HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG

Fakultät Design, Medien und Information
Department Medientechnik

Entwicklung eines Recommender Systems

Unterschiedliche Strategien anhand eines Travel- & Outdoor Online Shops

Bachelor-Thesis
zur Erlangung des akademischen Grades B.Sc.

Hoang Tuong Vy Nguyen
[redacted]

Erstprüferin: Prof. Dr.-Ing. Sabine Schumann
Zweitprüfer: Prof. Dr. Edmund Weitz

Hamburg, 17. 03. 2020

Inhaltsverzeichnis

Abkürzungsverzeichnis

ALS	Alternating Least Squares
AUC	Area Under The ROC Curve
CBF	Content-Based Filtering
CF	Collaborative Filtering
IB-CF	Item-Based Collaborative Filtering
IR	Information Retrieval
kNN	k-Nearest Neighbour
LOOCV	Leave One Out Cross Validation
LV	Latentes Variablenmodell
MAE	Mean Absolute Error
MF	Matrixfaktorisierung
NLP	Natural Language Processing
PCA	Principal Component Analysis
RAKE	Rapid Automatic Keyword Extraction
RMSE	Root-Mean-Square Error
ROC	Receiver Operating Characteristics
SGD	Stochastic Gradient Descent
SVD	Singular Value Decomposition
TF-IDF	Term Frequency-Inverse Document Frequency
UB-CF	User-Based Collaborative Filtering

Abbildungsverzeichnis

Tabellenverzeichnis

Keywords

Content-Based Filtering, Collaborative Filtering, Matrix Factorization, k-Nearest Neighbour, Leave One Out Cross Validation, Association Rules, Hybrid Systems, implicit data

Abstract

These days recommender systems are an almost fundamental tool for businesses in E-commerce. They enable automated mass customization and personalization of a site and help with increasing sales and customer loyalty.
This paper describes the development process of such a system. Various recommender strategies are being presented and implemented with different metrics and algorithms using implicit user data. The advantages and disadvantages of each approach are being discussed. Then, the recommender models are being compared to each other and analysed in an offline evaluation. Furthermore, hybrid methods are being introduced to solve the item cold start and the filter bubble problem. The paper completes with a conclusion, an approach for online testing and suggestions to further improve the recommendation engine.

Stichworte

Content-Based Filtering, Collaborative Filtering, Matrixfaktorisierung, k-Nearest Neighbour, Leave One Out Cross Validation, Assoziationsregeln, Hybride Systeme, Implizite Daten

Zusammenfassung

Recommender Systeme sind heutzutage ein fast fundamentales Tool für Unternehmen im E-Commerce. Sie ermöglichen eine automatisierte Massenpersonalisierung einer Seite und helfen dabei, Umsatz und Kundenloyalität zu steigern.
In dieser Arbeit wird der Entwicklungsprozess eines solchen Systems beschrieben. Es werden unterschiedliche Empfehlungsstrategien auf Basis von impliziten Userdaten vorgestellt, die mit verschiedenen Metriken und Algorithmen umgesetzt werden. Dabei wird auf die Vor- und Nachteile jedes Ansatzes eingegangen. Anschließend wird eine Offline Evaluation durchgeführt, bei der die Recommender Modelle auf Basis verschiedener Kriterien miteinander verglichen werden. Es werden außerdem hybride Methoden vorgestellt, die die Probleme des Item-Kaltstarts und der Filter Bubble lösen sollen. Schließlich werden eine Schlussfolgerung gezogen und Aussichten für weiteres Online Testing und Optimierungsvorschläge gegeben.

1 Einleitung

1.1 Motivation

Recommender Systeme sind im E-Commerce allgegenwärtig. Gerade durch den Aufstieg von Unternehmen wie Youtube, Netflix und Co. gewinnen sie in der heutigen Zeit immer mehr an Bedeutung und Relevanz. Als bekanntes Beispiel ist hier Amazon zu erwähnen, die bis heute Marktführer sind [MART] und dessen Produktempfehlungen den Standard im E-Commerce setzen. 35% ihrer Einkäufe stammen von Produktempfehlungen, bei Netflix basieren 75% der angesehenen Filme und Serien auf Recommender Systemen [MACMN13]. Weiterhin kommen 40% der Applikationsinstallationen des Google Play Stores und 60% der Watchtime bei Youtube ebenfalls von Empfehlungen [GOOGLEREC]. Es ist also leicht zu erkennen, warum Recommender Systeme eine so wichtige Rolle beim Unternehmensgewinn spielen: Sie helfen dem User dabei, interessante Inhalte inmitten einer großen Masse an Produkten zu finden und beeinflussen immens sein Verhalten und die damit verbundene User Experience. Dies hat nicht nur Vorteile für das Unternehmen sondern auch für die NutzerInnen [AHR11]: Der Betrieb kann damit seine Gewinne erhöhen, Kosten reduzieren, Kundenloyalität steigern und sein Image aufbessern. Usern wird bei der Kaufentscheidung geholfen, was Qualität und Effizienz steigert, sie genießen eine optimierte Benutzererfahrung und bauen damit Vertrauen zu dem System auf. Durch diese automatisierte Personalisierung wird damit eine Bindung zwischen User und Seite aufgebaut. Auch in dem Zusammenhang schätzt Netflix selbst den Wert seiner Recommendation Engine auf rund eine Milliarde Dollar jährlich ein [MCA16], was wortwörtlich den Wert eines solchen Systems noch einmal verdeutlicht.
Nichtsdestotrotz bringen Recommender Systeme auch einige Kritikpunkte mit sich: Sie sind, wie bereits erwähnt, ein sehr großer Einflussfaktor auf das Verhalten der Nutzer, was sie dadurch zu einem mächtigen Instrument der heutigen Zeit macht. Auch der ethische Aspekt ist damit ein sehr interessanter Gesichtspunkt, der betrachtet werden kann und sollte. Wie Lex Fridman, ein Forschungswissenschafter für menschenzentrierte künstliche Intelligenz, autonomem Fahren und Deep Learning am MIT, treffend formulierte [FRID20]:
"I think the most exciting, the most powerful AI system space for the next couple of decades is recommendation systems. Very little talked about, it seems like, but they are going to have the biggest impact on our society because they affect how [we see] information [...], how we learn, what we think, how we communicate. These algorithms are controlling us."

1.2 Struktur der Arbeit

In dieser Arbeit wird der Entwicklungsprozess eines Recommender Systems für einen Travel- & Outdoor Online Shop der novomind AG beschrieben. Sie lässt sich, zusammen mit der Einleitung, in fünf Teile gliedern. Ziel dabei ist es, auf Grundlage dieser Arbeit eine Recommendation Engine für die novomind AG zu entwickeln, die passende Empfehlungen zu den Produkten eines Shops liefert.

Zunächst wird erläutert, was ein solches System ist, was die Vorteile dabei im E-Commerce sind und welchen Nutzen Recommender Systeme haben. Außerdem werden theoretische Grundlagen erklärt, die für die Entwicklung relevant sind: Es werden die drei Haupttypen eines Recommender Systems vorgestellt und die Datensammlung sowie die verschiedenen Arten von Lernansätzen im Machine Learning geschildert, auf denen die später entwickelten Modelle basieren. Typische Metriken, die oft genutzt werden, werden ebenfalls vorgestellt und beschrieben, darunter k-Nearest Neighbour-Algorithmen, Matrixfaktorisierung, der Apriori Algorithmus und die Term Frequency-Inverse Document Frequency als Gewichtungsmaßnahme. Des Weiteren werden noch die notwendigen Schritte bei der Datenvorverarbeitung und dem Feature Engineering erwähnt und die unterschiedlichen Möglichkeiten und Kriterien bei der Evaluation eines Recommender Systems aufgeführt. Leave One Out Cross Validation (LOOCV) wird dabei als Evaluationsmetrik vorgestellt.

Im nächsten Kapitel wird die Entwicklung und Implementierung einer Recommendation Engine für den Online Shop beschrieben. Dazu werden die vorliegenden Datensätze dargelegt und in Trainings- und Testdaten aufgeteilt. Anschließend werden Modelle basierend auf den im vorherigen Kapitel erwähnten unterschiedlichen Lernansätzen und Algorithmen erstellt. Dabei wird die Implementierung erläutert, sowie die damit zusammenhängenden jeweiligen Herausforderungen und Vor- und Nachteile. Auf die erstellten Recommender Modelle wird LOOCV als Evaluationsmetrik angewendet und ein Zwischenfazit gezogen. Danach erfolgt eine Beschreibung zweier Ansätze für ein hybrides System, die darauffolgend umgesetzt werden.

Im vierten Kapitel werden diese wieder mit LOOCV evaluiert, es werden Vorteile und die Herausforderungen aufgeführt, die während des Entwicklungsprozesses aufgetreten sind und bewältigt werden konnten. Anschließend erfolgt eine kurze Beschreibung für die Online Evaluation und den damit verbundenen Erwartungen, dass das Recommender System den Umsatz und die Kundenzufriedenheit steigert.

Zum Schluss erfolgt ein Fazit über die Arbeit, Ergebnisse werden zusammengefasst und eine Schlussfolgerung gezogen.

2 Theoretische Grundlagen

2.1 Einführung in Recommender Systeme

Im Folgenden wird zunächst eine kurze Einführung in Recommender Systeme gegeben, welche Problemstellungen sie angehen und welchen Nutzen sie damit erfüllen.

2.1.1 Recommender Systeme im E-Commerce

Recommender Systeme sind Teil eines Informationssystems und Softwarewerkzeuge, die zur Bewältigung der Informationsüberflutung dienen. Sie sollen einem User dabei helfen, für ihn interessante Inhalte inmitten einer breiten Masse von Angeboten zu finden [RICCI11; GOOGLEREC]. Das Hauptziel ist hierbei, die Kundenzufriedenheit und damit den Umsatz des Unternehmens zu steigern. Die Entitäten, die ein System empfiehlt, werden dabei als Items bezeichnet: Bei Youtube sind dies beispielsweise Videos, bei einem App Store Applikationen usw.

Unter anderem können Recommender Systeme so bei der Lösung des sogenannten Long Tail Problems behilflich sein (siehe Abbildung 2.1). Anderson gibt dafür in seinem gleichnamigen Buch ein Beispiel [AND04]:
"IN 1988, A British mountain climber named Joe Simpson wrote a book called Touching the Void, a harrowing account of near death in the Peruvian Andes. It got good reviews but, only a modest success, it was soon forgotten. Then, a decade later, a strange thing happened. Jon Krakauer wrote Into Thin Air, another book about a mountain-climbing tragedy, which became a publishing sensation. Suddenly Touching the Void started to sell again.
Random House rushed out a new edition to keep up with demand. Booksellers began to promote it next to their Into Thin Air displays, and sales rose further. A revised paperback edition, which came out in January, spent 14 weeks on the New York Times bestseller list. That same month, IFC Films released a docudrama of the story to critical acclaim. Now Touching the Void outsells Into Thin Air more than two to one."
Der Grund für den beschriebenen Sachverhalt sind Amazons Produktempfehlungen: Aufgrund vom Kaufverhalten der User nahm das System an, dass Leute, die *Into Thin Air* mochten, auch *Touching The Void* mögen würden. Diese Empfehlung traf bei den Usern auf große Zustimmung, sie schrieben gute Bewertungen, was zu mehr Käufen, damit zu mehr Produktempfehlungen und schließlich zu einer sogenannten

positiven Feedbackschleife führte.
Das Beispiel verdeutlicht, dass mit Aufstieg der digitalen Wirtschaft Kunden und Kundinnen Produkte finden können, die speziell ihren Geschmäckern und Vorlieben angepasst sind. Long Tail ist die These, dass Unternehmen mit virtuellen Produkten den Großteil ihres Umsatzes mit Nischenprodukten machen und nicht mehr mit Bestsellern [AND04]: In klassischen Geschäftsmodellen wird das empfohlen, was beliebt oder neu ist. Die meisten wollen jedoch keine unpersonalisierten Hits empfohlen bekommen. Durch Recommender Systeme kann damit die Konsumierung generischer beliebter Produkte einem System mit vielfältigerem Angebot weichen, dass auch kleineren Zielgruppen passende Items empfehlen kann.

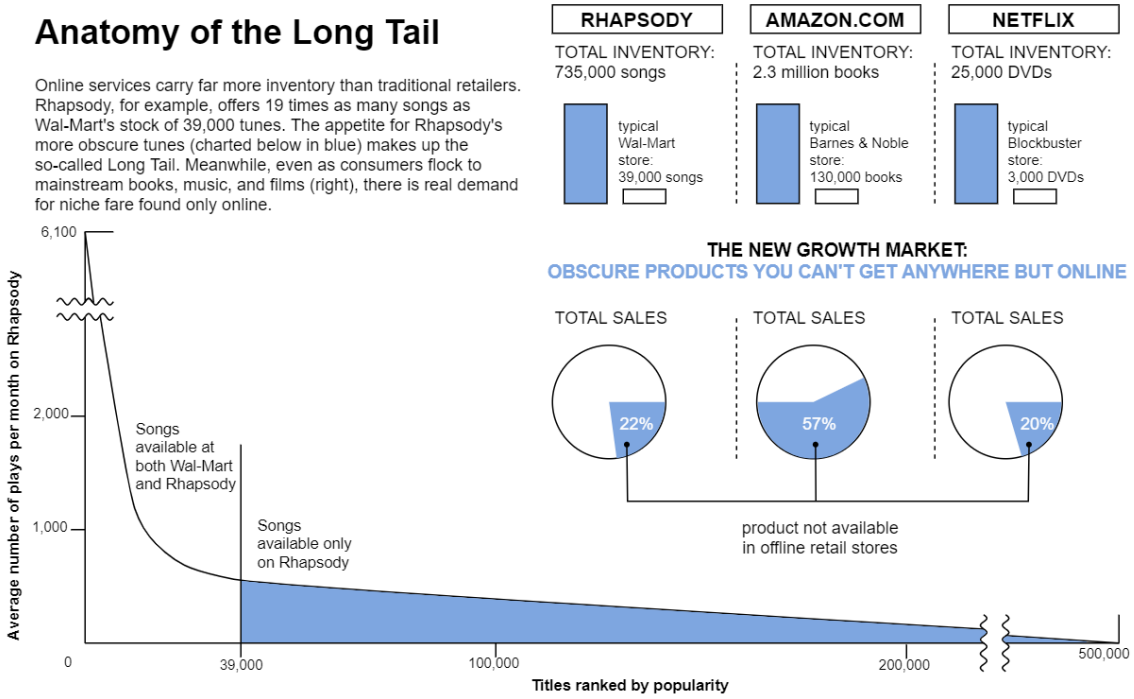


Abbildung 2.1: Anatomy of the Long Tail nach [AND04]

Ein weiterer wichtiger Punkt für den Umsatz und die Gewinnstrategie eines Unternehmens ist die Kundenbindung, die Reichheld und Sasser ansprechen [REISAS90]. Auch dabei spielen Recommender Systeme im E-Commerce eine große Rolle und können somit das Kaufverhalten eines Users stark beeinflussen (siehe Kapitel 2.1.2).

Die Relevanz eines Recommender Systems ist zudem abhängig von einem gegebenen Ziel, das verfolgt wird. Herlocker et al. [HERL04] nennen dabei elf Funktionen, bei deren Implementierung ein Recommender System für den User hilfreich sein kann (siehe Tabelle A.1). Die Rolle eines Recommender Systems kann also recht vielfältig sein, was zu einer Reihe von verschiedenen Techniken und Informationsquellen

führt, die genutzt werden können. Dies wird in den nachfolgenden Kapiteln näher untersucht.

2.1.2 Nutzen und Vorteile

Warum Recommender Systeme so nützlich und von Vorteil sind, ist in vielen Studien, Artikeln und anderen Arbeiten bereits erläutert und wird auch in der Praxis deutlich:

- Automatisierte Personalisierung, Steigerung der Kundenzufriedenheit und Kundenloyalität:
  Recommender Systeme ermöglichen eine schnelle und vor allem automatisierte Anpassung und Personalisierung einer Seite an den Nutzer/die Nutzerin. Damit zeigt man den Kunden/Kundinnen, dass man ihre Wünsche und Bedürfnisse zur Kenntnis nimmt. Das Unternehmen kann daraus lernen und damit besser verstehen, was der Kunde/die Kundin möchte. Die damit verbundene Steigerung der Kundenzufriedenheit und -loyalität kann den Umsatz des Unternehmens erhöhen [SSRM14; RICCI11].
  Recommender Systeme sind damit ein Schlüsselweg, um flexibilisierte Massenproduktion für E-Commerce Seiten zu automatisieren [SKR99].

- Nach Schafer et al. [SKR99] verbessern Recommender Systeme den Umsatz auf drei Wegen:
  1. „Von Browsern zu Buyern“ - Den Usern wird dabei geholfen, Produkte zu finden, die sie kaufen möchten. Dies kann dabei helfen, User, die sich eine Webseite nur ansehen bzw. sich durchklicken, zu Käufern umzuwandeln.
  2. Cross-sell - Recommender Systeme steigern diesen, indem sie zusätzliche Produkte vorschlagen. Wenn die Empfehlungen gut sind steigt damit auch die durchschnittliche Bestellmenge.
  3. Loyalty: Wie weiter oben bereits erwähnt, ist Kundenloyalität zu gewinnen eine essentielle Unternehmenstrategie. Man baut ein Verhältnis mit Mehrwert zwischen User und Seite auf, denn diese kehren zu den Seiten zurück, die am besten zu ihren Bedürfnissen passen.

- Anzahl der verkauften Items erhöhen [RICCI11]:
  Die wahrscheinlich wichtigste Funktion ist, dass mit Hilfe eines Recommender Systems zusätzlich Items verkauft werden können. Das Ziel ist, die Konversionsrate zu erhöhen, also die Anzahl an Usern, die eine Empfehlung nutzen und ein Item „konsumieren“, verglichen mit Besuchern bzw. Besucherinnen, die sich nur durch die Seite klicken.

- Vielfältigere Items verkaufen [RICCI11]:
  Ein weiterer Hauptvorteil ist, dass einem User durch ein Recommender System Produkte empfohlen werden, die er sonst möglicherweise nicht gefunden hätte. Dies hängt mit dem oben beschriebenen Long Tail Problem zusammen.

2.2 Die drei Haupttypen

In der Literatur werden Recommender Systeme im Allgemeinen meist in drei Hauptsysteme unterschieden [IFO15; SHAM13; SEV18]. Abbildung 2.2 veranschaulicht dies. Die unterschiedlichen Herangehensweisen bzw. Empfehlungstechniken, auf denen diese jeweils basieren, werden im Folgenden näher erläutert.

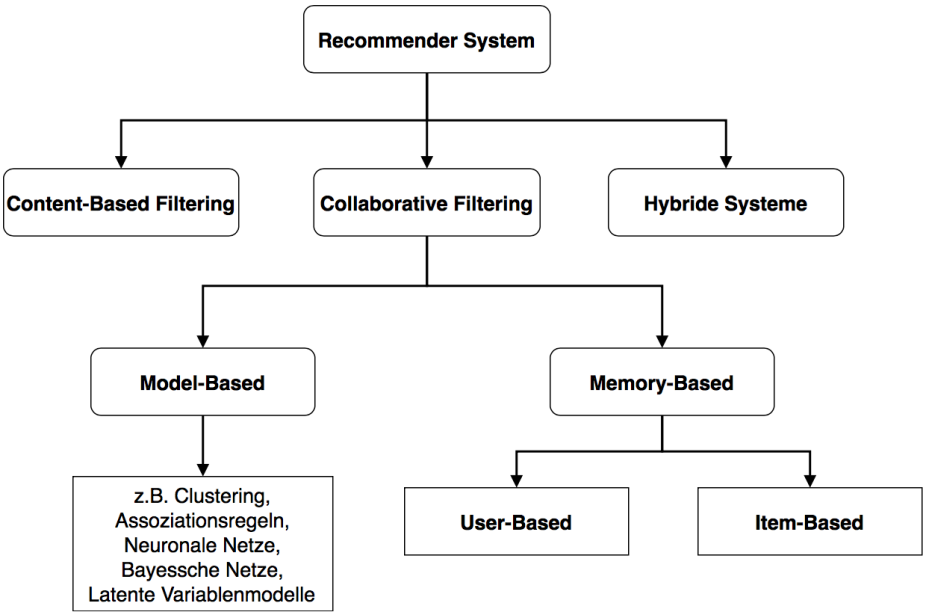


Abbildung 2.2: Recommendation Techniken in Anlehnung an [IFO15]

2.2.1 Content-Based Filtering

Content-Based Filtering (CBF) ist ein domain-abhängiger Algorithmus. Er basiert auf den Attributen und Eigenschaften eines Objektes, um Empfehlungen zu generieren und Vorhersagen zu treffen, also seinen sogenannten Features [RICCI11]. Beim CBF werden Items empfohlen, die ähnlich zu denen sind, mit denen ein User in der Vergangenheit bereits interagiert hat bzw. die er oder sie mag. Im Kontext eines Fashion Shops heißt das beispielsweise: Wenn ein User Hosen und Röcke gekauft hat, dann werden ihm basierend darauf weitere Hosen und Röcke angeboten. CBF erweist sich besonders bei Dokumenten wie Webseiten, Publikationen und Nachrichten als recht erfolgreich [IFO15; LOPS11].

In [LUK19] werden zwei Ansätze des CBF beschrieben:
1. Bei der ersten Variante wird der Content der Items analysiert. Es wird damit die Ähnlichkeit zwischen allen Itempaaren berechnet und dann zu den Items, die ein User bereits bewertet bzw. mit denen er interagiert hat, eine Liste von Empfehlungen generiert.

Um die Ähnlichkeit zwischen zwei Items zu berechnen gibt es verschiedene Möglichkeiten, die in den folgenden Kapiteln näher untersucht werden.

2. Bei der zweiten Variante werden User- und Itemprofile erstellt. Die Repräsentation der Items erfolgt anhand ihrer Beschreibung und Attribute bzw. Features (bei Büchern z.B. das Genre, der Autor usw.). Daraus ergibt sich das Itemprofil. Das Userprofil wird auf Grundlage der User-Historie und -bewertungen erstellt. Die gesammelte Userinformation wird mit den Content Features der Items verglichen und untersucht [SHAM13]. Darauf basierend werden dann Items gefunden, die am besten zu einem gegebenen Userprofil passen. Die Funktionsweise ist in Abbildung 2.3 veranschaulicht.

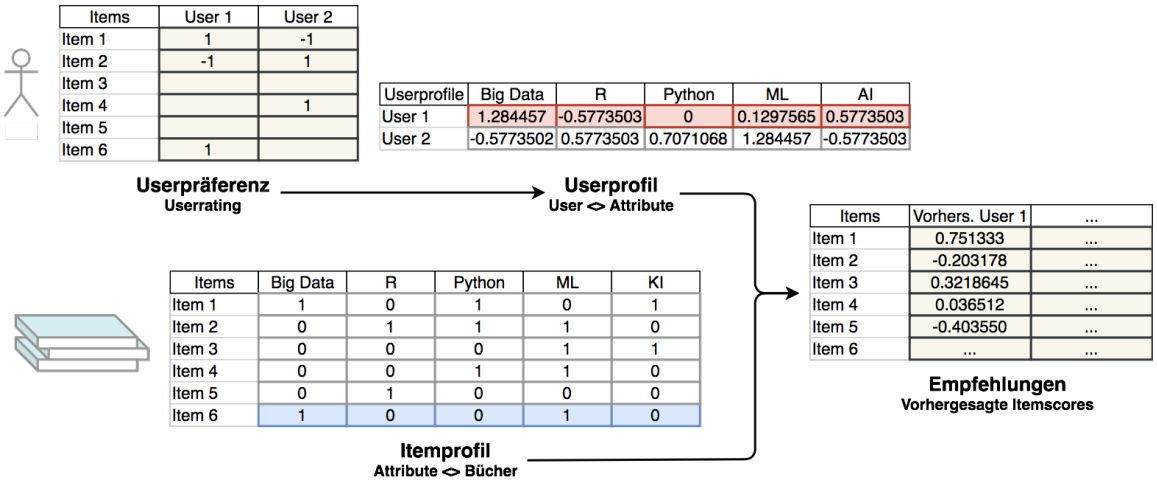


Abbildung 2.3: CBF mit Item- und Userprofilen in Anlehnung an [LUK19]

2.2.2 Collaborative Filtering

Collaborative Filtering (CF) ist domain-unabhängig und wird als die beliebteste und am meisten genutzte Technik für Empfehlungssysteme angesehen [RICCI11; SKKR01]. CF und CBF sind in Abbildung 2.4 gegenübergestellt.

Als CF werden im Allgemeinen alle Algorithmen bezeichnet, die nur auf Userverhalten basieren, beispielsweise Ratings, ähnlichen Usern (zur Ähnlichkeitsbestimmung im Folgenden mehr), User-Historie usw.: Die Grundidee von CF ist es, Empfehlungen oder Vorhersagen basierend auf der Meinung anderer gleichgesinnter User zu bieten [SKKR01]. Ziel ist es, eine Menge von Usern zu finden, die am ähnlichsten zu einem gegebenen User sind und eine Menge von Items zu finden, die am ähnlichsten zu einem gegebenen Item sind [SHAM13]. Die originale Implementation dieses Ansatzes empfiehlt dem aktuellen User Items, die Usern mit ähnlichen Interessen in der Vergangenheit gefallen haben [RICCI11].

Die Methode basiert auf Ähnlichkeiten in der Bewertungsmatrix von Usern und Items: Der Algorithmus repräsentiert den ganzen User-Item Raum als Ratingmatrix R. Jeder Eintrag R_{ij} in der Matrix steht dabei für das Rating vom i-ten User für das j-te Item. Ein großer Nachteil bei CF ist, dass bei einer geringen Menge an Daten keine zuverlässigen Recommender Modelle entwickelt werden können, da die Datengrundlage nicht ausreicht. Die Matrix ist dann zu spärlich besetzt (Data Sparsity Problem). Man unterscheidet im Allgemeinen zwischen memory-basierten und model-basierten CF-Methoden.

