar. Eine genauere Beschreibung beider Modelle entspricht nicht der Motivation dieses Abschnittes, sodass sie in einem späteren Abschnitt erfolgt.
Hidasi und Tikk [55] präsentierten ein Modell für Empfehlungssysteme mit impliziten Bewertungsdaten, das auf dem ALS-Verfahren beruht. Jedoch nahmen sie Änderungen zu den Grundprinzipien vor. Zum einen verwendeten sie vor dem Trainingsprozess die Saisonalität als Kontextdaten. Zum anderen führten sie einen neuen Algorithmus namens *Simfactor* ein, der eine Dimensionsreduzierung so durchführt, dass Ähnlichkeiten zwischen den Objekten auf effiziente Weise erhalten bleiben. Letztlich kommen sie zum Ergebnis, dass dieses Verfahren bessere Ergebnisse als andere Methoden der Matrixfaktorisierung liefert. Zudem stellt das Nutzen von Kontextdaten eine deutliche Verbesserung dar.

3.3 Inhaltsbasiertes Filtern

Inhaltsbasierte Systeme gehören neben den kollaborativen zu den am weitesten verbreiteten Arten von Empfehlungssystemen. Wie bereits erwähnt werden in der vorliegenden Arbeit lediglich KF-Methoden untersucht und evaluiert, weswegen dieser Abschnitt im Verhältnis zu vorherigem wesentlich kleiner ausfällt. Er umfasst die Beschreibung der generellen Funktionsweise sowie der Stärken und Schwächen solcher Systeme.
In inhaltbasierten Empfehlungsmethoden wird die Bewertung r_{ui} des Kunden u für das Produkt i aus seinen vorhandenen Bewertungen r_{uj} für jene Artikel $j \in \mathcal{N}_u(i)$ geschätzt, die i am ähnlichsten sind. Die Ähnlichkeit zweier Produkte berechnet sich jedoch nicht wie bei KF-Methoden über eine große Anzahl gemeinsamer Käufer, sondern über gemeinsame *Inhaltsdaten*. Bspw. versucht ein inhaltsbasiertes Empfehlungssystem für einen Filmanbieter, Ähnlichkeiten zu jenen Filmen zu berechnen, die von u in der Vergangenheit hoch bewertet wurden. Solche Ähnlichkeiten können bestimmte Schauspieler, Regisseure und/oder ein bestimmtes Genre oder Thema umfassen. Viele aktuelle inhaltsbasierte Systeme empfehlen Produkte mit *textuellen Informationen*, wie Dokumente, Webseiten oder Neuigkeiten. Um personalisierte Empfehlungen zu generieren, werden Kundenpräferenzen entweder explizit, bspw. durch Fragebögen, oder implizit, bspw. durch das Lernen von historischen Transaktionen, erhoben.
Zur Formalisierung dieses Ansatzes nehmen wir an, dass *prof(i)* ein Produktprofil darstellt, welches für Produkt i charakteristische Attribute enthält. Um seine Geeignetheit für Empfehlungen zu ermitteln, werden Inhalte, sog. *Schlüsselwörter*, aus i extrahiert. Als Beispiel ist die

inhaltsbasierte Komponente des *Fab-Systems* [9] zu nennen. Sie repräsentiert Webseiten durch ihre 100 informativsten Wörter. Der Informationsgehalt eines Wortes k_i im Dokument d_j wird durch das Gewicht w_{ij} ermittelt. Eines der bekanntesten Maße zur Messung dieser Gewichte ist der *term frequency/inverse document frequency* (Abk.: TF-IDF) [105]. Dazu nehmen wir an, dass die Anzahl aller Dokumente N beträgt, k_i in n_i von ihnen auftaucht und f_{ij} angibt, wie oft k_i in d_j auftaucht. Die *normalisierte Häufigkeit* dieses Schlüsselwortes TF_{ij} ist definiert als

$$TF_{ij} = \frac{f_{ij}}{\max_z f_{zj}}, \tag{3.40}$$

wobei das Maximum über die Häufigkeiten f_{zj} aller Schlüsselwörter k_z berechnet wird, die in d_j vorkommen. Häufig auftretende Wörter sind nicht hilfreich, um zwei Dokumente miteinander zu vergleichen. Aus diesem Grund wird die *inverse Dokumenthäufigkeit* IDF_i von k_i mit der Worthäufigkeit TF_{ij} kombiniert. Erstere berechnet sich wie folgt:

$$IDF_i = \log \frac{N}{n_i} \tag{3.41}$$

Das TF-IDF-Gewicht für Schlüsselwort k_i ist schließlich definiert als

$$w_{ij} = TF_{ij} \times IDF_i. \tag{3.42}$$

Das Profil $prof(j) = (w_{1j}, \dots, w_{kj})$ des Dokumentes d_j wird mithilfe der Gewichte ihrer Wörter repräsentiert. Um dem Nutzer u bisher unentdeckte Produkte empfehlen zu können, werden diese mit von ihm bewerteten Artikeln verglichen. Dabei wird aus den Profilen der bewerteten Produkte ein Kundenprofil $prof(u) = (w_{u1}, \dots, w_{uk})$ erstellt, in dem jedes Gewicht w_{ui} die Relevanz des Schlüsselwortes k_i für u definiert. Dieses Profil präsentiert die Präferenzen und Vorlieben eines Kunden und kann auf unterschiedliche Weise berechnet werden. Hier ist zum einen der *Rocchio-Algorithmus* [101] zu nennen. Dieser stellt ein *Relevance-Feedback-Verfahren* dar, bei dem Kunden ihre Empfehlungen bewerten können, um so eine ständige Verfeinerung ihres Profils zu erreichen. Des Weiteren hat sich der *Winnow-Algorithmus* [80] als ein geeignetes Verfahren für Fälle erwiesen, in denen Objekte viele unterschiedliche Eigenschaften haben.

Die Bewertungsfunktion r_{ui} in inhaltsbasierten Systemen wird definiert als

$$r_{ui} = score\big(prof(u), prof(i)\big), \tag{3.43}$$

wobei *score* eine Funktion darstellt, die aus den beiden Argumenten eine Bewertung erstellt. $prof(u)$ und $prof(i)$ können als Gewichtsvektoren $\vec{w}_u$ und $\vec{w}_i$ betrachtet werden, die Affinitäten zu einzelnen Schlüsselwörtern enthalten. Heuristische Methoden werden verwendet, um die Ähnlichkeit zwischen diesen Vektoren zu bestimmen. Wie in kollaborativen Systemen ist die Kosinus-Ähnlichkeit eine der bekanntesten Heuristiken:

$$r_{ui} = \cos(\vec{w}_u, \vec{w}_i) = \frac{\vec{w}_u \cdot \vec{w}_i}{\|\vec{w}_u\|_2 \times \|\vec{w}_i\|_2} = \frac{\sum_{x=1}^{K} w_{xu} \cdot w_{xi}}{\sqrt{\sum_{x=1}^{K} w_{xu}^2} \sqrt{\sum_{x=1}^{K} w_{xi}^2}} \quad (3.44)$$

Liest bspw. ein Kunde u oft Artikel über Themen aus der Bioinformatik, werden ihm weitere Artikel aus der Bioinformatik empfohlen, da solche Beiträge mehr themenspezifische Begriffe, wie z.B. *Genom*, *Sequenzierung* oder *Proteomik*, als andere enthalten und der Vektor $\vec{w}_u$ hohe Gewichten für diese Wörter enthält. Weitere Techniken, denen jedoch kein heuristischer Ansatz zugrunde liegt, sind *Bayessche Klassifikatoren* und zahlreiche Methoden aus dem Bereich maschinellen Lernen, wie etwa *Clustering*, *Entscheidungsbäume* und *künstliche Intelligenz*. Pazzani und Billsus [92] verwendeten zur Bewertung von Webseiten den naiven Bayes Ansatz. Die Seiten werden dazu in Klassen C_i eingeteilt, die angeben, ob sie relevant sind oder nicht. Diese Klasseneinteilung wird realisiert, indem Wahrscheinlichkeiten berechnet werden, zu welcher Klasse die Webseite p_j mit ihren Schlüsselwörtern $k_{1j}, \dots k_{nj}$ gehört. Aufgrund der *naiven* Annahme, dass die Wörter einer Seite unabhängig voneinander sind, ergibt sich folgende Gleichung:

$$P(C_i \mid k_{1j} \& \dots \& k_{nj}) = P(C_i) \prod_x P(k_{xj} \mid C_i) \quad (3.45)$$

$P(k_{xj} \mid C_i)$ und $P(C_i)$ können aus den zugrunde liegenden Trainingsdaten bestimmt werden. Für jede Webseite p_j und jede Klasse C_i werden die Wahrscheinlichkeiten analog zu Gleichung 3.45 berechnet, um sie jeweils der Klasse mit der höchsten Wahrscheinlichkeit zuzuordnen.

Stärken und Schwächen

Inhaltsbasierte Empfehlungssysteme haben gegenüber kollaborativen einen wesentlichen Vorteil: Sie können auch jene Produkte empfehlen, für die bislang noch keine Bewertungen vorliegen. Somit sind solche Systeme bezüglich neuer Artikel nicht vom Kaltstartproblem betroffen. Weiterhin ist der Empfehlungsprozess für einen Nutzer unabhängig von anderen Kunden. Es besteht also nicht wie bei KF-Methoden die Gefahr, dass mangels Nutzern mit ähnlichem Geschmack die entfernte Nachbarschaft zur Generierung von Empfehlungen herangezogen werden muss. Entscheidend sind vielmehr die Präferenzen des betrachteten Kunden. Ein weiterer Vorteil solcher Systeme liegt darin, dass das Zustandekommen der Empfehlungen transparent und für den Kunden nachvollziehbar ist. Solange keine neuen Produkte hinzukommen und sich Produktbeschreibungen nicht ändern, sind die Empfehlungen darüber hinaus deterministisch.

Um für einen bestimmten Kunden interessante von uninteressanten Artikeln zu unterscheiden, werden trennungswirksame Merkmale benötigt. Dazu müssen genügend Informationen über das Produkt vorhanden sein. Die Wahl der relevanten Eigenschaften und des Analyseverfahrens setzt Domänenwissen voraus. Zur Suche ähnlicher Artikel ist es nicht immer ausreichend, jedes Produkt durch ein Termvektor zu repräsentieren. So können Produktbeschrei-

bungen in verschiedenen Sprachen eine Schwierigkeit darstellen. In diesem Fall müssen externe Wissensquellen herangezogen werden. Zudem sind nicht alle Bereiche für automatisierte Merkmalsextraktion geeignet, wie es bei textuellen Dokumenten der Fall ist. Dies betrifft insb. Multimediadaten, wie Bilder, Audio- und Video-Streams.
Ein wesentlicher Nachteil inhaltsbasierter Empfehlungssysteme liegt in der Überspezialisierung. Da Empfehlungen auf den Vorlieben der Kunden beruhen, beinhalten sie keine überraschenden Artikel. Dieses Serendipitätsproblem führt dazu, dass Nutzern keine neuartigen Produkte empfohlen werden, von denen sie zu dem Zeitpunkt nicht wussten, dass diese interessant sein könnten. Zu Lösungsansätzen dieses Problems gehören die Einführung von Zufall durch genetische Algorithmen [116] sowie das Verwerfen zu ähnlicher Produkte [15].
Das Kaltstartproblem für Neukunden besteht auch in inhaltsbasierten Empfehlungssystemen. Um die Wahrscheinlichkeit ungenauer Empfehlungen zu verringern, muss ein Nutzer daher eine gewisse Anzahl von Produkten bewerten, bis das System signifikante Kundenpräferenzen ermitteln kann.

3.4 Wissensbasiertes Filtern

Wissensbasierte Empfehlungssysteme werden häufig im Kontext selten gekaufter Produkte eingesetzt. Beispiele für solche Produkte umfassen Immobilien, Automobile, Finanzdienstleistungen oder teure Luxusgüter. Da die wenigen Käufe auch noch unterschiedliche Bewertungen erzeugen, ist es schwer, eine ausreichende Zahl von Bewertungen für ein bestimmtes Meinungsmuster über ein Produkt zu erlangen. Diese Problematik besteht ebenfalls beim Kaltstartphänomen. In manchen Fällen entwickeln sich Kundenpräferenzen in Bezug auf diese Produkte mit der Zeit. In den meisten Fällen ist es jedoch sehr schwer, signifikante Kundenpräferenzen aus den historischen Bewertungsdaten zu gewinnen.
Kunden interessieren sich für solche Produkte nur dann, wenn sie bestimmte Eigenschaften besitzen. Autos haben bspw. verschiedene Hersteller, Modelle, Farben und Innenausstattungsoptionen, weshalb Kundeninteressen an einer bestimmten Kombination dieser Optionen geknüpft sind. Für solche Fälle, in denen sich die Produktdomäne mit ihren vielen Eigenschaften komplex darstellt und es aufgrund der vielen Kombinationen schwer ist, Bewertungen miteinander zu vergleichen, eignen sich wissensbasierte Empfehlungssysteme mit ihren Wissensdatenbanken, die Regel- und Ähnlichkeitsinformationen enthalten.
Solche Systeme werden in *beschränkungsbasiert* und *fallbasiert* kategorisiert. In beschränkungsbasierten Empfehlungssystemen [36] schränken Kunden explizit bestimmte Produkteigenschaften ein. Regeln stellen domainspezifisches Wissen dar und werden verwendet, um Kundenanforderungen bestimmten Produkteigenschaften zuzuordnen. Zudem produzieren beschränkungsbasierte Systeme oft Regeln zum Verhältnis von Kunden- zu Produktmerkmalen, wie z.B. „Ältere Investoren investieren nicht in sehr risikoreiche Produkte“. Der Suchprozess wird von Nutzern dahingehend beeinflusst, dass sie ihre Einschränkungen hoch- oder runterschrauben können, um weniger oder mehr Ergebnisse zu erhalten.

In fallbasierten Empfehlungssystemen [17] werden bestimmte Fälle oder Beispiele vom Kunden spezifiziert. Diese dienen für das System als Ziel oder Anknüpfungspunkt späterer Kundenanforderungen. Dazu werden Ähnlichkeitsmaße zwischen Produkteigenschaften definiert. Die angezeigten Ergebnisse werden meist als neue Zielfälle verwendet, die vom Nutzer jedoch je nach Vorstellung modifiziert werden können. Dieser interaktive Prozess führt den Kunden letztendlich zu seinem Wunschprodukt hin.

Wissensbasierte Empfehlungssysteme nutzen genauso wie inhaltsbasierte Systeme Produktattribute zur Empfehlungsgenerierung. Aus diesem Grund weisen sie ähnliche Nachteile auf. Auch in wissensbasierten Systemen kommt es durch die Nutzung von Schnittbewertungen zum Serendipitätsproblem. Der wesentliche Unterschied zwischen beiden liegt in ihrem Input: Inhaltsbasierte Systeme lernen von vergangenen Kundentransaktionen, während wissensbasierte mit Präferenzen aus aktiven Kundenspezifikationen arbeiten. Tabelle 3.3 fasst die Unterschiede der bisher vorgestellten Arten von Empfehlungssystemen nochmals zusammen.

Ansatz	Ziel	Input
Kollaborativ	Generiert Empfehlungen auf Basis von Bewertungen ähnlicher Kunden oder eigener Bewertungen ähnlicher Produkte	Bewertungen
Inhaltsbasierend	Generiert Empfehlungen auf Basis des Inhaltes/der Attribute von in der Vergangenheit präferierten Produkten	Bewertungen + Produktattribute
Wissensbasierend	Generiert Empfehlungen auf Basis der expliziten Spezifikationen der Attribute	Einschränkungen + Produktattribute + Domänewissen

Tabelle 3.3: Ziele der verschiedenen Arten von Empfehlungssystemen

3.5 Demographisches Filtern

Demografische Empfehlungssysteme empfehlen Produkte basierend auf soziodemographischen Kundendaten. Bspw. empfiehlt das System *Grundy* [100] Bücher auf Basis von Bibliotheken manuell zusammengestellter Stereotypen. Diese Daten werden über interaktive Dialoge gesammelt. In [91] werden Webseiten auf Basis vom demographischen Kundeneigenschaften empfohlen. In vielen Fällen werden demographische Informationen mit zusätzlichem *Kontext* kombiniert. Dieser Ansatz bezieht sich jedoch auf *kontextsensitive Empfehlungssysteme* und wird hier nicht weiter ausgeführt.

Die meisten aktuellen Verfahren verwenden Klassifikatoren. Eine bekannte Technik versucht, Kundeneigenschaften aus Homepages zu extrahieren, um zu berechnen, wie hoch die Wahrscheinlichkeit ist, dass einem Nutzer ein bestimmtes Restaurant gefällt. Regelbasierte Klassifikatoren, die wir bereits aus kollaborativen Filtern kennen, werden oft verwendet, um demographische Informationen in Verhältnis zum Kaufverhalten zu setzen.

Obwohl demographische Empfehlungssysteme auf Stand-alone Basis für gewöhnlich nicht die besten Ergebnisse liefern, verbessern sie andere Systeme als Teil einer hybriden Lösung. So werden sie aufgrund ihres Robustheitsfaktors mit wissensbasierten Systemen kombiniert.

3.6 Hybrides Filtern

In den vorherigen Abschnitten ist schon mehrfach angeklungen, dass es zur Erlangung besserer Ergebnisse von Vorteil sein kann, unterschiedliche Ansätze miteinander zu kombinieren. So hat sich im Laufe der Jahre eine Reihe von hybriden Ansätzen entwickelt, die in den meisten Fällen versuchen, Schwächen eines einzelnen Systems zu beheben. In einem Übersichtsartikel nennt Burke [21] sieben Möglichkeiten, unterschiedliche Empfehlungskomponenten miteinander zu kombinieren:

1. **Gewichtung:** Jede der Komponenten berechnet separat eine Reihe von Empfehlungen. Die Prognosen werden dann mit unterschiedlicher Gewichtung miteinander kombiniert, um ein endgültiges Ergebnis zu erhalten.

2. **Umschaltung:** Das hybride System wählt eine der Komponenten aus und gibt ihre Empfehlungen aus. Um Verbesserungen im Vergleich zur Verwendung eines einzelnen Verfahrens zu erzielen, ist ein geeignetes Auswahlverwahren erforderlich, das entweder externe Daten oder Maße aus Komponenten (z.B. p-Werte) zur Entscheidung heranzieht.

3. **Mischung:** Die Empfehlungen mehrerer Verfahren werden miteinander vermischt und dem Kunden präsentiert.

4. **Merkmalskombination:** Merkmale aus verschiedenen Quellen werden kombiniert und dann von einem einzelnen Empfehlungsalgorithmus bearbeitet. Ein inhaltsbasiertes Empfehlungssystem erhält als zusätzliche Eingabe Informationen wie „Kunde u und v mögen Produkt i“, die üblicherweise von kollaborativen Filtern verarbeitet werden. Es arbeitet mit diesen Daten auf dieselbe Weise wie mit allen anderen Eingaben.

5. **Merkmalserweiterung:** Um die Eingabedaten zu vervollständigen oder anzureichern, wird eines der Verfahren dazu verwendet, neue oder ergänzende Merkmale zu berechnen. Diese Daten erweitern die ursprüngliche Eingabe für die im nächsten Schritt verwendete Empfehlungstechnik.

6. **Kaskade:** Die Empfehlungsverfahren stehen in einer festen Hierarchie. Weiter unten stehende Verfahren kommen nur dann zum Einsatz, wenn höher priorisierte Methoden keine eindeutige Empfehlungsrangfolge generieren konnten.

7. **Meta-Level:** Das erste Verfahren erzeugt ein Modell, das dann als Eingabe für das nächste Verfahren dient. Burke bezeichnet diese Vorgehensweise als „*change of basis in the recommendation space*“, weil das zweite Verfahren nicht auf den Ausgangsdaten arbeitet.

Alle genannten Ansätze wurden bereits in Studien zu Empfehlungssystemen verwendet. Das *P-Tango-System* [25] nutzt einen gewichteten hybriden Ansatz in Form einer Linearkombina-

tion von numerischen *Scores*. Es initialisiert kollaborative und inhaltsbasierte Empfehlungen zunächst mit gleichen Gewichten, adjustiert diese dann jedoch schrittweise. Der Vorteil dieses Ansatzes ist der, dass die Potenziale aller Systeme auf einfache Weise ausgeschöpft werden. Jedoch wird implizit angenommen, dass der relative Wert verschiedener Verfahren mehr oder weniger gleichverteilt auf die Produktmenge ist. Wie bereits angeklungen, ist dies jedoch nicht immer der Fall. So weisen kollaborative Empfehlungsmethoden Schwächen bei Artikeln mit kleiner Anzahl an Bewertungen auf.

