Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Eduard Weigandt

Toolchain zur Untersuchung von Empfehlungssystemen basierend auf Matrix Factorization

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Eduard Weigandt

Toolchain zur Untersuchung von Empfehlungssystemen basierend auf Matrix Factorization

Projektbericht eingereicht im Rahmen der Umsetzung von Projekt 1

im Studiengang Master of Science Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuende Prüfer: Prof. Dr. Kai von Luck

Eingereicht am: 8. September 2015

Eduard Weigandt

Thema der Arbeit

Toolchain zur Untersuchung von Empfehlungssystemen basierend auf Matrix Factorization

Stichworte

Analytik, Statistik, Mahout, Lenskit, Matrix Factorization, MAE, RMSE, MovieTweeting, Book-Crossing, MovieLens

Kurzzusammenfassung

Inhaltsverzeichnis

1 Einleitung

In allen Bereichen des digitalen Lebens lassen sich Empfehlungssysteme wiederfinden. Angefangen bei redaktionell erstellten Vorschlägen für angesagte Filme oder automatisch berechneten Artikelempfehlungen mit Hilfe vom Kaufverhalten der Nutzer. Die richtige Strategie bei der Präsentation und Auswahl von Produkten kann entscheidend sein, um den Kunden besser zufriedenzustellen. Die einzelnen Daten zur Berechnung der Vorschläge können in aktiver oder passiver Form gesammelt werden. Für die neuen Kunden können bei der Filterung von Artikeln entweder Bewertungen und Erfahrungen von bestehenden Kunden oder Eigenschaften von gekauften Produkten einbezogen werden. Diese Variante betrachtet nicht den einzelnen Nutzer, sondern die ganze Gruppe und das Wissen über diese. Als weitere Möglichkeit können die individuellen Bedürfnisse der Nutzer betrachtet werden. Jemand der eine Affinität zu Elektronikartikeln zeigt, braucht dadurch nicht unbedingt Werbung für die Bademode. Außer, dass die Wahrscheinlichkeit für diese Ausnahme anhand des Kontextes[1] erhöht wird, weil es z.B. Sommer ist und die Badesaison anfängt. Das Hauptaugenmerk liegt meistens in den Beziehungen zwischen den Artikeln und Benutzern im derzeitigen Kontext. Die einzelnen Verfahren dafür haben je nachdem welches Ziel verfolgt wird gute oder schlechte Ergebnisse. Die richtige Kombination von Verfahren und die Evaluierung dieser stellt eine große Hürde dar. In den meisten Fällen ist das Versagen einer Methode eine gute Empfehlung zu finden, nicht auf Anhieb nachvollziehbar (Ekstrand u. Riedl, 2012). Aktuell hat sich *Matrix Factorization* (MF) (Koren u. a., 2009) als einer der vielversprechendsten und flexibelsten Ansätze zur Berechnung von Empfehlung bewehrt. Aus diesem Grund wird dieser Ansatz auch in dieser Ausarbeitung verwendet und untersucht.

1.1 Vorgehen und Ziele

In der Arbeit Weigandt (2015) wurden wichtige Komponenten aus der Theorie zur Erstellung eines Werkzeuges im Umfeld der Evaluierung von Empfehlungen erläutert. Das Ziel in der aktuellen Ausarbeitung besteht in erster Hinsicht in der Validierung von *Frameworks* in Kombination mit vorhandenen *Techniken*. Diese sollen dann zur Entwicklung einer Umgebung (Toolchain) für Experimente mit Hybriden-Empfehlungssystemen eingesetzt werden. Die daraus resultierenden Herausforderungen werden in einer anderen Projektarbeit zur Risikominimierung behandelt. Zum Testen der Toolchain werden Bewertungen von Filmen als Testdaten verwendet, da diese Daten in aufbereiteter Form zur Verfügung stehen. Die Absicht im Aufbau einer Toolchain

[1] Alle weiteren Metadaten, wie Jahreszeit, Alter, Geschlecht, Verhalten bei der Navigation, etc...

besteht in der Überprüfung der vorliegenden wissenschaftlichen Arbeiten im Gebiet der Empfehlungen. Die daraus gezogenen Ergebnisse sollen dann als Grundlage für ein Wunschsystem dienen, welches kontextabhängige Empfehlungen für Kunden von E-Commerce Plattform generieren soll. Die aufgebaute Umgebung soll darüber hinaus später zum Vergleich des Wunschsystems mit anderen Systemen genutzt werden. Als ein weiteres Ziel wird die Eignung von MF untersucht. Die erkannten Charakteristika dieses Verfahrens sind dabei von großem Interesse.