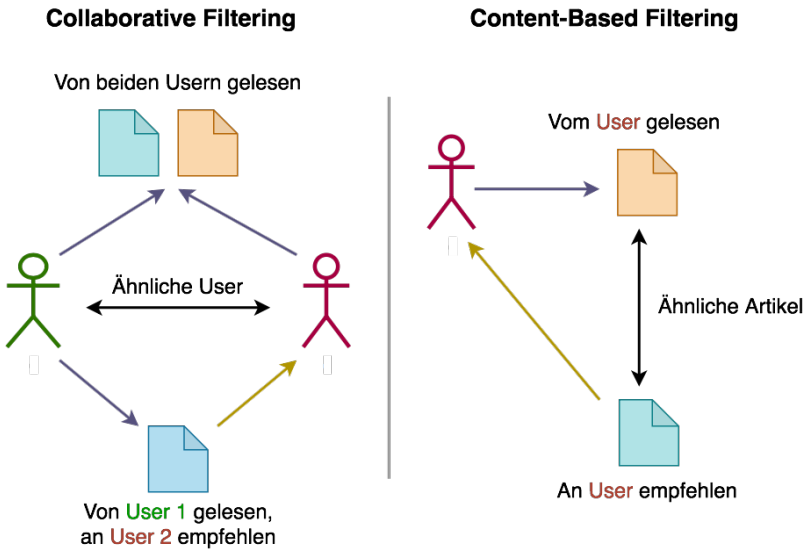


Abbildung 2.4: CF vs. CBF in Anlehnung an [LIAO19]

2.2.2.1 Memory-Based

Memory-Based Ansätze benutzen die ganze oder einen Teil der User-Item-Datenbank [SHAM13]. Es werden Userratingdaten verwendet, um Ähnlichkeiten zwischen Usern und Items zu berechnen. Die bekanntesten Techniken sind dabei Algorithmen, die auf Neighbourhood-Methoden basieren. Die Empfehlungsgenerierung lässt sich in zwei Schritte einteilen: Die Berechnung der Ähnlichkeiten zwischen User und Item und dem Treffen von Vorhersagen für unbekannte Ratings. Man unterscheidet beim Memory-Based Filtering zwischen zwei Herangehensweisen (siehe Abbildung 2.5).

1.User-Based Collaborative Filtering (UB-CF)
Bei UB-CF wird nach Usern geschaut, die dem aktiven User, basierend auf den Bewertungs- oder Bewegungsdaten, am ähnlichsten sind. Die Idee hierbei ist, dass User mit ähnlichem Kauf- und Nutzerverhalten auch ähnliche Interessen haben: Es liegt nahe, dass ein User U_1 die Items mögen wird, die ein anderer User U_2, der ähnlich zu U_1 ist, gekauft oder positiv bewertet hat [SSRM14].

Ähnlichkeiten zwischen Usern werden berechnet, indem ihre Ratings desselben Items verglichen werden [SHAM13]. Es werden dann Items vorgeschlagen, die diesen Usern gefallen haben. UB-CF basiert also auf der Korrelation zwischen einem User U und anderen Usern. Das Recommender System lernt somit nach und nach von seinen Nutzern bzw. Nutzerinnen [SKR99].

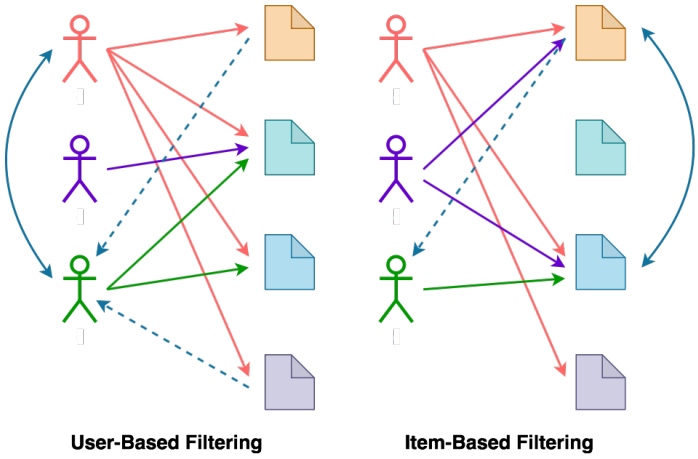


Abbildung 2.5: User-Based Filtering vs. Item-Based Filtering

2. Item-Based Collaborative Filtering (IB-CF)
IB-CF wurde 1998 von Amazon eingeführt und hat sich seit der Veröffentlichung zu einem beliebten Algorithmus durchgesetzt [SL17]. Dabei werden, wie der Name bereits vermuten lässt, Ähnlichkeiten zwischen den Items berechnet. Recommender Systeme, die auf solchen Algorithmen beruhen, empfehlen Items basierend darauf, bei welchen anderen Items der User bereits Interesse gezeigt hat [SSRM14]. Dies geschieht mittels Vergleichs der Ratings desselben Users U für die jeweiligen Items [SHAM13]. So können für einen User Empfehlungen basierend auf dessen derzeitigen Interessen generiert werden [GRUS19].
Empfehlungen beim IB-CF basieren also auf einem kleinen Set von Items, mit denen der User bereits interagiert oder an denen er Interesse gezeigt hat [SKR99]. Statt also die Empfehlungsgenerierung auf Ähnlichkeiten zwischen Usern zu stützen, werden hier Ähnlichkeiten zwischen den Bewertungsdaten der Items genutzt [ERK11], es wird also eine Item-Item-Ähnlichkeits-Matrix berechnet.

2.2.2.2 Model-Based

Model-Based Ansätze konstruieren ein Vorhersagemodell, um unbekannte Ratings zu schätzen, indem sie von Beobachtungsdaten lernen. Der Algorithmus erstellt also ein Modell zu den Userratings, es werden dabei probabilistische Ansätze verwendet [SKKR01]. Die Parameter der statistischen Modelle werden geschätzt, und es werden, anders als bei der memory-basierten Methode, nicht alle verfügbaren Informationen

genutzt, um Vorhersagen zu treffen [SHAM13]. Die Sammlung an Ratings wird genutzt, damit das Modell offline die Userpräferenzen lernt. Beispiele für zwei populäre model-basierte Techniken sind Bayessche Netze und Clustering. Eine weitere Methode sind latente Variablenmodelle und Matrixfaktorisierung (MF).
In Abbildung 2.6 ist die Model-Based CF Strategie mit einem latenten Variablenmodell als Empfehlungsmethode dargestellt.

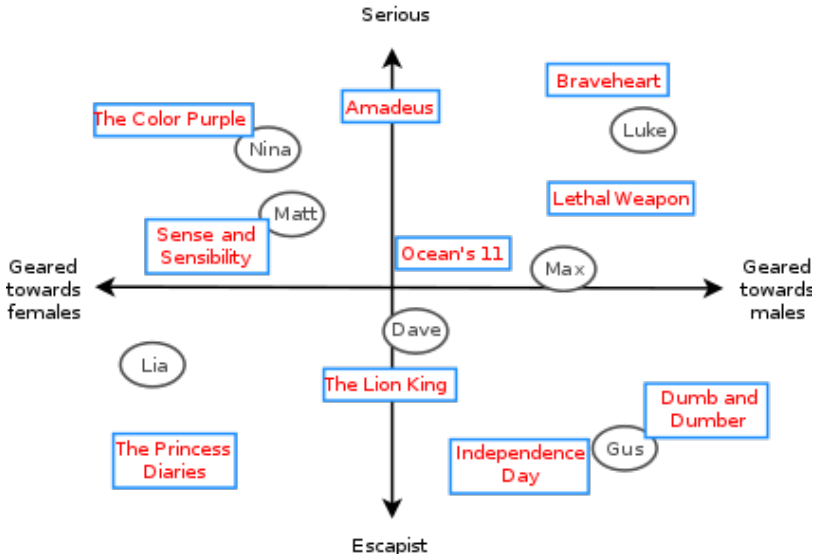


Abbildung 2.6: Model-Based Collaborative Filtering with Latent Factor Model in Anlehnung an [KBV09]

2.2.3 Hybrid Filtering

Elemente verschiedener Recommender Modelle können in einem System kombiniert werden, um die Vorteile einer Empfehlungstechnik optimal auszunutzen. Dies wird als ein hybrides System bezeichnet. Indem beispielsweise Technik A und B miteinander kombiniert werden, werden die Vorteile von A benutzt, um die Nachteile von B auszugleichen [RICCI11]. Allein Amazon verwendet beispielsweise mindestens 18 verschiedene Recommender Typen [AHR11].
Robin Burkes Artikel über hybride Recommender Systeme [BURKE02] ist eine bekannte Studie über die verschiedenen Möglichkeiten und Designs solcher Systeme. Jannach et al. [JZFF11] unterteilen die verschiedenen von Burke genannten Arten von Hybridisierungstechniken eines hybriden Systems (siehe Tabelle A.2) dabei in drei Kategorien:

2.2.3.1 Monolithic

Monolithische Hybridisierungstechniken sind Designs, bei denen Elemente von mehreren Recommender Strategien in eine Algorithmus-Implementation integriert werden

(siehe Abbildung 2.7).
Hierzu zählen die von Burke erwähnten Methoden Feature Combination und Feature Augmentation.

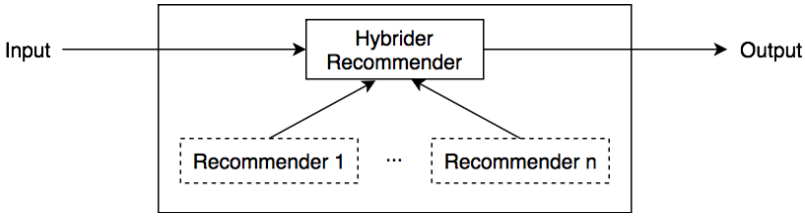


Abbildung 2.7: Monolithisches Hybridisierungsdesign in Anlehnung an [JZFF11]

2.2.3.2 Paralleled

Beim Paralleled-Ansatz werden mindestens zwei separate Recommender Implementationen benötigt, die dann kombiniert werden. Wie in Abbildung 2.8 zu sehen, werden unabhängig voneinander Empfehlungslisten erstellt. Im Hybridization Step werden diese zu einem finalem Output zusammengebracht.
Zum Paralleled Ansatz gehören die Weighted, Switching und Mixed Methode.

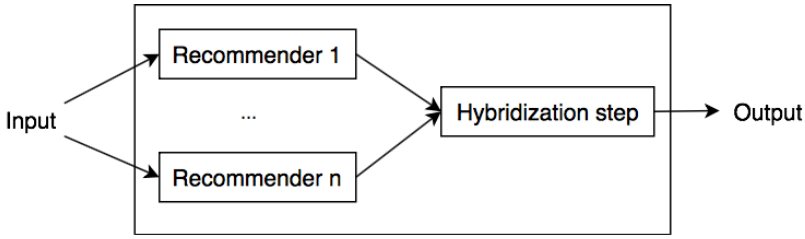


Abbildung 2.8: Parallelisiertes Hybridisierungsdesign in Anlehnung an [JZFF11]

2.2.3.3 Pipelined

Bei der Pipelined-Methode, die an dieser Stelle nur der Vollständigkeit halber genannt wird, wird der Output des einen Recommender Systems zum Input des nächsten (siehe Abbildung 2.9).

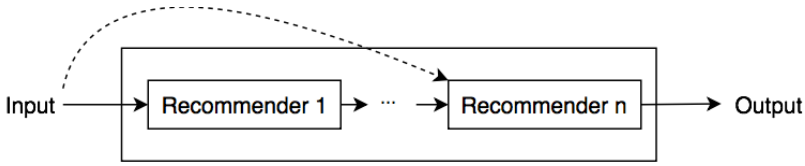


Abbildung 2.9: Pipelined Hybridisierungsdesign in Anlehnung an [JZFF11]

2.2.4 Weitere Techniken

Abgesehen von den bereits genannten Systemen sind in der Literatur noch weitere Techniken zu finden [RICCI11]:

- Knowledge-Based - Solche Systeme basieren auf dem Domainwissen, dass bestimmte Item Features zu bestimmten Userpräferenzen und -bedürfnissen passen. Man weiß damit, wie nützlich ein Item für einen Nutzer ist.
- Demographic - Diese Art von Recommender Systemen basieren auf dem demografischen Profil eines Users, z.B. ihrer Sprache, dem Land, Alter usw. Die Idee hierbei ist, dass unterschiedliche Zielgruppen auch unterschiedliche Präferenzen haben.
- Community-Based - Bei Community-Based Recommender Systemen basieren die Empfehlungen auf den Freunden des Users. Die Technik folgt damit dem Sprichwort „Zeig mir deine Freunde, und ich sag dir wer du bist.“. Gerade bei sozialen Netzwerken kann dies von Vorteil sein.

Für die in dieser Arbeit zur Verfügung stehenden Daten liegen keine persönlichen demografischen Informationen vor und Domainwissen zu bestimmten Items und Usern gibt es ebenfalls nicht. Es handelt sich hier auch um kein soziales Netzwerk oder ein System, bei dem die User anderweitig miteinander interagieren. Daher scheiden die drei oben genannten Techniken in diesem Anwendungsfall aus.

2.3 Datensammlung

Um Recommender Systeme - oder Allgemein Machine Learning-Systeme - erstellen zu können, werden Daten benötigt, auf denen man die Modellbildung basieren kann. Diese können von verschiedenen Quellen kommen und von unterschiedlicher Art sein. Für CBF werden z.B. Informationen über die zu empfehlenden Entitäten benötigt, um ein Recommender Modell basierend auf Produktattributen entwickeln zu können. Vor allem werden aber - besonders für CF - Userdaten benötigt. Bei diesen kann ebenfalls zwischen verschiedenen Datentypen unterschieden werden (siehe Tabelle 2.1). Bei der Datensammlung für Userdaten, den sogenannten Präferenzdaten, unterscheidet man grundsätzlich zwischen zwei verschiedenen Quellen: Expliziten und impliziten Daten [ERK11].
Um die erlernten Modelle auch testen zu können, werden die zur Verfügung stehenden Daten in einen Trainings- und einen Testdatensatz aufgeteilt. Mit Hilfe der Trainingsdaten werden die Modelle gebildet und trainiert, die dann im Anschluss mit den Testdaten evaluiert und validiert werden.

Datentyp	Beschreibung
Rating Data	- Rating Scores (diskrete oder kontinuierliche Bewertungsdaten) - latente Kommentare (z.B. am besten, gut, schlecht, schlechter) - Likes und Dislikes
Behaviour Pattern Data	- Dauer des Browsings, Anzahl der Klicks - Links von Webseiten - Speichern, Drucken, Scrollen, Löschen, Schließen, Aktualisieren von Webseiten - Auswahl, Bearbeitung, Suche, Kopieren, Einfügen, Markieren und Download von Webinhalten
Transaction Data	- Kaufdaten, Anzahl der Käufe - Preise - Rabatte

Tabelle 2.1: Datentypen für ein Recommender System in Anlehnung an [WEI16]

2.3.1 Explizite Daten

Bei expliziten Daten handelt es sich um Feedback vom User, welches er persönlich gegeben hat. Dieser User Input kann dabei in verschiedenen Formen auftreten: Als Likes und Dislikes, einem Bewertungssystem mit einem festgelegten Spektrum (z.B. eine Bewertungsskala von 1 bis 5), Formulare zum Ausfüllen oder als Reviews in Textform (z.B. Kundenbewertungen bei Amazon).
Bei expliziten Daten kommt das Feedback direkt vom User selbst. Der Nachteil ist allerdings, dass solche Informationen oft nicht zur Genüge vorliegen: User bringen selten Zeit dafür auf, um etwas zu bewerten oder persönliche Infos von sich preiszugeben [SHAM13]. Außerdem können auch Unstimmigkeiten zwischen dem, was der User angibt und dem, was tatsächlich wahr ist, vorliegen.
Bei solchen Präferenzdaten ist das Ziel eines Recommender Systems, für einen User die Bewertung von Items, die er noch nicht bewertet hat, vorherzusagen.

2.3.2 Implizite Daten

Bei impliziten Daten handelt es sich um Feedback, bei dem kein zusätzliches Eingreifen des Users benötigt wird. Beispiele hier wären seine Transaktionsdaten, also was er wann gekauft hat, und andere Bewegungs- bzw. Klickdaten: Welche Produkte er sich wie oft angesehen hat, was er gespeichert oder wonach er gesucht hat, was er in seinen Warenkorb gelegt hat usw. Es handelt sich hier also um die Aktivität eines Users. Der Vorteil hier gegenüber der expliziten Datensammlung ist, dass hier viel mehr Daten vorliegen. Nach Ekstrand et al. [ERK11] wird allerdings davon ausge-

gangen, dass implizite Daten verrauschter bzw. ungenauer sind. Allerdings können sie auf längere Sicht ein akkurateres Bild über die Interessen eines Users bilden.
Systeme, die implizites Feedback sammeln, werden häufiger in der Praxis angewendet [SHAM13]. Gerade im E-Commerce sind Page Views und Käufe die Hauptquelle für Präferenzdaten. Ein Recommender System ist allerdings nicht auf nur explizite oder nur implizite Daten beschränkt, sondern kann beide Arten als Informationsquelle nutzen.

2.4 Lernansätze

Beim Machine Learning gibt es verschiedene Arten, in die man Systeme einteilen kann, meist wird dabei zwischen drei verschiedenen Kategorien unterschieden: Überwachtem Lernen (Supervised Learning), unüberwachtem Lernen (Unsupervised Learning) und verstärkendem Lernen (Reinforcement Learning) (siehe Tabelle 2.2). Letzteres kommt in dieser Arbeit nicht zum Tragen.

Art	Beschreibung
Supervised	- Gekennzeichnete Daten - Direktes Feedback - Ergebnis/Zukunft vorhersagen
Unsupervised	- Keine Kennzeichnung/Ziele - Kein Feedback - Verborgene Strukturen in den Daten finden
Reinforcement	- Entscheidungsvorgang - Belohnungssystem - Aktionen erlernen

Tabelle 2.2: Lernansätze in Anlehnung an [RASCH18]

2.4.1 Supervised

Unter Supervised Learning, also überwachtem Lernen, versteht man ein von Menschen gesteuertes Lernen [NGZE18]. Die Trainingsdaten, also die Datensätze mit denen der Algorithmus lernen soll, enthalten bereits sogenannte Labels bzw. ein Target: Bei einem E-Mail-Filtersystem tragen die Datensätze beispielsweise bereits die Bezeichnungen „Spam“ oder „kein Spam“. Beispiele für das überwachte Lernen sind die Regression und die Klassifikation [RASCH18]:

- Klassifikation - Das gerade erwähnte E-Mail-Filtersystem ist ein Beispiel für eine binäre Klassifikation: Anhand der vorherigen Beobachtungen lernt der Algorithmus Regeln, um zwischen den zwei möglichen Klassen unterscheiden zu

können. Dadurch wird ein Vorhersagemodell geschaffen, das dann auf neue unbekannte Daten angewendet werden kann. Ziel ist hier also die Vorhersage von Klassenbezeichnungen. Es ist auch eine Mehrfachklassifizierung möglich.

- Regression - Algorithmen, bei denen es darum geht eine numerische Größe vorherzusagen, werden als Regression bezeichnet: Es geht also um die Vorhersage von stetigen Ergebnissen. Diese werden auf Basis von gegebenen Merkmalen und der Zielvariable, dem Ergebnis, getroffen. Es wird versucht, Beziehungen zwischen den Merkmalen zu finden, um die Ergebnisse vorhersagen zu können. Aus den Trainingsdaten kann so ein Modell hergeleitet werden, dass auf zukünftige Daten angewendet werden kann.
  Regressionsalgorithmen lassen sich zur Klassifikation einsetzen und umgekehrt (z.B. 10%ige Chance für Spam-Mail).

2.4.2 Unsupervised

Beim Unsupervised Learning, also dem unüberwachtem Lernen, gibt es keine Labels, die Daten sind also nicht gekennzeichnet. Ziel hierbei ist es, die Struktur der Daten zu untersuchen und so nützliche Informationen ohne eine Zielvariable zu gewinnen. Beispiele für Unsupervised Learning sind das Clustering und die Assoziationsregeln [NGZE18]:

- Clustering - Beim Clustering wird beispielsweise eine Gruppe von Usern in Gruppen, sogenannte Cluster, eingeteilt, ohne vorher Kenntnisse über ihre Gruppenzugehörigkeit o.Ä. gehabt zu haben. Die User in einem Cluster haben bestimmte Eigenschaften gemeinsam, die sie von den anderen Gruppen unterscheiden. So können z.B. Kunden mit gleichen Interessen zusammen gruppiert werden. Clustering wird auch als „unüberwachte Klassifizierung“ bezeichnet [RASCH18].

- Assoziationsregeln - Bei Assoziationsregeln werden große Datenmengen untersucht, um interessante Beziehungen zwischen Items, sogenannte Regeln, zu finden. Ein häufiges Beispiel hierfür sind Einkäufe in einem Supermarkt, bei denen Paare oder Gruppierungen von Produkten erkannt werden, die oft zusammen gekauft werden. Hier könnte eine dieser Regeln z.B. sein, dass Kunden, die Grillsoße und Kartoffelchips kaufen, oft auch Steaks mit einkaufen [GER18].

2.5 State of the Art Metriken

Für die Berechnung der Ähnlichkeiten zwischen Items und für die Umsetzung von Recommender Systemen haben sich, je nach dessen Ziel und Funktion, im Laufe der Jahre einige verschiedene Algorithmen als State of the Art Metriken durchgesetzt, die sich als effizient und erfolgreich erwiesen haben. Im Folgenden werden einige von ihnen, die für diese Arbeit relevant sind, näher untersucht.

2.5.1 K-Nearest Neighbour Algorithmen

Eine beliebte Klassifikation ist die k-Nearest Neighbour (kNN) Klassifizierung, eine nicht-parametrische „Lazy Learning“ Methode. Dies bedeutet, dass es bei kNN keine richtige Trainingsphase gibt, aus den Trainingsdaten wird keine Funktion erlernt sondern das Modell prägt sich quasi das Trainingsdatenset ein. Um Empfehlungen zu generieren, wird dann bei der Anfrage im Datenset nach den nächsten Nachbarn gesucht.
Entitäten werden hier als Punkte und Vektoren repräsentiert. Um z.B. ähnliche User zu einem gegebenen User oder die ähnlichsten Items zu einem bestimmten Item zu finden, werden die k-nächsten Punkte - also die *k-Nearest Neighbours* - zu einer bestimmten Entität aus dem Trainingsdatensatz gefunden [DEKA11]. Dafür werden die Abstände zwischen dem gegebenen Punkt und den anderen Punkten berechnet, um so die k-nächsten Nachbarn zu finden.
Für diese Neighbourhood-basierten Methoden werden folgende Distanzmetriken häufig verwendet [AGMI17; GOOGLEREC]:

1. Euklidischer Abstand:

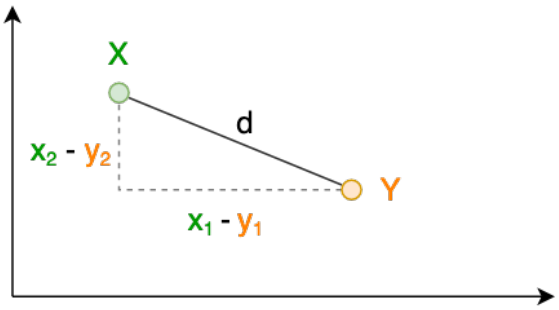


Abbildung 2.10: Euklidischer Abstand

Hier wird der Abstand d im Euklidischen Raum zwischen zwei Punkten gemessen, also die Länge der Strecke, die diese Punkte verbindet. Ein kleinerer Wert bedeutet dabei eine höhere Ähnlichkeit. Die Distanz d_2 in beispielsweise einem zweidimensionalen Raum zwischen zwei Punkten $X(x_1, x_2)$ und $Y(y_1, y_2)$, ist in Abbildung 2.10 dargestellt. d_2 lässt sich wie folgt berechnen:

$$d_2(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \tag{2.1}$$

Der euklidische Abstand zwischen zwei Punkten bzw. zwei Items X und Y in einem n-dimensionalen euklidischen Raum wird dementsprechend im Allgemeinen mit der in 2.2 dargestellten Formel berechnet.

$$d(X, Y) = \sqrt{\sum_{i=1}^{n} (x_\mathrm{i} - y_\mathrm{i})^2} \tag{2.2}$$

2. Kosinus-Ähnlichkeit:
Die Kosinus-Ähnlichkeit ist ein Maß für die Ähnlichkeit zweier Vektoren und ist eine oft genutzte Methode beim CF für Recommender Systeme [LOONY16]. Hier wird die Ähnlichkeit zwischen zwei Vektoren $\vec{u}$ und $\vec{v}$ innerhalb eines multidimensionalen Raumes bestimmt, indem der Kosinus des Winkels zwischen diesen berechnet wird (siehe Abbildung 2.11).

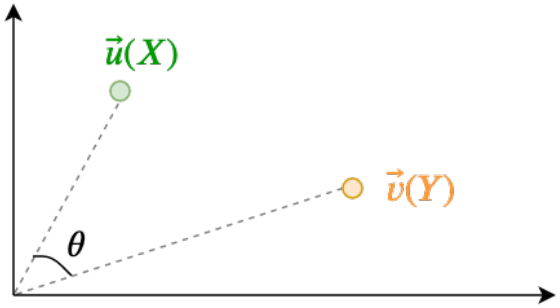


Abbildung 2.11: Kosinus-Ähnlichkeit

Je kleiner dieser Winkel also ist, desto mehr zeigen die Vektoren in die gleiche Richtung und desto ähnlicher sind sie sich dementsprechend. Es geht also um die Ausrichtung bzw. Orientierung und nicht, anders als beim euklidischen Abstand, um die Entfernung bzw. Größenordnung.
Ein Kosinuswert von 0 für θ bedeutet, dass die Vektoren orthogonal, also in einem 90°-Winkel, zueinander stehen und somit nicht miteinander übereinstimmen. Der Winkel θ hängt mit dem Standardskalarprodukt zusammen:

$$\vec{u} \cdot \vec{v} = ||\vec{u}|| \cdot ||\vec{v}|| \cdot \cos(\theta) \tag{2.3}$$

Stellt man diese Formel nun nach $\cos(\theta)$ um, so erhält man die Formel für die Kosinus-Ähnlicheit $CosSim_{u,v}$ zwischen zwei Vektoren $\vec{u}$ und $\vec{v}$ in einem n-dimensionalen Raum (siehe Formel 2.4).

$$\begin{aligned} CosSim_{\mathrm{u,v}} = \cos(\theta) &= \frac{\vec{u} \cdot \vec{v}}{||\vec{u}|| \cdot ||\vec{v}||} = \frac{<\vec{u}, \vec{v}>}{||\vec{u}|| \cdot ||\vec{v}||} \\ &= \frac{\sum_{i=1}^{n} u_{\mathrm{i}} v_{\mathrm{i}}}{\sqrt{\sum_{i=1}^{n} {u_{\mathrm{i}}}^2} \cdot \sqrt{\sum_{i=1}^{n} {v_{\mathrm{i}}}^2}} \end{aligned} \tag{2.4}$$

$||*||$ entspricht dabei der euklidischen Norm, also der Länge eines Vektors, $<\vec{u}, \vec{v}>$ ist das Skalarprodukt von $\vec{u}$ und $\vec{v}$. Bei Recommender Systemen repräsentieren die Vektoren die Ratings eines Users. Die Kosinus-Ähnlichkeit kann im Allgemeinen Werte zwischen -1 und 1 annehmen, da aber z.B. die Häufigkeit von Wörtern in Dokumenten (mehr dazu siehe Kapitel 2.5.4) nie negativ sein kann, liegen die Werte meist zwischen 0 und 1.