Ein umschaltendes hybrides Empfehlungssystem verwendet Kriterien, um zwischen verschiedenen Empfehlungsalgorithmen zu wechseln. Das System *Daily Learner* [15] nutzt einen hybriden Ansatz aus inhaltsbasiertem und kollaborativem Ansatz. Es führt zunächst eine Empfehlungsmethode ersterer aus. Erzeugt diese keine ausreichende Sicherheit, wird eine kollaborative Methode ausgeführt. Die kollaborative Komponente in einem hybriden System bietet eher die Fähigkeit, Gruppen semantisch ähnlicher Produkte zu überqueren und dennoch relevante Empfehlungen auszugeben (Serendipität). Die Schwierigkeit eines solchen Systems liegt darin, Umschaltkriterien festzulegen. Andererseits haben sie den Vorteil, flexibel mit Stärken und Schwächen der zugrunde liegenden Verfahren umgehen zu können.

Gemischte hybride Empfehlungssysteme geben Empfehlungen verschiedener Methoden aus. Das *PTV* System [118], welches ein TV-Programm zusammenstellt, basiert auf diesem Ansatz. Es wendet inhaltsbasierte Verfahren auf textuelle Beschreibungen der TV-Sendungen und KF-Methoden auf Kundenpräferenzen an. Die inhaltsbasierte Komponente generiert verlässliche Empfehlungen für neue Produkte und behebt somit das Kaltstartproblem. Auch bei diesem Ansatz sorgt die kollaborative Komponente für Empfehlungen von Nischenprodukten. Wenn ähnlich zum PTV System viele Empfehlungen benötigt werden, muss das System dementsprechend ausreichend Vorschläge von den unterschiedlichen Methoden erhalten. Dabei kann es zu Konflikten zwischen Empfehlungen für dasselbe Produkt führen, die mithilfe von Kombinationstechniken gelöst werden.

Das Nutzen kollaborativer Informationen als zusätzliche Eingabe für inhaltsbasierte Methoden stellt eine weitere Variante dar, beide Systeme miteinander zu kombinieren. Basu et al. [10] entwickelten den Regellerner *Ripper*, um Filmempfehlungen aus Kundenbewertungen und Inhaltseigenschaften zu generieren. Sie erzielten dabei wesentliche Verbesserungen bezüglich der Genauigkeit gegenüber der rein kollaborativen Variante. Ein solcher merkmalkombinierender hybrider Ansatz hat den Vorteil, dass das System nicht ausschließlich auf kollaborativen Daten beruht und somit nicht von der Anzahl der Kundenbewertungen für ein Produkt abhängig ist.

Bei merkmalerweiternden hybriden Empfehlungssystemen produziert eine Methode Daten, die als Eingabe einer anderen Methode verwendet werden. Als Beispiel ist das System *Libra* [86] zu nennen, welches mithilfe des naïven Bayes Klassifikators inhaltsbasierte Buchempfehlungen generiert. Die zugrunde liegenden Daten beinhalten textuelle Informationen über ähnliche Autoren und Titel und stammen aus den kollaborativen Systemen von Amazon. Ein sol-

ches System erweist sich deswegen als attraktiv, weil es einen Weg bietet, die Effizienz eines einzelnen Systems zu verbessern, ohne es zu modifizieren. Es unterscheidet sich von der merkmalskombinierenden Variante in dem Punkt, dass die erste Methode die Daten erweitert und nicht Rohdaten aus verschiedenen Quellen miteinander kombiniert werden.

Kaskadierte hybride Empfehlungssysteme arbeiten in einem stufenweisen Prozess. Ein Verfahren erstellt eine Rangfolge von kandidierenden Produkten, welche von einer zweiten verfeinert wird. Der Restaurantempfehler *EntreeC* [21] stellt ein kaskadiertes hybrides System dar und besteht aus einer wissensbasierten und einer kollaborativen Komponente. Es nutzt Informationen über Restaurants, um auf Kundeninteressen basierende Empfehlungen zu generieren. Diese Empfehlungen werden zunächst in *Buckets* gleicher Präferenzen platziert, um sie dann durch KF-Methoden weiter zu rangieren. Ein solches System verhindert, dass das nachrangige, zweite Verfahren auf jene Produkte angewandt wird, die bereits eindeutig durch die präferierte, erste Methode rangiert wurden. Da der zweite Schritt somit lediglich für untersuchungsbedürftige Artikel gedacht ist, müssten kaskadierte hybride Systeme effizienter sein als gewichtete hybride, wo alle eingesetzten Verfahren Einfluss auf die Prognose haben. Des Weiteren ist ersterer Ansatz unempfindlich gegen Rauschen in der Anwendung der nachrangigen Methode, da zunächst generierte Rangfolgen lediglich verfeinert, jedoch nicht aufgehoben werden.

Während beim merkmalserweiternden Ansatz ein erlerntes Model dazu genutzt wird, Merkmale zu generieren, die als Input für den zweiten Algorithmus dienen, stellt beim Meta-Level basierenden hybriden Ansatz das Modell selbst die Eingabe für das nächste Verfahren dar. Der auf KF-Methoden basierende Webseitenempfehler *Fab* [8] gehört zu dieser Kategorie hybrider Empfehlungssysteme. Er nutzt zur Berechnung der Ähnlichkeit zweier Kunden inhaltsbasierte Kundenprofile statt gemeinsam bewertete Produkte. Dieser Ansatz hat den Vorteil, dass das Spärlichkeitsproblem vermindert wird. Das erlernte Modell gleicht einer komprimierten bzw. dichteren Darstellung der Kundenpräferenzen.

3.7 Zu evaluierende Empfehlungsalgorithmen

Wie bereits erwähnt liegt der Schwerpunkt der vorliegenden Arbeit auf der Untersuchung kollaborativer Empfehlungsalgorithmen mit impliziten Bewertungen. Nachdem die Grundprinzipien der bekanntesten Verfahren dieser Art Empfehlungssysteme in Abschnitt 3.2. bereits beleuchtet wurden, widme ich mich in diesem Abschnitt jenen Verfahren ausführlicher, die im weiteren Verlauf experimentell untersucht werden. Dabei werden ggf. manche Aspekte wiederholt, um einige, neue Aspekte zu formalisieren und zu erweitern. Zudem wird im Falle verschiedener Implementierungsalternativen die von mir gewählte Lösung beschrieben. Die Erläuterungen der unterschiedlichen Methoden erfolgen in diesem Abschnitt im Hinblick auf implizite Daten. Dazu wurden repräsentative Algorithmen aus den beiden Hauptkategorien kollaborativer Filter gewählt. Repräsentativ für den speicherbasierten Ansatz habe ich mich für das produktbasierte k-NN-Verfahren entschieden. Die Wahl der produkt- statt der kunden-

basierten Variante ist aufgrund der in Abschnitt 3.2.3 genannten Vorteile bezüglich der Genauigkeit und Effizienz bei geringerer Anzahl Produkte gefallen. Für den modellbasierten Ansatz werden zwei MF-Methoden sowie ein regelbasiertes Verfahren analysiert: Repräsentativ für die Matrixfaktorisierung wählte ich die bereits in Abschnitt 3.2.4 kurz angerissenen Verfahren *Weighted Regularized Matrix Factorization* sowie *Bayesian Personalized Ranking Matrix Factorization*. Diese stellen zwei grundverschiedene Methoden dar, die beide jedoch auf dem SVD-Verfahren basieren und speziell für implizite Daten entwickelt wurden. Bei der Assoziationsregelanalyse fiel die Wahl auf das Verfahren der *bigram rules* aus [107]. Sie generierte in der Studie von Pradel et al. [94] mit Kaufdatensätzen die genausten Empfehlungen.

3.7.1 Assoziationsregelanalyse

Die Assoziationsregelanalyse stammt aus dem Data Mining und wird in der Regel zu Warenkorbanalysen verwendet. Dabei werden Produktassoziationen aus Kundenkäufen berechnet, weshalb diese Methode ebenfalls für Empfehlungssysteme sehr interessant ist. In ihrer einfachsten Form impliziert eine Assoziationsregel $i \rightarrow j$, dass ein Kunde, der Produkt i kauft, auch Produkt j mit einer bestimmten Wahrscheinlichkeit kauft. Konfidenz und Support messen die Stärke der Regel und werden wie folgt aus der Bewertungsmatrix P berechnet:

$$Konfidenz(i \rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum_{u \epsilon U} P_{ui}} \qquad (3.46) \qquad Support(i \rightarrow j) = \frac{P_{:i}^T P_{:j}}{\sum_{u \epsilon U} \sum_{i \epsilon I} P_{ui}} \qquad (3.47)$$

Anders formuliert misst die Konfidenz einer Regel die (bedingte) Wahrscheinlichkeit, dass j gekauft wird, gegeben, dass i gekauft wurde. Der Support misst jenen Anteil der Ereignisse, bei dem i und j gekauft wurden. Die Trainingsphase umfasst die Berechnung dieser beiden Interessantheitsmaße für alle in der Vergangenheit auftretenden Produktpaare. *Bigram Rules* beschränken sich hierbei auf einfache Assoziationsregeln, die sowohl im Body als auch im Head genau ein Produkt enthalten. Um Empfehlungen für den Kunden u zu generieren, werden zunächst alle mit seiner Kaufhistorie korrespondierenden Regeln identifiziert. Daraufhin werden die kandidierenden Produkte für üblich entweder nach der maximalen Konfidenz [107] oder der Summe der Konfidenzwerte [65] sortiert. Ich verwende in den Experimenten das Maximum aus der Multiplikation von Konfidenz und Support als Sortierkriterium. Der Rang eines Produktes j für den Kunden u mit der Transaktionshistorie I_u ist definiert als

$$\text{Rang}_u(\text{j}) = \max_{i \epsilon I_u} Konfidenz(i \rightarrow j) \times Support(i \rightarrow j) \qquad (3.48)$$

3.7.2 Produktbasiertes k-nächste-Nachbarn-Verfahren

Das produktbasierte k-NN-Verfahren berechnet Ähnlichkeiten zwischen Produkten auf Basis ihrer gemeinsamen Käufer. In der Trainingsphase steht die Berechnung heuristischer Ähnlichkeitsmaße im Vordergrund. In der Anwendungsphase wird für jedes Produkt i aus der Transaktionshistorie I_u eines Kunden u auf Grundlage der errechneten Gewichte Nachbarschaften $\mathcal{N}_u(i)$ aus den k ähnlichsten Produkten gebildet. Mehrere Ähnlichkeiten zum selben Produkt werden durch Summieren aggregiert. Somit lässt sich der Rang eines Produktes j für den Kunden u aus den Ähnlichkeitsgewichten w_{ij} analog zu Gleichung 3.6 berechnen:

$$Rang(j) = \sum_{\substack{i \in I_u \\ j \in \mathcal{N}_u(i)}} w_{ij} \qquad (3.49)$$

Die Genauigkeit und Effizienz dieser Empfehlungsmethode sind sowohl von der Größe der Nachbarschaft als auch vom gewählten Ähnlichkeitsmaß abhängig. Folgende Maße werden in den Experimenten untersucht: das *Kosinus-*, das *Bedingte Wahrscheinlichkeits-*, das *Jaccard-* sowie das *Log-Likelihood-Maß*.

Das Kosinus-Maß gehört zu den am häufigsten verwendeten Ähnlichkeitsmaßen in der Literatur [94],[107],[109]. Es ist einfach zu berechnen und liefert zudem gute Ergebnisse. Da es bereits im Abschnitt 3.2.3 vorgestellt wurde, sind hier keine weiteren Erläuterungen notwendig.

Ein weiteres Maß zur Berechnung von Produktähnlichkeiten ist das der bedingten Wahrscheinlichkeit von Deshpande und Karypis [31]. In ihrer Studie erzielte dieses Maß bessere Ergebnisse als die Kosinus-Ähnlichkeit. Es basiert auf der Wahrscheinlichkeit $P(j|i)$, dass ein Produkt j gekauft wird mit der Bedingung, dass Produkt i bereits gekauft wurde. Sie berechnet sich aus der Anzahl der Kunden, die sowohl i als auch j beschafften, dividiert durch die Anzahl der Kunden, die Produkt i beschafften. Diese Berechnung hat jedoch den Nachteil, dass häufig gekaufte Artikel dazu tendieren, ein sehr hohes Maß an Ähnlichkeit aufzuweisen. Um dieses Problem zu umgehen, wird die bedingte Wahrscheinlichkeit durch die mit α gewichtete Anzahl der j-Käufer dividiert:

$$sim_{Conditional}(i,j) = \frac{Freq(ij)}{Freq(i) \times \left(Freq(j)\right)^{\alpha}} \qquad (3.50)$$

Das Jaccard-Maß [63] kann ausschließlich für binäre Daten eingesetzt werden und vergleicht zur Messung der Ähnlichkeit die Kaufmuster zweier Produkte. Je höher die Anzahl der gemeinsamen Käufer ist, desto ähnlicher sind sich zwei Artikel. Diese Menge wird jedoch durch die Anzahl jener Käufer dividiert, die mindestens eins der beiden Produkte kauften. So wird erneut verhindert, dass Bestsellerartikel eine hohe Ähnlichkeit zu allen anderen Produkten aufweisen. Die Jaccard-Ähnlichkeit berechnet sich somit aus der Schnittmenge dividiert durch die Vereinigungsmenge zweier Produkte:

$$sim_{Jaccard} = \frac{|i \cap j|}{|i \cup j|} \tag{3.51}$$

Das letzte zu untersuchende Ähnlichkeitsmaß ist das Log-Likelihood-Maß von Dunning [32]. In der Studie von Owen et al. [88] erwies sich dieses Maß in KF-Methoden als erfolgsversprechende Variante. Es basiert auf Kombinationen zwischen dem Eintreten zweier Ereignisse A und B, die in Tabelle 3.4 zusammengefasst sind.

	Ereignis A	**Ereignis ¬A**
Ereignis B	A und B treten gemeinsam ein (k_{11})	B tritt ohne A ein (k_{12})
Ereignis ¬B	A tritt ohne B ein (k_{21})	Weder A noch B tritt ein (k_{22})

Tabelle 3.4: Ereigniskombinationen zur Berechnung des Log-Likelihood-Maßes

Das Log-Likelihood-Maß misst die *Unwahrscheinlichkeit*, dass die Überlappung zweier Produkte aus Zufall entstanden ist. Es berücksichtigt somit, ob und inwiefern eine *echte* Ähnlichkeit vorliegt. Die Formel zu ihrer Berechnung lautet

$$sim_{LogLikelihood} = 2\left(H\left[U_i, U_j\right] - H[U_i] - H\left[U_j\right]\right), \tag{3.52}$$

wobei H die Entropie von Shannon darstellt, die in der Informationstheorie ein Maß für den mittleren Informationsgehalt einer Nachricht ist. Sie wird wie folgt berechnet:

$$\begin{aligned} H\left[U_i, U_j\right] = {} & (k_{11} + k_{12} + k_{21} + k_{22}) \log k_{11} + k_{12} + k_{21} + k_{22} \\ & - \log k_{11} - \log k_{12} - \log k_{21} - \log k_{22} \end{aligned} \tag{3.53}$$

$$\begin{aligned} H[U_i] = {} & (k_{11} + k_{12}) \log k_{11} + k_{12} - \log k_{11} - \log k_{12} \\ & + (k_{21} + k_{22}) \log k_{21} + k_{22} - \log k_{21} - \log k_{22} \end{aligned} \tag{3.54}$$

$$\begin{aligned} H[U_i] = {} & (k_{11} + k_{21}) \log k_{11} + k_{21} - \log k_{11} - \log k_{21} \\ & + (k_{12} + k_{22}) \log k_{12} + k_{22} - \log k_{12} - \log k_{22} \end{aligned} \tag{3.55}$$

3.7.3 Gewichtete, regularisierte Matrixfaktorisierung

Wie bereits in Abschnitt 3.2.4 beschrieben, geht es bei der Matrixfaktorisierung darum, die Kunde-Produkt-Matrix P so in zwei Faktormatrizen W und H mit niedrigerem Rang zu zerlegen, dass ihr Skalarprodukt $\hat{P}_{ui} = W_{u:}H_{i:}^T$ das Interesse des Kunden u in Produkt i prognostiziert. Das WRMF-Modell gehört zu einer ihren gängigen Methoden [89]. Es basiert auf dem SVD-Verfahren und lässt sich als Problem der kleinsten Quadrate formulieren, welches aus zwei Hauptkomponenten besteht: Zum einem wird ein *Regularisierungsterm* zur Meidung von Überanpassung verwendet. Zum anderen berücksichtigt der WRMF-Algorithmus verschiedene *Konfidenzlevel*, die das Interesse eines Kunden in ein Produkt widerspiegeln. Die Idee ist dabei, dass der Konfidenzwert bekannter Bewertungen größer ist als der unbekannter. Eine Möglichkeit, diese Idee zu formalisieren, haben wir bereits in Abschnitt 3.2.4 (Formel 3.39)

beschrieben. Andere Möglichkeiten werden von Pan et al. [89] diskutiert, jedoch im weiteren Verlauf nicht berücksichtigt. Das WRMF-Optimierungskriterium lässt sich darstellen als

$$\min_{p^*,q^*} \sum_{(u,i)\in\varkappa} c_{ui}(r_{ui} - \mu - b_i - b_u + h_i^T w_u)^2 + \lambda(\|h_i\|^2 + \|w_u\|^2 + b_i^2 + b_u^2) \qquad (3.56)$$

wobei c_{ui} das Konfidenzlevel für Ereignis p_{ui} beschreibt, λ ein Regularisierungsparameter darstellt und $\|\cdot\|_F$ die Frobeniusnorm einer Matrix definiert. Letztere ist definiert als die Wurzel aus der Summe der Betragsquadrate aller Matrixelemente und berechnet sich für eine $(m\times n)$-Matrix $A\epsilon\mathbb{R}^{m\times n}$ wie folgt:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \qquad (3.57)$$

Die Faktormatrizen werden zunächst mit normal-verteilten, zufälligen Zahlen initialisiert. Dann werden sie abwechselnd mit folgenden Formeln aktualisiert [89]:

$$W_{u:} := P_{u:}C_{u:}H(H^T C_{u:}H + \lambda I)^{-1} \qquad (3.58)$$

$$H_{i:} := P_{:i}^T C_{:i}W(W^T C_{:i}W + \lambda I)^{-1} \qquad (3.59)$$

Der Vorteil des ALS-Verfahrens liegt darin, dass es einfach parallelisierbar und ihre Laufzeit linear zur Inputgröße ist [60].
Zusammenfassend und mit Hinblick auf die Experimente mit der verwendeten Open Source Bibliothek enthält der WRMF-Algorithmus folgende Parameter:

- **num_iter:** Anzahl Iterationen
- **num_factors:** Anzahl Faktoren
- **regularization:** Regularisierungsfaktor
- **alpha:** Konfidenz, die bei einem Kauf aufaddiert wird

3.7.4 Bayessche, personalisierte Ranking Matrixfaktorisierung

Das BPRMF-Modell, entwickelt von Rendle et al. [96], ist als Ranking-Problem formuliert worden, welches auf dem *Bayesian Personalized Ranking Optimierungskriterium* (Abk.: BPR-OPT) basiert. Anstatt wie der WRMF-Algorithmus ein einziges Produkt aus P im Scoring zu berücksichtigen, generiert der BPRMF-Algorithmus ein korrektes Ranking von Produktpaaren für jeden Kunden. Gegeben dem Tripel $\langle u, i, j\rangle$ ist die Grundidee hinter diesem Verfahren, dass wenn Kunde u Produkt i, jedoch nicht Produkt j konsumierte, $P_{uj} < P_{ui}$ gilt. In Abschnitt 3.2.4 wurde bereits erläutert, wie der BPRMF-Algorithmus seine Trainingsdaten kreiert.