Konferenz Als erster Anlaufpunkt wurden die aktuellsten Veröffentlichungen von der *ACM RecSys*[2] Konferenz durchsucht. Auf der *ACM RecSys* werden internationale Ergebnisse aus dem Forschungsfeld der Empfehlungs-Systeme präsentiert. Dadurch eignen sich die Veröffentlichungen gut, um eine Grundlage, auf der man weiter aufbauen kann, zu schaffen.

2 Umgebung und Methodik

Um eine Umgebung aufzubauen in der man auswertbare Experimente ausführen kann, braucht man nachvollziehbare und reproduzierbare Forschungsergebnisse. In Said u. Bellogín (2014) wird kritisch untersucht inwieweit man aktuelle Veröffentlichungen reproduzieren kann. Der Fokus dabei war auf die folgenden Aspekte gerichtet:

1. **Dataset:** öffentliche Datensätze (nicht zu spezialisierte Daten)
2. **Recommendation framework:** veröffentlichter Sourcecode, aktive Community
3. **Data details:** Angaben zur Reproduktion der Trainingsdaten (Splitting, Berechnungen)
4. **Algorithmic details:** Eingesetzte Werte für die Parameter

2.1 Aufbau

Die aufgezählten Aspekte werden ebenfalls in der hier präsentierten Projektarbeit in der Auswahl von Werkzeugen und dem Dokumentieren der erbrachten Eigenleistung weiter verfolgt. In Abbildung 2.1 sieht man die einzelnen Schritte, die von Said u. Bellogín (2014) genutzt wurden um drei bekannte Frameworks untereinander zu vergleichen. Eines der Ziele in dieser Projektarbeit besteht in der Validierung und Dokumentierung der erläuterten Beispielumgebung für weiterführende Arbeiten.

[2] Wissensdatenbank `http://www.recsyswiki.com/wiki/Main_Page` und Konferenz 2014 `recsys.acm.org/recsys14`

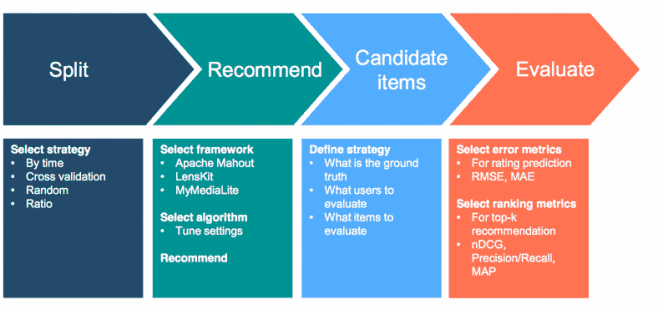

Abbildung 2.1: Evaluierungs-Pipeline von Empfehlungs-Systemen. (Quelle: Said u. Bellogín (2014))

Datensplitting Im ersten Schritt der Pipeline aus Abbildung 2.1 wird eine Auswahl zur Aufteilung der Daten getroffen. Dabei wird ein Teil der Daten für das Training eines Modells genutzt und der Rest zur Bewertung (Testset) der resultierenden Empfehlungen aus dem Modell. Die Experimente werden nicht auf Echtzeit-Daten ausgeführt, sondern auf den Datensätzen von *MovieLens*[1] Projekt, MovieTweetings[2] (Dooms u. a., 2013) und Book-Crossing(Ziegler u. a., 2005). Die häufigsten Strategien zu den Aufteilungen bestehen aus:

a. Jeder Nutzer bekommt unterschiedliche prozentuale Anteile.

b. Für jeden Nutzer wird ein fester Anteil definiert. (z.b. 80% / 20%)

c. Kreuzvalidierung: Eine Untermenge von allen Nutzern auswählen und daraus disjunkte echte Untermengen bilden, die jeweils nur einmal zur Evaluierung verwendet werden.

Ein Problem was bei solchen Offline-Untersuchungen auftreten kann, ist unter anderem dass man die Dynamik eines realen Systems nicht erfassen kann. Die Vorlieben von Menschen können sich verändern und sind dadurch schwer durch statische Datensätze zu untersuchen (Garcin u. a., 2014). Jedoch gelangt man nur sehr schwer an neue Datensätze, da diese wie Firmengeheimnisse behandelt werden.