3. Korrelationskoeffizient nach Pearson
Die Pearson-Korrelation ist ebenfalls eine oft genutzte Methode, um die Ähnlichkeit zwischen zwei Items zu bestimmen. Mit ihr wird eine lineare Korrelation zwischen zwei Vektoren gefunden [LOONY16]:

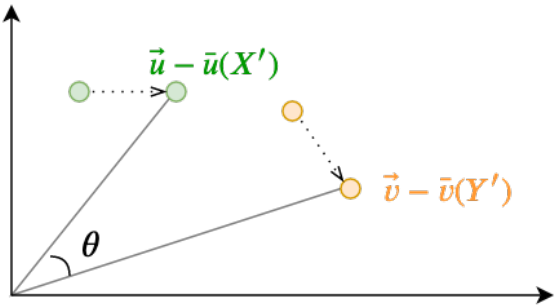


Abbildung 2.12: Pearson-Korrelation

Die Methode funktioniert im Prinzip wie die Kosinus-Ähnlichkeit, der Unterschied ist, dass hier nach dem Durchschnitt angepasst wird. Bedeutet: Jedes Rating eines Users U ist nach dem Mittelwert seiner Ratings normalisiert (siehe Abbildung 2.12). Ist dieser Null, so ist der Korrelationskoeffizient das Gleiche wie die Kosinus-Ähnlichkeit. Bei letzterer werden Skalarprodukte von Vektoren berechnet, während es sich bei der Pearson-Korrelation um die Kosinus-Ähnlichkeit zwischen standardisierten bzw. z-transformierten Vektoren handelt. Die Formel für die Berechnung der Pearson-Korrelation $\rho_{\mathrm{u,v}}$ ist in 2.5 dargestellt.

$$\begin{aligned} \rho_{\mathrm{u,v}} &= \frac{\sum_{i=1}^{n}(u_{\mathrm{i}} - \bar{u})(v_{\mathrm{i}} - \bar{v})}{\sqrt{\sum_{i=1}^{n}(u_{\mathrm{i}} - \bar{u})^2} \cdot \sqrt{\sum_{i=1}^{n}(v_{\mathrm{i}} - \bar{v})^2}} \\ &= \frac{< u - \bar{u}, v - \bar{v} >}{||u - \bar{u}|| \cdot ||v - \bar{v}||} \\ &= CosSim(u - \bar{u}, v - \bar{v}) \end{aligned} \tag{2.5}$$

Die Vektoren repräsentieren wieder die Ratings eines Users U, $\bar{u}$ bzw. $\bar{v}$ stehen für die Mittelwerte der Ratings der jeweiligen User.
Auch hier bewegen sich die Werte zwischen -1 und 1, wobei -1 eine perfekte negative Korrelation darstellt und 1 dementsprechend eine positive. Eine 0 bedeutet, dass keine Relation bzw. kein gerichteter Zusammenhang zwischen den beiden Vektoren vorliegt, dies wird auch Zero-Order Correlation (dt. Null-Korrelation) genannt.

2.5.2 Latentes Variablenmodell

Ein Latent Factor Model bzw. Latentes Variablenmodell (LV) ist eine beliebte model-based Methode für CF.

LVe arbeiten ebenfalls mit Userdaten und deren Bewertungen für verschiedene Produkte. Sie haben das Ziel, versteckte latente Faktoren zu identifizieren, die das Rating eines Users beeinflussen und somit die vorliegenden Bewertungen begründen [KOBE11]. Damit läuft der Vorgang analog zum CBF ab, mit dem Unterschied, dass die Faktoren vom Lernalgorithmus selbst identifiziert werden (siehe Abbildung 2.13). Diese können im echten Leben messbar sein (z.B. Genre oder Box-Office Erfolge bei Filmen), können aber auch abstrakte Variablen ohne eine konkrete Bedeutung sein.

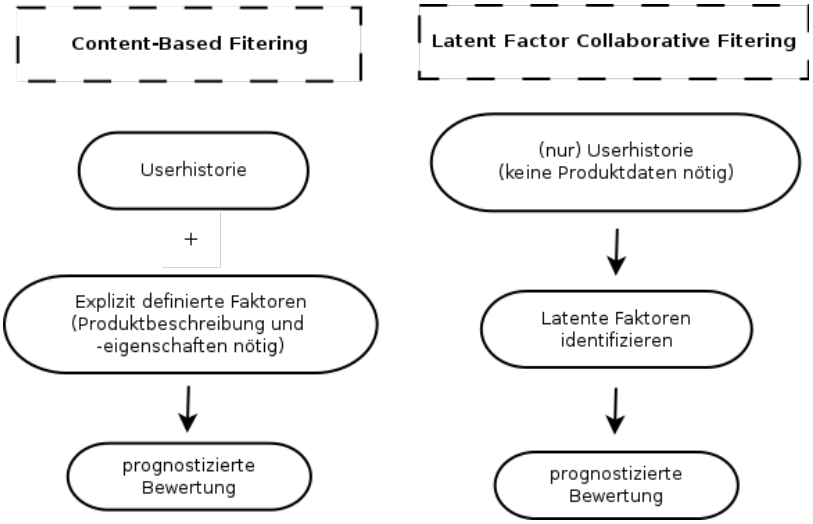


Abbildung 2.13: CBF und LV in Anlehnung an [LOONY16]

Viele erfolgreiche LVe basieren auf Matrixfaktorisierung (MF): Die Userdaten liegen, wie bei den kNN-Techniken, als User-Item-Rating-Matrix R vor. Um die Faktoren zu finden, wird R in eine User-Faktor-Matrix P und eine Item-Faktor-Matrix Q zerlegt (siehe Abbildung 2.14).
Diese Methode erlangte vor allem durch den Netflix Prize 2009 [NETFL09] große Bekanntheit und ist in Korens et al. Artikel beschrieben [KBV09]:

Jede Reihe in P beschreibt dabei einen User, die Spalten von P sind die latenten Faktoren. Bei Q repräsentiert jede Spalte ein Item, die Reihen sind hier die Faktoren.

Jedem Item I ist ein Vektor q_i zugehörig, jedem User ein Vektor p_u. Die Elemente von q_i beschreiben dabei, wie viel von jedem Faktor ein Item I besitzt. Die Elemente in p_u beschreiben die Affinität eines Users U für jedes der latenten Faktoren und damit ihr Interesse an solchen Items, die mit ebendiesen Faktoren übereinstimmen.
Das Skalarprodukt ${q_\mathrm{i}}^T p_\mathrm{u}$ beschreibt die Wechselbeziehung zwischen einem User U und einem Item I, also das Interesse von U an I. Mit folgender Formel kann damit das geschätzte Rating eines Users U für ein Item I berechnet werden:

$$\hat{r}_{\mathrm{u,i}} = {q_\mathrm{i}}^T \cdot p_\mathrm{u} \tag{2.6}$$

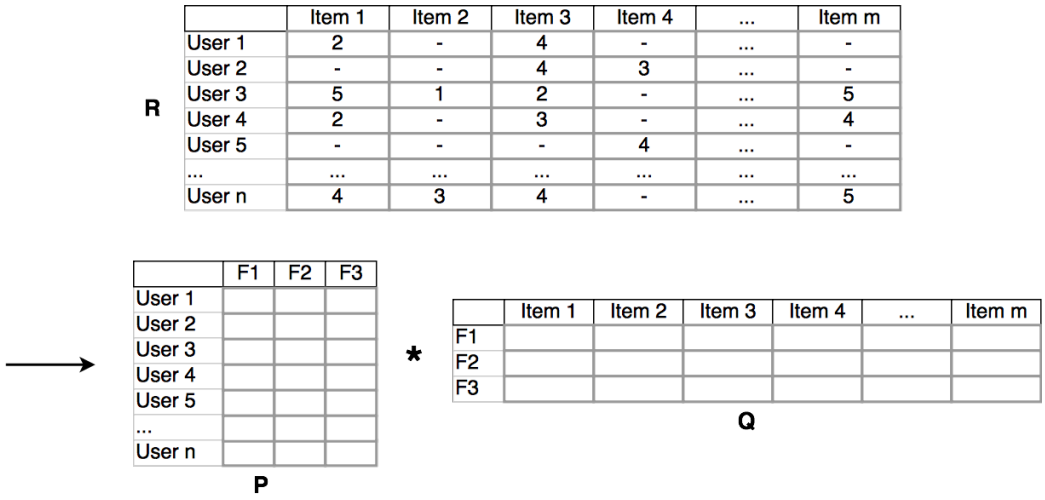


Abbildung 2.14: Matrixfaktorisierung

Damit kann das Recommender System, wenn jedem User und Item entsprechende Vektoren p_u und q_i zugeordnet sind, das Rating eines jeden Users für jedes Item vorhersagen bzw. schätzen.
Dieser Prozess ist vergleichbar mit dem, was bei der Singulärwertzerlegung einer Matrix (Singular Value Decomposition (SVD)) oder bei der Hauptkomponentenanalyse (Principal Component Analysis (PCA)) passiert. SVD ist eine bekannte Technik, die oft zur Identifizierung von latenten semantischen Faktoren im Bereich des IR genutzt wird. Diese Verfahren machen allerdings nur bei einer gut gefüllten Matrix Sinn, wenn also alle Ratings aller User für alle Items vorhanden sind.
Um das Verfahren der MF auch dann anwenden zu können, wenn die User-Item-Rating-Matrix viele fehlende Werte hat, wird nur für die Ratings gelöst, die auch zur Verfügung stehen. MF ist damit eine Methode zur Lösung des erwähnten Data Sparsity Problems. Eine Gleichung wie $r_{\mathrm{ui}} = p_{\mathrm{u}} \cdot q_{\mathrm{i}}$ wird also nur für alle vorhandenen Ratings in der Matrix R des Trainingsdatensatzes aufgestellt, mit Hilfe einer Regularisierung wird dann das sogenannte „Overfitting“ vermieden, welches auftritt, wenn das Modell den Trainingsdaten zu sehr angepasst ist:

$$\min_{q*,p*} \sum_{(u,i)\in\kappa} = (r_{\mathrm{ui}} - {q_{\mathrm{i}}}^T p_{\mathrm{u}})^2 + \lambda(\|q_{\mathrm{i}}\|^2 + \|p_{\mathrm{u}}\|^2) \tag{2.7}$$

κ ist hier die Menge an User-Item-Paaren, für die r_{ui} bereits bekannt ist. Der erste Teil des Terms ist damit der Fehler für ein Rating im Trainingsdatensatz, der zweite die Regularisierung, mit der Modelle mit zu vielen latenten Faktoren „bestraft“ (“penalized“) werden. Damit werden die Faktorvektoren p_u für jeden User U und q_i für jedes Item I gefunden, die den Fehler in den Trainingsdaten minimieren.
Zwei populäre Optimierungsverfahren für Formel 2.7 sind Stochastic Gradient Descent (SGD) und Alternating Least Squares (ALS). Für diese Arbeit wird letztere Methode verwendet.

Alternating Least Squares (ALS):
p_u und q_i sind beides Unbekannte, hält man eine der beiden Variablen, z.B. p_u, konstant, so wird die Gleichung quadratisch und kann nach q_i gelöst werden. In der nächsten Iteration wird q_i konstant gehalten und nach p_u gelöst. Es wird also abwechselnd nach p_u und q_i gelöst und der Vorgang so lange durchgeführt, bis p_u und q_i konvergieren. Bei ALS geht es also darum, folgende Gleichung zu lösen:

$$(r_{ui} - {q_i}^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) = 0 \tag{2.8}$$

ALS hat gegenüber SGD den Vorteil, dass jedes q_i unabhängig von den anderen q_i berechnet wird und ebenso jedes p_u unabhängig von den anderen p_u. Dies ermöglicht eine Parallelisierung des Algorithmus. Ein weiterer Vorteil ist außerdem, dass ALS sich besonders für implizite Daten eignet [KBV09].

2.5.3 Apriori Algorithmus

Assoziationsregeln sind eine weitere Möglichkeit, um Empfehlungen zu generieren. Sie sind vor allem dann von Vorteil, wenn Userinteraktionen anonym sind, es keine Features oder Ratings gibt oder eine hohe Privatsphäre gewährleistet werden soll [OPORMF19]. Dabei werden Regeln identifiziert, die die Anwesenheit von Items basierend auf der Anwesenheit von anderen Items innerhalb einer Transaktion vorhersagen [JZFF11] (siehe Kapitel 2.4.2).
Wenn also I die Menge aller Items und T ein Datensatz an Transaktionen ist, kann jede Transaktion t mit einem binären Vektor dargestellt werden. Dabei ist $t[k] = 1$, wenn das Item I_k in der Transaktion enthalten ist. Wenn es sich bei X und Y um eine Menge von Items, also Itemsets, handelt, wobei X und Y Teilmengen von I sind und keine gemeinsamen Elemente besitzen ($X \cap Y = \varnothing$), so sieht eine Regel folgendermaßen aus [AIS93; JZFF11]:

$$X \Longrightarrow Y \tag{2.9}$$

Bei einer Relation zwischen Items muss man zwischen Korrelation und Kausalität unterscheiden. Der Apriori Algorithmus ist dabei die bekannteste Technik, um signifikante Assoziationsregeln innerhalb eines großen Datensatzes an Kundentransaktionen zu finden.
Jede Regel muss ein gesetztes Minimum erfüllen, es gibt dabei zwei Werte, die berechnet werden [JZFF11]:

Support - Der Support beschreibt das Verhältnis von Transaktionen, die ein bestimmtes Itemset enthalten, in Relation zu allen vorliegenden Transaktionen. Es wird also ermittelt, wie häufig ein Itemset im Datensatz auftaucht. Dies lässt sich folgendermaßen berechnen:

$$\text{Support} = \frac{\text{Anzahl an Transaktionen, die } X \cup Y \text{enthalten}}{\text{Anzahl aller Transaktionen}} \tag{2.10}$$

Mit einem minimalen Support als Schwellwert wird damit eine minimale Anzahl von Transaktionen, die ein gegebenes Itemset enthalten, vorausgesetzt, damit eine Regel statistisch gesehen überhaupt signifikant ist. Wenn ein Item nicht oft genug gekauft wurde, gibt es nicht genug Daten, um die Regel zu unterstützen.

Confidence - Die Confidence zeigt, wie stark eine Assoziation ist, also das Verhältnis zwischen Transaktionen die $X \cup Y$ enthalten und denen, die X enthalten.

$$\text{Confidence} = \frac{\text{Anzahl an Transaktionen, die } X \cup Y \text{enthalten}}{\text{Anzahl der Transaktionen, die X enthalten}} \tag{2.11}$$

Es kann z.B. sein, dass die Items I_1 und I_2 zwar oft zusammen gekauft werden, dies aber daran liegen könnte, dass I_1 allgemein oft gekauft wird und daher in vielen Transaktionen enthalten ist. Somit würde keine Kausalität bei der Relation zwischen den beiden Items vorliegen. Für die Confidence wird ebenfalls ein Schwellwert festgelegt, der erreicht werden muss.

Das Ziel ist es also, Assoziationsregeln zu finden, die sowohl den Mindest-Support als auch die Mindest-Confidence erreichen. Der Apriori Algorithmus tut dies auf eine effiziente Art und Weise, da nicht alle möglichen Regeln generiert werden (siehe Abbildung 2.15):

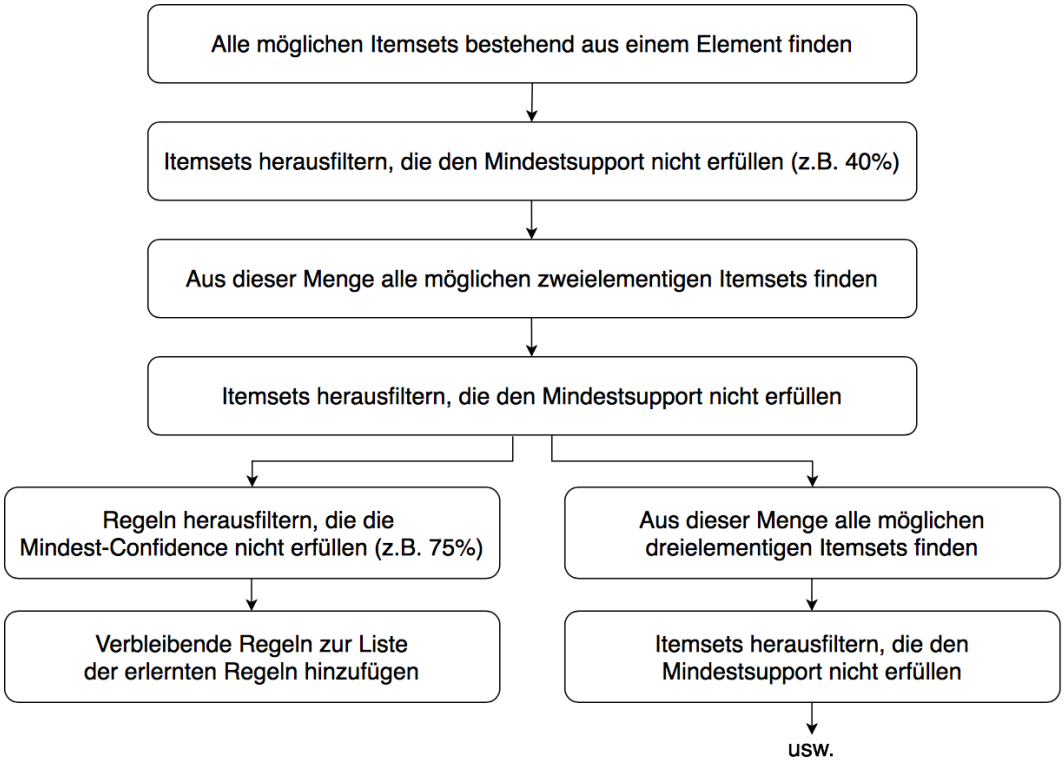


Abbildung 2.15: Apriori Algorithmus - Ablauf nach [LOONY16]

Es wird zunächst der Support für alle einelementigen Itemsets geprüft. Daraus werden dann alle möglichen zweielementigen Itemsets erstellt, die ebenfalls nach einem Mindest-Support gefiltert werden. Damit können dann bereits Assoziationsregeln erstellt werden, nachdem auf eine Mindest-Confidence geprüft wurde. Es können aber nach dem gleichen Prinzip Regeln für größere Itemsets gefunden werden, mit jedem Schritt fallen dabei Items weg, die den Mindest-Support nicht erfüllen. Der Vorgang kann so lange durchgeführt werden, bis keine größeren Itemsets mehr erstellt werden können [LOONY16].
Damit können Assoziationsregeln für Mengen unterschiedlicher Größen aufgestellt werden, die die gestellten Anforderungen erfüllen.

2.5.4 Term Frequency

Wie bereits erwähnt, sind die Features der Items bei den meisten CBF-Systemen textbasiert oder es handelt es sich um Entitäten wie Bücher und Artikel usw., was mit der Informationsgewinnung aus Dokumenten bei IR vergleichbar ist. Statt also kategorische Daten zu kodieren, können durch Natural Language Processing (NLP), also der maschinellen Verarbeitung natürlicher Sprache, ganze Texte verarbeitet werden. Recommender Systeme können die daraus gewonnenen Informationen nutzen und mit einbinden [FLHO03].
Content in textbasierten Systemen wird normalerweise mit Keywords beschrieben, die durch Methoden wie Rapid Automatic Keyword Extraction (RAKE) extrahiert werden könnnen [RECC10]. Die „Wichtigkeit“ eines solchen Wortes k_i in einem Dokument d_j wird mit Hilfe von Gewichtungsmaßnahmen bestimmt, eine der bekanntesten dabei ist die Term Frequency-Inverse Document Frequency (TF-IDF) [SHAM13]. Wörter, die häufig innerhalb eines Dokumentes vorkommen (Term Frequency), aber selten im Rest des Korpus auftauchen (Inverse Document Frequency), sind für ein Dokument wahrscheinlich relevanter. Die Wichtigkeit von Wörtern, die allgemein häufig vorkommen, wird damit vermindert. Bei einer Menge an Dokumenten bzw. einem Korpus $D = \{d_1, d_2, ..., d_N\}$ und einer Menge an Wörtern $T = \{t_1, t_2, ..., t_n\}$ wird TF-IDF folgendermaßen berechnet [LOPS11]:

$$\text{TF-IDF}(t_\mathrm{k}, d_\mathrm{j}) = \underbrace{\text{TF}(t_\mathrm{k}, d_\mathrm{j})}_{\text{TF}} \cdot \underbrace{\log \frac{N}{n_\mathrm{k}}}_{\text{IDF}} \tag{2.12}$$

N ist dabei die Anzahl der Dokumente im Korpus, n_k die Anzahl der Dokumente, in denen das Wort t_k mindestens einmal vorkommt. d_j wird als Vektor in einem n-dimensionalen Vektorraum repräsentiert, sodass $d_j = \{w_{1j}, w_{2j}, ..., w_{nj}\}$ gilt. w_kj ist dabei die Gewichtung eines Wortes t_k im Dokument d_j. TF wird mit folgender Funktion beschrieben:

$$\text{TF}(t_\mathrm{k}, d_\mathrm{j}) = \frac{f_{k,j}}{\max_z f_{z,j}} \tag{2.13}$$

Hier wird das Maximum der Häufigkeiten $f_{z,j}$ aller Wörter t_z, die im Dokument d_j vorkommen, berechnet. Dies steht in Relation zu der Häufigkeit von t_k in d_j.
Damit die Gewichtungen der Wörter im [0,1]-Intervall liegen und die Vektoren, die die Dokumente repräsentieren, die gleiche Länge haben, wird die Formel 2.12 noch normalisiert:

$$w_{k,j} = \frac{\text{TF-IDF}(t_k, d_j)}{\sqrt{\sum_{s=1}^{|T|} \text{TF-IDF}(t_s, d_j)^2}} \tag{2.14}$$

Für die anschließende Berechnung der Ähnlichkeit zwischen zwei Dokumenten wird meist die Kosinusähnlichkeit als Metrik verwendet (siehe Kapitel 2.5.1):

$$CosSim_{d_i,d_j} = \frac{\sum_{k=1}^{n} w_{ki} \cdot w_{kj}}{\sqrt{\sum_{k=1}^{n} w_{ki}^2} \cdot \sqrt{\sum_{k=1}^{n} v_{kj}^2}} \tag{2.15}$$

2.6 Data Preprocessing

Um Recommender Modelle zu erstellen, müssen die zur Verfügung stehenden Daten vorverarbeitet werden, beispielsweise müssen fehlende bzw. unvollständige Daten entweder vervollständigt oder entfernt werden. (Statistische) Ausreißer, die nicht zu den restlichen Daten passen, müssen ebenfalls behandelt werden, ebenso doppelte Einträge oder sonstige Ungereimtheiten in den Daten. Vor allem für CBF müssen aber die vorliegenden Produktdaten so transformiert werden, dass mit ihnen als Features gearbeitet werden kann.

2.6.1 Feature Engineering

Beim Feature Engineering werden rohe Daten auf Feature-Vektoren abgebildet. Im Allgemeinen kann man dort zwischen zwei Arten von Daten unterscheiden: Numerischen und kategorischen Daten.
Numerische Features bzw. Daten brauchen nicht kodiert zu werden, die Werte können meist direkt übernommen und bei Bedarf mit Hilfe von numerischen Faktoren und Gewichtungen angepasst werden.
Bei kategorischen Daten ist dies schwieriger. Sie besitzen eine diskrete Menge an möglichen Werten, z.B. gibt es ein Feature namens *geschlecht = {‚männlich', ‚weiblich', ‚unisex', ‚mädchen', ‚junge'}*. Da Zeichenketten in Recommender Modellen nicht verarbeitet werden können, wird Feature Engineering angewendet, um sie in numerische Werte umzuwandeln. Hale [HALE18] unterscheidet dabei zwischen sieben verschiedenen Arten von kategorischen Daten (siehe Tabelle 2.3).

Art	Beschreibung	Typ
Nutzlos	Für ML-Algorithmen nutzlos	diskret
Nominal	Gruppen ohne Reihenfolge bzw. Ordnung	diskret
Binär	Entweder/Oder	diskret
Ordinal	Gruppen mit Reihenfolge bzw. Ordnung	diskret
Count	Anzahl der Vorkommnisse	diskret
Zeit	zyklische Werte mit zeitlicher Komponente	kontinuierlich
Intervall	positive/negative Werte ohne zeitliche Komponente	kontinuierlich

Tabelle 2.3: Typen kategorischer Daten nach [HALE18]

Es gibt je nach Art verschiedene Möglichkeiten, wie diese kodiert bzw. transformiert werden können. Eine Variante ist es, jeden möglichen Wert eines Features auf einen numerischen Wert abzubilden, sogenannte Out-Of-Vocabulary-Kategorien (OOV), die also nicht zugeordnet werden können, werden in eine zusammengefasst [GOOGLEML]:

$$\begin{aligned} \text{‚männlich‘} &\rightarrow 0 \\ \text{‚weiblich‘} &\rightarrow 1 \\ \text{‚sonstiges‘ (OOV)} &\rightarrow 2 \end{aligned} \tag{2.16}$$

Diese Variante wird Label Encoding genannt, aus den Feature-Werten wird ein Vokabular erstellt, dem numerische Werte zugeordnet werden. Für ordinale Daten ist dies eine gute Kodierungsmöglichkeit, nicht jedoch für nominale. Diese Methode der Überführung in numerische Werte impliziert eine lineare Abhängigkeit zwischen den Kategorien, was aber nicht der Fall ist. In diesem Beispiel würde dies bedeuten *‚männlich‘ < ‚weiblich‘ < ‚sonstiges‘*, da $0 < 1 < 2$. Es gibt hier allerdings keine Ordnung oder bestimmte Reihenfolge. Label Encoding ist deswegen für den in dieser Arbeit vorgestellten Anwendungsfall ungeeignet.