Sei $\sigma(x) = 1/(1 + e^{-x})$ die *logistische Sigmoidfunktion*, $\hat{x}_{uij}$ eine beliebige Funktion, welche die Beziehungen in $\langle u, i, j \rangle$ spezifiziert, θ der Parametervektor eines beliebigen Modells und λ_θ ein modellspezifischer Regularisierungsparameter, stellt sich das allgemeine *BPR-OPT-Kriterium* wie folgt dar [96]:

$$BPR - OPT = \sum_{uij} \ln \sigma(\hat{x}_{uij}) - \lambda_\theta \|\theta\|^2 \qquad (3.60)$$

Im BPRMF-Verfahren erfolgt die Schätzung von $\hat{x}_{uij}$ durch ein MF-Modell. Da ein solches Modell im Allgemeinen lediglich einzelne Ereignisse und keine Paare prognostizieren kann, wird die Schätzung für das Ranking des Produktpaares i und j durch $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$ in einzelne Prognoseschritte zerlegt. Zur Optimierung des BPR-OPT-Kriteriums wurde das *LearnBPR-Verfahren* verwendet. Dieses auf *Bootstrapping*[5] basierende SGD-Verfahren arbeitet dabei mit der Aktualisierungsformel [96]

$$\theta := \theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1 + e^{-\hat{x}_{uij}}} \frac{\partial}{\partial \theta} \hat{x}_{uij} + \lambda_\theta \theta \right), \qquad (3.61)$$

bei der α für die Lernrate steht.
Folgende Parameter werden für den BPRMF-Algorithmus in der von uns eingesetzten Open Source Bibliothek verwendet:

- **num_iter:** Anzahl Iterationen
- **num_factors:** Anzahl Faktoren
- **learn_rate:** Lernrate für die Faktoraktualisierung
- **Reg_u:** Regularisierungsfaktor für u
- **Reg_i:** Regularisierungsfaktor für i
- **Reg_j:** Regularisierungsfaktor für j
- **BiasReg:** Regularisierungsfaktor für den Produktbias

[5] Bootstrapping ist eine Methode des Resampling, bei der wiederholt Statistiken auf der Grundlage lediglich einer Stichprobe berechnet werden. Zur Vertiefung dieser Methodik verweise ich auf [34].

4 Neighborhood based Added Value Supervized Discretization

Basierend auf der Forschung der bis hierhin vorgestellten Empfehlungsalgorithmen galt es ein auf implizite Bewertungen beruhendes Verfahren zu entwickeln. Implizite Daten umfassen im weiteren Verlauf Kundenklicks und -käufe. Das von mir als *Next Best Offer* (Abk.:NBO) benannte System, dem das eigen entwickelte Verfahren *Neighborhood based Added Value Supervized Discretization* (Abk.: NAVSD) zugrunde liegt, soll E-Commerce Unternehmen durch prognostizierte Kaufwahrscheinlichkeiten dabei unterstützen, ihre Kunden mit den richtigen Produkten anzusprechen. In diesem Kapitel wird dieser Algorithmus vorgestellt, indem ich insb. auf Unterschiede zu State-of-the-art Verfahren eingehe und erläutere, welche Vor- und Nachteile das NAVSD-Verfahren gegenüber diesen hat.

4.1 Idee und Motivation

Die Entwicklung des NAVSD-Verfahrens erfolgte unter folgenden Gesichtspunkten:

- **Genaue Empfehlungen in Echtzeit:** Gesucht wurde nach einem Verfahren, welches in Echtzeit möglichst genaue Empfehlungen generiert. MF-Methoden wurden wegen der vergleichsweise schlechten Laufzeit ausgeschlossen. Zudem kamen Forscher in der Mehrheit der Studien zu Empfehlungsalgorithmen auf Basis impliziter Bewertungsdaten zum Ergebnis, dass die einfachen Methoden, wie das Assoziationsregel- oder produktbasierte k-NN-Verfahren, bessere Ergebnisse liefern [5],[94].

- **Produktbasierte Assoziationen zwischen Vergangenheit und Gegenwart:** Aufgrund der geringeren Anzahl an Produkten wurde eine produktweise Berechnung vorgezogen. Der Fokus des NAVSD-Verfahrens liegt auf Assoziationen zwischen Produkten aus der Vergangenheit und der Gegenwart. Ziel ist es zu ermitteln, welche Auswirkungen bestimmte Produktkäufe oder -klicks aus der Vergangenheit auf Käufe in der Gegenwart haben. Aufgrund dieses Aufbaus erscheint es nicht sinnvoll Produktähnlichkeiten zu berechnen. Denn zwei Artikel sind sich dann ähnlich, wenn sie *innerhalb eines* Zeitraums und nicht in *zwei verschiedenen* Zeiträumen häufig von denselben Kunden gekauft werden.

- **Echte Wahrscheinlichkeiten statt Produktscores:** Das NAVSD-Verfahren stellt primär ein Algorithmus für Empfehlungssysteme dar. Zur Generierung und Sortierung von Empfehlungen genügen bei impliziten Bewertungen für üblich prognostizierte *Produktscores*. Dennoch sollen diese mithilfe eines Diskretisierungsverfahrens in echte Wahrscheinlichkeiten umgewandelt werden. Zum einen erwarte ich, dass der Diskretisierungsprozess einen Mehrwert gegenüber bestehenden Verfahren mit sich bringt. Zum anderen kann die Berechnung von Wahrscheinlichkeiten E-Commerce Unternehmen bei der Absatzplanung oder der Preisoptimierung unterstützen.

- **Problem der reduzierten Coverage:** Primäres Ziel war es, ein Verfahren zu entwickeln, welches den State-of-the-art Algorithmen bezüglich der Genauigkeit überlegen ist. Dennoch wurde bei der Entwicklung des NAVSD-Verfahrens ein weiteres Kriterium für Empfehlungsqualität berücksichtigt: die Coverage.

4.2 Trainingsphase

Der Trainingsprozess des NAVSD-Verfahrens besteht aus drei Komponenten: Diese umfassen ein von mir entwickeltes Maßberechnungsverfahren, ein Interessantheitsmaß sowie die Diskretisierung der aggregierten Interessantheitswerte. Erstere und letztere stellen ein Novum in der Forschung zu Empfehlungsalgorithmen dar.

Algorithmus 4.1 NAVSD-Trainingsprozess

Input: Trainingsbasisdatensatz $B\ \langle Kunde, Produkt\rangle$, Trainingszieldatensatz $Z\ \langle Kunde, Produkt\rangle$
Output: Interessantheitstabelle $I\ \langle Produkt\_A, Produkt\_B, Interessantheitswert\rangle$,
Gruppentabelle $G\ \langle Produkt, Min\_Score, Max\_Score, Wahrscheinlichkeit\rangle$

1: **function** Training(B, Z, k)
2: NAVSD_WeightComputation (B, Z) #Berechnet Interessantheitsmaße w_{ij}
3: #Output ist die Interessantheitstabelle I
4: **For** User $u \in \mathcal{U}_B$ #Menge der Kunden aus B
5: **For** Item $i \in \mathcal{I}_u^B$
6: $w_{ij} \leftarrow$ CorrespondedWeights(i, k, I) #Ermittelt korrespondierende Regeln
7: Add w_{ij} to $\mathcal{W}_u$
8: **For** $w_{xj} \in \mathcal{W}_u$ #für alle Werte mit demselben Zielprodukt
9: Aggregate w_{xj} to s_{uj}
10: Add s_{uj} to $\mathcal{S}_j$ #Menge der Scores für ein Zielprodukt
11: Add j to $\mathcal{J}_u$ #Menge der Zielprodukte für den Kunden u
12: **For** $j \in \mathcal{J}_u$
13: **if** $\langle u, j\rangle$ *in* Z **then**
14: $b_{uj} \leftarrow 1$
15: **else** $b_{uj} \leftarrow 0$
16: Add b_{uj} to $\mathcal{B}_j$ #Menge der Kaufvariablen für ein Zielprodukt
17: **For** $j \in \mathcal{I}_Z$ #Scores für Zielprodukte werden diskretisiert
18: Discretize$(\mathcal{S}_j, \mathcal{B}_j)$ #Output ist die Gruppentabelle G

Input des Trainingsprozesses stellen der Trainingsbasis- B und Trainingszieldatensatz Z dar. Während Z bspw. die Käufe der letzten 14 Tage beinhaltet, enthält B alle Klicks und Käufe der letzten 100 Tage vor Beginn des Trainingszielzeitraums. Die Trainingsphase kann in folgende Schritte unterteilt werden:

1. **Berechnung der Interessantheitswerte:** Zunächst werden die Interessantheitswerte aller Produktpaare berechnet. Wie dieser Schritt erfolgt und was die Unterschiede zur Standardtechnik sind, wird im nächsten Abschnitt erläutert. Ergebnis dieses Verfahrens ist letztlich die Tabelle I.

2. **Aggregation der korrespondierenden Interessantheitswerte:** Daraufhin werden für jeden Kunden u die korrespondierenden Interessantheitswerte w_{ij} ($Produkt\_A \rightarrow Produkt\_B$) aus I auf Basis seiner Transaktionen im Trainingsbasiszeitraum $\mathcal{I}_u^B$ ermittelt. Unterschiedliche Assoziationen zum selben Zielprodukt werden zu Produktscores s_{uj} summiert. Dabei werden für jedes Produkt lediglich die Artikel mit den k größten Interessantheitswerten berücksichtigt.

3. **Diskretisierung:** Das NAVSD-Verfahren wandelt die Scores durch einen Diskretisierungsprozess in Wahrscheinlichkeiten um. Die Wahrscheinlichkeitsberechnung der resultierenden Gruppen, bestehend aus Kunden und ihren Produktscores, erfolgt mithilfe der Kaufvariable b_{uj}. Diese gibt an, ob u das Produkt j im Zielzeitraum kaufte oder nicht. Die Diskretisierung erfolgt *produktweise*, d.h. es arbeitet nacheinander eine Liste von Scores unterschiedlicher Kunden für ein bestimmtes Produkt ab. Dabei nutzt es die Kaufinformation b_{uj} als Zielvariable, um statistisch signifikante Gruppen zu bilden.

Im Folgenden werden Motivation und Vorgehensweise der einzelnen Schritte ausführlich erläutert.

4.2.1 Trainingsverfahren

Das Assoziationsberechnungsverfahren des NAVSD-Algorithmus unterscheidet sich vom Standardtrainingsverfahren aus Abschnitt 3.2.3. Diese von Linden et al. [78] beschriebene Methodik betrachtet alle Produktkombinationen innerhalb eines vergangenen Zeitraums und berechnet ihre Interessantheits- oder Ähnlichkeitsmaße. Um die Laufzeit dieses Prozesses zu verkürzen, werden jedoch nur Produktkombinationen betrachtet, die bei mindestens einem Kunden auftreten.
Das *NAVSD-Trainingsverfahren* berechnet die Maße auf andere Weise. Es betrachtet Kombinationen zwischen Produkten aus der Vergangenheit bzw. aus dem Trainingsbasisdatensatz und der Gegenwart bzw. dem Trainingszieldatensatz und generiert letztlich Empfehlungen der Kategorie „Kunden, die Produkt A kauften/klickten, kaufen nun Produkt B“. Für jedes Produkt i_B aus dem Trainingsbasisdatensatz B wird geschaut, welche Nutzer $u \in \mathcal{U}_{i_B}$ es im Basiszeitraum kauften. Für jeden dieser Kunden wird dann ermittelt, welchen Artikel $j_Z \in \mathcal{I}_u$ sie im Zielzeitraum kauften. Der Zähler $n_{i_B j_Z}$ wird jedes Mal bei Auftreten der Kombination $\langle i_B, j_Z \rangle$ bei einem Kunden hochgezählt. Zudem wird das Zielprodukt der Liste $\mathcal{S}_{i_B}$ hinzugefügt, die aus allen Produkten j_Z mit einer Korrelation zu i_B besteht. Zum Schluss werden zwischen den Artikeln $j_Z \in \mathcal{S}_{i_B}$ und i_B mithilfe der Anzahl gemeinsamer Käufer $n_{i_B j_Z}$ die Interessantheitsgewichte $w_{i_B j_Z}$ berechnet. Das gewählte Interessantheitsmaß wird im nächsten Abschnitt erläutert.

Algorithmus 4.2 NAVSD-Trainingsverfahren

Input: Trainingsbasisdatensatz $B\ \langle Kunde, Produkt\rangle$, Trainingszieldatensatz $Z\ \langle Kunde, Produkt\rangle$
Output: Interessantheitsgewichte $w_{i_B j_Z}$ zwischen Produkt i_B und j_Z

1: **function** NAVSD_WeightComputation (B, Z)
2: **For** $i_B \in B$
3: **For** $u \in \mathcal{U}_{i_B}$
4: **For** $j_Z \in \mathcal{J}_u$
5: $n_{i_B j_Z} \leftarrow n_{i_B j_Z} + 1$
6: Add j_Z to $\mathcal{S}_{i_B}$
7: **For** $j_Z \in \mathcal{S}_{i_B}$
8: Compute $w_{i_B j_Z}$

Der Unterschied zwischen dem Standard- und dem NAVSD-Trainingsverfahren besteht darin, dass in ersterem Assoziationen zwischen zwei Produkten *innerhalb des Basiszeitraums*, während in letzterem Assoziationen zwischen Produkten *aus dem Basis- und dem Zielzeitraum* berechnet werden. Das Standardtrainingsverfahren ist eher in der Lage, Assoziationen zwischen zwei zeitlich nah aneinander liegenden Transaktionen zu finden. Es kann deswegen genauere Empfehlungen auf Basis aktueller Kundentransaktionen generieren. Das Trainingsverfahren des NAVSD-Algorithmus hingegen fokussiert eine langfristige Sicht auf die Kunden und ermittelt, welche Transaktionen aus der Vergangenheit mit Käufen aus der Gegenwart korrelieren. Somit wird durch diesen Aufbau ein gewisser Zeit- bzw. Trendeffekt berücksichtigt. Außerdem resultieren keine hohen Interessantheitswerte für jene Produkte, die innerhalb eines abgelaufenen Trends oder einer abgelaufenen Saison gekauft werden (z.B. Winterjacken, Halloweenkostüme), während das Standardtrainingsverfahren auch nach Ablauf des Trends oder der Saison solche Assoziationen hoch bewerten kann.
Ein weiterer Vorteil des NAVSD-Trainingsverfahrens besteht bezüglich der Serendipität. Während im Standardtrainingsverfahren eine große Anzahl gemeinsamer Käufe benötigt werden, um zwei Produkte als ähnlich zu klassifizieren, ist das NAVSD-Trainingsverfahren aufgrund der geringeren Anzahl an Assoziationen und des daraus resultierenden höheren Zufallsfaktors offener für neuartige Produkte. Der kurze Zielzeitraum und die kleine Assoziationszahl haben zudem den Vorteil, dass Laufzeit und Speicheraufwand geringer ausfallen. Zwar weist auch dieses Verfahren bei n Produkten eine Komplexität von $O(n^2)$ auf, jedoch ist diese aus eben genannten Gründen deutlich höher gegriffen als für das Standardtrainingsverfahren. Letztlich bleibt zu nennen, dass das NAVSD-Trainingsverfahren Wiederholungskäufe berücksichtigt.
Die berechneten Interessantheitswerte werden nach zwei Kriterien gefiltert: nach der *absoluten Häufigkeit* der Produktkombination und nach *Signifikanz der Unabhängigkeit*. Zum einen erscheint es nicht sinnvoll, Produktpaare zu berücksichtigen, die lediglich bei einem oder zwei Kunden auftreten. Diese Bedingung verschärft einerseits in geringem Ausmaß das Kaltstartproblem bei neuen Produkten. Andererseits soll es den Effekt verhindern, dass zufällig eingetretene Ereignisse bei selten gekauften Produkten ein hohes Interessantheitsmaß verursachen. Der Schwellenwert liegt in diesem Fall bei einer Anzahl von drei Kunden.

Das Interessantheitsmaß des NAVSD-Verfahrens, welches im nächsten Abschnitt vorgestellt wird, stellt ein Maß für die Abweichung von stochastischer Unabhängigkeit dar. Ob diese Abweichung statistisch signifikant ist, wird anhand eines Chi-Quadrat-Unabhängigkeitstests festgestellt [117], bei der die Prüfgröße durch die normalisierte Abweichung vom Erwartungswert definiert ist. Die Teststatistik

$$\mathcal{X}^2 = \sum_i \sum_j \frac{(n_{ij} - E(n_{ij}))^2}{E(n_{ij})} \tag{4.1}$$

ist annähernd $\mathcal{X}^2$-verteilt mit $2^l - l - 1$ Freiheitsgraden, wobei l für die Anzahl der Dimensionen steht. Der Wert dieser Prüfgröße korrespondiert je nach Anzahl der Freiheitsgerade (in unserem Fall 1) mit einem bestimmten p-Wert. Ist dieser p-Wert kleiner als das Signifikanzniveau $\alpha = 0{,}05$, kann die Nullhypothese H_0, dass beide Produkte stochastisch unabhängig voneinander sind, verworfen werden. Andernfalls wird die Assoziation entfernt und im weiteren Verlauf des Verfahrens nicht mehr berücksichtigt.

4.2.2 Interessantheitsmaß

Wie bereits dargestellt, wurde ein Maß gesucht, das den Einfluss eines historischen Kaufs oder Klicks auf einen gegenwärtigen Kauf berechnet. Dazu wurde zunächst die Konfidenz der Regel $A \rightarrow B$ herangezogen. Sie misst die bedingte Wahrscheinlichkeit von B gegeben A. Um dem Nachteil, der Vernachlässigung der Wahrscheinlichkeit von B, entgegenzuwirken, wird von der Regelkonfidenz der Support von B, die relative Häufigkeit des Ereignisses B, abgezogen. Zudem wird durch diese Subtraktion das *rare item* Problem angegangen. Häufig gekaufte Produkte werden bestraft, während selten gekaufte begünstigt werden. Dieses Maß tritt in der Literatur [75],[126] unter dem Namen *Added Value* (AV), *Pavillon Index* oder *Centered Confidence* auf. Es misst anhand folgender Gleichung, um wie viel höher oder niedriger die Wahrscheinlichkeit ist, dass ein Produktkauf in Kombination mit einer anderen Transaktion stattfindet, als die grundsätzliche Wahrscheinlichkeit, dass das Produkt gekauft wird:

$$AV(A \rightarrow B) = Konfidenz(A \rightarrow B) - Support(B) \tag{4.2}$$

Auch beim NAVSD-Verfahren werden lediglich einfache Assoziationsregeln betrachtet, deren Body und Head aus jeweils einem Produkt bestehen. Als Alternative zum AV ist der *Lift* [18] zu nennen. Dieser unterscheidet sich von ersterem lediglich darin, dass er keine Subtraktion, sondern eine Division berechnet. Da es somit einen Faktor anstatt einer absoluten Größe berechnet, wurde das AV-Maß hinsichtlich der additiven Aggregation während der Anwendungsphase als sinnvollere Alternative erachtet.

4.2.3 Diskretisierung

Nach der Berechnung der Interessantheitswerte werden für jeden Kunden basierend auf dem *Trainingsbasisdatensatz* die korrespondierenden Regeln bestimmt. Darunter sind solche Regeln zu verstehen, deren Body ein historisches Ereignis des Nutzers enthält. Daraufhin werden die k größten Interessantheitswerte selektiert und im Falle des gleichen Heads durch Summation aggregiert. Dieser Prozess tritt in ähnlicher Form in der Anwendungsphase auf und wird im weiteren Verlauf des Kapitels erneut thematisiert. Ergebnis ist letztlich eine Tabelle mit Kunden, kandidierenden Produkten und Summen der Interessantheitswerte bzw. Produktscores. Scores desselben Produktes fallen in der Regel für jeden Kunden unterschiedlich aus. Grund dafür sind verschiedene Transaktionshistorien und somit unterschiedliche Arten und Zahlen von Ereignissen mit Assoziation zum Zielprodukt aufweisen. Der Produktscore gibt somit an, in welchem Ausmaß historische Transaktionen des Kunden den Kauf des Zielproduktes beeinflussen.
An diese Tabelle wird weiterhin ein binäres Feld angefügt, welches berücksichtigt, ob der Nutzer u tatsächlich im Zielzeitraum das Produkt j kaufte oder nicht. Das NAVSD-Verfahren prüft dazu, ob die Kombination $\langle u, j \rangle$ im Trainingszieldatensatz vorhanden ist.
Daraufhin werden die kontinuierlichen Scores produktweise in Klassen eingeteilt. Im Bereich des maschinellen Lernens wird dieser Prozess *Diskretisierung* genannt. Zu unterscheiden sind zwei Arten von Diskretisierung: *überwacht* und *unüberwacht*. Während bei der überwachten Diskretisierung Zielvariablen einbezogen werden, erfolgt die Diskretisierung in der unüberwachten Form ohne Berücksichtigung weiterer Informationen. Das NAVSD-Verfahren nutzt ersteres, da es Scores mithilfe der Kaufinformation als Zielvariable in Gruppen teilt. Diese Information wird zudem verwendet, um die Wahrscheinlichkeit aus den in einer Gruppe enthaltenen Scores zu berechnen.
Die Grundidee hinter diesem Gruppierungsprozess ist die, dass die Schnittpunkte einer sortierten Scoreliste so gewählt werden, dass die Differenz zwischen den Klassen maximal ist. Durch rekursives Partitionieren werden die Scores kategorisiert. Dazu wird der von Hothorn et al. [58] entwickelte *Conditional Inference Tree* (Abk.: CTREE) verwendet, der Baum-basierte Regressionsmodelle mit einer wohldefinierten Theorie für bedingte Inferenzprozeduren verknüpft. Der Algorithmus setzt rekursives binäres Splitten in folgenden Schritten um:

1) Die globale Nullhypothese der Unabhängigkeit wird zwischen einer der m Inputvariablen $X_j, j = 1, \ldots, m$ und der Responsevariable Y getestet. Kann diese nicht verworfen werden, stoppt der Algorithmus. Andernfalls wird die Inputvariable X_{j^*} mit der stärksten Assoziation zu Y ausgewählt. Diese Assoziation wird über Teststatistiken bzw. p-Werte berechnet, welche die Abweichung von der partiellen Hypothese H_o^j erfassen.