Empfehlungen Die aufgeteilten Daten werden im nächsten Schritt zur Erzeugung von Empfehlungen verwendet. Das eingesetzte Verfahren in dieser Projektarbeit ist die *Matrix Factorization* (Koren u. a., 2009). Dabei wird versucht die Dimension des Problemraums[3] zu reduzieren und dabei versteckte Faktoren zwischen den Nutzern und zu empfehlenden Artikel aufzudecken. Als Ergebnis ergibt sich für jeden Nutzer und Gegenstand ein Vektor mit Faktoren, die in der Kombination einen Wert berechnen, der das Interesse vom Nutzer zum Artikel darstellt.

[1] `http://grouplens.org/datasets/movielens/`
[2] `https://github.com/sidooms/MovieTweetings`
[3] Kreuzprodukte von Nutzern zu Artikeln.

Leider ist die parallele Ausführung mehrerer Empfehlungs-Systeme in dieser Pipeline, so wie die Möglichkeit zwischen Algorithmen in einem Framework dynamisch zu wechseln nicht gegeben. Man hat nur eine Blackbox Ansicht auf das System. Dadurch kann man die Systeme nur anhand der Eingabe und Ausgabe analysieren.

Basis der Bewertung Damit man die Güte der späteren Empfehlungen beurteilen kann, braucht man Techniken um die Vorhersagegenauigkeit und die Relevanz bezüglich eines Kriteriums zu messen. Für das letztere wird eine Basis (*Ground Truth*) definiert anhand welcher man die späteren Ergebnisse bewerten kann. Dazu gehört eine Auswahl an Kriterien aufzustellen bei denen bestimmte Nutzer und Gegenstände im Fokus stehen. Dieser Schritt ist nicht notwendig, wenn man nur die Fehlerrate bzw. Abweichung zu den bekannten Bewertungen und den Berechneten überprüfen will. In dieser Projektarbeit wird dieser Schritt zurückgestellt, da keinen expliziten Szenarien verglichen werden sollen. Darüber hinaus braucht man mehr Erfahrung die richtige *Ground Truth* zu definieren, was nicht Gegenstand der aktuellen Arbeit ist und zu möglichen Fehlinterpretationen führen kann.

Evaluierung Zum Schluss wird die Güte der berechneten Scores über statistische Formeln zur Bewertung der Genauigkeit (MAE[4], RMSE[5]) und Relevanz (F-Maß[6]) der Ergebnisse untersucht (Herlocker u. a., 2004). Diese Formeln sind gut bekannt und werden oft bei solchen Analysen genommen, jedoch spiegeln sie nicht die Realität wider, da Menschen durch viele Faktoren beeinflusst werden. Wie in der Einleitung erwähnt ist der Kontext sehr entscheidend. Die Kombination aus Präsentation und Wahrnehmung spielt eine entscheidende Rolle wie man anhand von Hollywood Blockbustern sehen kann. Die Filme mit den größten Budgets und den bekanntesten Schauspielern werden häufiger angesehen. Durch das Extrahieren solcher Informationen aus z.B. der IMDB[7] lassen sich Empfehlungen weiter anreichern. In Said u. a. (2012) wird die Evaluierung von drei unterschiedlichen Anforderungsgebieten vorgeschlagen, bei denen jeweils unterschiedliche Ziele verfolgt werden. Dazu zählen die Nutzer- und Unternehmenssicht sowie die des Systems. In McNee u. a. (2006) wird unter anderem argumentiert, dass die Fixierung auf die berechnete Genauigkeit dem Forschungsgebiet schadet und der Nutzer im Vordergrund stehen sollte.

[4] **Mean absolute error**: Durchschnitt der absoluten Fehler zwischen Vorhersage und Testdaten.
[5] **Root mean square error**: Mittlere quadratische Abweichung der Vorhersage aggregiert auf einen Wert.
[6] Mittel über Genauigkeit und Trefferquote.
[7] `www.imdb.com`