Eine weitere Möglichkeit, mit kategorischen Features zu arbeiten, ist One-Hot Encoding [GOOGLEML]. Hier werden binäre Vektoren für jeden einzelnen kategorischen Wert erstellt. Für das Beispiel mit dem Feature *geschlecht* ist diese Variante in Abbildung 2.16 dargestellt.
Eine Spalte wird in mehrere Spalten aufgeteilt, jeder individuelle Feature-Wert von *geschlecht* bekommt eine eigene Spalte, der Wert ist dann 1, wenn dies für ein Item I zutrifft, ansonsten 0. Haben mehrere Spalten eines Features den Wert 1, so wird dies Multi-Hot-Encoding genannt.
Der Vorteil hier ist, dass One-Hot Encoding sowohl für ordinale als auch für nominale Daten funktioniert. Allerdings ist diese Variante bei Features, die eine hohe Kardinalität haben - wenn ein Feature sehr viele eindeutige Werte besitzt -, problematisch, da dies zu Speicherproblemen führt [HALE18].

	Geschlecht
Item 1	maedchen
Item 2	jungs
Item 3	weiblich
Item 4	maennlich
Item 5	unisex
...	...
Item n	maennlich

	maennlich	weiblich	unisex	maedchen	jungs
Item 1	0	0	0	1	0
Item 2	0	0	0	0	1
Item 3	0	1	0	0	0
Item 4	1	0	0	0	0
Item 5	0	0	1	0	0
...	...	...	...	...	...
Item n	1	0	0	0	0

Abbildung 2.16: One-Hot Encoding Beispiel

Es ist außerdem noch *Feature Weighting* möglich, um bestimmten Features eine höhere Gewichtung und damit mehr Wichtigkeit zu geben [DEB08].

2.6.2 Feature-Auswahl

Eine weitere Herausforderung bei der Datenvorverarbeitung ist die richtige Feature-Auswahl. Dies kann vor allem bei Datensätzen mit einer hohen Anzahl an Eigenschaften schwierig sein, da man nicht zu viele Attribute nutzen sollte. Eine Reduzierung von Features ist aus folgenden Gründen sinnvoll [NGZE18]:

- Vermeidung von Overfitting:
  Benutzt man viele Features, so optimiert man in einem Raum mit sehr hoher Dimension. Damit läuft man Gefahr, das Modell den Trainingsdaten zu sehr anzupassen (Overfitting), was mit dem sogenannten „Fluch der Dimensionalität“ zusammenhängt: Je höher die Dimensionalität, desto unbedeutender werden einzelne Beobachtungen bzw. Werte. Dementsprechend werden also mehr Daten benötigt, damit eine gewisse statistische Signifikanz erhalten bleibt.

- Vermeidung von Underfitting:
  Das Gegenteil zum genannten Overfitting ist das Underfitting. Wenn zu wenig Features genutzt werden bzw. allgemein zu wenig Daten vorliegen, können keine qualitativ hochwertigen und effizienten Modelle entwickelt werden.

- Reduzierung der Trainingszeit:
  Die Trainingszeit eines Modells hängt unter anderem auch von der Anzahl der Features ab. Je nachdem welche Metrik man hier wählt, kann dies stärker oder schwächer ins Gewicht fallen.

- Bessere Interpretierbarkeit der Daten:
  Je weniger Features genutzt werden, desto einfacher ist es im Nachhinein, die Ergebnisse eines Modells nachzuvollziehen. Die Kausalität lässt sich leichter erkennen.

Gut geeignete Features zu finden ist ebenfalls eine Herausforderung, vor allem wenn man die Anzahl dieser reduziert [GOOGLEML]:
Zunächst einmal sollte man keine Eigenschaften als Features benutzen, die nur selten vorkommen. Basiert man das Recommender Modell auf solchen Werten, können keine verlässlichen Vorhersagen getroffen werden: Das Feature ist kein geeigneter Indikator, da es den Großteil der Items nicht repräsentiert, das Modell könnte also nichts Nützliches daraus lernen. Außerdem sollten die Attribute klar und verständlich sein und sogenannte „magische Zahlen“ sollten nicht verwendet werden. Beispiel: Ein Feature erwartet nur Werte im Intervall von 0.0 bis 1.0. Für Items, bei denen der Wert unbekannt ist, wird -1.0 eingesetzt. Dies würde allerdings das Modell beeinflussen und muss dementsprechend beachtet werden.
Um eine gute Feature-Auswahl zu treffen, kann außerdem die Korrelation zwischen den Itemattributen und dem Target untersucht werden. Eine hohe Korrelation bedeutet, dass ein Feature relevant und informativ ist [NGZE18].

2.7 Evaluation eines Recommender Systems

Allein Recommender Modelle zu erstellen reicht nicht. Einer der wichtigsten Schritte bei der Entwicklung eines Recommender Systems ist die Evaluation, bei der untersucht wird, wie effektiv die erstellten Modelle überhaupt sind.

2.7.1 Offline vs. Online

Recommender Systeme können auf verschiedene Weisen evaluiert werden, man unterscheidet dabei zwischen Offline und Online Evaluation.

1. Offline Evaluation
Offline Experimente, also Tests vor der womöglichen Aufnahme in den Live-Betrieb, sind die einfachste Variante, um Modelle zu evaluieren: Hier werden die bereits vorliegenden Ratingdaten als Grundlage genutzt. Ein großer Vorteil hier ist, dass die Interaktion mit realen Usern nicht nötig ist [SHANI11]. So können mehrere unterschiedliche Modelle mit wenig verbundenen Kosten evaluiert werden.
Andererseits wird bei der Offline Evaluation von vergangenen Beobachtungen ausgegangen. Basierend auf diesen wird angenommen, dass sich User auch genau so verhalten hätten, wenn der neue Empfehlungsalgorithmus genutzt worden wäre, um Empfehlungen zu generieren [GOHU15]. Der Einfluss des Recommender Systems auf das Userverhalten kann daher nicht beurteilt werden. Damit gibt es neben der Vorhersagekraft nicht viel mehr Kritikpunkte, die bei der Offline Evaluation ebenfalls bewertet werden können. Das Ziel hier ist es, Modellansätze, die gar nicht geeignet sind, herauszufiltern und so die Menge an möglichen Kandidaten zunächst zu reduzieren [SHANI11].

2. User Studies
Eine weitere aufwändigere Möglichkeit der Evaluation sind User Studies [SHANI11]: Kleine Gruppen von Usern bekommen verschiedene Aufgaben (z.B. „Suche dir ein neues Wander-Outfit heraus.“), die eine Interaktion mit dem neuen Recommender System erfordern.
Das Verfahren ist sehr aufwändig und ist mit Mehrkosten verbunden und wird in dieser Arbeit deswegen nicht eingehender untersucht.

3. Online Testing
Als letzte Variante gibt es noch das Online Testing, bei dem das neue Recommender System z.B. in den laufenden Online-Shop implementiert wird [SHANI11]: Bei solchen Experimenten wird das System an realen Usern getestet (die nicht wissen, dass sie Teil der Evaluation sind), sodass der Einfluss des Systems auf Userverhalten direkt getestet werden kann.
Die Auswirkung eines Recommender Systems hängt von vielen Faktoren ab, Online Evaluation liefert die realistischsten Ausgangsbedingungen und damit die verlässlichsten Ergebnisse bezüglich der Effizienz und Wirkung eines Systems in der Realität. Einfluss auf den Umsatz und die Kundenbindungsrate können ebenfalls direkt gemessen werden. Es können zudem mehrere Modelle getestet werden, indem ein kleiner Teil des Datenverkehrs zu einer anderen Recommender Engine geleitet wird. Ablauf und Verhalten können wieder protokolliert und anschließend miteinander verglichen werden.
Zu beachten ist hierbei, dass die User dafür zufällig gewählt werden, um eine repräsentative Gruppe zu erhalten. Außerdem ist Online Testing auch mit einem gewissen Maß an Risiko verbunden: Stellt sich ein neues Recommender Modell nämlich als ineffizient heraus, könnte dies die User dazu demotivieren, das System in Zukunft weiterhin zu nutzen. Daher sollte vor Online Experimenten erst eine Offline Evaluation durchgeführt werden, um gänzlich ungeeignete Ansätze bereits vorher herauszufiltern.

2.7.2 Accuracy als Hauptmetrik

Ein Hauptaspekt, unter dem Recommender Systeme meist beurteilt werden, ist die Accuracy bzw. Genauigkeit. Bei dieser Metrik geht es darum, zu messen, wie sehr die generierten Empfehlungen oder ein vorhergesagtes Rating für ein Item I mit den Interessen eines gegebenen Users oder dem Userrating für I übereinstimmen [HERL04]. Es wird also untersucht, wie treffend die Empfehlungen sind, die das System generiert. Gerade bei Offline Evaluation ist Genauigkeit ein wichtiges Bewertungskriterium, da andere Kriterien offline nur schwer zu beurteilen sind.
Genauigkeit ist zwar ein wichtiger Kritikpunkt, jedoch alleine nicht ausreichend, um eine allgemeingültige Aussage über die Effizienz und Relevanz eines Recommender Systems treffen zu können [MCRJ16]. Diese hängt unter anderem auch davon ab, welche Funktionen ein Recommender System überhaupt erfüllen soll (siehe Kapitel 2.1.1). Außerdem spielt auch die Nützlichkeit eine Rolle: User Experience und

Kundenzufriedenheit sind zwei wichtige Faktoren, die nicht zwangsläufig von der Genauigkeit abhängen. Neben der Accuracy gibt es weitere Punkte, unter denen Empfehlungen evaluiert werden können, Herlocker et al. [HERL04] nennen als weitere Kriterien beispielsweise die Serendipity und Novelty: Hierbei wird ein Recommender System danach evaluiert, wie „neu und überraschend“ die Empfehlungen sind. Bei Novelty werden einem User neue Items empfohlen, mit denen er noch nicht interagiert hat, während es bei Serendipity darum geht, dem User überraschende aber trotzdem interessante Items zu empfehlen, die er ansonsten nicht entdeckt hätte. Beispiel: Ein Recommender System empfiehlt einem User Filme basierend auf seinem Lieblingsregisseur. Ihm wird ein Film von diesem Regisseur empfohlen, den er noch nicht gesehen hat, die Empfehlung ist also neu aber nicht überraschend: Der User hätte den Film vermutlich früher oder später selbst entdeckt. Eine Empfehlung mit einer hohen Serendipity wäre ein Film mit einem neuen Regisseur, den der User aber trotzdem interessant findet. Hier geht es also nicht um Genauigkeit oder Richtigkeit, sondern um eine verbesserte User Experience.
Ein weiterer Punkt ist die Diversity [SZLLM17]: Hier wird ein Recommender System danach beurteilt, wie vielfältig die Empfehlungen sind. Eine Liste ohne viel Abwechslung und nur ähnlichen Items ist für einen User eintönig und möglicherweise nicht interessant. Unter dem Kriterium (Item Space) Coverage wird gemessen, von wie vielen Items das Recommender System Empfehlungen anbieten kann, also die Menge an Items, mit denen ein System arbeitet.

Es wird ersichtlich, dass die Effizienz eines Recommender Systems von mehreren Faktoren abhängt und unter verschiedenen Gesichtspunkten evaluiert werden kann, je nachdem auf welche Kriterien (z.B. Accuracy, Serendipity, Novelty, Coverage, Diversity,...) mehr Wert gelegt wird. Für diese Arbeit wird lediglich eine Offline Evaluation durchgeführt und daher Accuracy als Hauptmetrik verwendet.

2.7.3 Leave One Out Cross Validation

Für die Evaluation unter dem Gesichtspunkt der Accuracy sind in der Literatur eine Vielzahl an Metriken zu finden, der Mean Absolute Error (MAE), der Root-Mean-Square Error (RMSE), Precision und Recall, die Receiver Operating Characteristics (ROC)-Kurve und die Area Under The ROC Curve (AUC) sind einige beliebte Verfahren. Die meisten dieser Techniken sind jedoch auf explizite Daten ausgelegt und bewerten, wie gut ein Rating vorhergesagt wird.
Eine weitere Metrik, mit der Recommender Systeme im Bezug auf Genauigkeit evaluiert werden können und die ebenfalls oft in der Literatur auftaucht, ist die sogenannte Leave One Out Cross Validation (LOOCV) [JEUNEN18]. Dieses Verfahren ist für die Evaluation bei implizitem Feedback geeignet, hier wird ein Recommender Modell auf seine Trefferquote untersucht.

Dazu wird der Datensatz wie bereits erwähnt in einen Trainings- und einen Testdatensatz eingeteilt. Für gewöhnlich werden dazu zufällig ausgewählte Reihen aus dem Datensatz entfernt und in einem neuen separaten Datensatz zusammengefügt (siehe Abbildung 2.17).

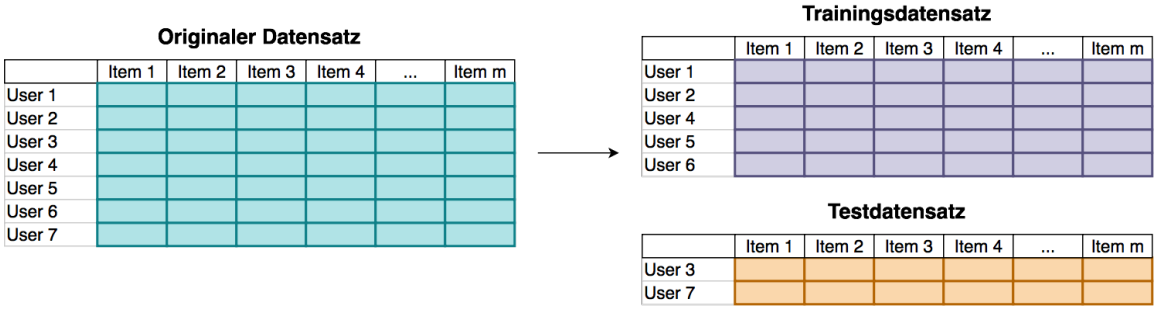


Abbildung 2.17: Aufteilung in Trainings- und Testdaten

Bei CF ist dies jedoch problematisch, da alle User-Item-Interaktionen benötigt werden, um ein effizientes Recommender Modell zu entwickeln. Für das LOOCV-Verfahren wird für jeden User eine Item-Interaktion zufällig ausgewählt, die aus dem Datensatz entfernt wird und zu den Testdaten geordnet wird. Die verbleibenden Daten bilden den Trainingsdatensatz. Eine Interaktion pro User wird also versteckt (siehe Abbildung 2.18).

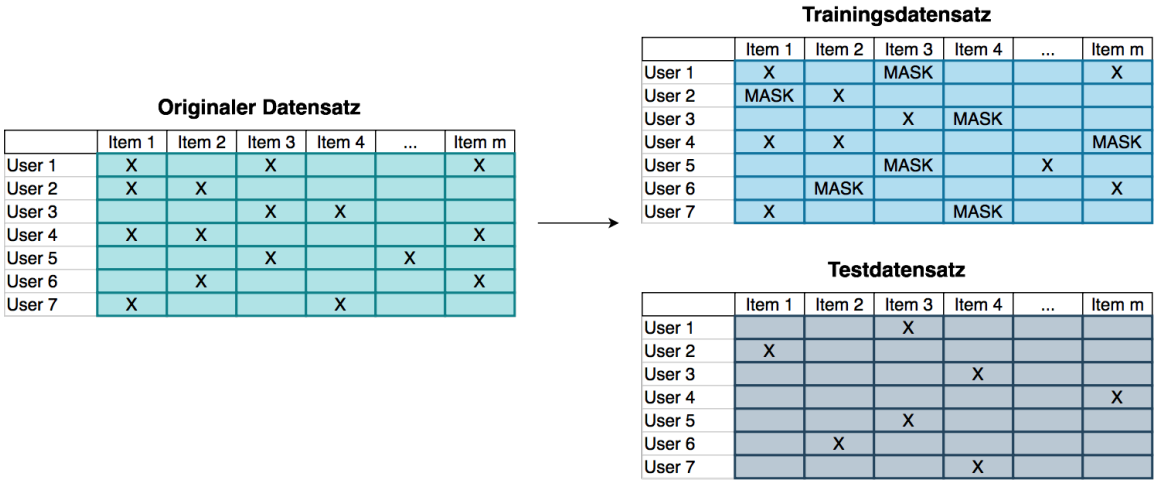


Abbildung 2.18: Aufteilung in Trainings- und Testdaten bei LOOCV

Das Ziel ist es, das Recommender System danach zu beurteilen, ob das weggelassene Item während der Testphase richtig vorhergesagt wird: Für einen User U wird, basierend auf den Daten aus dem Trainingsdatensatz, eine Liste von Top N Empfehlungen erstellt und dann geschaut, ob bei dieser Liste das Item aus dem Testdatensatz dabei bist. Dies wird entweder für alle vorhandenen User durchgeführt oder für mehrere

kleinere Teilmengen an Usern. Aus den Ergebnissen wird eine Trefferquote berechnet:

$$\text{HitRate@N}_k = \frac{\text{Anzahl an Treffern für k}}{\text{k}} \tag{2.17}$$

N steht hier für die Anzahl an generierten Empfehlungen, also der Top N Liste, k für die Anzahl der User, für die getestet wird. Die Trefferquote liegt dabei in einem Intervall von [0,1], wobei bei einem Wert von 1.0 die versteckte Item-Interaktion für jeden User k_i richtig vorhergesagt wurde. Ein Wert von 0.0 bedeutet, dass das Item für keinen User in der generierten Empfehlungsliste aufgetaucht ist. Je größer N gewählt wird, desto größer ist dementsprechend auch die Chance, dass unter den Empfehlungen das Item aus dem Testdatensatz vorhanden ist. Daher wird die Trefferquote immer in Abhängigkeit von N angegeben.
Dieses Verfahren wird für mehrere unterschiedliche Gruppen und Größen von Usern und verschiedenen Aufteilungen von Trainings- und Testdaten durchgeführt, um zuverlässige Ergebnisse zu erzielen. Daraus wird dann ein Mittelwert berechnet.

2.8 Tools

Für die Entwicklung der verschiedenen Empfehlungsstrategien wird in dieser Arbeit Python verwendet. Diese Wahl wurde aus folgenden Gründen getroffen:
Python ist eine der populärsten und am weitesten verbreitetsten Programmiersprachen für maschinelles Lernen. Sie ist recht leicht zu lernen und anzuwenden. Außerdem bietet sie gerade durch ihre Beliebtheit eine große Community, die einem weiterhelfen kann und es findet sich viel Material im Zusammenhang mit Machine Learning und künstlicher Intelligenz. Das Ausführen und Testen von Code gestaltet sich ebenfalls simpel und es existieren viele nützliche Bibliotheken.
Folgende werden dabei in dieser Arbeit verwendet: Pandas, Numpy, SciPy, Scikit, Rake und Implicit. Außerdem werden Microsoft Excel sowie Pythons Seaborn und Matplotlib für die Visualisierung der Daten benutzt.

3 Entwicklungsprozess

3.1 Datensammlung

3.1.1 Datensätze

Zur Entwicklung der unterschiedlichen Strategien in dieser Arbeit werden die gleichen Datensätze als Grundlage genutzt. Diese stammen aus der Domain eines Outdoor & Travel iSHOPs der novomind AG. Für die produktattributbasierten Empfehlungen, also CBF, wird auf die Stammdaten bzw. Produktattribute aus dem Objektspeicher des Shops zugegriffen. Die daraus generierte csv-Datei umfasst 12461 Artikel mit 17 verschiedenen Eigenschaften (Stand: 16.10.19, siehe Tabelle A.4). Das daraus resultierende Dataframe wird *df_productfeed* genannt.
Für CF sowie die Assoziationsregeln werden implizite Ratings zur Auswertung verwendet, da für diesen Shop keine expliziten Ratingdaten zur Verfügung stehen. Diese bestehen aus Eventtrackingdaten, die aus dem Hadoop Cluster der novomind AG stammen, also Bewegungs- bzw. Klickdaten. Zur Verfügung stehen dabei zwei verschiedene Datensätze: Die productview-Daten, also wann immer sich ein User etwas auf der Artikeldetailseite angesehen hat, und die addtobasket-Events, also jedes Mal, wenn ein User ein Produkt seinem Warenkorb hinzugefügt hat. Die productview-Datei umfasst 1572552 Einträge, die addtobasket-Datei 52843 (Stand: 23.10.19) (siehe Tabelle A.3). Gesammelt wurden die Daten über einen Zeitraum von ca. fünf Monaten.

In dieser Arbeit wird sowohl nur mit den productview-Daten gearbeitet, als auch mit beiden Datensätzen zusammen. Der folgende Prozess beschreibt die Filterung der Daten aus beiden Dateien, die Filterung nur aus den productview-Daten erfolgt analog dazu. Alle in dieser Arbeit angegebenen beispielhaften Dateiauszüge sind außerdem so angepasst, dass sie keine personenbezogenen Daten enthalten und damit kein Rückschluss auf eine konkrete reale Person mehr möglich ist.
Da timestamps hier nicht berücksichtigt werden, werden aus den beiden Dateien nur die *userId*- und die *articleId*-Spalten benötigt. Die Werte darin werden in numerische Werte umgewandelt, um das Arbeiten mit diesen zu erleichtern. Duplikate (wenn sowohl *userId* als auch *articleId* einer Zeile mit anderen Zeilen übereinstimmen) werden aus den Datensätzen entfernt. Den Dataframes wird eine weitere Spalte *eventType* hinzugefügt, 'VIEW' bei den productview-Einträgen und 'ADD_BASKET' bei addtobasket. Dann werden die beiden Dataframes miteinander konkateniert.
Angelehnt an [LI19] werden die Klickdaten als implizite Ratings der User genutzt.

Um auf diese die Techniken anwenden zu können, die in der Literatur häufig für explizite Daten eingesetzt werden, wird den vorliegenden Daten eine Gewichtung für die verschiedenen Events folgendermaßen zugeordnet:

```
event_type_strength = {
    'VIEW': 1.0,
    'BOOKMARKED':3.0,
    'ADD_BASKET': 4.0,
    'ORDERED': 5.0
}
```

Ausschnitt 3.1: Eventtype Ratings

Da für diesen Shop nur die productview- und addtobasket-Daten vorliegen, sind die ‚BOOKMARKED' und ‚ORDERED' Events hier nicht nötig, jedoch der Vollständigkeit halber aufgeführt, da sie später bei anderen Shops nützlich sein könnten. Auf den weiteren Verlauf der Arbeit haben sie keinen Einfluss.
Die Werte werden nun für jede [*userId*, *articleId*]-Gruppierung aufsummiert. Der neue Datensatz wird nach Schwellenwerten gefiltert, da beispielsweise Artikel mit zu wenig Transaktionsdaten nicht aussagekräftig genug sind. Genau so auch User, die nur mit einem Artikel interagiert haben. Bei Untersuchung der Daten nach Liaos Vorgehensweise [LIAO19] werden folgende Dinge ersichtlich: Es gibt 350583 verschiedene User, von denen sich nur 25% zwei oder mehr Artikel angesehen bzw. in den Warenkorb gelegt haben. Der Höchstwert für die Anzahl an verschiedenen Artikeln, mit denen ein einzelner User interagiert hat, liegt bei 846. Veranschaulicht ist dies am Graphen in Abbildung 3.1.

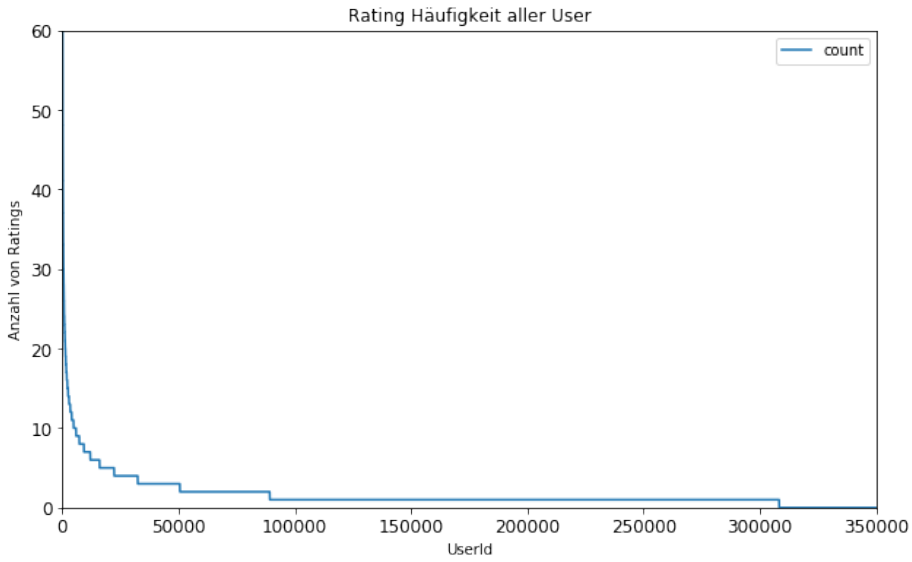


Abbildung 3.1: Rating Häufigkeit aller User

Es gibt außerdem 10632 verschiedene Artikel, von denen ca. 75% mit 63 oder mehr Usern Interaktionen hatten. Der Höchstwert hier liegt bei 4031, sprich: Mit einem einzelnen Artikel haben bis zu 4031 unterschiedliche User interagiert (Abbildung 3.2).