2) In der selektierten Inputvariable wird ein binärer Split durchgeführt. Der beste Split wird im Rahmen von Permutationstests basierend auf standardisierten linearen Statistiken bestimmt.

3) Schritt 1) und 2) werden rekursiv wiederholt.

Als Stopp- und Selektionskriterium für den CTREE-Algorithmus wird die von Strasser und Weber vorgeschlagene Teststatistik des *Permutationstests* [119] verwendet. Da der Produktscore die einzige Inputvariable darstellt, wird hier nicht weiter auf das Kriterium zur Selektion der Splitvariable im ersten Schritt eingegangen. Liegt der p-Wert dieser Teststatistik unterhalb eines vorher definierten Wertes α, wird die Nullhypothese H_0 abgelehnt. Ist das nicht der Fall, stoppt CTREE an dieser Stelle. α kann in diesem Sinne als Parameter zur Bestimmung der Baumgröße betrachtet werden.
Im zweiten Schritt wird die Güte eines Splits ebenfalls über lineare Teststatistiken für alle möglichen Teilmengen A von X_{j^*} ermittelt. Die Teststatistiken erfassen dabei den Unterschied zwischen den beiden Stichproben

$$\{Y_i | w_i > 0 \wedge X_{ij} \in A; i = 1, \dots, n\} \text{ und}$$
$$\{Y_i | w_i > 0 \wedge X_{ij} \notin A; i = 1, \dots, n\}.$$

Es wird die Teilmenge A^* von X_{j^*} mit der größten Teststatistik gewählt. Weitere Details zur genauen Berechnung der Teststatistiken und der p-Werte in den ersten beiden Schritten des Algorithmus finden sich in der Studie von Hothorn et al. [58].
Ergebnis des Diskretisierungsprozesses sind aussagekräftige, statistisch unterschiedliche Gruppen, für die Wahrscheinlichkeiten auf Basis der Anzahl der in ihr enthaltenden Kunden und Käufe berechnet werden. Sei n_i die Zahl der Kunden sowie k_i die Anzahl der Käufer in Klasse i, berechnet sich die Wahrscheinlichkeit p_i wie folgt:

$$p_i = \frac{k_i}{n_i} \tag{4.3}$$

Das beschriebene Diskretisierungsverfahren lässt sich in *R* mit dem Paket *smbinning*[6] realisieren. Die Tabellen 4.1 und 4.2 zeigen beispielhaft, wie die Ein- und Ausgabe dieses Prozesses für ein Produkt aussehen könnten. Die Quote von 0,5 für die Klasse 2 des Produktes 1 ergibt sich aus zwei Käufern unter den vier Kunden.
Die Diskretisierung von Produktscores wurde bisher in keiner Studie zu Empfehlungssystemen thematisiert und stellt somit ein Novum dar. Dennoch haben wir sie im NAVSD-Verfahren aus zwei Gründen eingesetzt: Zum einen kontrolliert bzw. mindert sie den Einfluss von Ausreißern. Dies betrifft insb. von der Allgemeinheit gekaufte Produkte, wie z.B. USB-Sticks, die aufgrund ihres unverhältnismäßig großen Scores abgeschwächt werden. Zum anderen löst sie

[6] R Package 'smbinning' Version 0.2 (2015), http://www.scoringmodeling.com/rpackage/smbinning/.

das Problem der unterschiedlichen Größenordnungen der Produktscores. So sind resultierende Wahrscheinlichkeiten für verschiedene Produkte miteinander vergleichbar.

Kunde	Produkt	Score	Kauf
101	1	1,3	1
102	1	1,1	0
103	1	0,8	0
104	1	0,6	1
105	1	0,5	0
106	1	0,4	0
107	1	0,3	1
108	1	0,2	0

Tabelle 4.1: Beispiel für den Input des Diskretisierungsprozesses

Produkt	Klasse	Min_Score	Max_Score	Käufer	Potentielle Käufer	Quote
1	2	0,6	100	2	4	0,5
1	1	-100	0,6	1	4	0,25
2	3	1,6	100	3	5	0,6
2	2	1,2	1,6	3	10	0,3
2	1	-100	1,2	2	20	0,1

Tabelle 4.2: Beispiel für den Output des Diskretisierungsprozesses

Output des Trainingsprozesses und gleichzeitig Grundlage für den Anwendungsprozess sind zwei Tabellen: Einerseits eine Tabelle mit allen Produktkombinationen samt ihren Interessantheitswerten, die im weiteren Verlauf als *Interessantheitstabelle* bezeichnet wird, und zum anderen jene Tabelle, die analog zu Tabelle 4.2, Scoregruppen enthält und im weiteren Verlauf als *Gruppentabelle* bezeichnet wird.

4.3 Anwendungsphase

Zur Eingabe des NAVSD-Anwendungsverfahrens gehört neben der Ausgabe des Trainingsprozesses der *Anwendungsdatensatz*. Dieser enthält alle Kundentransaktionen aus einem bestimmten vergangenen Zeitraum, der dieselbe Länge wie der Trainingsbasiszeitraum umfasst, jedoch um den Trainingszielzeitraum nachgelagert verläuft. Die Anwendungsphase durchläuft folgende drei Schritte:

1. **Aggregation der korrespondierenden Interessantheitswerte:** Zunächst werden die korrespondierenden Regeln mit den k größten Interessantheitswerten ermittelt, die dann zu Produktscores addiert werden. Dieser Schritt unterscheidet sich von der Regelfindung im der Trainingsphase lediglich darin, dass er auf Basis des *Anwendungsdatensatzes* erfolgt[7]. Aus diesem wird entnommen, welche Produkte $i \in \mathcal{I}_u^A$ der Kunde u in seiner Historie kaufte oder klickte.

[7] Zur Erinnerung: In der Trainingsphase erfolgte sie auf Grundlage des Trainingsbasisdatensatzes.

2. **Umwandlung der Scores zu Wahrscheinlichkeiten:** Daraufhin werden die Produktscores s_{uj} mithilfe der Gruppentabelle G in Wahrscheinlichkeiten p_{uj} umgewandelt. Das NAVSD-Verfahren prüft, in welcher produktspezifischen Gruppe sich der Score befindet, um die Wahrscheinlichkeit dieser Gruppe zu erlangen.

3. **Empfehlungsausgabe:** Für jeden Kunden werden die kandidierenden Produkte nach ihrer Wahrscheinlichkeit sortiert und schließlich die Top-N Produkte empfohlen.

Algorithmus 4.3 NAVSD-Anwendungsprozess

Input: Anwendungsdatensatz $A\ \langle Kunde, Produkt\rangle$,
Interessantheitstabelle $I\ \langle Produkt\_A, Produkt\_B, Interessantheitswert\rangle$,
Gruppentabelle $G\ \langle Produkt, Min\_Score, Max\_Score, Wahrscheinlichkeit\rangle$
Output: Top-N Produkte

```
1:  function Apply(A, I, G, k)
2:      For User u ∈ U_A                          #Menge der Kunden aus A
3:          For Item i ∈ J_u^A
4:              w_ij ← CorrespondedWeights(i, k, I)   #Ermittelt korrespondierende Regeln
5:              Add w_ij to W_u
6:          For w_xj ∈ W_u                        #für alle Werte mit demselben Zielprodukt
7:              Aggregate w_xj to s_uj
8:              Add s_uj to S_j                   #Menge der Scores für ein Zielprodukt
9:              Add j to J_u                      #Menge der Zielprodukte für einen Kunden
10:         For s_uj ∈ S_j
11:             p_uj ← Lookup(s_uj, G)            #wandelt Score in Wahrscheinlichkeit um
12:         Sort p_uj
13:         Return Top-N Items
```

4.3.1 k-Nächste-Nachbarn-Verfahren

Damit lediglich die wesentlichen Einflussfaktoren in die Prognose einfließen, werden im NAVSD-Verfahren die Assoziationen mit den k größten Interessantheitswerten eines Produktes betrachtet. Dieses Prinzip gleicht dem k-NN-Verfahren in traditionellen KF-Methoden, in denen die Nachbarschaft aus den k ähnlichsten Produkten besteht. In Kombination mit der Assoziationsregelanalyse wurde es bereits in Studien zur Lösung des Coverage-Problems verwendet [42],[66]. Die Vor- und Nachteile des k-NN-Verfahrens wurden in Abschnitt 3.2.3 diskutiert. Ein Einschränken auf die k größten Interessantheitswerte hat den Vorteil, dass häufig gekaufte Produkte, die eine große Anzahl an unbedeutenden Assoziationen aufweisen und meistens nicht-empfehlenswerte Produkte darstellen, seltener empfohlen werden. Auch hier hat die Wahl des kritischen Faktors k einen wesentlichen Einfluss auf die Genauigkeit und Effizienz des Verfahrens.

Die Anwendungsphase beginnt damit, dass basierend auf den Käufen und Klicks aus dem Anwendungsdatensatz alle korrespondierenden Regeln generiert werden. Für jedes Basisprodukt werden dabei jene k Zielprodukte selektiert, auf die ersteres den größten Einfluss hat bzw.

den größten AV-Wert aufweist. Wie bereits durch Gleichung 3.6 dargestellt, werden bei impliziten Daten die Gewichte durch Summieren aggregiert.
Abbildung 4.1 illustriert ein Beispiel mit $k = 3$ für einen Kunden, der die Produkte i_1, i_5 und i_8 in der Vergangenheit kaufte bzw. klickte. Die berechneten Scores stellen Pseudo-Prognosen dar, nach denen die Produkte für gewöhnlich sortiert werden. Aus ihnen können keine Aussagen bezüglich der Kaufwahrscheinlichkeit getroffen werden. Das NAVSD-Verfahren wandelt daher im nächsten Schritt die Produktscores in echte Wahrscheinlichkeiten um.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
i_1	0,1	0,1	0	**0,3**	**0,2**	**0,4**	0	0,1
i_2	0,1	0,2	**0,8**	**0,9**	0	**0,2**	0,1	0
i_3	0	**0,8**	0,3	0	**0,4**	0,1	0,3	**0,5**
i_4	**0,3**	**0,9**	0	0,2	0	**0,3**	0	0,1
i_5	**0,2**	0	**0,4**	0	0,1	**0,1**	0	0
i_6	**0,4**	**0,2**	0,1	**0,3**	0,1	0,2	0	0,1
i_7	0	**0,1**	**0,3**	0	0	0	0	0
i_8	**0,1**	0	**0,5**	**0,1**	0	0,1	0	0,1
	0,3	0	0,9	0,4	0,2	0,5	0	0

Abbildung 4.1: Beispiel einer Aggregation der Interessantheitswerte im NAVSD-Verfahren

4.3.2 Empfehlungsgenerierung

Während des Trainingsprozesses wurden die Scores produktweise klassifiziert und Wahrscheinlichkeiten für die resultierenden Klassen berechnet. Um mithilfe der Gruppentabelle die im Anwendungsprozess berechneten Produktscores in Wahrscheinlichkeiten zu transformieren, ermittelt das NAVSD-Verfahren, in welcher produktspezifischen Klasse sich der betrachtete Score befindet. Nehmen wir als Beispiel einen Kunden, der für das Produkt *2* einen Score von *1,4* aufweist. So kauft er das Produkt laut Tabelle 4.2 mit einer Wahrscheinlichkeit von 0,3.
Für jeden Nutzer werden schließlich die Top-N Produkte, sortiert nach ihrer berechneten Wahrscheinlichkeit, ausgegeben. Jedoch wird vorher sichergestellt, dass das Produkt nur dann empfohlen wird, wenn die Kaufwahrscheinlichkeit für den Kunden größer ist als die allgemeine Kaufwahrscheinlichkeit des Artikels. Trotz eventuell hoher Wahrscheinlichkeit soll so die Empfehlung eines Produktes nicht zustande kommen, wenn es ohnehin von den Kunden gekauft wird.

4.4 Implementierung

Nachdem das NAVSD-Verfahren ausreichend erläutert wurde, widme ich mich in diesem Abschnitt wenigen Implementierungsaspekten und technischen Details. Im Vordergrund steht hierbei der Input des NBO-Systems, der sich aufgrund des NAVSD-Trainingsverfahrens von Eingaben bestehender Systeme unterscheidet.

4.4.1 Systembeschreibung

Das NBO-System wurde aus zwei Gründen mit dem Statistikprogramm *R*[8] implementiert: Zum einen können statistische Kennzahlen, wie der p-Wert, einfach berechnet werden. Zum anderen bietet es sich aufgrund des einfachen Zusammenspiels mit der In-Memory Datenbank *Exasol*[9] an. Diese Datenbank wurde zur Datenhaltung verwendet, weil sie zu den schnellsten und intelligentesten gehört. Letztere Eigenschaft zählt jedoch gleichzeitig zu ihren Schwächen: Aufgrund ihrer komplexen Optimierungsprozesse werden triviale Befehle im Vergleich zur Alternativlösung, der weit verbreiteten Open-Source-Datenbank *mySQL*[10] relativ langsam ausgeführt. Die mySQL-Datenbank hingegen erwies sich bei komplexeren Befehlen mit großen Datenmengen, die zur Implementierung des NAVSD-Verfahrens benötigt werden, als deutlich langsamer. Des Weiteren stellt das Erstellen von Funktionen oder Skripten in den Programmiersprachen *R*, *Python*, *Java* und *Lua* ein hilfreiches Feature der Exasol-Datenbank dar.

4.4.2 Datenbasis

Grundlage für den Trainings- und Anwendungsprozess des NAVSD-Verfahrens sind drei Datensätze: der Trainingsbasis-, der Trainingsziel- und der Anwendungsdatensatz. Die impliziten Bewertungsdaten werden als Textdateien aufbereitet, die lediglich die Felder *User* und *Item* aufweisen. Um Klicks von Käufen unterscheiden zu können, wird im Falle eines Kaufes auf das Feld *Item* eine im Voraus festgelegte Konstante addiert. Der Wert dieser Konstante sollte größer als der maximale Wert des Feldes sein (in diesem Fall 5.000.000). Ein Klick und ein Kauf desselben Produktes werden vom System somit als unterschiedliche *Items* erfasst.
Da im NAVSD-Verfahren Assoziationen zwischen Produkten aus dem Trainingsbasis- und dem Trainingszieldatensatz berechnet werden, enthalten diese Datensätze lediglich jene Kunden, die in beiden Zeiträumen aktiv waren. Der Zeitraum des Trainingsbasisdatensatzes gleicht dem des Anwendungsdatensatzes. Der Zeitraum des Trainingszieldatensatzes entspricht dem Prognosezeitraum der Empfehlungen. Dieser liegt für gewöhnlich zwischen zwei Wochen und zwei Monaten. Im Trainingszieldatensatz werden nicht nur Nutzer gefiltert, die vorher nicht aktiv waren, sondern auch alle Klicks, da letztlich Käufe prognostiziert werden. Im Anwen-

[8] https://www.r-project.org.
[9] http://www.exasol.com.
[10] https://www.mysql.de.

dungsdatensatz werden keine Daten gefiltert. Er enthält alle historischen Kundentransaktionen und bildet letztlich die Grundlage des Empfehlungsprozesses.

Abbildung 4.2 illustriert die drei Datensätze beispielhaft an einem Zeitstrahl: Während die Kunden (2), (3) und (5) aus dem Trainingsbasisdatensatz entfernt werden, weil sie keine Käufe im Trainingszielzeitraum aufweisen, wird der Kunde (4) aus dem Trainingszieldatensatz entfernt, weil er nichts im Trainingsbasiszeitraum kaufte. Die Nutzer (2), (4) und (5) bleiben jedoch im Anwendungsdatensatz und erhalten somit Empfehlungen. Die Transaktionen der Kunden (1) und (7) werden aus keinem Datensatz entfernt.

Aufbereitet werden die Dateien mit dem Programm *QlikView*[11], welches sich aufgrund der schnellen, in-Memory Verarbeitung von sehr großen Datenmengen anbietet. Die Datenvorverarbeitung erscheint zunächst komplexer als bei anderen Empfehlungssystemen, ist jedoch mit dem Befehl „*Inner Keep*“[12] in QlikView einfach zu bewältigen.

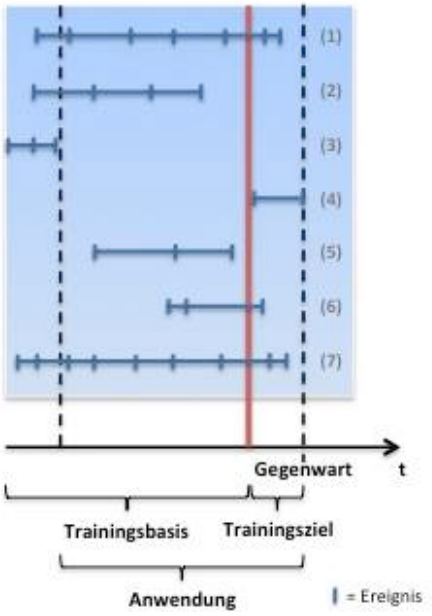

Abbildung 4.2: Graphische Darstellung der Datenbasis für das NBO-System

[11] http://www.qlik.com.

[12] Dieser Befehl löscht aus zwei Tabellen die nicht zur Schnittmenge gehörenden Datensätze.

5 Experimente

In diesem Kapitel werden das NAVSD-Verfahren sowie die in Abschnitt 3.7 erläuterten Methoden experimentell untersucht. Während die produktbasierten k-NN-Verfahren (Abk.: Item-KNN) und das Assoziationsregelverfahren (engl.: Association Rule Mining, Abk.: ARM) von mir implementiert wurden, nutze ich zur Evaluierung der MF-Methoden BPRMF und WRMF die Open Source Bibliothek *MyMediaLite*[13] [40]. Die Experimente werden auf einem Intel Core i7 Prozessor mit 2,40 GHz sowie einem 32 GB Arbeitsspeicher durchgeführt. Bevor ich die Ergebnisse präsentiere und diskutiere, werden die verwendeten Datensätze, Evaluationsmaße und Splitmethoden beschrieben sowie ein Verfahren zur Parameteroptimierung vorgestellt. Zunächst ist jedoch die Motivation der Experimente zu klären.

5.1 Parameter und Fragestellungen

Pradel et al. [94] entwickelten mit ihrer Fallstudie ein besseres Verständnis für Kaufdatensätze und kritische Faktoren von Empfehlungssystemen für E-Commerce Unternehmen. An diesem Ansatz knüpft die vorliegende Arbeit an. Beim Durchführen der Experimente stehen folgende Parameter und Fragestellungen im Vordergrund:

- **Größe der Nachbarschaft und Ähnlichkeitsmaß**
  Wie wirkt sich die Wahl der Nachbarschaftsgröße und des Ähnlichkeitsmaßes auf die Empfehlungsqualität jener Verfahren aus, die dem k-NN-Prinzip folgen? Welches Ähnlichkeitsmaß und welche Nachbarschaftsgröße liefern die genausten Empfehlungslisten?

- **Optimierung der Parameter**
  Um vergleichbare Ergebnisse zu erhalten, stellt sich die Frage, welche Parameterwerte optimal für die einzelnen Empfehlungsalgorithmen sind.

- **Vergleiche zwischen unterschiedlichen Methoden**
  Wie stellt sich die Rangfolge der Algorithmen bezüglich der Genauigkeit dar? Ist diese Rangfolge konsistent über verschiedene Trainings- und Splitverfahren? Welche Zeit wird jeweils beansprucht?

- **Länge der Transaktionshistorie**
  Wie wirkt sich die Länge der Transaktionshistorie des Trainingsbasisdatensatzes auf die Empfehlungsqualität aus? Gibt das System bei Trainingsdaten mit kürzerer Historie genauere Empfehlungen aus, weil Produkte bzw. Käufe Trends oder saisonalen Effekten unterliegen?