Ergebnisse und Fazit Die beschriebene Pipeline wird in Form einer modularen Bibliothek (Java) unter dem Projektnamen RiVal[8] von Said u. Bellogín (2014) bereitgestellt. Die einzelnen Module bieten Hilfsklassen, um z.B. Daten von einem Schritt zum nächsten zu transportieren. Die Anbindung zu Lenskit[9] und Apache Mahout[10] sind gegeben, so dass man nur noch die Konfiguration eintragen muss. Die Einstellungen bestehen jeweils aus den Details zum Algorithmus, Datensets und den gewünschten Metriken. Eine wichtige Erkenntnis der Autoren die bei der Auswertung der Frameworks mit Hilfe von RiVal festgestellt wurde ist, dass sich die Implementierungen zu sehr voneinander unterscheiden, was zu einer schlechten Vergleichbarkeit führt. Trotzdem ist die Pipeline und der Grundgedanke dahinter interessant, um die Umsetzungen von anderen zu verstehen und deren Forschungsergebnisse nachzustellen. Dadurch kann man leichter auf anderen Arbeiten aufbauen. In dieser Projektarbeit konnte dadurch die Ausführung der Ergebnisse leichter nachgestellt werden, jedoch bleiben noch viele offene Fragen bezüglich der Untersuchung auf Offline-Daten, sowie den eingesetzten statistischen Formeln die nicht ein komplettes reales System überprüfen. Zudem muss die Pipeline in der Implementierung erweitert werden, um auch andere Datenquellen für die Modelle zuzulassen.

3 Matrix Factorization

Dieses Verfahren ist im Bereich der Collaborative Filtering Algorithmen angesiedelt und charakterisiert z.B über Bewertungen die Beziehung zwischen Artikeln und Nutzern. Die Grundidee besteht in der Berechnung zweier oder mehr Matrizen, die über das Produkt die ursprüngliche Matrix wiedergeben (siehe Tabelle 3.1). Durch die Zerlegung einer solchen Matrix in Faktoren, kann man unterschiedliche Eigenschaften (Feature) für die Beschreibung einer Beziehung nutzen. Jedes erkannte Feature stellt eine eigenständige Dimension (f) dar, welche z.B. Nutzerverhalten, Genres oder Formen beschreiben kann. Die richtige Anzahl an Features entscheidet, ob man alle Aspekte der Daten erfasst oder sich zu sehr an die gegebenen Daten anpasst (Overfitting).

Beispiel Matrix Zu jedem Film i existiert ein Vektor q_i von Faktoren, welcher die positive oder negative Charakteristik eines Films darstellt. Dem entsprechend gibt es auch für jeden Nutzer u einen Vektor, welcher das Interesse des Nutzers zu den Faktoren widerspiegelt (siehe Gleichung 3.2). Die Wechselwirkungen zwischen diesen beiden Vektoren werden durch das Produkt in der Gleichung 3.3 dargestellt. Die Herausforderung liegt nun jeweils in der Berechnung von passenden Faktoren, mit deren Hilfe dann die Bewertungen von nicht gesehenen Filmen vorher-

[8] https://github.com/recommenders/rival
[9] https://github.com/lenskit/lenskit
[10] http://mahout.apache.org/

$$
\begin{array}{c}
\\
Frank \\
Anna \\
Peter \\
\vdots \\
User_n
\end{array}
\begin{array}{c}
\begin{array}{cccc} Terminator & Matrix & \cdots & Movie_n \end{array} \\
\begin{pmatrix}
2 & 3 & \dots & r_1 \\
1 & 5 & \dots & r_2 \\
5 & & \dots & r_2 \\
\vdots & \vdots & \ddots & \vdots \\
r_n & r_n & \dots & r_n
\end{pmatrix}
\end{array}
$$

Tabelle 3.1: Matrix: Nutzer Bewertungen für Filme

gesagt werden können (Koren u. a., 2009). Ein mögliches Ergebnis einer solchen Berechnung sieht man in Tabelle 3.2, wo aus der Tabelle 3.1 eine kleiner dimensioniertes Modell berechnet wurde.

$$
\begin{array}{ccc}
\text{Nutzer} & & \text{Filme} \\
 & * & \\
\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{array}
\begin{array}{c}
p'_u = \\
\\
\begin{pmatrix} 2 \\ 1 \\ 2.5 \\ \vdots \\ r_n \end{pmatrix}
\end{array}
 & &
\begin{array}{c}
\begin{array}{cccc} f_1 & f_2 & \cdots & f_n \end{array} \\
q_i^T = \begin{pmatrix} 1 & 1 & \dots & r_1 \end{pmatrix}
\end{array}
\end{array}
$$

Tabelle 3.2: Das Beispielmodell einer Dimension (Bewertungen) auf den gegebenen Daten.

$$q_i \in \mathbb{R}^f \qquad (3.1) \qquad\qquad p_u \in \mathbb{R}^f \qquad (3.2)$$

Matrixzerlegung Ein häufig genutztes Verfahren zur Ermittlung von Faktoren kommt aus dem Bereich von Information Retrieval. *Singular Value Decomposition* (SVD) (Paterek, 2007) zerlegt im Groben eine gegebene Matrix in eine faktorisierte Darstellung[1]. Damit kann man unterschiedliche Vektoren von Faktoren berechnen. Der Vorteil von MF ist die Geschwindigkeit zur Laufzeit neue Bewertungen zu berechnen. Diese Eigenschaft wird sich durch eine kostspielige Modellberechnung erkauft.