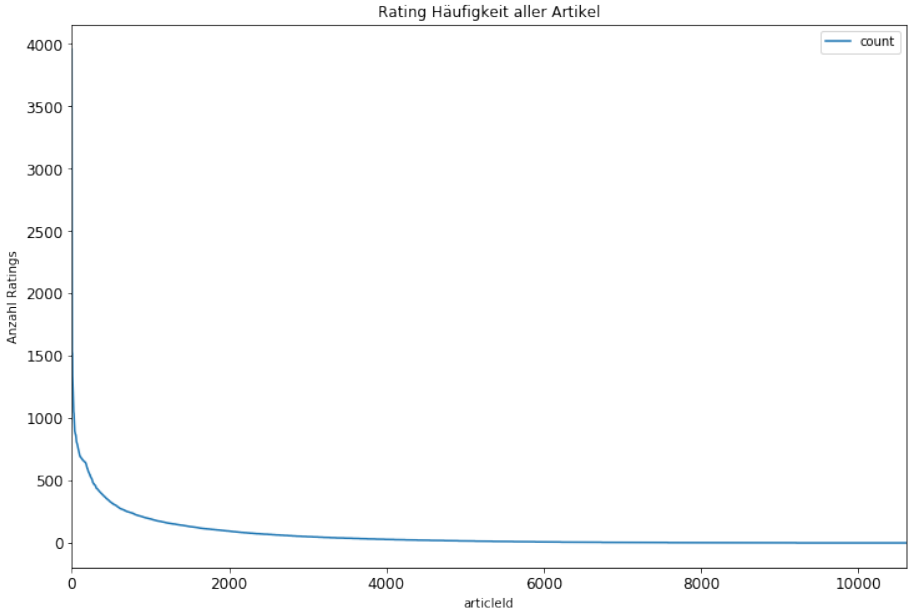


Abbildung 3.2: Rating Häufigkeit aller Artikel

Der „Beliebtheits“-Grenzwert, also der Schwellenwert für die Rating Häufigkeit aller Artikel, wird auf 63 festgelegt, der „Ratings“-Grenzwert der User auf zwei, sodass von den insgesamt 717440 Einträgen noch 391073 mit 89490 verschiedenen Usern und 2674 verschiedenen Artikeln übrig bleiben. Mit diesem Datensatz (nachfolgend *df1* genannt) wird im Folgenden weitergearbeitet (siehe Tabelle 3.1).

	userId	articleId	title	rating
50776	5g656-456j65	123-12345	Schale Infinity	5.0
586208	1234a5-6789b	345-67891	Powder Winter Pant	1.0

Tabelle 3.1: Auszug gefilterte Bewegungsdaten aus productview- und basket-Dateien

Der Datensatz mit den gefilterten Bewegungsdaten, die nur aus der productview-Datei stammen, wird genau so erstellt und umfasst 383545 Einträge mit 87598 verschiedenen Usern und 2609 verschiedenen Artikeln (nachfolgend *df2* genannt).

3.1.2 Datenaufteilung

In Vorbereitung für das bereits erwähnte LOOCV-Evaluierungsverfahren, werden die gefilterten Bewegungsdaten in einen Trainings- und einen Testdatensatz eingeteilt.

Dazu werden im Datensatz *df1* bzw. *df2* für jede eindeutige *userId* die Artikel zusammen gruppiert, mit denen der User interagiert hat. Von diesen Gruppierungen wird jeweils ein Eintrag entfernt, der dem Testdatensatz hinzugefügt wird. Die restlichen Einträge des Users kommen in den Trainingsdatensatz. So wird für jeden User, wie der Name des Verfahrens impliziert, ein Artikel ausgelassen, der dem User dann im besten Fall beim Evaluationsprozess vom Recommender Modell wieder vorgeschlagen wird.
Im Folgenden ist mit User der Kunde/die Kundin des Travel- & Outdoor Shops gemeint, also die NutzerInnen der Webseite. Mit den Items, also den Entitäten, die das System empfiehlt, sind die verschiedenen Artikel dieses Shops gemeint. Die Begriffe Item, Produkt und Artikel werden synonym verwendet.

3.2 Candidate Generation

Für die Generierung der Empfehlungen gibt es viele verschiedene Ansätze und Techniken. Im Folgenden werden vier unterschiedliche State of the Art Verfahren implementiert und untersucht.

3.2.1 Memory-Based Collaborative Filtering

3.2.1.1 Item-Based vs. User-Based

Beim Memory-Based CF unterscheidet man, wie vorher erwähnt, zwischen IB-CF und UB-CF (siehe Kapitel 2.2.2.1). Dabei wird bei der Auswahl zwischen einer userbasierten und itembasierten Strategie nach Desrosiers und Karypis [DEKA11] u.a. auf die folgenden Kriterien geschaut:

- Effizienz/Laufzeit:
  Wie auch in der Publikation zum Recommender System bei Amazon [LSY03] erläutert, beträgt die Laufzeit von UB-CF im schlechtesten Falle $\mathcal{O}(nm)$ für n User und bis zu m Items für jeden User. Aufgrund der Spärlichkeit der User-Item-Matrix liegt die Laufzeit praktisch gesehen aber eher bei $\mathcal{O}(n + m)$: Da User meist nur mit einem kleinen Teil der Items tatsächlich interagieren bzw. diese bewerten, liegt die Laufzeit-Performance des Algorithmus' näher bei $\mathcal{O}(n)$. User, die sehr viele Items bewertet haben, benötigen $\mathcal{O}(m)$ Berechnungszeit.
  Verglichen damit liegt die Laufzeit bei IB-CF für das Berechnen der Item-Item-Matrix im schlechtesten Falle bei $\mathcal{O}(m^2n)$, in der Realität aufgrund der spärlichen Daten vermutlich eher bei $\mathcal{O}(mn)$ [LSY03].
  Im Vergleich zu IB-CF ist die Laufzeit für UB-CF zwar geringer, die Item-Item-Matrix bei letzterem kann jedoch offline berechnet werden, was bei UB-CF nicht möglich ist. Laut Desrosiers und Karypis hängt die Performance vom Verhältnis zwischen Usern und Items ab: Gibt es mehr User als Items - was meist der Fall ist - können itembasierte Empfehlungen von Vorteil sein.

- Genauigkeit:
  Wie die Effizienz hängt auch die Genauigkeit bzw. Präzision der Empfehlungen vom oben genannten Verhältnis ab. In Fällen, in denen die Useranzahl die der Itemanzahl weit übersteigt, können itembasierte Methoden genauere Empfehlungen liefern. Andersherum kann in der Hinsicht der userbasierte Ansatz bei Systemen mit weniger Usern als Items bessere Ergebnisse erzielen.

- Stabilität:
  Die Auswahl zwischen UB-CF und IB-CF beruht u.a. auch auf den Veränderungen bei Usern und Items: Ist die Liste der verfügbaren Items relativ statisch, kann hier wieder die itembasierte Methode vorteilhafter sein. Ändert sich diese jedoch konstant, ist die userbasierte Variante zu bevorzugen.

- Serendipity:
  Bei Item-Based Methoden werden Ratings für Items, basierend auf den Ratings, die ähnliche Items erhalten haben, vorhergesagt. Folglich führt dies zwar zu „sicheren“ Empfehlungen, hilft dem User allerdings nicht dabei, viele verschiedene Typen von Items zu entdecken. Bei der Serendipity schneidet im Durchschnitt also das UB-CF besser ab.

3.2.1.2 Implementierung des IB-CFs

Für die Candidate Generation wird als erstes ein Memory-Based CF Verfahren angewendet. Dazu wird ein IB-CF Modell entwickelt, dass auf Nearest-Neighbour-Algorithmen beruht. Die Entscheidung fällt hier auf IB-CF statt UB-CF, da IB-CF aufgrund der im vorherigen Abschnitt erwähnten Punkte vorteilhafter erscheint. UB-CF wird aufgrund der höheren Zeit- und Speicherkosten in der Praxis selten verwendet [BCT11]. Bei dem hier vorliegenden Datensatz gibt es viel mehr User als Items, außerdem liegt der Fokus hier auf passenden Empfehlungen zu unterschiedlichen Produkten und weniger auf personalisierten Empfehlungen, die speziell auf einen Nutzer/eine Nutzerin zugeschnitten sind. Item-Item-Beziehungen sind zudem intuitiver zu erklären als User-User-Beziehungen.

Ausgangslage ist der Trainingsdatensatz *df1* bzw. *df2*. Aus diesem Dataframe wird mittels der *pivot_table*-Funktion aus der Programmbibliothek Pandas eine Item-User-Rating-Matrix R erstellt, wobei mit r_{ij} das Rating des i-ten Items vom j-ten User gemeint ist. Es wird außerdem ein zweites Dataframe *df_products* erstellt, das *articleId*, *Titel* und andere Merkmale enthält, um später nachvollziehen zu können, um welchen Artikel es sich handelt. Der Index dieses zweiten Dataframes ist dabei die *articleId*, um das Finden dieser einfacher und vor allem schneller zu gestalten.
Die erstellte Item-User-Rating-Matrix ist Grundlage, um für das IB-CF eine Item-Item-Matrix zu generieren. Für die Berechnung der Ähnlichkeiten zwischen den Produkten werden verschiedene Distanzmetriken verwendet.

- Euklidischer Abstand:
  Die erste zu untersuchende Metrik ist der euklidische Abstand. Berechnet werden die Distanzen mit Hilfe der SciPy-Bibliothek.

```
from scipy.spatial import distance
euclidean_distances = distance.cdist(itemUserRatingMatrix,
    itemUserRatingMatrix, metric='euclidean')
itemItemMatrix = pd.DataFrame(euclidean_distances,
    index=itemUserRatingMatrix.index,
    columns=itemUserRatingMatrix.index)
```

Ausschnitt 3.2: Berechnung euklidischer Abstände

distance.cdist() berechnet den Abstand zwischen jedem Paar aus den beiden gegebenen Datensammlungen mit der Formel aus 2.2. Die Werte x_i und y_i sind dabei die Ratingwerte der verschiedenen User für die Items X und Y.
Die Funktion gibt ein Numpy Array zurück, welches dann in ein Dataframe bzw. eine Item-Item-Matrix umgewandelt wird, mit den *articleIds* der Produkte als Indizes für die Zeilen und Spalten (siehe Tabelle 3.2).

articleId	0	1	2	5	6
0	0.000000	20.248457	19.493589	20.074860	19.235384
1	20.248457	0.000000	10.862780	11.180340	9.899495
2	19.493589	10.862780	0.000000	10.246951	9.055385
5	20.074860	11.180340	10.246951	0.000000	9.433981
6	19.235384	9.899495	9.055385	9.433981	0.000000

Tabelle 3.2: Ausschnitt der Item-Item-Matrix (Distanzen) mit Euklid

Um mit der erstellten Matrix nun Empfehlungen für ein gegebenes Produkt zu generieren, wird die Spalte mit der entsprechenden *articleId* herausgesucht. Da es sich bei den Werten um Distanzen handelt, wird nach aufsteigenden Werten sortiert. Das Item mit dem geringsten Abstand zu dem gegebenen Produkt steht also an erster Stelle, was im besten Falle das Produkt selbst sein sollte. Für die Top N Empfehlungen werden also die ersten N *articleIds* dieser Liste benutzt, wobei die erste Zeile nicht mit berücksichtigt wird, da es sich hier um das Produkt selbst handelt. Um direkt sehen zu können, um welchen Artikel es sich handelt, kann *df_products* genutzt werden. Hier kann über den Index direkt auf das entsprechende Produkt zugegriffen werden. Heraus kommt damit eine Liste von N Empfehlungen zu einer gegebenen *articleId*.
Möchte man nun Empfehlungen generieren, die auf einen bestimmten User zugeschnitten sind, wird ähnlich zu dem in [GRUS19] beschriebenen Prinzip auf die Artikel, mit denen der User bereits interagiert hat (also die entsprechenden

Items aus *df1* bzw. *df2*), das obige Verfahren angewendet, um passende Empfehlungen zu diesen zu erhalten. Diese neue Liste wird dann wieder nach den Distanzen in aufsteigender Reihenfolge sortiert, von denen dann die ersten N Empfehlungen zurückgegeben werden. Für die Produktinformationen werden die entsprechenden Zeilen aus *df_products* herausgefiltert.

- Kosinus-Ähnlichkeit:
  Für die Berechnung der Ähnlichkeiten auf Basis der Kosinus-Ähnlichkeit wird wieder die SciPy-Bibliothek verwendet.

```
from scipy.spatial import distance
distances = distance.pdist(itemUserRatingMatrix, 'cosine')
itemItemMatrix = pd.DataFrame(distance.squareform(distances),
    index=itemUserRatingMatrix.index,
    columns=itemUserRatingMatrix.index)
```

Ausschnitt 3.3: Berechnung der Kosinus-Ähnlichkeit

distance.pdist() berechnet den Kosinus-Abstand $d_{\cos}$ zwischen jedem Paar mit der Formel aus 3.1. u_{i} und v_{i} (siehe Formel 2.4) sind dabei die Ratingwerte der verschiedenen User für die Items u und v.

$$d_{\cos}(u,v) = 1 - CosSim(u,v) = 1 - \frac{\vec{u} \cdot \vec{v}}{||\vec{u}|| \cdot ||\vec{v}||} \tag{3.1}$$

Auch hier wird ein Array zurückgegeben, mit dem dann, wie beim Verfahren zum euklidischen Abstand, ein Dataframe erstellt wird. Indizes für die Zeilen und Spalten sind hier wieder die *articleIds*. Da das Gewicht der Ratingwerte hier niemals negativ sein kann, liegt die Kosinus-Ähnlichkeit stets zwischen 0 und 1. Die Distanzwerte bewegen sich damit in einem Intervall von [0.0, 1.0]: Der Wert liegt bei 0.0, wenn es sich um das gleiche Produkt handelt, bei 1.0, wenn die Items komplett verschieden sind (Tabelle 3.3).

articleId	0	1	2	5	6
0	0.000000	0.955784	0.916361	0.972045	0.925878
1	0.955784	0.000000	0.983046	0.968834	0.938022
2	0.916361	0.983046	0.000000	0.981242	1.000000
5	0.972045	0.968834	0.981242	0.000000	0.977142
6	0.925878	0.938022	1.000000	0.977142	0.000000

Tabelle 3.3: Ausschnitt der Item-Item-Matrix (Distanzen) mit Kosinus

Für die Generierung von N Empfehlungen zu einem bestimmten Item, wird die entsprechende Spalte aus der Item-Item-Matrix herausgesucht und nach aufsteigender Distanz sortiert, wobei wieder die erste Zeile nicht mit berücksichtigt

wird.
Die Erstellung von N Empfehlungen für einen gegebenen Nutzer/eine gegebene Nutzerin funktioniert wie oben beschrieben: Für jeden Artikel, mit dem der User interagiert hat, werden Empfehlungen generiert, die dann zusammen in einer Liste gespeichert und wieder nach Distanzwerten sortiert werden.

- Pearson-Korrelation:
  Fast genau so wie bei den beiden anderen Metriken wird für den Ansatz mit Korrelationskoeffizienten nach Pearson ebenfalls die Matrix berechnet. Hier wird allerdings eine User-Item-Rating-Matrix R' verwendet, also ist mit r_{ij} das Rating des i-ten Users für das j-te Item gemeint.

```
itemItemMatrix = userItemRatingMatrix.corr(method='pearson')
```

Ausschnitt 3.4: Berechnung der Ähnlichkeiten auf Basis von Pearson

Mit der *dataframe.corr()*-Funktion aus der Pandas-Bibliothek wird die paarweise Korrelation aller Spalten des Dataframes berechnet. Mit dem Parameter *pearson* wird der Standard-Pearsonkorrelationskoeffizient nach Formel 2.5 als Methode benutzt.
Dies gibt direkt ein Dataframe bzw. eine Item-Item-Matrix mit den *articleIds* als Indizes zurück. Hier handelt es sich bei den Werten um Ähnlichkeiten, bei einem Wert von 1.0 handelt es sich also um das gleiche Produkt, bei -1.0 um komplett unterschiedliche Produkte (Tabelle 3.4).

articleId	0	1	2	5	6
0	1.000000	0.042512	0.082326	0.026383	0.073051
1	0.042512	1.000000	0.016292	0.030456	0.061462
2	0.082326	0.016292	1.000000	0.018161	-0.000459
5	0.026383	0.030456	0.018161	1.000000	0.022370
6	0.073051	0.061462	-0.000459	0.022370	1.000000

Tabelle 3.4: Ausschnitt der Item-Item-Matrix (Ähnlichkeitsmatrix) mit Pearson

Für einen bestimmten Artikel werden, anders als bei den ersten beiden Fällen, die Werte der entsprechenden Spalte also nach absteigenden Ähnlichkeitswerten sortiert. Ansonsten funktioniert der restliche Vorgang wie vorher beschrieben.

3.2.1.3 Herausforderungen

Beim IB-CF (und teilweise auch CF allgemein) sind einige Herausforderungen bzw. Biases zu beachten:

- Popularity Bias:
  Da beliebte Artikel meist auch die am meisten bewertetsten sind, neigen CF-Systeme dazu, diese auch eher zu empfehlen. Produkte, mit denen noch kaum interagiert wurde, werden möglicherweise nur selten empfohlen, da deren vorhergesagtes Rating eventuell nicht zuverlässig ist [BCT11].

- First-Rater-Problem [BCT11]:
  Weil das System Items empfiehlt, die am meisten mit dem aktuellen User korrelieren, können neue Produkte nicht empfohlen werden, da noch kein User mit ihnen interagiert hat und somit kein Modell für diese definiert werden kann. Daraus resultiert dann auch folgender Punkt:

- Cold Start:
  Das Kaltstartproblem kann in zwei unterschiedliche Varianten unterschieden werden [GASP15; DEKA11]: Beim Item-Kaltstartproblem wird ein neues Produkt dem Katalog hinzugefügt, zu dem es anfangs keine oder nur geringe Bewegungsdaten gibt. Gerade CF ist auf diese Daten jedoch angewiesen, um passende Empfehlungen zum genannten Produkt generieren bzw. um dieses weiterempfehlen zu können. Beim User-Kaltstartproblem gibt es einen neuen User, der noch nicht oder kaum mit den Items interagiert hat. Auch hier ist es somit nicht möglich, passende auf den User zugeschnittene Empfehlungen zur Verfügung zu stellen.

- Sparsity:
  Mit geringerer Dichte der Daten (also spärliche, dünne Besetzung bzw. Datenmangel) fällt auch die Qualität der Empfehlungen kleiner aus, da für ein brauchbares System nicht genügend Daten vorliegen. Gerade bei großen E-Commerce-Seiten mit vergleichsweise wenig Einkäufen stellt dies ein Problem dar [SSRM14].

- Skalierbarkeit:
  kNN-Algorithmen wachsen mit der Anzahl an Usern und Items, die verfügbar sind, dies führt damit zu einem Problem der Skalierbarkeit mit steigender User- bzw. Itemmenge [SSRM14].

- Gray Sheep:
  Für einen User, dessen Interessen und Ratings inkonsistent sind, ist es schwierig, passende Empfehlungen zu generieren. Reines CF funktioniert für solche NutzerInnen nicht [LOONY16].

- Synonymität:
  Manche Produkte sind praktisch identisch, z.B. ein Kleidungsstück in verschiedenen Größen oder das gleiche Buch einmal als E-Book und als „richtiges“ Buch. Auch, wenn dies im Grunde die gleichen Produkte sind, haben die meisten Shops für die Artikel unterschiedliche Ids, so auch in diesem Fall. Da CF

keine Produktinformationen verwendet, könnte diese Information eventuell untergehen [LOONY16].

- Shilling attacks [LOONY16]:
  Ein Beispiel hier wäre ein Autor, der für seine eigenen Werke lauter positive Bewertungen hinterlässt und andere negativ bewertet. Dies ist hier allerdings nicht ganz so relevant, da keine expliziten Bewertungsdaten sondern implizite verwendet werden, die auf dem Userverhalten basieren.

3.2.2 Latent factor based Matrixfaktorisierung mit ALS

Als nächstes wird ein Model-Based CF Verfahren mit Latent Factor Modellen angewendet: Die bereits beschriebene Matrixfaktorisierung (MF), hier mit ALS als Optimierungsverfahren. In vielen Forschungsarbeiten und auch in der Praxis hat sich die Methode der MF als effektiv bewiesen [QFZM14; GOHU15; HZKC16].

3.2.2.1 Implementierung

Zur Implementierung wird Lis [LI19] beschriebener Prozess zur Empfehlungsgenerierung verwendet.
Wie vorher beim IB-CF handelt es sich beim Trainingsdatensatz um *df1* bzw. *df2*. Es wird ebenfalls ein zweites Dataframe *df_products* mit den entsprechenden Eigenschaften erstellt, um das Heraussuchen der Items zu vereinfachen.
Es werden zwei dünnbesetzte Matrizen erstellt: Eine für die Anpassung des Modells an die Daten (*ItemUserMatrix*) und eine für die Empfehlungen (*UserItemMatrix*). Diese werden mit Hilfe der Implicit-Bibliothek einem ALS Recommender Modell angepasst (siehe Ausschnitt 3.5). Für die Menge der latenten Faktoren wird hier 64 gewählt, als Regularisierungsfaktor 0.1 und für die Anzahl der Durchläufe 50.

```
import implicit
itemUserMatrix = sparse.csr_matrix((df_train['rating']
    .astype(float), (df_train['articleId'], df_train['userId'])))
userItemMatrix = sparse.csr_matrix((df_train['rating']
    .astype(float), (df_train['userId'], df_train['articleId'])))

model = implicit.als.AlternatingLeastSquares(
    factors=64, regularization=0.1, iterations=50)
alpha = 15
data = (itemUserMatrix * alpha).astype('double')
model.fit(data)
```

Ausschnitt 3.5: ALS Recommender Modell

Für das Finden von passenden bzw. ähnlichen Artikeln zu einem bestimmten Produkt A wird wie folgt vorgegangen: Es werden zuerst die Itemvektoren des Modells herausgesucht, also die Arrays von latenten Faktoren eines jeden Items im Trainingsdatensatz. Von diesen werden jeweils die Normen, also die Beträge der Vektoren, berechnet:

$$|\vec{v_i}| = \sqrt{{x_1}^2 + {x_2}^2 + ... + {x_{64}}^2} \tag{3.2}$$

Aus der Liste der Itemvektoren wird das entsprechende Array des gewünschten Produktes A herausgesucht, mit dem dann das Skalarprodukt zu jedem anderen Array berechnet wird, geteilt durch die entsprechende Vektornorm. Dadurch erhält man eine Liste mit Scores für jedes Item in Beziehung zu A. Im nächsten Schritt werden diese passend sortiert und dann die ersten N Indexe (wieder ohne die erste Zeile) genommen, also die *articleIds*. In einer Liste L werden dann die endgültigen Ähnlichkeitswerte wie folgt gespeichert: Der Index zusammen mit dem entsprechenden Score, geteilt durch die Vektornorm des Produktes A. Für die Produktinformationen werden wieder die entsprechenden Zeilen aus *df_products* herausgefiltert.
Für die Generierung von Empfehlungen zu einem bestimmten User U werden mit den User- und Itemvektoren des Modells die *UserFactorMatrix* und *ItemFactorMatrix* erstellt (siehe Ausschnitt 3.6).

```
import scipy.sparse as sparse
userFactorMatrix = sparse.csr_matrix(model.user_factors)
itemFactorMatrix = sparse.csr_matrix(model.item_factors)
```

Ausschnitt 3.6: ALS Matrizen

Aus der *UserItemMatrix* werden die Ratingdaten von U geholt, dann wird zu diesen jeweils 1 addiert, damit Items, mit denen der User noch nicht interagiert hat, den Wert 1 erhalten. Produkte, die dann einen Wert höher als 1 haben, also mit denen U bereits interagiert hat, werden auf 0 gesetzt. Die daraus resultierenden Werte werden in einem Array *interactions* gespeichert. Im nächsten Schritt werden die Skalarprodukte zwischen allen Itemvektoren und dem entsprechenden Uservektor zu User U berechnet. Diese Werte werden nun zu einem Wertebereich zwischen 0 und 1 skaliert und dann mit *interactions* multipliziert, sodass Items, mit denen bereits interagiert wurde, mit 0 multipliziert werden. Im letzten Schritt werden die Werte in absteigender Reihenfolge sortiert und dann die ersten N Einträge für die Top N Empfehlungen genommen (ohne die erste Zeile). Für die entsprechenden Produktinformationen wird wieder *df_products* verwendet.

3.2.2.2 Herausforderung bei der Matrixfaktorisierung

Berechnet man den Prozentwert für die Spärlichkeit der Daten, also wie dünnbesetzt die *UserItemMatrix* ist, sieht man, dass dieser sowohl bei *df1* als auch bei *df2* bei

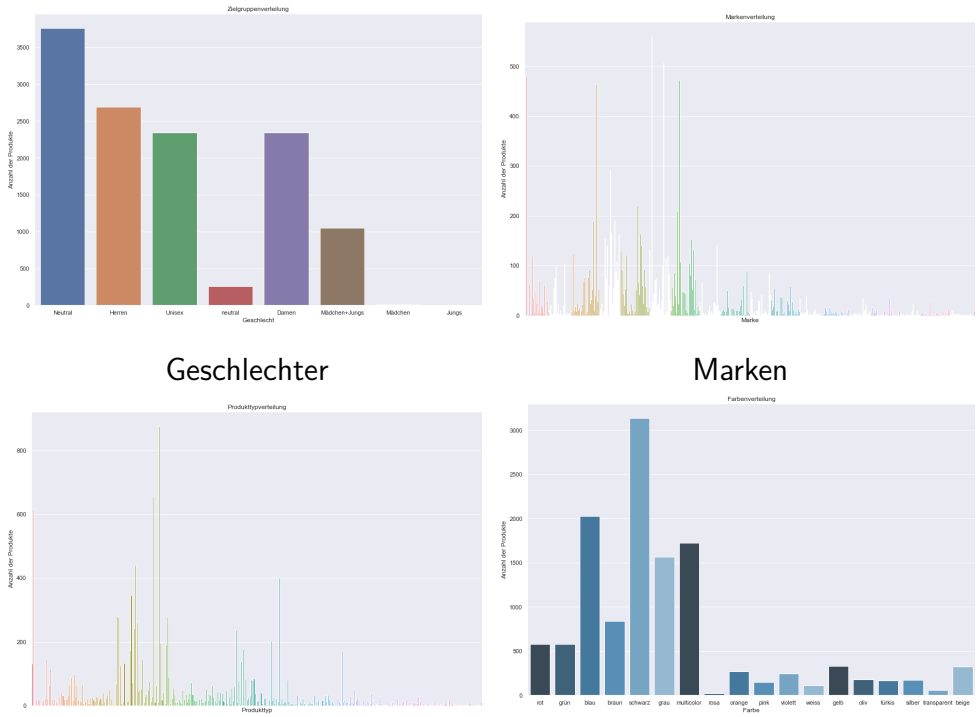
ca. 99.99% liegt. CF-Algorithmen beruhen, wie bereits erwähnt, auf den vorliegenden Daten. Wenn diese nicht ausreichen, leidet auch die Qualität der Empfehlungen darunter. Der iShop ist noch nicht lange online, dies ist später bei der Evaluation zu berücksichtigen. Mit der Zeit wird die Anzahl der verfügbaren Daten steigen und damit dann auch die Effizienz des Recommender Systems. Das Spärlichkeitsproblem spielt beim IB-CF jedoch eine größere Rolle und fällt stärker ins Gewicht als bei der MF mit ALS.