[13] http://www.mymedialite.net/.

- **Hinzunahme der Klickhistorie**
  Welche Auswirkungen hat die Hinzunahme der Klickhistorie auf die Genauigkeit?

- **Warengruppen statt Artikel**
  Wie wirkt sich das Ersetzen der Artikel durch Warengruppen auf die Genauigkeit aus? Erhöht sich durch die dichter besetzte Warengruppenmatrix die Coverage auf Produktebene, wenn nach dem Training und der Anwendung auf Warengruppendaten ein zufällig ausgewähltes Produkt aus der prognostizierten Warengruppe empfohlen wird?

5.2 Datensätze

Für die Experimente stellten mir zwei E-Commerce Unternehmen die Datensätze *DS01* und *DS02* zur Verfügung. Beide Datensätze basieren auf impliziten Bewertungen und enthalten historische Kaufdaten. Ihre Eigenschaften sind in der Tabelle 5.1 zusammengefasst. Hierbei ist insb. die geringere Spärlichkeit von DS02 zu nennen. Die Abbildungen 5.1 und 5.2, welche die Verteilung der Käufe pro Kunde darstellen, verdeutlichen nochmals diesen Unterschied. Sie zeigen, dass lediglich ein kleiner Teil der Kunden verhältnismäßig viele Produkte kauft. Dieses Phänomen ist für DS01 ausgeprägter als für DS02.
Für DS01 stehen mir Klick- und Warengruppendaten zur Verfügung. Für diesen Datensatz wurde zudem ein Trainingsbasis- bzw. Anwendungszeitraum von *100 Tagen* und einen Trainingsziel- bzw. Testzeitraum von *14 Tagen* gewählt. Bei DS02 fallen diese Zeiträume mit *12* und *2 Monaten* deutlich größer aus.
Die Datensätze wurden als Textdateien aufbereitet, die lediglich die Felder *User* und *Item* aufweisen. Auch hier stellen Kauf und Klick desselben Produktes unterschiedliche Items dar, da im Falle eines Kaufes eine vorab definierte Konstante aufaddiert wird.

	DS01	**DS02**
Branche	Elektronik	Gesundheit
Zeitspanne	128 Tage	16 Monate
#Kunden	843.257	49.481
#Produkte	190.774	1.521
#Warengruppen	8296	-
#Käufe	2.816.476	319.586
#Klicks	6.298.937	-
Spärlichkeit (%)	99,9982	99,5754

Tabelle 5.1: Eigenschaften der Datensätze DS01 und DS02

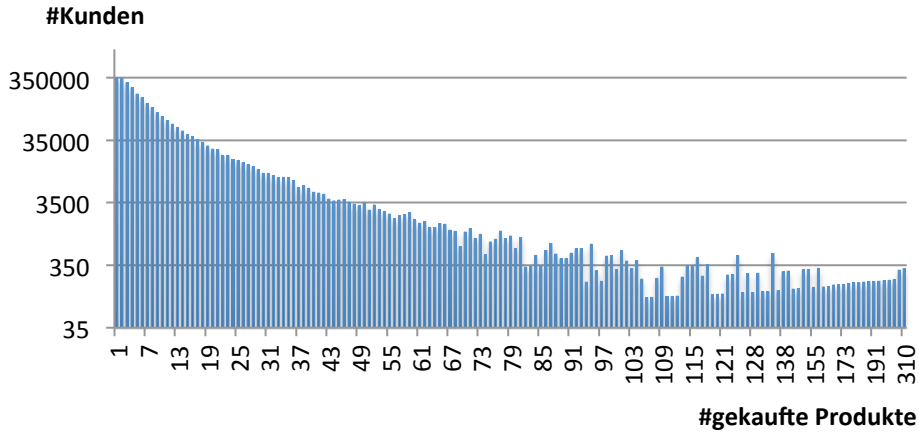

Abbildung 5.1: Verteilung der gekauften Produkte pro Kunde für DS01

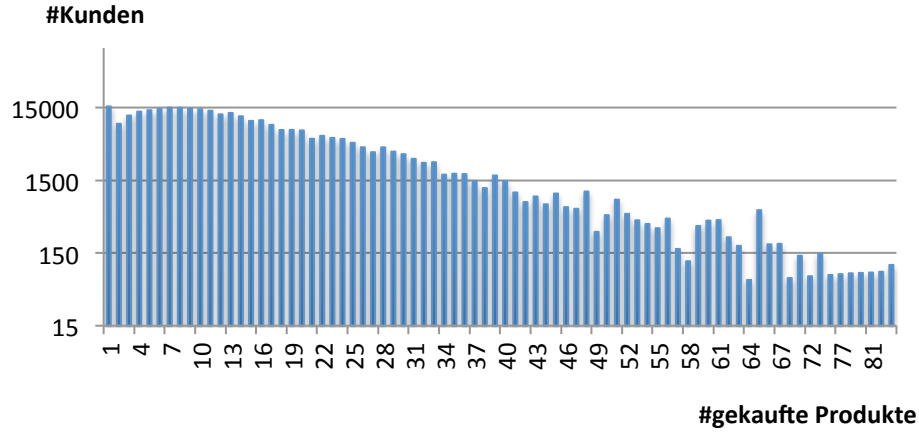

Abbildung 5.2: Verteilung der gekauften Produkte pro Kunde für DS02

5.3 Splitmethoden

Die Experimente werden *offline* durchgeführt. Dabei werden mit bereits vorhandenen Käufen und Klicks Interaktionen der Kunden mit dem Empfehlungssystem simuliert. Dazu müssen die Daten in *Trainings-* und *Testdatensätze* aufgeteilt werden. Das Empfehlungssystem erlernt Muster aus den Trainingsdaten und generiert letztlich eine Top-N-Empfehlungsliste. Die Testdaten enthalten Käufe, mit denen die ausgegebenen Empfehlungen evaluiert werden. Die Aufteilung in Trainings- und Testdaten kann auf unterschiedlichen Wegen erfolgen. Im Folgenden werden zwei bewährte Methoden aufgegriffen, die aufgrund des Trainingszieldatensatzes und dem NAVSD-Trainingsverfahrens leichte Änderungen erfahren. Das Motiv, das hinter der Verwendung verschiedener Splitmethoden steht, ist die Erlangung konsistenter Ergebnisse und die Überprüfung, ob und inwieweit sich die Rangfolge der Algorithmen unter ihnen ändert.

5.3.1 Random-Split

Der nicht-chronologische Split wird am häufigsten in der Literatur verwendet [65], [96],[103],[107]. Für üblich wird zunächst jeder Kunde mit einer Anzahl historischer Käufe unter einem Schwellenwert verworfen, um das Kaltstartproblem zu meiden. In diesem Fall werden nur Kunden mit mindestens fünf Käufen ausgewertet. Anschließend werden die Käufe jedes Nutzers *in zufälliger Reihenfolge* in fünf gleich große Gruppen aufgeteilt, um eine *Kreuzvalidierung* durchzuführen: Hierzu wird jede Gruppe einmal als Testdatensatz verwendet, während die jeweils vier anderen Gruppen die Trainingsbasisdaten bilden. Das Ergebnis ergibt sich aus dem Durchschnitt der Resultate aus den fünf Durchläufen. Der Trainingszieldatensatz, der für das NAVSD-Trainingsverfahren benötigt wird, bleibt in allen Durchläufen unverändert. Diese Splitmethode wird im Folgenden in Kombination mit dem Standardtrainingsverfahren aus Abschnitt 3.2.3 unter *Option O1* zusammengefasst.

5.3.2 Chronologischer Split

Der chronologische Split bietet einen realistischeren Aufbau: Bei diesem Split wird mit vergangenen Käufen und Klicks trainiert, um künftige Käufe zu prognostizieren. Aus diesem Grund wird diese Splitmethode in den Experimenten sowohl in Kombination mit dem Standard- als auch mit NAVSD-Trainingsverfahren verwendet. Der Testdatensatz besteht aus Kaufdaten über eines bestimmen Zeitraum unmittelbar nach dem Trainingszeitraum. Der chronologische Split wird im Folgenden in Kombination mit dem Standardtrainingsverfahren unter *Option O2* und in Kombination mit dem NAVSD-Trainingsverfahren unter *Option O3* zusammengefasst.

5.4 Evaluationsmaße

Die Evaluation von Empfehlungssystemen im industriellen Kontext verdient besondere Aufmerksamkeit. Während die Prognosegenauigkeit in wissenschaftlichen Arbeiten im Vordergrund steht, sind Empfehlungen in der Industrie dafür bestimmt, Umsätze zu maximieren. Aiolli [5] weis darauf hin, dass die Wahl eines genauen Verfahrens nicht alles sei, was ein gutes Empfehlungssystem ausmache. Andere Evaluationskriterien, wie die Coverage, die Utility oder die Serendipität spielen ebenfalls eine wesentliche Rolle im E-Commerce. Da sich die vorliegende Arbeit in erster Linie mit dem Vergleich von Empfehlungsalgorithmen beschäftigt und kein gründliches Testen eines bestimmten Verfahrens als Ziel setzt, werden Genauigkeit und Laufzeit die wesentlichen Evaluationskriterien darstellen. In den Studien zogen es Forscher vor, die Genauigkeit anstatt der Laufzeit zu untersuchen. Laut Beel et al. [12] berichten nur 11% der Forschungsartikel zum Thema Empfehlungssysteme über die Laufzeit der Algorithmen. Diese Statistik erscheint verwunderlich. Denn die Laufzeit stellt gerade in Umgebungen, in denen in Echtzeit Interaktionen mit dem System ausgeführt werden, einen wesentlichen Faktor dar.

Output jedes Empfehlungsalgorithmus ist eine Top-N-Empfehlungsliste für jeden Kunden. Um die Genauigkeit dieser Listen zu evaluieren, werden häufig die Metriken *Precision*, *Recall* und *F1* verwendet [54]. Sie gehören zu den *klassifizierenden Genauigkeitsmaßen* und messen die Häufigkeit, mit der ein Empfehlungsalgorithmus korrekte oder falsche Empfehlungen ausgibt. Die Precision misst jenen Anteil an den empfohlenen Produkten, der *relevant* ist. Relevant ist eine Empfehlung des Produktes *j* für den Kunden *u* dann, wenn *j* von *u* gekauft wurde bzw. der Eintrag (*u*, *j*) im Testdatensatz vorhanden ist. Sei *L(u)* die Liste der Empfehlungen und *T(u)* die Liste der Produkte aus dem Testdatensatz für *u*, ist die Precision definiert als

$$Precision@N = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{L(u) \cap T(u)}{N} \tag{5.1}$$

Recall gehört neben Precision zu den am weitesten verbreiteten Metriken zur Evaluierung von Empfehlungssystemen. Sie misst jenen Anteil der relevanten an den empfohlenen Produkten:

$$Recall@N = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{L(u) \cap T(u)}{T(u)} \tag{5.2}$$

Sowohl Recall als auch Precision nehmen einen Wert zwischen 0 und 1 an. Die F1-Metrik berücksichtigt beide Maße. Sie stellt das harmonische Mittel von Precision und Recall dar und berechnet sich wie folgt:

$$F1@N = \frac{2 \times Precision@N \times Recall@N}{Precision@N + Recall@N} \tag{5.3}$$

In den folgenden Experimenten werden für jeden Kunden die Top-10 Empfehlungen generiert, sodass die Maße für *N=10* berechnen werden.
Auch ein genaues Empfehlungssystem ist oft nicht in der Lage bestimmte Produkte zu empfehlen. So sind manche Artikel aufgrund ihrer Spärlichkeit in keiner Empfehlungsliste enthalten. Um dieses Phänomen formal zu beschreiben, wird die *Coverage* eines Empfehlungssystems berechnet. Die am weitesten verbreitete Form von Coverage stellt die *Catalog Coverage* (Abk.: CC) dar. Sie definiert jenen Anteil der Produkte, der mindestens einem Kunden empfohlen wurde [41]. Für *n* Artikel und *m* Nutzer wird die CC wie folgt berechnet:

$$CC = \frac{\left|\bigcup_{u=1}^{m} L(u)\right|}{n} \tag{5.4}$$

Nach Good et al. [46] müsse zwischen Genauigkeit und Coverage abgewogen werden. Primärer Zweck der Experimente ist es, verschiedene Verfahren bezüglich ihrer Prognosegenauigkeit miteinander zu vergleichen. Vergleiche der Coverage stehen vielmehr bei der Untersuchung der Auswirkungen von Warengruppen als Inputdaten im Vordergrund.

5.4.1 Parameteroptimierung via *Golden Section Search*

Abhängig davon welches Verfahren verwendet wird, gibt es verschiedene Parameter, die das Ergebnis in unterschiedlichem Maße beeinflussen. Um aus jeder Methode, das beste Ergebnis zu erlangen, müssen die Parameter optimiert werden. Dazu wird üblicherweise eine Kreuzvalidierung durchgeführt. Die von mir verwendete Methodik hingegen ist die *Golden Section Search* (Abk.: GSS) [49]. Sie sucht die Zielfunktion *f(x)* maximierende Stelle *x* im Konfigurationsraum *[a,b]*:

$$x^* = \arg\max_{a \le x \le b} f(x) \qquad (5.5)$$

Während die Funktion f den Empfehlungsalgorithmus darstellt, wird die F1-Metrik als Rückgabewert $f(x)$ definiert. Der GSS-Algorithmus kann als Maximierungsproblem formuliert werden: Er findet x^* durch iteratives Verkleinern des Konfigurationsraumes. Die Grenzen werden dabei durch den *inversen Goldenen Schritt*

$$\varphi = \frac{-1 + \sqrt{5}}{2} \approx 0{,}618$$

solange näher aneinander gerückt, bis das resultierende Intervall kleiner als die vordefinierte Toleranz ε ist. Algorithmus 5.1 beschreibt, wie das GSS-Verfahren auf mehrere Parameter angewendet werden kann [23]. Da diese abhängig voneinander sind, werden die Parameter nicht einzeln, sondern stückweise, iterativ optimiert.

Algorithmus 5.1 Golden Section Search

Input: Liste $\Phi(param_1\langle a_1, b_1, \varepsilon_1\rangle, \ldots, param_n\langle a_n, b_n, \varepsilon_n\rangle)$
Output: Optimale Werte $x_1^*, \ldots, x_n^*$ der Parameter in Φ

```
1:  function MULTI-GSS(Φ)
2:      while |Φ| > 0 do                    #nicht optimierte Parameter existieren
3:          for i ← 1, |Φ| do
4:              x_1 ← b_i − φ(b_i − a_i)
5:              x_2 ← a_i + φ(b_i − a_i)
6:              if f(x_1) < f(x_2) then     #x_i* muss im Intervall [x_1, b] sein
7:                  a_i ← x_1
8:              else                        #x_i* muss im Intervall [a, x_2] sein
9:                  b_i ← x_2
10:             if b_i − a_i ≤ ε_i then     #param_i wurde optimiert
11:                 x_i* ← (b_i − a_i)/2
12:                 Entferne param_i aus Φ
13:                 return x_i*
```

5.5 Ergebnisse und Diskussion

Basierend auf den Fragestellungen aus Abschnitt 5.1 sind Experimente mit den in Abschnitt 5.2 beschriebenen Datensätzen durchgeführt worden. Um die Konsistenz und Vergleichbarkeit der Ergebnisse zu wahren, wurden die Parameter jedes Empfehlungsalgorithmus optimiert.

5.5.1 Verschiedene Ähnlichkeitsmaße und Größe der Nachbarschaft

Zunächst wurde untersucht, wie sich verschiedene Ähnlichkeitsmaße und Nachbarschaftsgrößen auf die Prognosegenauigkeit des ItemKNN-Verfahrens auswirken. Durchgeführt wurde diese Versuchsreihe unter der Option O1. Als Evaluationsmaß wurde die F1-Metrik gewählt, da diese sowohl Precision als auch Recall berücksichtigt. Die Abbildungen 5.3 und 5.4 präsentieren die Ergebnisse für die Datensätze DS01 und DS02. Die Größe der Nachbarschaft hat erwartungsgemäß einen signifikanten Einfluss auf die Genauigkeit. Jedoch weisen das ItemKNN- und das NAVSD-Verfahren unterschiedliche Verläufe auf: Während die Kurve des ersteren bis zu einer bestimmten Nachbarschaftsgröße steigt und danach verflacht, sinkt die Genauigkeit des letzteren mit steigender Nachbarzahl und erreicht bei einer kleinen Nachbarschaft ihr Maximum. Zu ähnlichen Ergebnissen für das ItemKNN-Verfahren kamen Sarwar et al. [109]. Wird k zu groß gewählt, erhöht sich das Risiko des Rauschens in den Daten, da nicht alle Nachbarprodukte gute Prädiktoren darstellen. Dieser Effekt tritt beim NAVSD-Verfahren bereits bei einer kleinen Nachbarschaftsgröße ein. Den umgekehrten Fall, dass bei zu kleinem k gute Prädiktoren verworfen werden, zeigt der anfängliche Kurvenverlauf des ItemKNN-Verfahrens. Zwar unterscheiden sich die beiden Verfahren im Wesentlichen durch das Ähnlichkeits- bzw. Interessantheitsmaß sowie durch den Diskretisierungsprozess[14], jedoch sind die unterschiedlichen Verläufe insb. auf letztere Komponente zurückzuführen. Nach der Diskretisierung der aggregierten Ähnlichkeitswerte erzeugte das modifizierte ItemKNN- einen ähnlichen Verlauf zum NAVSD-Verfahren.

Zur Rangfolge der Ähnlichkeitsmaße lässt sich Folgendes sagen: Das Log-Likelihood-Maß liefert gegenüber dem Kosinus-, Jaccard- und bedingten Wahrscheinlichkeitsmaß für beide Datensätze bessere Ergebnisse, was angesichts ihrer intelligenten Metrik zu erwarten war. Das NAVSD-Verfahren jedoch ist allen genannten Verfahren überlegen. In den weiteren Analysen verwenden wir repräsentativ für die produktbasierte KF-Methode jene Variante, die auf dem Log-Likelihood-Ähnlichkeitsmaß basiert.

[14] Zur Erinnerung: Im ItemKNN-Verfahren findet keine Diskretisierung statt.

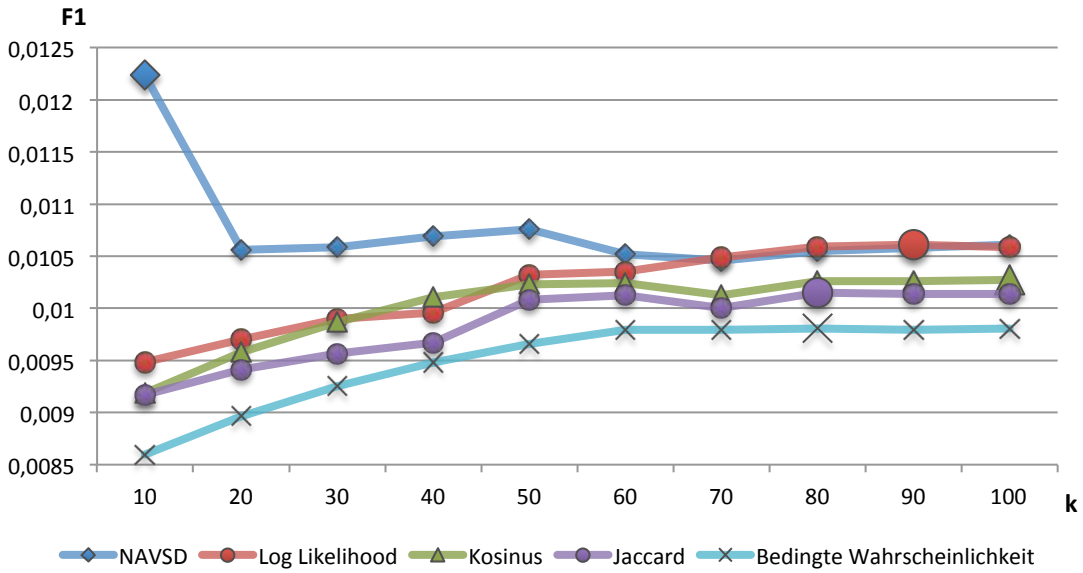

Abbildung 5.3: Auswirkung des Ähnlichkeitsmaßes und der Nachbarschaftsgröße auf die F1-Metrik für DS01

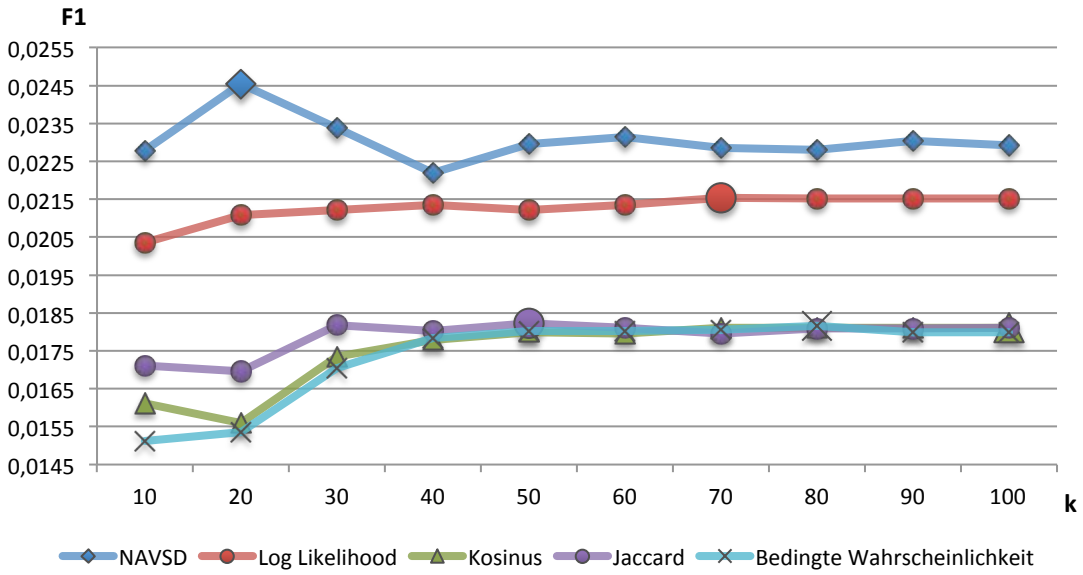

Abbildung 5.4: Auswirkung des Ähnlichkeitsmaßes und der Nachbarschaftsgröße auf die F1-Metrik für DS02

5.5.2 Optimierte Parameter

Neben den Parametern des NAVSD- und des ItemKNN-Verfahrens wurden die der MF-Methoden nach dem GSS-Verfahren optimiert. Auch dieser Optimierungsprozess erfolgte unter der Option O1. Tabelle 5.2 fasst die optimierten Parameter zusammen. Es ist auffällig, dass der WRMF- weniger Iterationen benötigt als der BPRMF-Algorithmus. Dies ist darauf zurückzuführen, dass ersterer jeweils eine der Faktorenmatrizen P und Q^T aktualisiert und das Optimierungsproblem somit quadratisch ist. Des Weiteren fiel während des Optimierungsprozesses auf, dass in manchen Fällen die Zwischenergebnisse besser sind als die Endergebnisse. Der Grund dafür könnte sein, dass die jeweilige Funktion bzw. Methode für be-

stimmte Parameter nicht unimodal ist und somit mehr als ein Maximum aufweist. Dieses Problem wurde gelöst, indem der aktuell optimale Parameterwert samt seiner Genauigkeit zwischengespeichert wird. Dies erklärt, warum die optimierten Werte der Parameter *Reg_u*, *Reg_i*, *Reg_j* und *bias_reg* für DS01 Standardwerte sind.

		DS01	DS02
BPRMF	num_iter	39	31
	num_factors	33	7
	learn_rate	0,0248	0,0143
	bias_reg	0	0
	Reg_u	0,0025	0,0021
	Reg_i	0,0025	0,0045
	Reg_j	0,0025	0,0027
WRMF	num_iter	23	15
	num_factors	78	16
	regularization	0,0144	0,0141
	alpha	13	2
ItemKNN	k	87	72
	Correlation Type	Log Likelihood	Log Likelihood
NAVSD	k	9	21

Tabelle 5.2: Optimierte Parameter der zu vergleichenden Algorithmen

5.5.3 Vergleich der Methoden

Um zu ermitteln, welcher Algorithmus die genausten Empfehlungen generiert, wurden Experimente unter verschiedenen Optionen durchgeführt. Dabei wurden zunächst ausschließlich Kaufdatensätze verwendet. Die Tabellen 5.3-5.8 zeigen die Ergebnisse, zu denen zunächst zu sagen ist, dass Option O3 nicht auf MF-Methoden angewandt werden kann. Der große Unterschied zwischen den Ergebnissen für O1 und O2 ist auf die unterschiedlichen Splitmethoden zurückzuführen. Während beim nicht-chronologischen Split das Kaltstartproblem durch das Entfernen von Kunden mit weniger als fünf Käufen beseitigt wird, ziehen diese das Ergebnis beim chronologischen Split runter. Zudem fand ich in weiteren Tests heraus, dass das Auswerten der Käufe in zufälliger Reihenfolge ebenfalls für bessere Ergebnisse sorgt: Dazu wurden die Kunden mit weniger als fünf Käufen entfernt und ein chronologischer sowie nicht-chronologischer Split durchgeführt. Dies führte zu einem Genauigkeitsunterschied von 200-300% zu Gunsten der nicht-chronologischen Splitmethode. Diese Beobachtung zeigt uns, dass das Einhalten der Sequenz während der Evaluation ein kritischer Faktor ist, und dass der Random-Split Empfehlungssysteme mit einer zu optimistischen Prognosequalität beschreibt.

Methode	Prec@10	Rec@10	F1@10
NAVSD	0,0351	**0,0921**	**0,0508**
ItemKNN	0,0356	0,0822	0,0497
ARM	**0,0360**	0,0852	0,0506
BPRMF	0,0063	0,0305	0,0104
WRMF	0,0105	0,0494	0,0173

Tabelle 5.3: Genauigkeit der Verfahren für DS01 unter O1

Methode	Prec@10	Rec@10	F1@10
NAVSD	0,0626	**0,1870**	**0,0938**
ItemKNN	0,0597	0,1780	0,0894
ARM	**0,0630**	0,1785	0,0931
BPRMF	0,0567	0,1643	0,0843
WRMF	0,0610	0,1769	0,0907

Tabelle 5.4: Genauigkeit der Verfahren für DS02 unter O1

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0104**	0,0155	**0,0124**
ItemKNN	0,0073	**0,0194**	0,0106
ARM	0,0061	0,0181	0,0091
BPRMF	0,0017	0,0068	0,0028
WRMF	0,0033	0,0108	0,0052

Tabelle 5.5: Genauigkeit der Verfahren für DS01 unter O2

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0185**	0,0304	**0,0230**
ItemKNN	0,0136	**0,0517**	0,0215
ARM	0,0099	0,0328	0,0153
BPRMF	0,0090	0,0315	0,0140
WRMF	0,0096	0,0389	0,0154

Tabelle 5.6: Genauigkeit der Verfahren für DS02 unter O2

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0587**	**0,0917**	**0,0716**
ItemKNN	0,0565	0,0871	0,0685
ARM	0,0583	0,0891	0,0705
BPRMF	-	-	-
WRMF	-	-	-

Tabelle 5.7: Genauigkeit der Verfahren für DS01 unter O3

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0702**	**0,1753**	**0,1003**
ItemKNN	0,0502	0,1632	0,0768
ARM	0,0643	0,1715	0,0935
BPRMF	-	-	-
WRMF	-	-	-

Tabelle 5.8: Genauigkeit der Verfahren für DS02 unter O3

Des Weiteren sind große Unterschiede zwischen den Ergebnissen für O2 und O3 zu sehen, welche auf die unterschiedlichen Trainingsverfahren zurückzuführen sind. Das Maßberechnungsverfahren des NAVSD-Algorithmus, das NAVSD-Trainingsverfahren, welches ebenfalls in Kombination mit der ARM- sowie der ItemKNN-Methode angewandt wurde, ist gegenüber dem Standardtrainingsverfahren für beide Datensätze deutlich im Vorteil.

Bezüglich der Rangfolge der Algorithmen stellen wir Folgendes fest: Das NAVSD-Verfahren schneidet gefolgt vom ARM-, ItemKNN- und den MF-Verfahren am besten ab. Zwar weist das ItemKNN-Verfahren für O2 einen höheren Recall oder das ARM-Verfahren für O1 eine höhere Precision auf. Jedoch ist das NAVSD-Verfahren allen anderen Algorithmen bezüglich dem letztendlich entscheidenden Kriterium, der F1-Metrik, überlegen. Unter den traditionellen KF-Methoden liefert das ARM-Verfahren die besten Ergebnisse, da sie für zwei von drei Optionen

besser als das ItemKNN-Verfahren abschneidet. Unter den MF-Methoden liefert das WRMF-Verfahren durchgängig bessere Ergebnisse. Zusammengefasst hat sich eine bis auf eine Ausnahme über alle Optionen konsistente Verfahrensrangfolge gebildet.

Des Weiteren ist auffällig, dass der Unterschied zwischen den traditionellen KF- und den MF-Methoden bezüglich der Ergebnisse für Datensatz DS01 viel größer ausfällt als für DS02. Für DS02 ist der WRMF-Algorithmus unter O2 sogar genauer als das ARM-Verfahren (siehe Tabelle 5.6). Dies könnte daran liegen, dass die KF-Methoden vom spärlicheren Datensatz DS01 nicht in dem Ausmaß negativ beeinflusst werden, wie es die MF-Methoden werden. Es scheint, als ob die Effizienz ersterer nicht durch die Spärlichkeit an sich verschlechtert wird. Solange es zu erlernende Muster gibt, ignorieren sie fehlendende Werte in der Matrix. Die MF-Methoden hingegen arbeiten auch mit diesen. Je höher der Anteil der Nullwerte ist, desto *verdünnter* sind die zu untersuchenden Daten.

Zu der Erkenntnis, dass die ARM-Methode die restlichen Algorithmen bezüglich der Empfehlungsgenauigkeit übertrumpft, kamen auch Pradel et al. [94]. Auch Aiollis Studie [5] geht in diese Richtung: Er kam zum Ergebnis, dass traditionelle KF-Methoden gegenüber MF-Methoden bei impliziten Daten im Vorteil sind. Koren et al. [74] kamen mit expliziten Daten zu gegenteiligem Ergebnis. Somit ist ersichtlich, dass sich die Rangfolge der Algorithmen bezüglich der Genauigkeit für unterschiedliche Arten von Bewertungsdaten unterscheiden. Es scheint, als seien MF-Methoden für binäre Daten ungenauer als für nicht-binäre. Der Grund dafür liegt darin, dass die Werte binärer Daten undifferenzierter sind. Da MF-Methoden auf Datenapproximation basieren, profitieren sie von Werten innerhalb eines bestimmten Intervalls, wie z.B. 1-10. Dabei führen Werte außerhalb dieses Intervalls zu entgegenwirkendem Effekt, da sie keine Datenmuster repräsentieren können. Für implizite und insb. binäre Daten sind Werte außerhalb des Bereiches 0-1 generell kein Problem. Sie deuten auf eine starke oder schwache Kundenpräferenz hin. Da das ARM- sowie ItemKNN-Verfahren nicht darauf beschränkt sind, Daten innerhalb eines bestimmten Bereiches zu approximieren, könnte dies ein Grund sein, weshalb sie bessere Ergebnisse als die MF-Methoden liefern.

Unter den MF-Methoden schneidet das WRMF-Algorithmus besser ab. Auffällig ist, dass die beiden Algorithmen für Datensatz DS02 deutlich näher aneinander liegen als für DS01. Als Grund ist hier die höhere Spärlichkeit von DS01 zu nennen. Da der BPRMF-Algorithmus mit Produktpaaren trainiert, scheint die riesige Menge an Produkten in Kombination mit der hohen Spärlichkeit von DS01 der Grund für die schlechten Ergebnisse zu sein. Denn die Annahme, dass bekannte Bewertungen unbekannten vorgezogen werden, ist umso weniger berechtigt, je mehr Produkte im Datensatz vorhanden sind. Da der größte Teil der Produkte für die Nutzer unbekannt ist, haben Datenmuster, die durch paarweises Vergleichen zwischen vorhandenen und fehlenden Bewertungen erlangt werden, eine geringe Signifikanz. Es ist ebenfalls berechtigt zu sagen, dass das ALS-Verfahren im WRMF-Algorithmus besser mit sehr spärlichen Daten zurechtkommt als das SGD-Verfahren im BPRMF-Algorithmus. Ersteres aktualisiert eine der beiden Faktorenmatrizen P und Q^T während einer Iteration, was zu einer uni-

modalen Optimierungsfunktion führt. Je spärlicher eine Matrix ist, desto mehr mögliche Annäherungslösungen hat sie. Aus diesem Grund führt eine multimodale Optimierung, wie sie im Falle des SGD-Verfahrens erfolgt, mit höherer Wahrscheinlichkeit zu einer nicht-optimalen Lösung.

Des Weiteren untersuchte ich die Algorithmen bezüglich ihrer Laufzeiten. Die Tabellen 5.9 und 5.10 präsentieren die Ergebnisse für die verschiedenen Optionen und Datensätze. Die notierten Zeiten ergeben sich aus der Summe von Trainings- und Anwendungszeit. Sie sind abseits der Implementierungsdetails abhängig von ihren Parameterwerten, wie *k* für die produktbasierten KF-Methoden oder die Anzahl Iterationen und Faktoren für die MF-Methoden. Erwartungsgemäß weisen die MF-Methoden deutlich höhere Laufzeiten als der Rest auf. Trotz der Tatsache, dass der WRMF-Algorithmus in allen Fällen mit weniger Iterationen als der BPRMF-Algorithmus trainiert wird, sieht man, dass er deutlich langsamer ist. Der Grund dafür könnte darin liegen, dass das ALS-Verfahren eine Lösung der kleinsten Quadrate liefert, was deutlich teurer als eine SGD-Iteration ist. Die unterschiedlichen Zeiten zwischen dem NAVSD- und dem ItemKNN-Verfahren sind auf den Diskretisierungsprozess des ersteren zurückzuführen. Der ARM-Algorithmus übertrifft beide aufgrund ihres einfacheren Anwendungsprozesses. Anhand der unterschiedlichen Laufzeiten zwischen den Optionen O2 und O3 wird deutlich, dass unter dem NAVSD-Trainingsverfahren deutlich weniger Interessantheits- bzw. Ähnlichkeitsmaße berechnet werden. Dieses Verfahren ist somit gegenüber dem Standardtrainingsverfahren sowohl bezüglich der Genauigkeit als bezüglich der Laufzeit im Vorteil.

Methode	Laufzeit O1	Laufzeit O2	Laufzeit O3
NAVSD	00:06:24	00:06:39	00:04:42
ItemKNN	00:05:23	00:05:08	00:04:03
ARM	00:04:37	00:04:50	00:03:58
BPRMF	01:42:19	00:11:20	-
WRMF	02:58:43	00:12:38	-

Tabelle 5.9: Laufzeiten der Algorithmen unter den verschiedenen Optionen für DS01 im Format [*hh:mm:ss*]

Methode	Laufzeit O1	Laufzeit O2	Laufzeit O3
NAVSD	00:00:26	00:00:22	00:00:20
ItemKNN	00:00:15	00:00:14	00:00:13
ARM	00:00:13	00:00:13	00:00:13
BPRMF	00:00:55	00:00:14	-
WRMF	00:01:08	00:00:16	-

Tabelle 5.10: Laufzeiten der Algorithmen unter den verschiedenen Optionen für DS02 im Format [*hh:mm:ss*]

5.5.4 Sensitivität der Transaktionshistorie

In diesem Abschnitt werden die einzelnen Verfahren mit reduzierten Trainingsdatensätzen evaluiert. Ähnliche Experimente wurden bereits in [62] und [94] durchgeführt. Während in [62] der reduzierte Trainingsdatensatz zufällig ausgewählte Transaktionen enthielt, wurden in [94] vergangene Transaktionen aus zwei Wochen verwendet. In der vorliegenden Arbeit wird ein flexibler Ansatz gewählt: Die Algorithmen werden mit variierender Länge der Transaktionshistorie trainiert und ausgewertet. Der Zeitraum des Trainingsbasisdatensatzes wird stetig verkleinert. Indes bleiben die restlichen Datensätze unverändert. Die Auswertung findet erneut unter der Option O2 statt.

Die Abbildungen 5.5 und 5.6 zeigen die Ergebnisse der Versuchsreihe für die Datensätze DS01 und DS02. Die Ergebnisse für DS01 deuten darauf hin, dass Käufe innerhalb eines kürzeren Zeitraums stärker miteinander korrelieren. Zum Ergebnis, dass die Aktualität der Käufe einen kritischen Faktor für die Empfehlungen darstellen, kamen auch Pradel et al. [94], Koren et al. [74] sowie Lee et al. [76]. Dieser Trend ist bei Datensatz DS02 nicht zu verzeichnen. Hier schwanken die Ergebnisse zwar bei gewissen Methoden (z.B. beim NAVSD- und BPMRF-Verfahren), weisen jedoch weder einen positiven noch einen negativen Trend auf.

Grundsätzlich stehen sich dabei zwei Effekte gegenüber: Auf der einen Seite werden durch immer kleiner werdende Trainingsdatensätze immer mehr Korrelationen verworfen, die einen positiven Beitrag zur Empfehlungsgenerierung leisten. Auf der anderen Seite kann dieser negative Effekt dann zum Vorteil werden, wenn saisonale oder im Trend stehende Produkte bessere Prognosen erhalten. Je nach Ausprägung dieser Effekte können einzelne Methoden mit kleiner werdenden Trainingsdatensätzen unterschiedlich verlaufen. Dies ist jedoch ebenfalls von der Produktart abhängig. So ist anzunehmen, dass der Zeitfaktor bspw. für Empfehlungen von Produkten aus der Modebranche einen großen Einfluss hat.

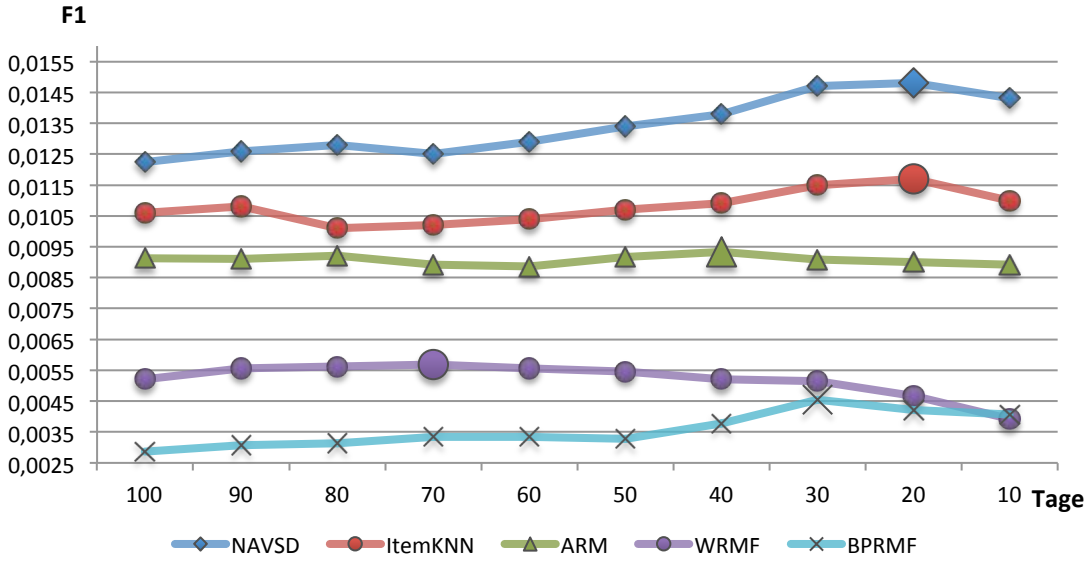

Abbildung 5.5: Sensitivität der Transaktionshistorie für Datensatz DS01

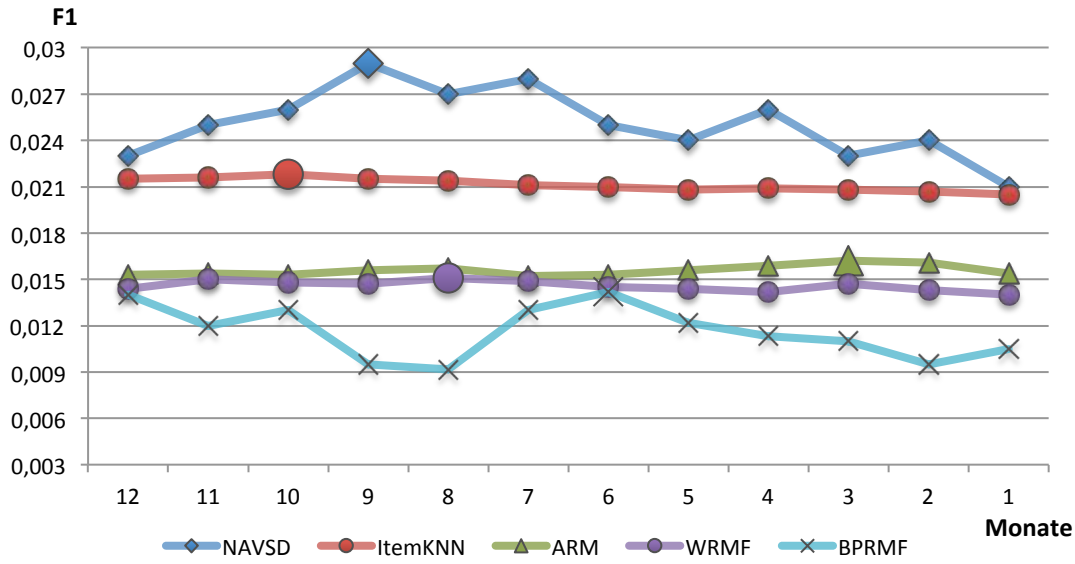

Abbildung 5.6: Sensitivität der Transaktionshistorie für Datensatz DS02

Zusammenfassend ist zu sagen, dass Empfehlungsalgorithmen durch zeitliche Aspekte der Transaktionen beeinflusst werden. Somit empfiehlt es sich, im E-Commerce Zeitberücksichtigende Methoden in Empfehlungssystemen zu übernehmen. Dass solche Verfahren zu besseren Ergebnissen führen können, zeigt eine Studie mit Filmbewertungsdaten von Koren [72].