3.2.3 Content-Based Filtering

Als nächstes Verfahren wird das Content-Based Filtering (CBF) angewendet. Der große Vorteil hierbei ist die Vermeidung des erwähnten Item-Kaltstartproblems.

3.2.3.1 Datenvorverarbeitung

Um Empfehlungen auf der Grundlage von Produktattributen generieren zu können, müssen passende Eigenschaften bzw. Features ausgewählt werden. Die vorliegenden Produkteigenschaften eines Shopartikels sind in Tabelle A.4 aufgeführt.
Im ersten Schritt werden alle Felder, in denen Werte fehlen, durch die Zeichenkette *‚NaN‘* ersetzt.



Geschlechter | Marken

Produkttypen | Farben

Abbildung 3.7: Produktattributverteilungen

1. Wie bereits erläutert ist es wichtig und sinnvoll, nur eine begrenzte Anzahl an Features zu benutzen [NGZE18]. Für den Beginn werden hier zunächst nur vier verwendet.
Untersucht man *df_productfeed*, liegt nahe, dass Attribute wie *Geschlecht* - also die Zielgruppe - und *Produkttyp* wichtig sein könnten. Genau so auch die Eigenschaften *Marke* und *Farbe*. *Geschlecht* hat acht verschiedene Werte von denen zwei zusammengefasst werden können: ‚Neutral' und ‚neutral', sodass die Eigenschaft noch sieben eindeutige Werte besitzt. Es gibt des Weiteren 423 unterschiedliche Marken, 323 verschiedene Produkttypen und 18 Farbwerte. Eine Visualisierung der Attributverteilungen ist in Abbildung 3.7 zu sehen.
Bei allen vier Eigenschaften handelt es sich, wie eigentlich auch bei fast allen anderen vorliegenden Merkmalen, um kategorische Daten. Die Werte müssen also so umgewandelt werden, dass mit ihnen gearbeitet werden kann. Sie sind außerdem nominal, da keine der Attribute für ihre Werte eine bestimmte Reihenfolge oder Ordnung besitzt. Damit kommen als Encodingvarianten One-Hot Encoding, Hashing, LeaveOneOut oder Target bzw. Mean Encoding in Frage [HALE18].
Für die Produktattribute *Farbe* und *Geschlecht* wird das One-Hot Encoding Verfahren angewendet. *Marke* und *Produkttyp* haben eine höhere Kardinalität, also viele verschiedene eindeutige Werte. Wie bereits in Kapitel 2.6.1 erläutert, hängt dies mit dem „Fluch der Dimensionalität" zusammen, was zu Overfitting oder schlechten Ergebnissen führen kann. Die Spannweite ist bei beiden aber noch akzeptabel, daher wird sowohl für beide Attribute dasselbe Encodingverfahren verwendet. Die Bearbeitungsschritte für Outlier (dt. Ausreißer), zum Binning und Scaling fallen hier weg, da keine numerischen Daten vorliegen.

Mit den, durch das One-Hot Encoding, neu gewonnenen Spalten wird eine *ItemFeatureMatrix* erstellt, die für jede Zeile - also jedes Item - eine Spalte für jeden eindeutigen Wert zum Vergleich zu allen anderen Werten besitzt (siehe Tabelle 3.5).

	Marke_1	Marke_2	...	Produkttyp_1	...	Farbe_1	...	Farbe_18
0	1	0	...	1	...	0	...	0
1	0	1	...	0	...	1	...	0
2	0	0	...	0	...	0	...	1
3	0	1	...	0	...	0	...	0

Tabelle 3.5: Beispielhafter Auszug aus der ItemFeatureMatrix

2. Nach der Implementierung mit den vier Attributen werden im Folgenden weitere Features mit einbezogen. Betrachtet man z.B. das Merkmal *Nachhaltigkeit*, sieht man, dass es elf verschiedene Werte gibt (unter anderem z.B. „Bio", „Biologisch ab-

baubar“, „Recycled“, „Vegan“). Die einzelnen Werte sind hier allerdings nicht von großer Wichtigkeit, für die Eigenschaft ist eher bedeutend, ob das Feld gefüllt ist. Es wird also überall der Wert 1 dort gesetzt, wo das Feld nicht gleich *‚NaN‘* ist und sonst 0.
Weitere Merkmale, die nun mit berücksichtigt werden, sind *Länge*, *Saison*, *Kragenform*, *Aktivität*, *Schichtenprinzip* und *MigrationModul* (siehe Tabelle 3.6). Sie können aufgrund ihrer geringen Kardinalität ebenfalls mit dem One-Hot Encoding Verfahren kodiert werden. Daraus wird dann eine neue *ItemFeatureMatrix* erstellt.

Attribut	Unique_Count	Beispiele
Nachhaltigkeit	2	1,0
Saison	18	Frühling/Sommer 2019, Herbst/Winter 2014, ...
Länge	8	kurz, lang, ärmellos, ...
Kragenform	4	Rundhals, Stehkragen, ...
Aktivität	56	Backpacking, Camping, Klettern, ...
Schichtenprinzip	4	1. Schicht, 2. Schicht, ...
MigrationModul	7	Bike, Snow Sports, Travel/Reisen, ...

Tabelle 3.6: Weitere Features für CBF

3. Die perfekte Feature-Kombination - oder zumindest eine Lösung, die nah dran ist - zu finden und Produkteigenschaften miteinander zu vergleichen, wenn nur Attribute vorliegen, die zudem fast ausschließlich als kategorische Daten vorliegen, gestaltet sich als recht schwierig und komplex. Ein weiterer Ansatz für das Empfehlen auf Basis von Produktdaten ist die in Kapitel 2.5.4 beschriebene Methode, die NLP und TF-IDF für textbasierte Entitäten nutzt. In Produktattributen wie beispielsweise der *Beschreibung* kann sehr viel nützliche Information stecken, die so mit dem bisherigen Verfahren nicht zugänglich ist. In diesem Schritt wird daher der Ansatz mit NLP nach dem von Grimaldi beschriebenen Vorgang [GRIM18] näher untersucht.
Abgesehen von der Produktbeschreibung stehen in den Eigenschaften *HighlightText*, *Merkmale* und *Titel* weitere Angaben, die von Bedeutung sein könnten. Beim *HighlightText* und der *Beschreibung* handelt es sich um Textblöcke, aus denen zunächst die Schlüsselwörter heraus gefiltert werden müssen. Dies kann mit der Multi-Rake-Bibliothek bewerkstelligt werden, die RAKE als Methode zur Keyword-Extrahierung verwendet: Es wird durch die Zeilen von *df_productfeed* iteriert und für jedes Item die *apply*-Funktion auf die *Beschreibung* und den *HighlightText* angewendet. Hierbei werden Stoppwörter der deutschen Sprache bereits mit berücksichtigt, die gewonnenen Schlüsselwörter werden als Array in einer neuen Spalte *key_words* des Dataframes *df_productfeed* gespeichert (siehe 3.7).

```
df_productfeed['key_words'] = ""
for index, row in df_productfeed.iterrows():
    highlight = row['HighlightText']
    r = Rake(language_code='de')
    keywords = r.apply(highlight)
    keywords_words = []
    for keyword in keywords:
        keywords_words.append(keyword[0])
    df_productfeed.at[index,'key_words'] = keywords_words
```

Ausschnitt 3.7: Schlüsselwörter-Generierung aus dem HighlightText

Der Inhalt des Produktattributes *Merkmale* liegt bereits in Form von eindeutigen Begriffen vor, getrennt sind sie mit dem Zeichen ‚|'. Hier wird also für jedes Item der gegebene String bei ‚|' getrennt und die daraus gewonnenen Begriffe ebenfalls als Array gespeichert. Damit stehen nun Schlüsselwörter aus der Produktbeschreibung, dem HighlightText und den Merkmalen zur Verfügung, die alle zusammen in einer Spalte *bag_of_words* zusammengefasst werden. Für das nur auf NLP und TF-IDF basierende Verfahren werden die Inhalte von *Titel* sowie der anderen Felder *Geschlecht*, *Saison* usw. ebenfalls in das Feld von *bag_of_words* aufgenommen, die restlichen Spalten werden dann bei der weiteren Bearbeitung nicht mehr benötigt und können weggelassen werden, sodass das Dataframe so aussieht wie in Tabelle 3.7.

	bag_of_words
12209	Nammatj 2 Neutral Durchläufer kompaktes expedi...
11639	Puma Gore WS Unisex Durchläufer daunen winters...
4787	Classic Messenger Dip Neutral Herbst/Winter 20...
4905	Mini Backpack Unisex Durchläufer handgepäck ta...
850	Belorado II Low Bunion Lady GTX Damen Durchläu...

Tabelle 3.7: Beispielhafter Auszug der „Bag of Words"

Für die eigentliche Konvertierung der „Bag of Words" in eine Matrix mit TF-IDF-Features wird schließlich der TfidfVectorizer der Scikit-Bibliothek verwendet. Hier wird erst ein CountVectorizer angewendet, bei dem der Text tokenisiert, also in geeignete Einheiten zerlegt wird. Darauf wird dann der TfidfTransformer verwendet, der TF-IDF als Gewichtungsmaßnahme nach den in 3.3 genannten Formeln nutzt [SCIKIT].

$$\begin{aligned} \text{TF-IDF}(t_k, d_j) &= \text{TF}(t_k, d_j) \cdot \text{IDF}(t_j) \\ \text{IDF}(t_j) &= \log \frac{n}{\text{df}(t)} + 1 \\ \nu_{norm} = \frac{\nu}{\|\nu\|_2} &= \frac{\nu}{\sqrt{\nu_1^2 + \nu_2^2 + ... + \nu_n^2}} \end{aligned} \tag{3.3}$$

n ist dabei die Gesamtanzahl an Dokumenten bzw. hier den einzelnen *bag_of_words* der Items, $df(t)$ ist die Anzahl der Dokumente, die den Term t_j enthalten. Die daraus entstehenden TF-IDF Vektoren werden dann mit der euklidischen Norm normalisiert. Daraus lässt sich dann die *ItemWordMatrix* gewinnen.

4. Als vierte und letzte Variante werden der textbasierte Ansatz sowie das Verfahren der Kodierung der kategorischen nominalen Daten zusammen angewendet.
Hier werden wieder Schlüsselwörter aus der *Beschreibung* und dem *HighlightText* erzeugt, die zusammen mit den Begriffen aus den Feldern *Merkmale* und *Titel* als „Bag of Words“ genutzt werden. Die daraus erstellte *ItemWordMatrix* und die aus dem zweiten Schritt resultierende *ItemFeatureMatrix* werden in einer Matrix zusammengefasst, mit der dann weitergearbeitet werden kann (siehe Tabelle 3.8).

	aufrollbar	...	thermoisoliert	...	Marke_1	Marke_2	...
3	0	...	0	...	1	0	...
4	0	...	1	...	1	0	...
5	0	...	0	...	0	1	...
6	1	...	0	...	0	0	...

Tabelle 3.8: Beispielhafter Auszug der ItemMatrix für Variante 4

3.2.3.2 Ähnlichkeitsbestimmung und Empfehlungen

Für die Bestimmung der Ähnlichkeit zweier Items kann wieder eine Distanzmetrik angewendet werden, hier die Kosinus-Ähnlichkeit. Aus der *ItemFeatureMatrix* bzw. der *ItemWordMatrix* werden die Kosinus-Ähnlichkeiten für jedes Item in Korrelation mit den anderen Items berechnet (siehe Auschnitt 3.8).

```
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarities = cosine_similarity(itemFeatureMatrix)
```

Ausschnitt 3.8: Berechnung der Kosinus-Ähnlichkeit bei CBF

u_i und v_i (siehe Formel 2.4) sind dabei die verschiedenen Features zweier Items u und v bei der *ItemFeatureMatrix* bzw. die Häufigkeit der Schlüsselwörter bei der *ItemWordMatrix*.
Aus diesen Ähnlichkeitswerten kann wieder eine Ähnlichkeitsmatrix erstellt werden, mit der dann wieder genau so verfahren wird wie beim IB-CF in 3.2.1.2, um passende Top N Empfehlungen zu einem bestimmten Item oder User zu generieren. Der Unterschied hier ist nur, dass keine Entfernungen zurückgegeben werden, sondern die Kosinusähnlichkeit selbst. Es wird also nach absteigenden Ähnlichkeitswerten sortiert.

3.2.3.3 Vor- und Nachteile

Der Vorteil beim CBF ist, wie bereits erwähnt, dass dabei vor allem das Item-Kaltstartproblem vermieden wird, da das Verfahren nicht auf Userdaten basiert. CBF-Methoden haben das First-Rater Problem nicht.
Damit ist CBF auch User-unabhängig: Während beim CF Ratingdaten anderer User gebraucht werden, um überhaupt einen k-Nearest Neighour für einen aktuellen User U zu finden, werden beim CBF nur Daten vom User U benötigt, um für diesen passende Empfehlungen zu generieren.
Eine weitere Stärke ist die Transparenz: Man kann schnell erkennen, warum etwas zu einem bestimmten Produkt empfohlen wird, indem man die Produktattribute und -eigenschaften der jeweiligen Artikel gegenüberstellt. Im Vergleich dazu ist CF eher wie eine Blackbox zu sehen [LOPS11].

CBF hat jedoch auch Nachteile. Eine Problematik, die es beim CF beispielsweise nicht gibt, ist die sogenannte „Filter Bubble“ und „Overspecialization“ [PAR11; NGZE18]: Da beim CBF Items empfohlen werden, die ähnlich zu dem aktuellen Produkt sind, ist das System restriktiv. Als Beispiel: Klickt ein User sich durch viele Hosen, werden diesem dementsprechend auch mehr Hosen empfohlen bzw. angezeigt. Das System ist dabei nicht in der Lage zu erkennen, dass die Interessen des Users auch über Hosen hinausgehen könnten.
Ein weiterer Nachteil ist die begrenzte Content-Analyse [LOPS11]: Die Anzahl und Art an Features, die für das CBF genutzt werden kann, ist limitiert. Außerdem ist Wissen über die Domain, also die Produkteigenschaften, notwendig. Wenn hier nicht genug Informationen zur Verfügung stehen können auch keine guten Empfehlungen generiert werden.
Auch das New-User Problem ist mit CBF nicht gelöst, ohne ausreichende Userdaten können auch hier keine zuverlässigen Empfehlungen zur Verfügung gestellt werden.

3.2.4 Assoziationsregeln

Als letztes Verfahren zur Empfehlungsgenerierung werden Assoziationsregeln angewendet.
Da bei den zur Verfügung stehenden Daten keine Order-Events existieren, werden als

Datengrundlage die addtobasket-Daten verwendet. Jede dieser Interaktionen besitzt jeweils eine *sessionId*. Events mit der gleichen *sessionId* werden als eine Order bzw. Bestellung interpretiert, für andere iShops können analog die Order-Events verwendet werden.

Von der addtobasket-Datei werden nur die Spalten *sessionId*, *userId* und *articleId* benötigt. Zusammen mit *articleId* und *title* aus *df_productfeed* werden die beiden Datensätze zu einem neuen Dataframe *df_association* zusammengefügt, dabei werden sie über die *articleId* als gemeinsames Merkmal miteinander verschmolzen. Artikel, die nicht mehr im aktuellen Objektspeicher sind - also dessen *articleIds* nicht in *df_productfeed* zu finden sind - werden damit entfernt. Der neue Datensatz, nachfolgend *df_association* genannt, hat 30113 Einträge, von denen 17357 SessionIds einzigartig sind. Damit liegen also 17357 verschiedene Bestellungen vor. Außerdem gibt es im Datensatz 6077 unterschiedliche Artikel und 15958 verschiedene User. *articleId*, *sessionId* und *userId* werden wieder in numerische Daten umgewandelt, um das Arbeiten mit diesen zu erleichtern.
df_association wird eine Spalte *rating* hinzugefügt, jede Zeile bekommt dabei den Wert 1.0. Damit wird eine Session-Item-Rating-Matrix M erstellt, wobei die *sessionIds* als Indizes und *articleIds* als Spaltennamen genutzt werden. Die *ratings* werden als Werte zum Füllen der Matrix verwendet. Felder, die dabei leer bleiben, werden mit 0.0 aufgefüllt.

Für die Erstellung der Assoziationsregeln wird der Apriori Algorithmus angewendet. Mit Hilfe der Matrix wird nun der in Kapitel 2.5.3 erwähnte Support für ein Itemset berechnet: Also der Anteil an Usern, die dieses Set von Items ihrem Warenkorb bei einer Bestellung hinzugefügt haben. Aus M wird eine Liste *sessionList* mit allen eindeutigen *sessionIds* erstellt, N_{all} ist dabei die Länge dieser Liste.
Für ein gegebenes Itemset S mit beispielsweise drei Items werden zunächst für das erste darin enthaltene Item I_1 alle Sessions bzw. Bestellungen - also Reihen - aus der *SessionItemRatingMatrix* herausgefiltert, die für I_1 einen Ratingwert größer als 0 haben. Dieses neu erstellte Matrix wird nun als Grundlage für I_2 genutzt: Wieder werden die Reihen entfernt, für die der Wert bei I_2 nicht größer als 0 ist. Schließlich wird auf diese neue Matrix dasselbe noch einmal für I_3 durchgeführt. Übrig bleibt damit eine RatingMatrix M_S, die nur Bestellungen enthält, in denen das gegebene Itemset S enthalten ist. Der Support für S lässt sich damit aus der Division von N_S, der Anzahl der Sessions in M_S, und N_{all} berechnen (siehe 3.4).

$$\text{Support}_{\text{S}} = \frac{N_{\text{S}}}{N_{\text{all}}} \tag{3.4}$$

Im ersten Durchlauf wird der Support für jedes Produkt berechnet, ein Itemset besteht damit aus einem einzelnen Item. Der Mindestsupport wird hier auf 0.5% festgelegt. Der Wert wird normalerweise höher gesetzt, da der vorliegende Datensatz jedoch nicht so groß ist und der Prozess zur Erstellung von Assoziationsregeln mehr im Fokus

liegt als die Ergebnisse selbst, wird der Mindestsupport so gewählt, um sichtbare Ergebnisse zu erzielen. Alle Artikel, die den Schwellenwert überschreiten, werden dann in einer Liste *allItems* gespeichert. Für *df_association* sind das 64 Items.
Für diese Arbeit werden nun Assoziationsregeln für Itemsets bestehend aus zwei Items erstellt, generell kann man den Vorgang für n Items durchführen. Aus allen Artikeln in *allItems* werden zunächst alle möglichen Permutation mit der Länge zwei, also jede mögliche Kombination eines Produktes mit einem anderen Produkt, mit Hilfe der Bibliothek itertools generiert. Damit erhält man 4032 Paare: Dies sind also alle Regeln, die infrage kommen. Für solch eine Regel r_i wird nun nach Formel 2.11 ihre Confidence berechnet.
Für das Paar (4937, 1699) berechnet sich dies beispielsweise folgendermaßen (3.5):

$$\mathrm{Confidence}_{\mathrm{r_i}} = \frac{\mathrm{Support}(4937, 1699)}{\mathrm{Support}(4937)} \tag{3.5}$$

Für die Confidence jeder Regel r_i wird nun ebenfalls geprüft, ob diese einen gesetzten Mindestwert erreicht. Für sichtbare Ergebnisse wird dieser hier ebenfalls niedriger gewählt, nämlich 1.0%. Außerdem muss der berechnete Support des Paares ebenfalls einen Mindestwert erreichen, dieser wird auf 0.1% gesetzt.
Mit dem beschriebenen Vorgang erhält man für *df_association* schließlich eine Liste mit 14 Assoziationsregeln, die die Restriktionen bezüglich Support und Confidence erfüllen.

3.3 Zwischenevaluation

3.3.1 Anwendung des LOOCV Verfahrens

3.3.1.1 Zu beachtende Biases

Für die Evaluation wird das beschriebene LOOCV Verfahren genutzt. Dabei sind allerdings einige Dinge zu berücksichtigen, die von Nachteil sind:

- Da es sich hier um ein Offline Evaluationsverfahren handelt, ist es nicht möglich, Rückschlüsse über den Einfluss des Recommender Systems auf das Userverhalten zu ziehen.
  Das bedeutet unter anderem, dass für Items, mit denen ein User nicht interagiert hat, angenommen werden muss, dass diese für ihn auch dann nicht interessant gewesen wären, wenn sie ihm empfohlen worden wären. Auch, wenn dies andere Gründe haben könnte, z.B. weil der User nichts von der Existenz des Artikels wusste. Dies fällt für das Evalationsergebnis dementsprechend negativ ins Gewicht.

- Die Technik ist zwar (bzgl. der Datenverteilung und Userdaten) unvoreingenommen, hat aber eine große Schwankungsbreite, da es nur eine Interaktion

(pro User) für die Vorhersage gibt. Beispiel: Ein User hat sich viele Kleidungsstücke und eine Trinkflasche angesehen bzw. in seinen Warenkorb gelegt. Wird nun genau die Trinkflasche in den Testdatensatz gelegt, wird das Recommender System ihm diese aller Wahrscheinlichkeit nach nicht empfehlen können.

- Es ist schwierig, genau ein bestimmtes Item bei N Empfehlungen zu treffen, besonders wenn die Menge an vorhandenen Items sehr hoch ist. Die Trefferquote bei LOOCV ist daher relativ gering.
- LOOCV hat einen hohen Rechen- und Zeitaufwand, vor allem, wenn die Methode mehrmals durchgeführt werden soll.
- LOOCV kann auf die CF-, CBF- und hybriden Modelle angewendet werden. Für die Evaluation der Assoziationsregeln ist das Verfahren jedoch nicht geeignet.
- Recommender Systeme im Live-Betrieb sind nicht so statisch wie bei LOOCV angenommen [JEUNEN18].
- Timestamps werden hier nicht mit einbezogen, die chronologische Reihenfolge der Events wird also nicht berücksichtigt. Items, mit denen ein User anfangs interagiert hat, sind für ihn jetzt möglicherweise gar nicht mehr relevant.

LOOCV kann nicht als alleiniges ausschlaggebendes Verfahren genutzt werden, es sind viele andere Faktoren zu berücksichtigen und neben der Genauigkeit spielen auch andere Kriterien eine wichtige Rolle. LOOCV kann jedoch als ein erster Anhaltspunkt für die Qualität der Recommender Modelle gesehen werden und ist für die vorliegenden Daten eine gute Möglichkeit für eine erste Offline Evaluation, um erste Ansätze herauszufiltern, die ungeeignet scheinen.
Ob und welche der Verfahren tatsächlich gewinnbringender und effektiver sind, kann erst durch Online Testing festgestellt werden.

3.3.1.2 Ergebnisse

Das LOOCV Verfahren wird für eine verschiedene Anzahl von Top N Empfehlungen benutzt, für die Evaluation werden unter anderem Trainingsdatensätze mit und ohne den addtobasket-Daten entsprechend mit ihren dazugehörigen Testdatensätzen verwendet.
Das LOOCV wird auf Grundlage der in Kapitel 3.1.1 genannten verfügbaren Daten angewendet, die Aufteilung in Trainings- und Testdatensätze wird dabei variiert. Für die Erstellung dieser wird dabei jedes Mal so verfahren wie für *df1* und *df2*. Innerhalb dieser Datensätze wird das Evaluationsverfahren mehrfach für eine Gruppe von zufällig gewählten Usern durchgeführt, aus deren Ergebnissen dann ein Durchschnittswert gezogen wird. Aus diesen Ergebnissen der verschiedenen Trainings- und Testdatensätze wird ebenfalls ein gemeinsamer Durchschnittswert berechnet.

Zuerst werden die Evaluationsergebnisse beim IB-CF betrachtet (siehe Tabellen 3.9 und 3.10). Visualisiert sind sie außerdem in Abbildung 3.10.

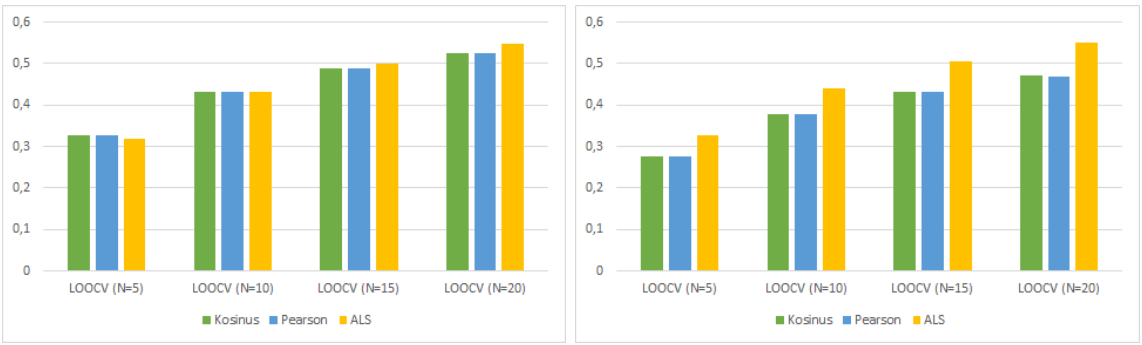
	IB-CF			LV/MF
	Euklid	Kosinus	Pearson	ALS
LOOCV (N=5)	0.004	0.327	0.327	0.319
LOOCV (N=10)	0.009	0.432	0.431	0.432
LOOCV (N=15)	-	0.490	0.489	0.500
LOOCV (N=20)	-	0.526	0.525	0.548

Tabelle 3.9: Evaluationsergebnisse CF im Vergleich

	IB-CF		LV/MF
	Kosinus	Pearson	ALS
LOOCV (N=5)	0.277	0.276	0.328
LOOCV (N=10)	0.378	0.377	0.441
LOOCV (N=15)	0.433	0.432	0.506
LOOCV (N=20)	0.471	0.470	0.552

Tabelle 3.10: Evaluationsergebnisse CF mit Basket-Daten im Vergleich

Zu erkennen ist, dass das IB-CF mit dem euklidischen Abstand als Distanzmetrik mit Abstand am schlechtesten abschneidet. Nach den ersten paar Durchläufen wurde dieser Ansatz daher nicht weiter verfolgt, um weiteren Rechenaufwand zu vermeiden.