5.5.5 Integration der Klickhistorie

Für den Datensatz DS01 steht neben der Kauf- auch die Klickhistorie zur Verfügung. Bisher stellt der Umgang mit Klickdaten in kollaborativen Empfehlungssystemen im E-Commerce ein eher unerforschtes Thema dar. Wie in Kapitel 5.2 beschrieben wurde, ist ein Klick in dieser Versuchsreihe als ein eigenständiges Item zu verstehen, welches analog zu Käufen behandelt wird. Die Abbildungen 5.7 und 5.8 zeigen, wie sich die Integration der Klickdaten auf die Empfehlungsgenauigkeit der einzelnen Methoden auswirkt. Wie erwartet steigt die F1-Metrik für alle Algorithmen. Je mehr Items zur Verfügung stehen, desto höher ist die Wahrscheinlichkeit für hoch korrelierende Produkte bzw. gute Prädiktoren. Während das NAVSD-, ItemKNN- sowie WRMF-Verfahren einen großen Zuwachs erfahren (ca. 20-30%), weisen das ARM- sowie BPRMF-Verfahren einen kleinen Gewinn auf (ca. 5%). Dass der ARM-Algorithmus keinen hohen Zuwachs erfährt, begründet sich mit der Tatsache, dass maximal zehn historische Käufe bzw. Klicks, welche die stärksten Assoziationen aufweisen, in die Prognose für einen Kunden einfließen. Im NAVSD- sowie ItemKNN-Verfahren hingegen werden eine höhere Anzahl an Interessantheits- bzw. Ähnlichkeitswerten aggregiert. So fließen mehr Klicks und Käufe in die Prognose ein. Dass die Prognosegenauigkeit des BPRMF-Algorithmus nicht wesentlich steigt, könnte daran liegen, dass die Trainingsdaten durch die Hinzunahme der Klicks noch spärlicher besetzt sind. Das Verhältnis der vorhandenen zu den

fehlenden Bewertungen ist somit kleiner geworden. So kommt den Vergleichen zwischen bekannten und unbekannten Bewertungen weniger Bedeutung zu, worunter letztlich die Prognostizierbarkeit der Trainingsdaten leidet.

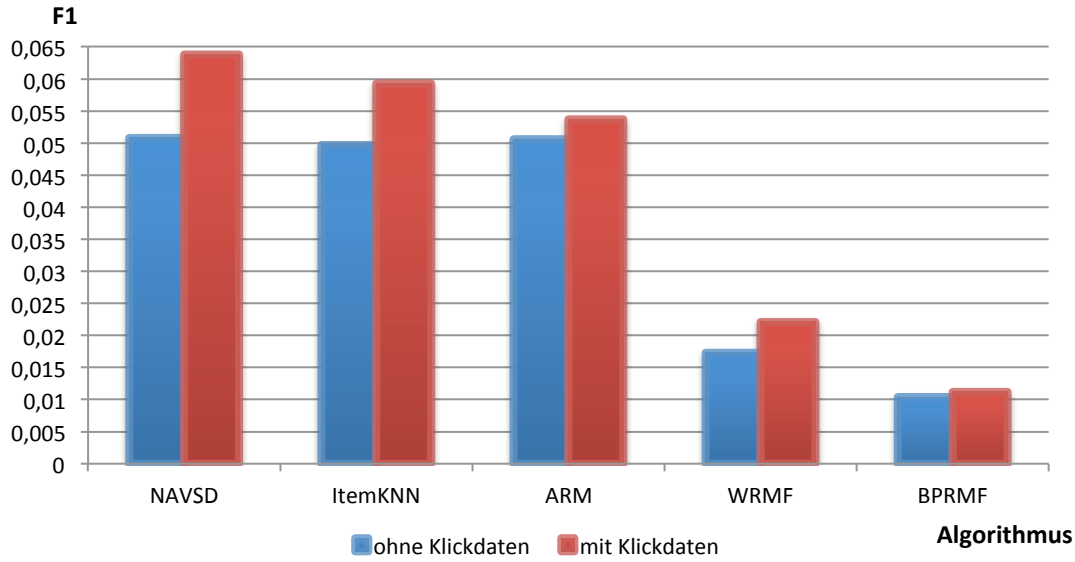

Abbildung 5.7: Auswirkung der Klickdaten auf die F1-Metrik für DS01 unter O1

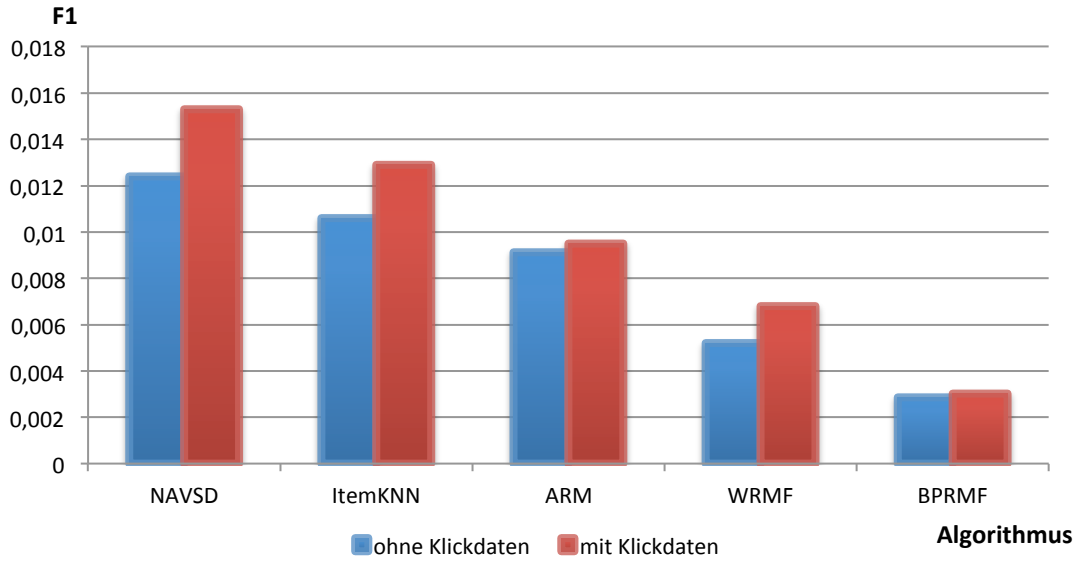

Abbildung 5.8: Auswirkung der Klickdaten auf die F1-Metrik für DS01 unter O2

5.5.6 Warengruppen statt Artikel

In diesem Abschnitt untersuche ich, wie sich das Ersetzen von Produkten durch Warengruppen auf die Empfehlungsqualität auswirkt. Dazu stehen mir Warengruppendaten für den Datensatz DS01 zur Verfügung. Die Evaluierung erfolgt unter Option O2 und kann auf zwei unterschiedlichen Wegen erfolgen: auf Warengruppen- oder auf Produktebene. Während bei der Evaluierung auf Warengruppenebene in allen Datensätzen das Feld *Item* mit Warengruppen

gefüllt ist, besteht bei der Evaluierung auf Produktebene der Testdatensatz aus Produkten. Für letztere Evaluationsvariante wird am Ende des Anwendungsprozesses jede Warengruppe der Top-10-Liste in ein jeweils aus ihr stammendes Produkt umgewandelt. Der Produktselektionsalgorithmus berücksichtigt bei der Wahl eines Artikels die Wahrscheinlichkeiten, mit denen Produkte einer Warengruppe in der Vergangenheit gekauft wurden.

Die Tabellen 5.11 und 5.12 zeigen die Ergebnisse für beide Evaluationsarten. Die Prognosegenauigkeit auf Warengruppenebene ist deutlich höher als die in Abschnitt 5.5.3 ermittelten Werte (siehe Tabelle 5.5). Dies erscheint jedoch angesichts der Daten und der Evaluationsart wenig überraschend. Schließlich genügt der Kauf eines Produktes aus der empfohlenen Warengruppe, um von einer relevanten Empfehlung zu sprechen. Die Genauigkeit auf Produktebene ist deutlich niedriger, da letztlich auf zufällige Weise Artikel aus einer Warengruppe selektiert werden, die dann empfohlen werden. Ob Empfehlungsverfahren auf Basis von Warengruppen jedoch tatsächlich schlechter sind als auf Basis von Produkten kann durch Offline-Experimente nicht ermittelt werden. Vielmehr müssten dazu Online-Experimente anhand von Kunden-Fallstudien durchgeführt werden, in denen Nutzer mit dem Empfehlungssystem interagieren sollen, um herauszufinden, ob und inwieweit sie von Empfehlungen beeinflusst werden [99]. Solche Experimente würden den Rahmen dieser Arbeit sprengen und wurden aus diesem Grund nicht weiter verfolgt.

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0292**	0,0737	**0,0418**
ItemKNN	0,0263	**0,0909**	0,0407
ARM	0,0223	0,0687	0,0337
BPRMF	0,0225	0,0698	0,0340
WRMF	0,0249	0,0712	0,0369

Tabelle 5.11: Genauigkeit der Verfahren auf Warengruppenebene

Methode	Prec@10	Rec@10	F1@10
NAVSD	**0,0021**	0,0040	**0,0028**
ItemKNN	0,0015	**0,0043**	0,0022
ARM	0,0008	0,0027	0,0012
BPRMF	0,0007	0,0024	0,0011
WRMF	0,0011	0,0033	0,0017

Tabelle 5.12: Genauigkeit der Verfahren auf Produktebene

Bezüglich der Rangfolge zeichnet sich dasselbe ab, was sich bereits auf Basis von Produkten andeutete: MF-Methoden profitieren am meisten von der sinkenden Spärlichkeit. Diese kommt daher, dass die Warengruppenmatrix weniger Spalten bzw. Items beinhaltet, die zudem dichter besetzt sind. Während das NAVSD- und das ItemKNN-Verfahren die genausten Empfehlungen generieren, ist das WRMF-Verfahren vor dem BPRMF- sowie ARM-Algorithmus anzusiedeln.

Des Weiteren wurde in diesem Zusammenhang die Coverage untersucht. Dazu wurde der CC-Wert sowohl auf Basis von Produkten als auch auf Basis von Warengruppen ermittelt. Beide Varianten wurden letztlich auf Artikelebene evaluiert. Die Ergebnisse in Abbildung 5.9 deuten darauf hin, dass durch das Arbeiten mit Warengruppendaten eine deutlich größere Bandbreite an Produkten in Empfehlungslisten abgedeckt wird. Die Coverage erfährt einen Anstieg von ca. 40%. Der Hauptgrund liegt in der geringeren Spärlichkeit der Warengruppendaten. Selbst

selten gekaufte Warengruppen weisen immerhin so viele Käufe auf, dass sie hohe Scores erzeugen können. Dies ist ein großer Vorteil gegenüber Produktdaten, in denen einige Korrelationen aufgrund zu kleiner Schnittmengen entweder verworfen oder zu klein geschätzt werden.

Auffällig ist zudem, dass die Rangfolge bezüglich der Coverage in nahezu umgekehrtem Verhältnis gegenüber der Rangfolge bezüglich der Prognosegenauigkeit steht. Die Schwäche traditioneller kollaborativer Empfehlungsalgorithmen, die reduzierte Coverage, konnte das NAVSD-Verfahren nicht beseitigen. Ihr Coverage-Level befindet sich ungefähr auf dem des ItemKNN-Verfahrens. Eine Abwägung zwischen Genauigkeit und Coverage, wie sie bereits Ge et al. [41] beschrieben, können durch die vorliegenden Ergebnisse somit bestätigt werden.

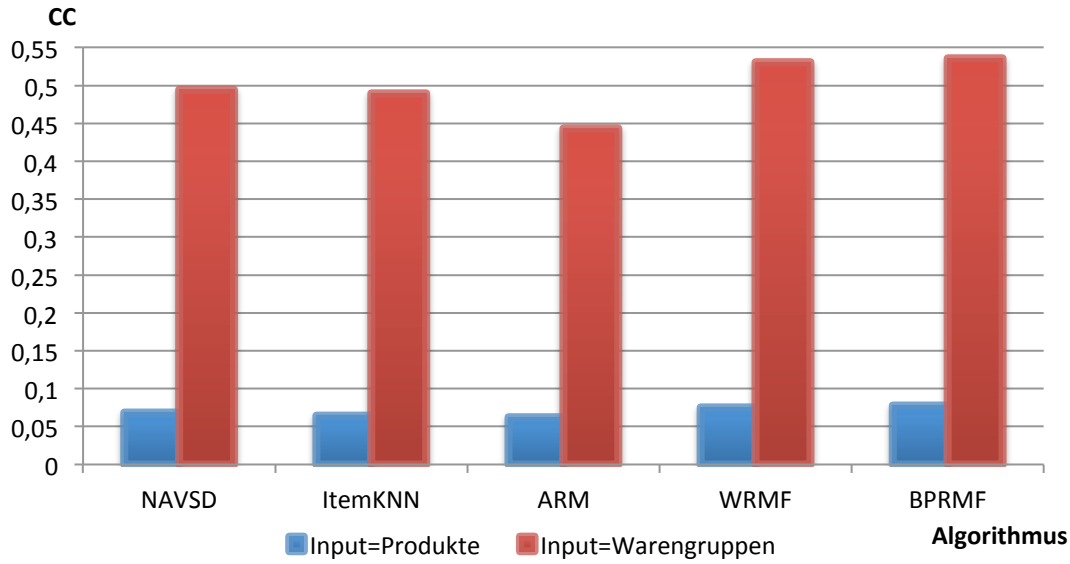

Abbildung 5.9: Auswirkung des Ersetzens von Produkten durch Warengruppen auf die Coverage der verschiedenen Algorithmen

Zusammenfassend ist zu sagen, dass Warengruppendaten durchaus einen Mehrwert für Genauigkeit und Coverage von Empfehlungsalgorithmen bieten können. Durch ihre dichter besetzte Bewertungsmatrix können sie einige Nachteile der spärlicheren Produktdaten aushebeln. Auf der anderen Seite können produktspezifische Beziehungen ebenfalls sehr gewinnbringend sein. Welche Daten vorzuziehen sind oder wie sie gewinnbringend miteinander kombiniert werden können, wird nicht weiter diskutiert. Besitzen Warengruppen die passende Granularität, können sie unabhängig davon, ob neben oder anstatt von Produktdaten, die Empfehlungsqualität erhöhen.

6 Fazit und Ausblick

Empfehlungssysteme stellen ein wichtiges Themengebiet in der Forschung dar und dienen gleichzeitig als umsatzgenerierendes Werkzeug im E-Commerce. Aus der Kategorie kollaborativer Systeme gelten zwei Verfahren als State-of-the-art: nachbarschaftsbasierte KF- sowie MF-Methoden. Des Weiteren wurde das von mir entwickelte NAVSD-Verfahren beleuchtet, welches zwei neue Komponenten enthält: Hier sind das NAVSD-Trainingsverfahren und der Diskretisierungsprozess zu nennen. Während ersteres Assoziationen zwischen Produkten aus der Vergangenheit und der Gegenwart berechnet, wandelt letzteres Produktscores in echte Wahrscheinlichkeiten um.

Das Ziel der vorliegenden Arbeit war es, zu ermitteln, welche Faktoren kritisch für die Empfehlungsqualität sind. Der Fokus lag hierbei auf der Untersuchung verschiedener Empfehlungsalgorithmen. Eine Frage, die es zu klären gab, war, welche dieser Algorithmen bezüglich der Prognosegenauigkeit die besten Ergebnisse für typische E-Commerce-Daten liefern, die keine expliziten, sondern spärliche implizite Bewertungen enthalten. Wie die Untersuchung gezeigt hat, generiert das NAVSD-Verfahren die genausten Empfehlungslisten. Zudem rechtfertigen die dargestellten Ergebnisse die Aussage, dass im Falle impliziter Daten *traditionelle, einfache Verfahren* genauer sind als MF-Methoden. Als kritische Faktoren für die Rangfolge der Algorithmen erwiesen sich neben den Parametern der einzelnen Verfahren, die Spärlichkeit der Daten sowie die Splittechnik bei der Evaluation. Der bekannte nicht-chronologische Split eruiert zu optimistische Genauigkeiten. Der chronologische Split, welcher vergangene Ereignisse trainiert, um zukünftige vorherzusagen, bietet einen deutlich realistischeren Aufbau, und liefert deutlich geringere Quoten. Zudem erwies sich das NAVSD-Trainingsverfahren gegenüber dem Standardtrainingsverfahren als deutlich bessere Variante der Assoziationsberechnung. Über die Rangfolge bezüglich der Laufzeit lässt sich ähnliches festhalten. Jedoch ist das NAVSD-Verfahren aufgrund ihres Diskretisierungsprozesses langsamer als die traditionellen KF-Methoden. Bezüglich der Coverage ergab sich jedoch umgekehrtes Bild: Das BPRMF- sowie WRMF-Verfahren decken mit ihren Empfehlungen die meisten Produkte ab. Der NAVSD-Algorithmus konnte letztlich keinen Beitrag zur Lösung des Problems der reduzierten Coverage leisten. Dennoch stellt er aufgrund seiner hohen Prognosegenauigkeit mehr als eine Alternative zu den State-of-the-art Verfahren dar und bietet mit ihren Komponenten neue Ansätze, die es Zukunft weiter zu erforschen gilt.

Der Faktor Zeit ist ebenfalls in Form von Aktualität und Sequenz der Transaktionen ein wesentlicher Faktor. Abhängig von den Handelswaren weisen E-Commerce-Datensätze unterschiedliche Effekte der Saisonalität auf. Die Ergebnisse deuten an, dass das Trainieren der gesamten Transaktionshistorie nicht immer optimal sein muss.

Untersucht wurden weiterhin die Auswirkungen der Klickhistorie und der Warengruppen. Insgesamt lässt sich sagen, dass die Empfehlungsqualität durch solche Datensätze erhöht werden kann. Die Prognosegenauigkeit kollaborativer Empfehlungssysteme scheint mit der Menge der Daten zu steigen. Mithilfe von Warengruppendaten kann aufgrund ihrer dichter besetz-

ten Matrix das Coverage-Problem angegangen werden. Die Experimente mit solchen Daten hatten primär den Zweck, ihre Spezifika aufzuzeigen und Auswirkungen auf die Empfehlungsqualität zu untersuchen. Ob und inwieweit Produkt- und Warengruppendaten gemeinsam effizient eingesetzt werden können, ist in dieser Arbeit nicht geklärt worden. Während der Untersuchungen mit den Klickdaten wurden Kauf und Klick desselben Produktes als unterschiedliche *Items* behandelt. Der Ansatz, anhand der Anzahl der Klicks das Konfidenzlevel eines Nutzers zu berechnen, sofern dieser das Produkt nicht gekauft hat, stellt eine lohnenswerte Aufgabe für zukünftige Untersuchungen dar.

Für zukünftige Studien sind zwei weitere Aspekte zu nennen: Zum einen sind effiziente Lösungen für die untersuchten Faktoren wünschenswert. Zum anderen stellt die Online-Evaluation von auf impliziten Daten basierenden Empfehlungssystemen ein erforschenswertes Thema dar.

Literaturverzeichnis

[1] G. Adomavicius, J. Bockstedt, S. Curley & J. Zhang. Do recommender systems manipulate consumer preferences? A study of anchoring effects. *SSRN*, 2013.

[2] G. Adomavicius & A. 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.

[3] C. Aggarwal: Recommender Systems: The Textbook. Springer, 2016. ISBN 978-3-319-29659-3.

[4] C. Aggarwal & J. Han. Frequent pattern mining. Springer, New York, 2014. ISBN 978-3-319-07821-2.

[5] F. Aiolli. Efficient top-n recommendation for very large scale binary rated datasets. In *Proceedings of the 7th ACM Conference on Recommender Systems*, 273–280, 2013.

[6] X. Amatriain & J. Basilico. Netflix recommendations: Beyond the 5 stars. 2012. Unter http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html (abgerufen am 12.08.2016).

[7] X. Amatriain, N. Lathia, J. M. Pujol, H. Kwak & N. Oliver. The wisdom of the few: A collaborative filtering approach based on expert opinions from the web. In *Proc. of SIGIR '09*, 2009.

[8] M. Balabanovic. An Adaptive Web Page Recommendation Service. In *Agents 97: Proceedings of the First International Conference on Autonomous Agents*, Marina Del Rey, CA, 378-385, 1997.

[9] M. Balabanovic & Y. Shoham. Fab: Content-Based, Collaborative Recommendation. Comm. ACM, vol. 40, no. 3, 66-72, 1997.

[10] C. Basu, H. Hirsh & W. Cohen: Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, 714-720, 1998.

[11] R. Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex Systems*, 3(4), 331–342, 1989.

[12] J. Beel, S. Langer, M. Genzmehr, B. Gipp, C. Breitinger & A. Nürnberger. Research Paper Recommender System Evaluation: A Quantitative Literature Survey. *Proceedings of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys)*, 2013.

[13] R. Bell & Y. Koren. Improved neighborhood-based collaborative filtering. In *Proceedings of KDD Cup and Workshop*, 2007.

[14] D. Billsus & M. Pazzani, Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning (ICML '98)*, 1998.

[15] D. Billsus & M. Pazzani. User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, vol. 10, nos. 2- 3, pp. 147-180, 2000.

[16] J. Breese , D. Heckerman & C. Kadie: Empirical analysis of predictive algorithms for collaborative filtering. In *Proc. of the 14th Annual Conf. on Uncertainty in Artificial Intelligence*, 43–52, 1998.

[17] D. Bridge, M. Goker, L. McGinty & B. Smyth. Case-based recommender systems. *The Knowledge Engineering Review*, 20(3), 315–320, 2005.

[18] S. Brin, R. Motwani, J. D. Ullman & S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data* (ACM SIGMOD '97), 265-276, 1997.

[19] J. Buder & C. Schwind. Learning with personalized recommender systems: A psychological view. *Computers in Human Behavior, 28*(1), 207–216, 2012.

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

[21] R. Burke. Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web*, LNCS 4321, 377–408. Berlin Heidelberg, Germany: Springer, 2007.

[22] S. Castagnos, N. Jones & P. Pu. Recommenders' influence on buyers' decision process. In *Proceedings of the Third ACM Conference on Recommender Systems (RecSys '09)*, 361–364. New York, NY, USA: ACM, 2009.

[23] Y. C. Chang. N-dimension golden section search. Its variants and limitations. In *Biomedical Engineering and Informatics, 2009. BMEI'09. 2nd International Conference,* 1-6. IEEE, 2009.

[24] S. H. S. Chee, J. Han & K. Wang. RecTree: an efficient collaborative filtering method, In *Proceedings of the 3rd International Conference on Data Warehousing and Knowledge Discovery*, 141–151, 2001.

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

[26] M. O'Connor & J. Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR Workshop on Recommender Systems (SIGIR '99)*, 1999.

[27] T. Cover & P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions*, 13(1):21–27, 1967.

[28] P. Cremonesi, F. Garzotto & R. Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *ACM Transactions on Interactive Intelligent Systems, 2*(2), Article No. 11, 1-41, 2012.

[29] P. Cremonesi, F. Garzotto & R. Turrin. User effort vs. accuracy in rating-based elicitation. In *Proceedings of the sixth ACM conference on Recommender systems (RecSys '12)*, 27–34. New York, NY, USA: ACM, 2012.

[30] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer & R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, vol. 41, no. 6, 391–407, 1990.

[31] M. Deshpande & G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143-177, 2004.

[32] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Comput. Linguist.*, vol. 19, no. 1, 61–74, 1993.

[33] E-Commerce-Center Handel in Kooperation mit der prudsys AG. Produktempfehlungen als Erfolgsfaktor im Online-Handel – Aktuelle Studienergebnisse und Handlungsempfehlungen. Unter https://www.prudsys.de/nc/webinar.html?tx_drblob_pi1%5BdownloadUid%5D= 586 (abgerufen am 09.08.2016).

[34] B. Efron. Bootstrap Methods: Another Look at the Jackknife. In *The Annals of Statistics*. 7, Nr. 1, 1-26, 1979.

[35] M. Ester, H. P. Kriegel, J. Sander & X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD '96)*, 1996.

[36] A. Felfernig & R. Burke. Constraint-based recommender systems: technologies and research issues. *International conference on Electronic Commerce*, 2008.

[37] F. Fouss, J. M. Renders, A. Pirotte, M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering* 19(3), 355–369, 2007.

[38] N. Friedman, D. Geiger & M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, vol. 29, no. 2-3, 131–163, 1997.

[39] S. Funk. Netflix Update: Try This at Home. 2006. Unter http://sifter.org/~simon/journal/ 20061211.html (abgerufen am 28.07.2016).

[40] Z. Gantner, S. Rendle, C. Freudenthaler & L. Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the 5th ACM Conference on Recommender Systems*, 2011.

[41] M. Ge, C. Delgado-Battenfeld & D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *RecSys*, 2010.

[42] F. Gedikli & D. Jannach. Neighborhood-restricted mining and weighted application of association rules for recommenders. *Web Information Systems Engineering–WISE 2010*. Springer Berlin Heidelberg, 157-165, 2010.

[43] S. Geman & D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, no. 6, 721–741, 1984.

[44] D. Goldberg, D. A. Nichols, B. M. Oki & D. B. Terry. Using Collaborative Filtering to Weave an Information Tapestry. In *Communications of the ACM* 35 (1992), Nr. 12, 61–70, 1992.

[45] K. Goldberg, T. Roeder, D. Gupta & C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2), 133–151, 2001.

[46] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. L. Herlocker & J. Riedl. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of the Conference of the American Association of Artificial Intelligence (AAAI-99)*, 439-446, Orlando, Florida, 1999.

[47] R. Greinemr, X. Su, B. Shen & W. Zhou. Structural extension to logistic regression: discriminative parameter learning of belief net classifiers. *Machine Learning*, vol. 59, no. 3, 297–322, 2005.

[48] U. Gretzel & D. Fesenmaier. Persuasion in recommender systems. *International Journal of Electronic Commerce, 11*(2), 81–100, 2006.

[49] R. Hamming. Numerical methods for scientists and engineers. Courier Dover Publications, 2012.

[50] J. Han & M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, Calif, USA, 2011. ISBN 978-0123814791.

[51] R. Hangartner. What is the recommender industry? Unter http://www.mobilegroove.com/guest-column-what-is-the-recommender-industry-750 (abgerufen am 09.07.2016).

[52] F. M. Harper, X. Li, Y. Chen & J. A. Konstan. An economic model of user rating in an online recommender system. In L. Ardissono, P. Brna, & A. Mitrovic (Eds.), *Proceedings of the 10th International Conference on User Modeling (UM'05)*, LNCS 3538, 307–316. Berlin Heidelberg, Germany: Springer, 2005.

[53] J. L. Herlocker, J. A. Konstan, A. Borchers & J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proc. of the 22nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, 230–237. ACM, New York, NY, USA, 1999.

[54] J. L. Herlocker, J. A. Konstan, L. G. Terveen & J. T. Riedl. Evaluating Collaborative Filtering Recommender Systems. *ACM Trans. on Inf. Sys.*, 22(1):5–53, 2004.

[55] B. Hidasi & D. Tikk. Enhancing matrix factorization through initialization for implicit feedback databases. In *Proceedings of the 2nd Workshop on Context-awareness in Retrieval and Recommendation*, 2-9, ACM, 2012.

[56] W. Hill, L. Stead, M. Rosenstein & G. Furnas. Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, 194–201. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[57] T. Hofmann. Latent Semantic Models for Collaborative Filtering. *ACM Transactions on Information Systems* 22, 89–115, 2004.

[58] T. Hothorn, K. Hornik & A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[59] A. E. Howe & R. D. Forbes. Re-considering neighborhood-based collaborative filtering parameters in the context of new data. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, 1481–1482. ACM, New York, NY, USA, 2008.

[60] Y. Hu, Y. Koren & C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, 263–272, 2008.

[61] N. Hu, L. Liu & J.J. Zhang. Do online reviews affect product sales? The role of reviewer characteristics and temporal effects. *Information Technology & Management, 9*(3), 201–214, 2008

[62] Z. Huang, D. Zeng & H. Chen. A comparative study of recommendation algorithms for e-commerce applications, *IEEE Intelligent Systems*, 2006.

[63] P. Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. Bulletin de la Société Vaudoise des Sciences Naturelles, Vol. 37, 241-272, 1901.

[64] G. Karypis. Evaluation of item-based top-N recommendation algorithms. In ACM CIKM '01, 247–254, ACM, 2001.

[65] C. Kim & J. Kim. A recommendation algorithm using multi-level association rules. In *Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, ser. WI '03, 2003.

[66] R. Kiran, R. Uday & M. Kitsuregawa. An improved neighborhood-restricted association rule-based recommender system. *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*. Australian Computer Society, Inc., 2013.

[67] B. P. Knijnenburg, N. J. M. Reijmer & M. C. Willemsen. Each to his own: How different users call for different interaction methods in recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys '11)*, 141–148. New York, NY, USA: ACM, 2011.

[68] S. Y. X. Komiak & I. Benbasat. The effects of personalization and familiarity on trust and adoption of recommendation agents. *MIS Quarterly, 30*(4), 941–960, 2006.

[69] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, J. Riedl. GroupLens: applying collaborative filtering to usenet news. Communications of the ACM 40(3), 77–87 1997.

[70] J. A. Konstan & J. Riedl. Recommender systems: From algorithms to user experience. *User Modeling and User-Adapted Interaction, 22*, 101– 123, 2012.

[71] Y. Koren. Tutorial on recent progress in collaborative filtering. In *Proceedings of the the 2nd ACM Conference on Recommender Systems*, 2008.

[72] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, 2009.

[73] Y. Koren. The BellKor Solution to the Net ix Grand Prize. KorBell Team's Report to Net ix, 2009.

[74] Y. Koren, R. Bell & C. Volinsky. Matrix Factorization Techniques for Recommender Systems. Computer, 42(8):30–37, 2009.

[75] N. Lavrac, P. A. Flach & B. Zupan. Rule evaluation measures: A unifying view. In Dzeroski, S., Flach, P.A. (eds.) ILP 1999. LNCS (LNAI), vol. 1634, 174–185, 1999.

[76] T. Q. Lee, Y. Park & Y.-T. Park. An Empirical Study on Effectiveness of Temporal Information as Implicit Ratings. *Expert Syst. Appl.*, 36(2):1315–1321, 2009.

[77] D. Lemire. Scale and translation invariant collaborative filtering systems. *Information Retrieval*, vol. 8, no. 1, 129–150, 2005.

[78] G. Linden, B. Smith & J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing*, IEEE, 7(1), 76-80, 2003.

[79] R. J. A. Little. Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics*, vol. 6, no. 3, 287–296, 1988.

[80] N. Littlestone & M. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, vol. 108, no. 2, pp. 212- 261, 1994.

[81] I. MacKenzie, C. Meyer & S. Noble. How retailers can keep up with consumers. 2013. Unter http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers (abgerufen am 27.08.2016).

[82] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Symposium on Math, Statistics, and Probability*, 281–297, Berkeley, Calif, USA, 1967.

[83] P. Melville, R. J. Mooney & R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI '02)*, 187–192, Edmonton, Canada, 2002.

[84] K. Miyahara & M. J. Pazzani. Collaborative filtering with the simple Bayesian classifier. In *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, 679–689, 2000.

[85] K. Miyahara & M. J. Pazzani. Improvement of collaborative filtering with the simple Bayesian classifier. *Information Processing Society of Japan*, vol. 43, no. 11, 2002.

[86] R. J. Mooney & L. Roy. Content-Based Book Recommending Using Learning for Text Categorization. SIGIR ’99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA, 1999.

[87] A. Ng & M. Jordan. PEGASUS: a policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI ’00)*, Stanford, Calif, USA, 2000.

[88] S. Owen, R. Anil, T. Dunning & E. Friedman, *Mahout in Action*. Manning Publications Co., 2011.

[89] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz & Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM ’08, 502–511, 2008.

[90] B. Pathak, R. Garfinkel, R. Gopal, R. Venkatesan & F. Yin. Empirical analysis of the impact of recommender systems on sales. *Journal of Management Information Systems, 27*(2), 159–188, 2010

[91] M. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13, 5–6, 1999.

[92] M. Pazzani & D. Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, vol. 27, 313-331, 1997.

[93] A. Pommeranz, J. Broekens, P. Wiggers, W. P. Brinkman & C. M. Jonker. Designing interfaces for explicit preference elicitation: a user-centered investigation of preference representation and elicitation process. *User Modeling and User-Adapted Interaction*, 22(4–5), 357–397, 2012.

[94] B. Pradel, S. Sean, J. Delporte, S. Guerif, C. Rouveirol, N. Usunier, F. Fogelman-Soulie & F. Dufau-Joel. A case study in a recommender system based on purchase data. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, 377–385, 2011.

[95] M. M. Recker, A. Walker & K. Lawless. What do you recommend? Implementation and analyses of collaborative information filtering of web resources for education. *Instructional Science, 31*, 299–316, 2003.

[96] S. Rendle, C. Freudenthaler, Z. Gantner & L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, ser. UAI '09, 2009.

[97] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom & J. Riedl. GroupLens: An open archit ecture for collaborative filtering of netnews. In *CSCW '94: Proc. of the 1994 ACM Conf. on Computer Supported Cooperative Work*, 175–186. ACM, New York, NY, USA, 1994.

[98] P. Resnick & H. R. Varian. Recommender systems. *Communications of the ACM*, vol. 40, no. 3, 56–58, 1997.

[99] F. Ricci, L. Rokach, B. Shapira & P. B. Kantor (Herausgeber): Recommender Systems Handbook. Springer, 2015. ISBN 978-1489976369.

[100] E. Rich. User modeling via stereotypes. Cognitive Science, 3(4), 329–354, 1979.

[101] J. J. Rocchio. Relevance Feedback in Information Retrieval. *SMART Retrieval System-Experiments in Automatic Document Processing*, G. Salton, ed., chapter 14, Prentice Hall, 1971.

[102] D. B. Rubin. Multiple Imputation for Nonresponse in Surveys, John Wiley & Sons, New York, NY, USA, 1987.

[103] A. Said, A. Bellogın & A. de Vries. A top-n recommender system evaluation protocol inspired by deployed systems. *LSRS Workshop at ACM RecSys*, 2013.

[104] R. Salakhutdinov, A. Mnih & G. Hinton. Restricted Boltzmann Machines for Collaborative Filtering. *Proc. 24th Annual International Conference on Machine Learning*, 791–798, 2007.

[105] G. Salton. Automatic Text Processing. Addison-Wesley, 1989.

[106] G. Salton & M. McGill. Introduction to Modern Information Retrieval. McGraw-Hill, New York, NY, USA, 1983.

[107] B. Sarwar, G. Karypis, J. Konstan & J. Riedl. Analysis of Recommendation Algorithms for e-Commerce. In *Proc. of the 2nd ACM Conf. on Electronic Commerce*, 158–167, 2000.

[108] B. Sarwar, G. Karypis, J. Konstan & J. Riedl. Application of dimensionality reduction in recommender system-a case study. *Minnesota Univ Minneapolis Dept of Computer Science*, 2000.

[109] B. Sarwar, G. Karypis, J. Konstan & J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference*, 285-295, 2001.

[110] B. Sarwar, G. Karypis, J. Konstan & J. Riedl. Incremental SVD-based algorithms for highly scaleable recommender systems. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT '02)*, 2002.

[111] B. Sarwar, G. Karypis, J. Konstan & J. Riedl. Recommender systems for large-scale E-commerce: scalable neighborhood formation using clustering. In *Proceedings of the 5th International Conference on Computer and Information Technology (ICCIT '02)*, 2002.

[112] J. B. Schafer, D. Frankowski, J. Herlocker & S. Sen. Collaborative filtering recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The Adaptive Web*, LNCS 4321, 291–324. Berlin Heidelberg, Germany, 2007.

[113] J. B. Schafer, J. Konstan & J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce (EC '99)*, 158–166. New York, NY, USA, 1999.

[114] J. B. Schafer, J. Konstan & J. Riedl. E-Commerce recommendation appcations. *Data Mining and Knowledge Discovery, 5*(1–2), 115–153, 2001.

[115] U. Shardanand & P. Maes. Social information filtering: Algorithms for automating "word of mouth". In CHI '95: *Proc. of the SIGCHI Conf. on Human factors in Computing Systems*, 210–217. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[116] B. Sheth & P. Maes. Evolving Agents for Personalized Information Filtering. *Proc. Ninth IEEE Conf. Artificial Intelligence for Applications*, 1993.

[117] C. Silverstein, S. Brin & R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2:39–68, 1998.

[118] B. Smyth & P. Cotter. A Personalized TV Listings Service for the Digital TV Age. *Knowledge-Based* Systems 13: 53-59, 2000.

[119] H. Strasser & C. Weber. On the asymptotic theory of permutation statistics. *Mathematical Methods of Statistics*, 8:220–250, 1999.

[120] X. Su & T. M. Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. In *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI '06)*, 497–504, 2006.

[121] X. Su, T. M. Khoshgoftaar & R. Greiner. A mixture imputation-boosted collaborative filter. In *Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference (FLAIRS '08)*, 312–317, Coconut Grove, Fla, USA, 2008.

[122] X. Su, T. M. Khoshgoftaar, X. Zhu & R. Greiner. Imputation-boosted collaborative filtering using machine learning classifiers. In *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC '08)*, 949–950, Ceará Fortaleza, Brazil, 2008.

[123] X. Su, M. Kubat, M. A. Tapia & C. Hu. Query size estimation using clustering techniques. In *Proceedings of the 17th International Conference on Tools with Artificial Intelligence (ICTAI '05)*, 185–189, Hong Kong, 2005.

[124] M. Svensson, K. Höök & R. Cöster. Designing and evaluating Kalas: A social navigation system for food recipes. *ACM Transactions on Computer-Human Interaction, 12*(3), 374–400, 2005.

[125] G. Takacs, I. Pilaszy, B. Nemeth & D. Tikk. Matrix factorization and neighbor based algorithms for the Netflix prize problem. *ACM Conference on Recommender Systems*, 267–274, 2008.

[126] P. Tan, V. Kumar & J. Srivastava. Selecting the right interestingness measure for association patterns. *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining*, 32-41, 2002.

[127] L. H. Ungar & D. P. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshopon Recommendation Systems*, AAAI Press, 1998.

[128] C. P. Wei, M. J. Shaw & R. F. Easley. A survey of recommendation systems in electronic commerce. In Rust, R. T., & Kannan, P. K. (Eds.), *E-Service: New Directions in Theory and Practice*. Armonk, NY, USA, 2002.

[129] N. Xia, C. Desrosiers & G. Karypis. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. *Recommender Systems Handbook*. Springer US, 37-76, 2015.

[130] B. Xiao & I. Benbasat. E-commerce product recommendation agents: Use, characteristics, and impact. *MIS Quarterly, 31*(1), 137–209, 2007.

[131] J. Xu, K. Johnson-Wahrmann & S. Li. The development, status and trends of recommender systems: a comprehensive and critical literature review. In *Proceedings of International Conference Mathematics and Computers in Science and Industry*. 117–122, 2014.

[132] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the ACM SIGIR Conference*, 114–121, Salvador, Brazil, 2005.

[133] K. Yu, A. Schwaighofer, V. Tresp, X. Xu & H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, 56–69, 2004.

[134] T. Zhang, R. Ramakrishnan & M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, vol. 25, 103–114, Montreal, Canada, 1996.

[135] Y. Zhou, D. Wilkinson, R. Schreiber & R. Pan. Large-scale parallel collaborative filtering for the Netflix prize. In *AAIM '08*, 337–348, Berlin, Heidelberg, Springer-Verlag, 2008.