LOOCV mit Datensätzen wie df1 LOOCV mit Datensätzen wie df2

Abbildung 3.10: Evaluationsergebnisse CF im Vergleich

Zwischen den Strategien mit jeweils der Kosinus-Ähnlichkeit und dem Korrelationskoeffizienten nach Pearson als Metrik gibt es keine großen Unterschiede, die Variante

mit der Kosinus-Ähnlichkeit schneidet zwar minimal besser ab, es liegt allerdings kein signifikanter Unterschied vor.
Vergleicht man diese Ergebnisse nun mit denen des latenten Variablenmodells, schneidet IB-CF für ein kleines N besser ab, mit steigendem N wird jedoch auch die Genauigkeit bei MF besser und ist dem IB-CF Modell etwas überlegen. Betrachtet man die LOOCV-Ergebnisse des CF, bei denen die addtobasket-Daten mit einbezogen sind, ist der Unterschied zwischen IB-CF und MF größer. Hier ist bei dem latenten Variablenmodell außerdem eine kleine Verbesserung zu den Datensätzen ohne die Basketdaten zu sehen, während die Modelle basierend auf IB-CF schlechter abschneiden als vorher. Nennenswert ist außerdem, dass MF auch bei kleinem N bessere Ergebnisse als IB-CF erzielt.

Als nächstes werden die LOOCV Ergebnisse des CBF evaluiert. Im Vergleich stehen hier vier verschiedene Varianten: CBF mit vier Features (Farbe, Geschlecht, Produkttyp und Marke), mit mehreren Produktattributen, mit dem textbasierten NLP Ansatz und als viertes eine Kombination der genannten Varianten. Aufgeführt sind die Ergebnisse in den Tabellen 3.11 und 3.12, die entsprechenden Diagramme sind in Abbildung 3.13 zu sehen.

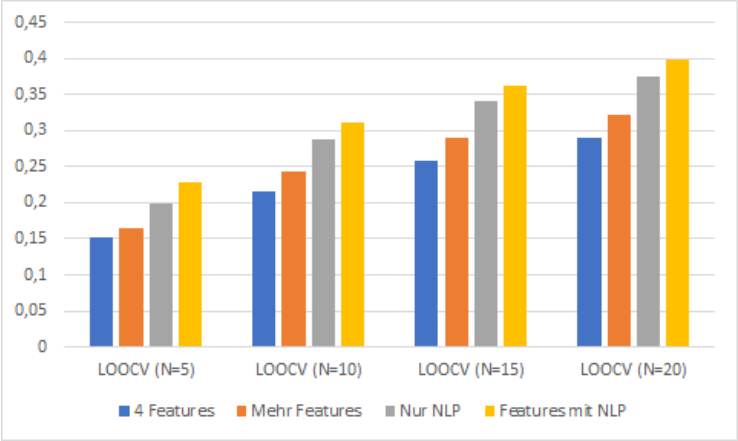
	4 Features	Mehr Features	Nur NLP	Features mit NLP
LOOCV (N=5)	0.151	0.165	0.199	0.229
LOOCV (N=10)	0.216	0.243	0.287	0.311
LOOCV (N=15)	0.259	0.289	0.340	0.362
LOOCV (N=20)	0.290	0.321	0.376	0.399

Tabelle 3.11: Evaluationsergebnisse CBF im Vergleich

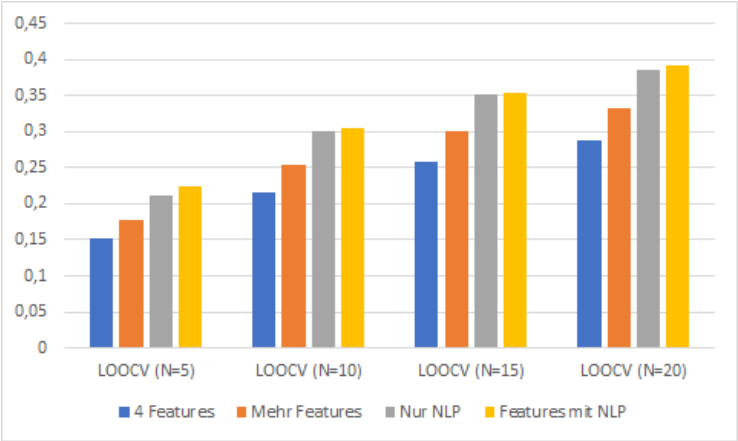
	4 Features	Mehr Features	Nur NLP	Features mit NLP
LOOCV (N=5)	0.151	0.177	0.212	0.224
LOOCV (N=10)	0.215	0.254	0.300	0.304
LOOCV (N=15)	0.258	0.301	0.352	0.354
LOOCV (N=20)	0.287	0.332	0.386	0.392

Tabelle 3.12: Evaluationsergebnisse CBF mit Basket-Daten im Vergleich

Man erkennt an den LOOCV-Ergebnissen, warum CBF vor allem für textbasierte Entities verwendet wird. Die beiden Techniken, die kategorische Daten verwenden und transformieren, erzielen schlechtere Ergebnisse als der rein auf NLP und TF-IDF basierte Ansatz. Am besten schneidet die Kombination zwischen den Features und NLP ab. Für die verschiedenen Datensätze und unterschiedlich gewählten N bleibt diese Rangfolge gleich.



LOOCV mit Datensätzen wie df1



LOOCV mit Datensätzen wie df2

Abbildung 3.13: Evaluationsergebnisse CBF im Vergleich

3.3.2 Ergebnisse des Apriori Algorithmus

Mit LOOCV können Assoziationsregeln, wie bereits erwähnt, nicht evaluiert werden. Die Ergebnisse des Apriori Algorithmus werden hier daher nur kurz angesprochen. Nach Durchführung der in Kapitel 3.2.4 beschriebenen Schritte erhält man 14 Assoziationsregeln. Aus diesen geht hervor, dass Geschenk- und Gutscheinkarten scheinbar oft zusammen bestellt werden. Eine Schüssel wird außerdem oft mit einem Teller gekauft und andersherum. Diese Regeln scheinen erst einmal schlüssig, ob diese aber auch tatsächlich effektiv sind kann erst beim Online Testing herausgefunden werden. Auch sind die Produkte der jeweiligen Regeln noch recht ähnlich zueinander (gleicher Produkttyp usw.), mit steigender Datenmenge und anderen Mindestwerten für Support und Confidence können möglicherweise bessere und vielfältigere Assoziationsregeln gebildet werden.

3.3.3 Zwischenfazit

1. Collaborative Filtering:
Beim Vergleich der CF-Methoden, MF und IB-CF, können folgende Schlussfolgerungen gezogen werden bzw. sind folgende Punkte zu beachten:

- MF bietet sich vor allem an, wenn man eine dichte Domäne mit nicht zu vielen Items hat, was hier der Fall ist.
- MF erlaubt die Einbeziehung von zusätzlichen Informationen [KBV09]. Da es sich hier um implizite Userdaten handelt, können diese daher oft mit einer dicht gefüllten Matrix repräsentiert werden. Ansonsten ist MF für spärliche Daten auch weniger empfindlich als IB-CF.
- MF eignet sich gut für personalisierte Empfehlungen, beispielsweise bei Newslettern oder Ähnlichem [TIKK15].
- Beim latenten Variablenmodell werden die Faktoren, die das Userverhalten beeinflussen, vom Algorithmus selbst bestimmt, es handelt sich um einen semi-unsupervised Learning Ansatz.
- MF hat eine gute Skalierbarkeit und Vorhersagengenauigkeit [KBV09; SAMN08; HU08]. Beim durchgeführten LOOCV Verfahren wurde ebenfalls ersichtlich, dass MF bessere Ergebnisse im Bezug auf Accuracy erzielt.
- Das Problem der Synonymität (siehe Kapitel 3.2.1.3) scheint vor allem bei latenten Variablenmodellen gut gelöst zu sein und tritt nicht oder kaum auf.
- Aus dem Entwicklungsprozess wird ersichtlich, dass die Empfehlungsgenerierung bei MF schneller abläuft als beim IB-CF. Es ist außerdem ein memory-effizientes Modell, dass Systeme recht schnell lernen können [KBV09].
- MF-Methoden haben offline zwar einen hohen Rechenaufwand, die resultierenden Modelle sind aber für den Online-Einsatz aufgrund ihrer Schnelligkeit von Vorteil [ERK11].
- Es ist allerdings zu berücksichtigen, dass MF für Offline Evaluationsmetriken normalerweise besser geeignet ist [TIKK15], beim Online Testing kann das Ergebnis anders ausfallen.
- Bei LOOCV handelt es sich um Evaluation auf userbezogene Empfehlungen. Bei Item-Item-Empfehlungen, also bei solchen, die nicht für einen bestimmten User sondern zu einem Produkt sind (z.B. Artikeldetailseite oder beim Checkout), könnte das IB-CF möglicherweise besser geeignet sein [TIKK15].

Geht man also von auf einen User zugeschnittene Empfehlungen aus, so ist die Methode des CF mit MF zu bevorzugen. In den Punkten Genauigkeit, Spärlichkeit und Skalierbarkeit ist MF dem IB-CF überlegen. Dennoch könnte sich letztere Technik gerade bei Item-Item-Empfehlungen besser anstellen, was online getestet werden kann. Außerdem bleibt weiterhin das Kaltstartproblem sowohl bei MF als auch beim IB-CF erhalten.

2. Content-Based Filtering:
Aufgrund der Evaluationsergebnisse ist für das CBF die vierte Variante zu bevorzugen. Mit der Kombination aus *ItemFeatureMatrix* und *ItemWordMatrix* wurden hier die besten Ergebnisse erzielt. Die Strategie auf Grundlage von TF-IDF wird dabei für die Produktattribute *Merkmale*, *Beschreibung* und *Titel* verwendet. Für die anderen wird One-Hot Encoding genutzt.
Mit CBF sind Hindernisse wie die Popularity Bias und das Kaltstartproblem gelöst, allerdings kommt, wie bereits erwähnt, eine neue Schwierigkeit hinzu: CBF ist restriktiv, es könnte also zu „Overspecialization“ kommen.

Um sowohl die Vorteile des CF als auch die von CBF miteinander zu vereinen, wird im nächsten Abschnitt das hybride Recommender Modell untersucht.

3.4 Hybrides System

Mit einem hybriden Recommender System können Elemente aus mehreren Modellen verwendet werden, um Empfehlungen zu generieren. Es wurden in der Literatur beispielsweise bereits viele Empfehlungstechniken zur Vermeidung des Kaltstartproblems erforscht, die dazu einen content-basierten Ansatz integrieren [CFHR10; JINSI14; LENHE16; MMN02]. In der Ausarbeitung von Schein et al. [SPUP02] werden unterschiedliche Methoden und Metriken allein für diesen Zweck untersucht.
Im Folgenden werden der Paralleled und der Monolithic Approach für ein hybrides System genauer untersucht.

3.4.1 Paralleled Hybrid Approach

Wie in Kapitel 2.2.3 beschrieben gibt es beim Paralleled Hybrid Ansatz getrennte Recommender Implementationen, hier das CF- und das CBF-Modell, dessen Ergebnisse dann miteinander kombiniert werden. Der Vorteil hierbei ist, dass das Item-Kaltstartproblem vermieden wird: Es kann sich darauf verlassen werden, dass die CBF-Komponente auf Grundlage der Produktattribute Empfehlungen generieren kann, auch, wenn für einen gegebenen Artikel keine oder nicht genug Bewegungsdaten vorliegen. Allerdings lässt sich damit nicht das User-Kaltstartproblem lösen, da sowohl das CF- als auch das CBF-Modell Userdaten benötigen, um für diesen Items empfehlen zu können.

Bei diesem Ansatz werden für die Top N Empfehlungen für ein gegebenes Produkt also sowohl CF als auch CBF verwendet. Das auf Produktdaten basierte Modell ist die in Kapitel 3.2.3 erarbeitete Variante, bei der *ItemFeatureMatrix* und *ItemWordMatrix* zusammen verwendet werden. Beim CF handelt es sich um das latente Variablenmodell mit MF und ALS als Optimierungsverfahren. Die generierten Empfehlungen, also die möglichen Kandidaten, werden zusammengefasst und dann nach ihren Scores sortiert. Es handelt sich hier also um die Weighted Hybridisierungsmethode. Da sich die Wertebereiche der Ähnlichkeitswerte beim CBF und beim CF in einem ähnlichen Rahmen bewegen werden sie nicht gesondert gewichtet. Items, die von beiden Modellen empfohlen werden, kommen doppelt in der Liste vor und werden dementsprechend entfernt. Zurückgegeben werden dann die ersten N Empfehlungen. Für die Top N Empfehlungen für einen bestimmten User wird der gerade beschriebene Vorgang auf die Items angewendet, mit denen der User bereits interagiert hat. Diese neue Liste wird dann wieder nach Duplikaten gefiltert und dann nach Ähnlichkeitswerten sortiert. Anschließend werden wieder die Top N ausgegeben.

3.4.2 Monolithic Hybrid Approach

Für den Monolithic Hybrid Ansatz wird die Feature Augmentation Methode verwendet: Der Output der einen Technik wird als Input für eine andere Technik verwendet. Die Idee ist folgende: Als Hauptkomponente zur Empfehlungsgenerierung soll hier CF verwendet werden. Für den Anfang, wenn für ein neues Item A noch nicht genug Daten vorliegen, dann soll auf CBF zurückgegriffen werden, um einen Artikel B zu finden, der ähnlich zu dem eigentlichen Produkt A ist und für den genug Bewegungsdaten vorliegen. B wird dann als Input für CF genutzt, sodass für A trotz unzureichender Daten Empfehlungen auf Basis von CF generiert werden können. Mit dieser Methode kann die Lücke zwischen bereits existierenden und neuen Items überbrückt werden, also wenn ein Item sozusagen noch „kalt“ ist, solange, bis es „warm“ genug ist.
Neue Artikel können, da CF auf Userdaten beruht, nicht empfohlen werden. Um diese ins System zu bringen, wird auf die ersten drei generierten Kandidaten für A wieder CBF angewendet und Items ausgewählt, die noch keine oder kaum Bewegungsdaten besitzen. Diese werden dann zusammen mit den für A generierten Kandidaten ausgegeben. Mehrfach vorkommende Produkte werden aus der Liste entfernt.
Für die Top N Empfehlungen für einen User wird das obige Verfahren wieder auf jedes Item angewendet, mit dem der User interagiert hat. Dabei werden Empfehlungen, die mit CBF generiert wurden, um Items ohne Transaktionsdaten zu finden, der Ähnlichkeitswert 0 zugewiesen. Für eine Liste mit den Top N werden dann 60% aus der Liste mit den von CF erzeugten Empfehlungen ausgegeben. Die restlichen 40% bestehen dann aus den aus CBF generierten Empfehlungen, also den Items ohne Bewegungsdaten.

3.4.3 Re-Ranking

Beim Re-Ranking, der letzten Etape eines Recommender Systems, können auf Grundlage von zusätzlichen Kriterien und Restriktionen einige der Empfehlungen entfernt werden und anders sortiert werden, bevor sie einem User gezeigt werden. Dieser Post-Processing Schritt ist in der Regel dafür verantwortlich, dass ein Recommender System, wie Ekstrand es ausdrückt, „nicht dumm aussieht“ [EKS16]. In diesem Zusammenhang sind Freshness und Diversity als Schlüsselbegriffe zu nennen, es geht nicht um Genauigkeit sondern um eine bessere User Experience: Wenig Diversity, also wenn kaum Vielfältigkeit bei den Empfehlungen existiert, kann zu einer schlechten User Experience führen. Genauso ist Freshness wichtig, das Recommender System sollte immer versuchen, die neuesten vorliegenden Informationen zu nutzen.
Folgende Kriterien könnten hier beim Re-Ranking z.B. angewendet werden:

- Vom User bereits gekaufte Produkte (wenn es Order-Events gibt) werden herausgefiltert.
- Produkte, die identisch zueinander sind - also quasi doppelt vorkommen - werden entfernt, falls das Problem der Synonimität vorher nicht vermieden werden konnte.
- Falls explizite Ratingdaten existieren dann werden Items, die vom User explizit negativ bewertet wurden, ebenfalls herausgefiltert.
- Artikel, die ausverkauft sind, sollen dem User in der Auflistung nicht gezeigt werden. Vom User bereits gekaufte oder ausverkaufte Items sind möglicherweise zwar gute Empfehlungen, für den Nutzer/die Nutzerin aber nicht sehr nützlich.
- Dem User sollen nur Produkte empfohlen werden, die seiner Zielgruppe bzw. seinem Geschlecht entsprechen.

Ein Re-Ranking erfolgt bei den hier in dieser Arbeit entwickelten Systemen nicht, ist jedoch bei der Umsetzung in der Praxis ein wichtiger Schritt, der durchgeführt werden sollte.

4 Evaluation

4.1 Systeme im Vergleich

4.1.1 LOOCV Ergebnisse

Um einen ersten Vergleich ziehen zu können, wird das LOOCV Verfahren auf die hybriden Modelle angewendet. Folgende Varianten werden evaluiert:

- Hybrid 1.1:
  Es handelt sich hierbei um den Paralleled Hybrid Approach mit CBF und CF mit MF und ALS als Optimierungsverfahren.

- Hybrid 1.2:
  Dies ist wieder der beschriebene Paralleled Hybrid Approach mit CBF und IB-CF als CF-Modell. Die Kosinusähnlichkeit wird hier als Distanzmetrik verwendet.

- Hybrid 2:
  Hierbei handelt es sich um den beschriebenen Monolithic Hybrid Approach, es wird als Hauptmodell CF mit MF verwendet. Ist dies nicht möglich wird CBF benutzt, um ähnliche Produkte zu finden, die Transaktionsdaten besitzen und auf die das CF Verfahren angewendet werden kann. Außerdem werden Produkte ohne Bewegungsdaten unter die Empfehlungen mit untergemischt.

Zu erwarten ist hier, dass die hybriden Recommender Systeme bei der Evaluation schlechter abschneiden, da das LOOCV Verfahren auf Genauigkeit beruht, die hybriden Modelle aber darauf ausgelegt sind, neue Items in das System zu bringen und eine bessere User Experience zu bieten. Hauptkriterium ist bei diesen also nicht die Accuracy. Gerade beim *Hybrid 2*-Modell werden sogar Items ohne Transaktionsdaten hinzugefügt, die bei LOOCV natürlich negativ ins Gewicht fallen. Daher ergibt die Anwendung von LOOCV hier eigentlich weniger Sinn, wird aber der Vollständigkeit halber und zur Evaluation hier ebenfalls angewendet.
Das LOOCV Verfahren wird genauso ausgeführt wie bereits in Kapitel 3.3.1, allerdings werden hier der Einfachheit halber und um zusätzlichen Rechenaufwand zu vermeiden, nur Datensätze verwendet, die so wie *df2* aussehen, die also keine addtobasket-Daten enthalten. Ergebnisse sind in Tabelle 4.1 aufgeführt, visualisiert sind sie in Abbildung 4.1. Zum Vergleich sind die Ergebnisse vom CBF (mit *ItemFeatureMatrix* und *ItemWordMatrix*), dem CF mit MF und ALS als Optimierungs-

verfahren und IB-CF mit Kosinusähnlichkeit als Distanzmetrik mit angegeben.

	CBF	MF	IB-CF	Hybrid 1.1	Hybrid 1.2	Hybrid 2
LOOCV (N=5)	0.229	0.319	0.327	0.260	0.158	0.192
LOOCV (N=10)	0.311	0.432	0.432	0.355	0.212	0.304
LOOCV (N=15)	0.362	0.500	0.490	0.409	0.247	0.369
LOOCV (N=20)	0.399	0.548	0.526	0.446	0.272	0.405

Tabelle 4.1: Evaluationsergebnisse Hybrid im Vergleich

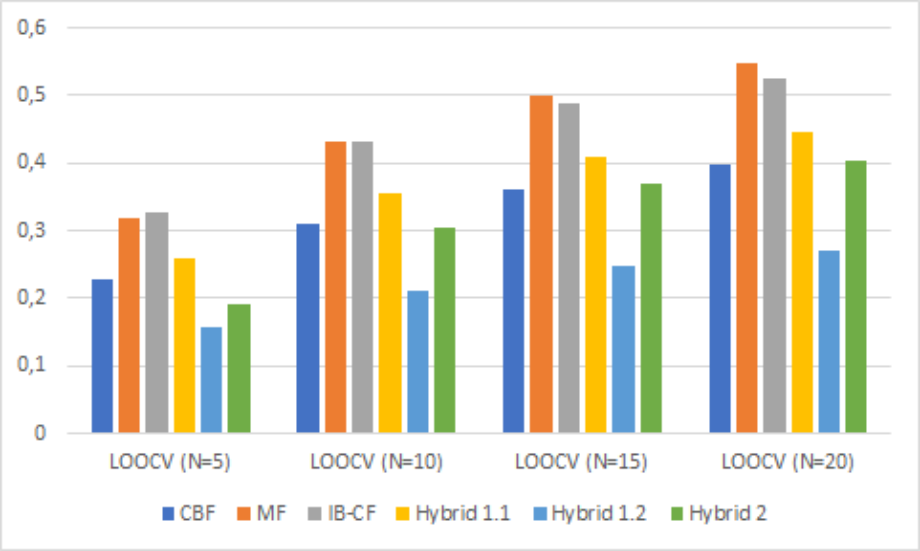


Abbildung 4.1: Evaluationsergebnisse Hybrid im Vergleich

Insgesamt schneiden die CF Recommender Modelle am besten ab. Verwunderlich ist hier, dass *Hybrid 1.1* und *Hybrid 1.2* so unterschiedliche Ergebnisse erzielen, obwohl sie den gleichen hybriden Ansatz verfolgen. Allerdings wurde bei der Evaluation der CF Modelle vorher auch ersichtlich, dass bei Datensätzen wie *df1* zwischen den IB-CF Methoden und CF mit MF keine signifikanten Unterschiede bei den Ergebnissen zu erkennen sind, während die Differenz bei *df2*-ähnlichen Daten offenbar größer ist. Damit schneidet *Hybrid 1.1* hier als drittbeste Methode ab, *Hybrid 1.2* am schlechtesten. Das reine CBF und *Hybrid 2* schneiden für höhere N ungefähr gleich ab, was ebenfalls überraschend ist. Außerdem ist auffällig, dass *Hybrid 2* bessere Ergebnisse erzielt als *Hybrid 1.2*, obwohl 40% der Empfehlungen bei *Hybrid 2* Produkte ohne bzw. nicht genügend Transaktionsdaten sind.

4.1.2 Vorteile

Hybride Systeme bringen viele Vorteile mit sich, einige dieser Punkte sind z.B.:

- Hybride Systeme ermöglichen eine Kombination verschiedener Techniken, um bessere Empfehlungen zu generieren. „Besser“ bedeutet entsprechend, je nachdem nach welchen Kriterien optimiert werden soll.
- Es können Informationen aus mehreren unterschiedlichen Quellen genutzt werden, hier z.B. aus dem Objektspeicher und den Userdaten.
- Ein großer Vorteil bei den hier vorliegenden hybriden Systemen ist, dass die Probleme des Item-Kaltstartproblems und der Filter Bubble gelöst werden können.
- Hybride Systeme können zu einer verbesserten User Experience und höherer Kundenzufriedenheit führen. Auch, wenn die Accuracy darunter leidet, unter den Kritikpunkten Novelty und Serendipity können sie dafür möglicherweise bessere Ergebnisse erzielen.

4.2 Herausforderungen

Folgende Herausforderungen, die teilweise auch schon in Kapitel 3.2.1.3 erwähnt wurden, galt es, in dieser Arbeit zu lösen bzw. zu optimieren:

- Neues Item - Das erwähnte Item-Kaltstartproblem konnte, wie bereits erläutert, gelöst werden, indem sich bei Auftreten von diesem dem CBF bedient wird. Zusätzlich werden im zweiten hybriden Ansatz „kalte“ Items mit unter die Empfehlungen gemischt, um diese in das System zu bringen.
- Filter Bubble - Das Problem der Restriktion tritt hauptsächlich beim CBF auf. Da CF hier als Hauptkomponente für das Recommender System genutzt wird und CBF als Starthilfe für neue Shopartikel, konnte auch die Problematik der Filter Bubble gelöst werden.
- Accuracy - Die Genauigkeit ist ein wichtiges Kriterium bei der Entwicklung eines Recommender Systems. Durch das LOOCV-Verfahren wurde ersichtlich, dass MF den Neighbourhood-basierten Methoden sowie den hybriden Systemen in diesem Punkt überlegen ist.
- Skalierbarkeit - Die Skalierbarkeit stellt bei kNN-Algorithmen ein Problem dar (siehe Kapitel 3.2.1.3), was bei MF optimaler verläuft (siehe 3.3.3).
- Interpretability - Bei Recommender Systemen ist es meist von Vorteil, zu wissen, warum einem User ein bestimmtes Item empfohlen wird. Es muss also eine gute Interpretierbarkeit der Daten gewährleistet sein. Dies war z.B. bei CBF ein Aspekt, der bei der Feature-Auswahl zu beachten war.

- Sparsity - Die Spärlichkeit der Daten ist ebenfalls eine Herausforderung, die vor allem beim CF zum Problem wird. MF ist hier eine Methode, die zur Lösung des Data Sparsity Problems genutzt wird.
- User Experience - Wie bereits erwähnt hängt eine gute User Experience nicht nur von der Genauigkeit der Empfehlungen ab. Durch die vorgestellten hybriden Ansätze werden verschiedene Modelle miteinander kombiniert, um im besten Fall eine bessere Nutzererfahrung zu bieten und so die Kundenzufriedenheit zu steigern.
- Performance - Die Performance eines Recommender Systems wird in dieser Arbeit nur angeschnitten, es wird allerdings ersichtlich, dass MF schneller abläuft als die anderen Techniken. Hier werden Empfehlungen für einen User innerhalb von Millisekunden generiert.
- Neuer User - Das Problem des neuen Users steht in dieser Arbeit ebenfalls im Hintergrund, Lösungsvorschläge hierfür wären, einem neuen User für den Anfang die beliebtesten oder am besten bewerteten Artikel des Shops zu empfehlen. Man kann ihn außerdem zu Beginn nach Vorlieben fragen, um so gleich Feedback bzw. ein Rating zu erhalten, mit dem gearbeitet werden kann.

4.3 Online Testing

In dieser Arbeit werden die entwickelten Modelle nur offline bewertet. Um jedoch zuverlässige Aussagen über ihre Relevanz treffen zu können, müssen diese noch anderweitig evaluiert werden.

4.3.1 Weiteres Vorgehen

Im weiteren Verlauf sollten die hybriden Systeme - und je nach Funktion, die erfüllt werden soll - auch die auf CBF und MF basierenden Modelle getestet werden, ob durch Online Testing oder User Studies. CBF könnte beispielsweise auf der Artikeldetailseite angewendet werden, CF unter dem Punkt: *„Kunden, die das gekauft haben, haben auch folgendes gekauft“*. Assoziationsregeln sind bei der Marktanalyse ebenfalls wichtig und können ebenfalls eingesetzt werden.
Durch Online Testing kann der Einfluss der Recommendation Engine auf das Verhalten der User bewertet und evaluiert werden, und, ob diese tatsächlich den Umsatz und die Kundenzufriedenheit steigert. Auf diese Weise können die verschiedenen Modelle in einer realen Ausgangssituation miteinander verglichen werden.

4.3.2 Erwartungen

Zu erwarten ist, dass die hybriden Systeme vor allem das Item-Kaltstartproblem lösen, welches bei CF auftritt. Basierend auf den LOOCV-Ergebnissen sollten die

auf MF-basierten Modelle besser abschneiden als die kNN-Methoden. Außerdem sollten die hybriden Ansätze eine bessere User Experience gewährleisten, vor allem der monolithische Ansatz, da hier neue vom CF-System noch nicht aufgefasste Items mitempfohlen werden.
Allgemein wird erwartet, dass das Recommender System den Umsatz und die Kundenzufriedenheit steigert.

5 Fazit und Ausblick

5.1 Zusammenfassung

Recommender Systeme sind mächtige Tools für Unternehmen, um den Umsatz zu steigern und Kundenzufriedenheit zu erhöhen. Auch für die User sind sie eine große Entscheidungshilfe und beim Finden neuer und interessanter Produkte sehr nützlich.

In dieser Arbeit wurden verschiedene Recommender Modelle für einen Online Shop der novomind AG entwickelt. Vorgeschlagen wurden hierbei zwei hybride Ansätze, die auf CF- und CBF-basierten Methoden aufbauen. Diese wurden zuvor ebenfalls entwickelt und implementiert. Untersucht wurde dabei IB-CF basierend auf kNN-Algorithmen. Ausgetestet wurden der euklidische Abstand, die Kosinusähnlichkeit und die Pearson-Korrelation als Distanzmetriken. Hier erwiesen sich die Unterschiede zwischen den beiden letzteren Methoden bei der Evaluation mit LOOCV als recht unerheblich, die Kosinusähnlichkeit erzielte minimal bessere Ergebnisse. Der euklidische Abstand erwies sich jedoch als sehr ungeeignet. Ein CF-Ansatz mit einem LV basierend auf MF wurde ebenfalls implementiert. Für CBF wurde als Distanzmetrik die Kosinusähnlichkeit verwendet und vier verschiedene Varianten getestet: Für den Algorithmus wurden erst nur vier Features verwendet, dann mehr, dann ein Ansatz nur auf NLP basierend und zum Schluss eine Kombination aus dem zweiten und dritten Ansatz. Hier schnitt die letzte Variante am besten ab. Weiterhin wurde ein Empfehlungsansatz basierend auf Assoziationsregeln mit Hilfe des Apriori Algorithmus umgesetzt.
Aus den gesamten Evaluationsergebnissen wird ersichtlich das das CF-Modell basierend auf MF am besten abschneidet. Die Empfehlungsgenerierung läuft bei MF schneller ab als bei den Neighbourhood-basierten Methoden und ist auch in den Punkten Skalierbarkeit und Spärlichkeit zu bevorzugen. Allerdings taucht, anders als bei CBF das Item-Kaltstartproblem auf. Es wurden daher hybride Ansätze vorgeschlagen, welche für neue Items CBF anwenden, um dieses Problem zu umgehen.

Während des Entwicklungsprozesses konnten einige Ansätze für Recommender Modelle durch die Offline Evaluation mit LOOCV bereits wieder verworfen werden. Es ist allerdings weitere Arbeit und Online Testing nötig, um herauszufinden, ob die Erwartungen sich auch bewahrheiten und welche der vorgestellten Modelle letztendlich eine bessere Nutzererfahrung bieten und den Umsatz steigern.

5.2 Aussichten

Wie bereits erwähnt sind Online Experimente notwendig, um die entwickelten Modelle zu testen. Das Recommender System ist außerdem eine Grundlage, auf der weiter aufgebaut und die optimiert werden kann, z.B. können neben den bereits erwähnten Distanzmetriken weitere zur Ähnlichkeitsbestimmung getestet werden. Außerdem gibt es weitere Algorithmen und Strategien, die das System ebenfalls optimieren könnten und möglicherweise bessere Ergebnisse erzielen (z.B. Decision Trees, Clustering, Bayessche Netze,...). Eine weiterer Ansatz sind neuronale Netze, die auf Deep Learning basieren.
Neben den neuen Methoden könnte auch das Feature Engineering noch weiter optimiert und ausgebaut werden, es können noch mehr Produkteigenschaften berücksichtigt und die Werte alle normalisiert werden.
Auch andere hybride Ansätze sind ein weiterer Weg zur Optimierung. Je nach User können verschiedene Algorithmen genutzt werden, um personalisierte Empfehlungen zu generieren.
Das bereits genannte Re-Ranking könnte in Zukunft umgesetzt und die Timestamps der Userhistorien mit berücksichtigt werden, sodass nur aktuelle Interessen des Users bei der Empfehlungsgenerierung verwendet werden. Dies würde auch die Liste der Items eines jeden Users, zu denen jeweils Empfehlungen gegeben werden, verringern und hätte damit Einfluss auf einen weiteren Gesichtspunkt, der in dieser Arbeit vernachlässigt wurde: Die Performance der verschiedenen Modelle. Diese ist gerade bei Online-Systemen sehr wichtig, in Zukunft könnte also auch im Hinblick auf die Fähigkeit zu einem Real-Time Recommender System, also einem System, dass Eingabedaten innerhalb von Millisekunden verarbeitet, getestet und optimiert werden.
Für Shops mit mehr Kundendaten könnten außerdem die genannten Knowledge- und Demographic-Based Recommender Techniken eine valide Option sein.

Anhang

A.1 Tabellen

Funktion	Beschreibung
Find Some Good Items	Dem User Items empfehlen, die er, basierend auf den Vorhersagewerten, vermutlich mögen wird. (Hauptfunktion eines Recommender Systems)
Find all good items	Es werden alle Items empfohlen, die die Userbedürfnisse befriedigen können.
Annotation in context	Bei einer gegebenen Itemliste werden einige von ihnen, basierend auf den Präferenzen eines Users, hervorgehoben.
Recommend a sequence	Statt einer einzelnen Empfehlung wird eine Sequenz von Items empfohlen.
Recommend a bundle	Es wird eine Gruppe von Items empfohlen, die gut zusammenpassen.
Just browsing	Dem User wird dabei geholfen, nach Items zu browsen, die eher zu seinen Interessen passen.
Find credible recommender	Manche User spielen mit Recommender Systemen herum, es werden Funktionen angeboten, mit denen dieser das System testen kann.
Improve the profile	Der User gibt Feedback darüber, was er mag und was nicht, um die Empfehlungen zu optimieren und zu personalisieren.
Express self	Einigen Usern ist das Recommender System egal, ihnen ist es wichtig, ihre Meinung zu teilen.
Help others	Es gibt User, die gerne Informationen teilen, weil sie damit anderen Usern helfen möchten.
Influence others	Es gibt User, deren Hauptziel es ist andere User bei ihren Käufen zu beeinflussen.

Tabelle A.1: Funktionen eines Recommender Systems nach Herlocker [HERL04]

Hybridisierungsmethode	Beschreibung
Weighting	Die Scores mehrerer Empfehlungstechniken werden kombiniert, um eine Empfehlung zu generieren.
Switching	Das System wechselt, abhängig von der jeweiligen Situation, zwischen verschiedenen Empfehlungstechniken.
Mixed	Empfehlungen mehrerer verschiedener Recommender werden gleichzeitig angezeigt.
Feature Combination	Features verschiedener Empfehlungsdatenquellen werden zusammen in einen einzigen Empfehlungsalgorithmus integriert.
Cascade	Ein Recommender verfeinert die Empfehlungen die von einem anderen generiert werden.
Feature Augmentation	Der Output der einen Technik wird als Input Feature für eine andere Technik genutzt.
Meta-level	Das Modell eines Recommenders wird als Input für einen anderen Recommender genutzt.

Tabelle A.2: Hybridisierungsmethoden nach Burke [BURKE02]

	channel	*brand*	*country*	*store*	*language*	*currency*
1	web	brand	de	-	de	EU
2	web	brand	de	-	de	EU

	sessionId	*datetime*	*userId*
1	12345SESSION1	2019-09-06 01:58:20.155000000	1234a5-6789b
2	45678SESSION2	2019-09-06 07:44:47.137000000	5g656-456j65

	articleId	*productId*	*sourceId*	*date*	*year*	*month*
1	34512-67891	34512	PRODUCT_VIEW	2019-09-06	2019	9
2	12386-12345	12386	PRODUCT_VIEW	2019-09-06	2019	9

Tabelle A.3: Beispielhafter Auszug der addtobasket-Datei

	articleId	*Titel*	*Marke*	*Produkttyp*
3964	34512-67891	Powder Winter Pant	Marke1	Hose
11506	12386-12345	Schale Infinity	Marke2	Besteck

	Merkmale	*Nachhaltigkeit*
3964	wasserdicht|winddicht|atmungsaktiv	Bio
11506	-	Recycled

	Geschlecht	*Farbe*	*Aktivitaet*	*MigrationModul*
3964	Mädchen+Jungs	türkis	Skitouren	-
11506	Neutral	multicolor	Wandern	Trekking/Outdoor

	Laenge	*Saison*	*Kragenform*	*Schichtenprinzip*
3964	lang	Herbst/Winter 2019	-	3. Schicht
11506	-	Durchläufer	-	-

	HighlightText	*productID*
3964	Hoch geschnittene Kinder-Schneehose mit Hosent...	34512
11506	Schale aus Lexan.	12386

	Beschreibung
3964	Mit der Powder Winter Pant von Marke1 kann...
11506	Als Muesli-Schale, Suppenteller oder für Desserts...

Tabelle A.4: Beispielhafter Auszug der Produktattribute

A.2 Git

Der Code für diese Arbeit sowie die genutzten Produkt- und Userdaten (in veränderter Form), sind unter folgendem Link zu finden:
https://gitlab.com/vynguyen/ba_recsys_2020

Literaturverzeichnis

[AGMI17] Agarwal, Ajay & Chauhan, Minakshi: „Similarity Measures used in Recommender Systems: A Study“, *International Journal of Engineering Technology Science and Research (IJETSR)*, Vol. 4, No. 6, Ohne Ort, Juni 2017

[AIS93] Agrawal, Rakesh & Imielinski, Tomasz & Swami, Arun: „Mining Association Rules between Sets of Items in Large Databases“, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, Vol. 22, No. 2, S. 207-216, Ohne Ort, Juni 1993

[AHR11] Ahrens, Sophie Charlotte: *Recommender systems - relevance in the consumer purchasing process*, epubli, Berlin 2011

[AND04] Anderson, Chris: *The Long Tail*, https://www.wired.com/2004/10/tail/#, Oktober 2004, letzter Zugriff: 16.12.2019

[BCT11] Bambini, Riccardo & Cremonesi, Paolo & Turrin, Roberto: „A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment“, in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 9, S.299-331, Springer, New York [u.a.], 2011

[BURKE02] Burke, Robin: „Hybrid Recommender Systems: Survey and Experiments“, *User Modeling and User-Adapted Interaction*, Vol. 12, No. 4, S.331-370, Ohne Ort, November 2002

[CFHR10] de Campos, Luis M. & Fernández-Luna, Juan M. & Huete, Juan F. & Rueda-Morales, Miguel A.: „Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks“, *International Journal of Approximate Reasoning 51*, S.785-799, Ohne Ort, 2010

[DEB08] Debnath, Souvik & Ganguly, Niloy & Mitra, Pabitra: „Feature weighting in content based recommendation system using social network analysis“, *Proceedings of the 17th International Conference on World Wide Web (WWW 2008)*, Beijing, April 2008

[DEKA11] Desrosiers, Christian & Karypis, George: „A Comprehensive Survey of Neighborhood-based Recommendation Methods“, in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 4, S.107-144, Springer, New York [u.a.], 2011

[EKS16] Ekstrand, Michael: *A PRACTICAL GUIDE TO BUILDING RECOMMENDER SYSTEMS - Don't look stupid*, https://buildingrecommenders.wordpress.com/2016/02/01/dont-look-stupid/, Februar 2016, letzter Zugriff: 05.12.2019

[ERK11] Ekstrand, Michael D. & Riedl, John T. & Konstan, Joseph A.: „Collaborative Filtering Recommender Systems", *Foundations and Trends in Human Computer Interaction*, Vol. 4, No. 2, S.81-173, Ohne Ort, Februar 2011

[FLHO03] Fleischmann, Michael & Hovy, Eduard: „Recommendations without user preferences: a natural language processing approach", *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, S. 242-244, vermutl. Miami, Januar 2003

[FRID20] Fridman, Lex: *Deep Learning State of the Art (2020) | MIT Deep Learning Series*, https://www.youtube.com/watch?v=0VH1Lim8gL8, Januar 2020, letzter Zugriff: 23.01.2020

[GASP15] Gaspar, Huba: *The Cold Start Problem for Recommender Systems*, https://www.yuspify.com/blog/cold-start-problem-recommender-systems/, Juli 2015, letzter Zugriff: 09.12.2019

[GER18] Géron, Aurélien: *Praxiseinstieg Machine Learning mit Scikit-Learn und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme* , 1. Aufl., O'Reilly, Heidelberg, 2018

[GOHU15] Gomez-Uribe, Carlos A. & Hunt, Neil: „The Netflix Recommender System: Algorithms, Business Value and Innovation", *ACM Transactions on Management Information Systems (TMIS)*, Vol. 6, No. 4, Artikel Nr. 13, Dezember 2015

[GOOGLEREC] Google LLC: *Machine Learning Courses Recommendation - Candidate Generation Overview*, https://developers.google.com/machine-learning/recommendation/overview/candidate-generation, Ohne Datum, letzter Zugriff: 05.12.2019

[GOOGLEREC] Google LLC: *Recommendations: What and Why?*, https://developers.google.com/machine-learning/recommendation/overview, Ohne Datum, letzter Zugriff: 16.12.2019

[GOOGLEML] Google LLC: *Representation: Feature Engineering*, https://developers.google.com/machine-learning/crash-course/representation/feature-engineering, Ohne Datum, letzter Zugriff: 31.12.2019

[GRIM18] Grimaldi, Emma: *How to build a content-based movie recommender system with Natural Language Processing*, https://towardsdatascience.com/how-to-build-from-scratch-a-content-based-movie-recommender-with-natural-language-processing-25ad400eb243, Oktober 2018, letzter Zugriff: 13.12.2019

[GRUS19] Grus, Joel: *Data science from Scratch: First Principles with Python*, 2. Aufl., O'Reilly, Beijing 2019

[HALE18] Hale, Jeff: *Smarter Ways to Encode Categorical Data for Machine Learning*, https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159, 2018, letzter Zugriff: 28.10.2019

[HZKC16] He, Xiangnan & Zhang, Hanwang & Kan, Min-Yen & Chua, Tat-Seng: „Fast Matrix Factorization for Online Recommendation with Implicit Feedback", *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, S.549-558, Pisa, Juli 2016

[HERL04] Herlocker, Jonathan L. & Konstan, Joseph A. & Terveen, Loren G. & Riedl, John T.: „Evaluating Collaborative Filtering Recommender Systems", *ACM Transactions on Information Systems (TOIS)*, Vol. 22, No. 1, S.5-53, vermutl. New York, Januar 2004

[HU08] Hu, Yifan & Koren, Yehuda & Volinsky, Chris: „Collaborative Filtering for Implicit Feedback Datasets", *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, S.263-272, Pisa, Dezember 2008

[IFO15] Isinkaye, F.O. & Folajimi, Y.O. & Ojokoh, B.A.: „Recommendation systems: Principles, methods and evaluation", *Egyptian Informatics Journal*, Vol. 16, No. 3, S.261 273, Kairo, Ägypten, November 2015

[JZFF11] Jannach, Dietmar & Zanker, Markus & Felfernig, Alexander & Friedrich, Gerhard: *Recommender Systems: An Introduction*, Cambridge University Press, Cambridge [u.a.], 2011

[JEUNEN18] Jeunen, Olivier & Verstrepen, Koen & Goethals, Bart: „Fair Offline Evaluation Methodologies for Implicit-Feedback Recommender Systems with MNAR Data", *Proceedings of the REVEAL '18 Workshop on Offine Evaluation*, Vancouver, Oktober 2018

[JINSI14] Jindal, Honey & Singh, Sandeep Kumar: „A HYBRID RECOMMENDATION SYSTEM FOR COLD START PROBLEM USING ONLINE COMMERCIAL DATASET", *International Journal of Computer Engineering and Applications*, Vol. 7, No. 1, Juli 2014

[KOBE11] Koren, Yehuda & Bell, Robert: „Advances in Collaborative Filtering", in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 5, S.145-186, Springer, New York [u.a.], 2011

[KBV09] Koren, Yehuda & Bell, Robert & Volinsky, Chris: „MATRIX FACTORIZATION TECHNIQUES FOR RECOMMENDER SYSTEMS“, *Computer*, Vol. 42, No. 8, S.30-37, vermutl. Washington, August 2009

[LENHE16] Lenhart, Philip & Herzog, Daniel: „Combining Content-based and Collaborative Filtering for Personalized Sports News Recommendations“, *Proceedings of the 3rd Workshop on New Trends in Content-Based Recommender Systems co-located with ACM Conference on Recommender Systems (RecSys 2016)*, Vol. 1673, S.3-10, Boston, September 2016

[LI19] Li, Susan: *Building a Collaborative Filtering Recommender System with ClickStream Data*, https://towardsdatascience.com/building-a-collaborative-filtering-recommender-system-with-clickstream-data-dffc86c8c65, 2019, letzter Zugriff: 28.10.2019

[LIAO19] Liao, Kevin: *Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering*, https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea, 2018, letzter Zugriff: 28.10.2019

[LOONY16] Loony Corn: *Byte-Sized-Chunks: Recommendation Systems*, https://www.udemy.com/course/recommendation-systems/, März 2016, letzter Zugriff: 20.11.2019

[LOPS11] Lops, Pasquale & de Gemmis, Marco & Semeraro, Giovanni: „Content-based Recommender Systems: State of the Art and Trends“, in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 3, S.73-105, Springer, New York [u.a.], 2011

[LUK19] Luk, Kevin: *Introduction to TWO approaches of Content-based Recommendation System*, https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c, 2019, letzter Zugriff: 25.11.2019

[LSY03] Linden, Greg & Smith, Brent & York, Jeremy: „Amazon.com Recommendations: Item-to-Item Collaborative Filtering“, *IEEE Internet Computing*, Vol. 7, No. 1, S.76-80, Ohne Ort, Januar 2003

[MACMN13] MacKenzie, Ian & Meyer, Chris & Noble, Steve: *How retailers can keep up with consumers*, https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers, Oktober 2013, letzter Zugriff: 16.12.2019

[MART] Martinez, Michael: *Amazon: Everything you wanted to know about its algorithm and innovation*, https://www.computer.org/publications/tech-

news/trends/amazon-all-the-research-you-need-about-its-algorithm-and-innovation, Ohne Datum, letzter Zugriff: 16.12.2019

[MCA16] McAlone, Nathan: *Why Netflix thinks its personalized recommendation engine is worth $1 billion per year*, https://www.businessinsider.in/Why-Netflix-thinks-its-personalized-recommendation-engine-is-worth-1-billion-per-year/articleshow/52754724.cms, Juni 2016, letzter Zugriff: 23.01.2020

[MCRJ16] McNee, Sean M. & Riedl, John & Konstan, Joseph A.: „Being Accurate is Not Enough: How Accuracy Metrics have hurt Recommender Systems“, *CHI EA '06: CHI '06 Extended Abstracts on Human Factors in Computing Systems*, S.1097-1101, Montréal-Québec, April 2016

[MMN02] Melville, Prem & Mooney, Raymond J. & Nagarajan, Ramadass: „Content-Boosted Collaborative Filtering for Improved Recommendations“, *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, S. 187-192, Edmonton-Alberta, Juli 2002

[NETFL09] Netflix Prize: *The Netflix Prize*, https://www.netflixprize.com/, 2009, letzter Zugriff: 18.12.2019

[NGZE18] Nguyen, Chi Nhan & Zeigermann, Oliver: *Machine Learning: kurz & gut*, 1. Aufl., O'Reilly, Heidelberg, 2018

[OPORMF19] Osadchiy, Timur & Poliakov, Ivan & Olivier, Patrick & Rowland, Maisie & Foster, Emma: „Recommender system based on pairwise association rules“, *Expert Systems With Applications*, Vol. 115, S. 535-542, Ohne Ort, Januar 2019

[PAR11] Pariser, Eli: *The Filter Bubble - What the Internet Is Hiding from You*, The Penguin Press, New York, 2011

[QFZM14] Qian, Xueming & Feng, He & Zhao, Guoshuai & Mei, Tao: „Personalized Recommendation Combining User Interest and Social Circle“, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 7, S.1763-1777, Ohne Ort, Juli 2014

[RASCH18] Raschka, Sebastian & Mirjalili, Vahid: *Machine Learning mit Python und Scikit-Learn und TensorFlow: Das umfassende Praxis-Handbuch für Data Science, Deep Learning und Predictive Analytics*, 2. Aufl., mitp, Frechen 2018

[REISAS90] Reichheld, Frederick F. & Sasser, Jr, W. Earl: „Zero Defections: Quality Comes to Services“, *Harvard Business School Review*, Vol. 68, No. 5, S.105-111, Ohne Ort, September-Oktober 1990

[RICCI11] Ricci, Francesco & Rochach, Lior & Shapira, Bracha: „Introduction to Recommender Systems Handbook“, in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 1, S.1-35, Springer, New York [u.a.], 2011

[RECC10] Rose, S. Rutherfoord & Engel, Dave & Cramer, Nick & Cowley, Wendy: „Automatic keyword extraction from individual documents“, in: Berry, Michael W. & Kogan, Jacob (Hrsg.): *Text Mining: Applications and Theory*, 1. Aufl., S.3-20, Wiley, Ohne Ort, März 2010

[SAMN08] Salakhutdinov, Ruslan & Mnih, Andriy: „Probabilistic Matrix Factorization“, *Proceedings of the 20th International Conference on Neural Information Processing Systems*, S.1257 1264, Vancouver, Dezember 2007

[SKKR01] Sarwar, Badrul & Karypis, George & Konstan, Joseph & Riedl, John: „Item-based Collaborative Filtering Recommendation Algorithms“, *Proceedings of the 10th International Conference on World Wide Web (WWW10)*, S.285-295, Hong Kong, April 2001

[SKR99] Schafer, J. Ben & Konstan, Joseph & Riedl, John: „Recommender Systems in E-Commerce“, *EC'99 Proceedings of the 1st ACM conference on Electronic commerce*, S.158-166, Denver-Colorado, November 1999

[SPUP02] Schein, Andrew I. & Popescul, Alexandrin & Ungar, Lyle H. & Pennock, David M.: „Methods and Metrics for Cold-Start Recommendations“, *Proceedings of the 25'th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, S.253-260, Tampere, August 2002

[SCIKIT] Scikit 0.22 Dokumenation: *6.2. Feature extraction*, https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction, Ohne Datum, letzter Zugriff: 02.01.2020

[SEV18] Severt, Natalie: *An Introduction to Recommender Systems (+9 Easy Examples)*, https://www.iteratorshq.com/blog/an-introduction-recommender-systems-9-easy-examples/, November 2018, letzter Zugriff: 05.12.2019

[SHANI11] Shani, Guy & Gunawardana, Asela: „Evaluating Recommendation Systems“, in: Ricci, Francesco & Rokach, Lior & Shapira, Bracha & Kantor, Paul B. (Hrsg.): *Recommender Systems Handbook*, Kap. 8, S.257-297, Springer, New York [u.a.], 2011

[SHAM13] Sharma, Meenakshi & Mann, Sandeep: „A Survey of Recommender Systems: Approaches and Limitations“, *International Journal of Innovations in Engineering and Technology (ICAECE 2013)*, Vol. 2, No. 2, S.8-14, März 2013

[SZLLM17] Silveira, Thiago & Zhang, Min & Lin, Xiao & Liu, Yiqun & Ma, Shaoping: „How good your recommender system is? A survey on evaluations in recommendation“, *International Journal of Machine Learning and Cybernetics*, Vol. 10, No. 5, S.813 831, Ohne Ort, Mai 2019

[SSRM14] Sivapalan, Sanjeevan & Sadeghian, Alireza & Rahanam, Hossein & Madni, Asad M.: „Recommender Systems in E-Commerce“, *2014 World Automation Congress (WAC)*, Vol. Proceedings, Kona-Hawaii, August 2014

[SL17] Smith, Brent & Linden, Greg: „Two Decades of Recommender Systems at Amazon.com“, *IEEE Internet Computing*, Vol. 21, No. 3, S.12-18, vermutl. USA, Mai 2017

[TIKK15] Tikk, Domonkos: *Neighbor methods vs matrix factorization - case studies of real-life recommendations*, https://de.slideshare.net/domonkostikk/neighbor-methods-vs-matrix-factorization-case-studies-of-reallife-recommendations-gravity-lsrs2015-recsys-2015, 2015, letzter Zugriff: 12.12.2019

[WEI16] Wei, Kangning & Huang, Jinghua & Fu, Shaohong: „A Survey of E-Commerce Recommender Systems“, *2007 International Conference on Service Systems and Service Management*, S.1-5, Chengdu, Juni 2007

[YANG16] Yang, Zhe & Wu, Bing & Zheng, Kan & Wang, Xianbin & Lei, Lei: „A Survey of Collaborative Filtering Based Recommender Systems for Mobile Internet Applications“, *IEEE Access*, Vol. 4, S.3273-3287, Mai 2016

Code-Ausschnitte

Ich versichere, die vorliegende Arbeit selbstständig ohne fremde Hilfe verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt zu haben. Die aus anderen Werken wörtlich entnommenen Stellen oder dem Sinn nach entlehnten Passagen sind durch Quellenangaben eindeutig kenntlich gemacht.

Ort, Datum Hoang Tuong Vy Nguyen