In Gleichung 3.4 sieht man die Formel, welche durch die Minimierung der Abweichung zwischen bekannten Bewertungen der Nutzer (u, i) aus dem Trainingsset (κ) und der Vorhersagten

[1] Auf unser Beispiel angewandt ergibt sich folgende Darstellung: $M = q_i^T \cdot S \cdot p'_u$.

$$\widehat{r}_{ui} = q_i^T p_{u'} \tag{3.3}$$

$$\min_{q',p'} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{3.4}$$

zur Optimierung genutzt werden kann. Dabei steht $r_{ui} - q_i^T p_u$ für die Abweichung zwischen dem realen Wert und der Vorhersage. Das λ soll in diesem Kontext die Überanpassung (Overfitting) minimieren, sonst wird das Modell nur für die aktuellen Daten angepasst und die Übertragbarkeit auf neue Daten ist nicht mehr gegeben. Diese Gefahr besteht, weil viele Einflussfaktoren in der Berechnung einbezogen werden, die aber möglicherweise keine Auswirkungen auf alle Bewertungen haben.

Präferenzen Wie schon erwähnt erfasst Gleichung 3.3 die Wechselwirkungen zwischen den Nutzern und Artikeln. Das darauf aufbauende Modell muss jedoch nicht von beiden Teilen gleich abhängig sein. Ein Beispiel dafür sieht man in einem Nutzer der unabhängig vom Film immer sehr gute Bewertungen abgibt. Damit auch diese Eigenschaft berücksichtigt wird, kann man die Formel auf die Gleichung 3.5 erweitern.

$$\widehat{r}_{ui} = \mu + b_u + b_i + q_i^T p_{u'} \tag{3.5}$$

Das μ steht für die globale Bewertung im Durchschnitt und b_i sowie b_u für die jeweiligen Offsets, die über alle Bewertungen zum Film oder des aktuellen Nutzers berechnet werden.

Beispiel 1 *Bei einer durchschnittlichen Bewertung von $\mu = 4.0$ Punkten für den Film The Shawshank Redemption addiert man den Unterschied gegenüber anderen Film $b_i = 0.5$ auf, da dieser Film besonders hoch bewertet wird. Zum Schluss addiert man für den Nutzer $b_u = 0.5$ hinzu, weil dieser im Gegensatz zu anderen um 0.5 Punkte höher bewertet. Die daraus berechnete Bewertung von diesem Nutzer für diesen Film beträgt damit 5.0 Punkte. Damit ist dieser Film für diesen Nutzer auf einer Skala von 0 bis 5 sehr empfehlenswert.*

Weitere Datenquellen Um weitere Informationen über die Person zu Berücksichtigen muss man die Gleichung 3.5 soweit erweitern, dass man aus der Menge von Eigenschaften[2] $I(u)$, die Summe aller Vektoren mit allen Faktoren zu den Eigenschaften $\gamma_a \in \mathbb{R}^f$ bildet und diese dann mit dem Nutzer Vektor p'_u addiert (siehe Gleichung 3.6).

[2]Diese Informationen sind indirekt über den Nutzer zu bekommen und werden in Form von booleschen Werten dargestellt.

$$\widehat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_{u'} + \sum_{a \in I(u)} \gamma_a) \tag{3.6}$$

Zeitlich Dynamik der Daten Die bisher berechneten Daten im System sind statisch und berücksichtigen keinen Wandel im Verhalten der Nutzer über die Zeit. Dazu zählt z.B. die Veränderung der Nutzervorlieben zu einem Filmgenre. Daraus folgernd muss man die Gleichung 3.6 und die dynamischen Komponenten[3] von der Zeit abhängig machen. Eine gute Berechnung ist schwer Nachzustellen, da man kein Echtzeitsystem beobachtet, sondern nur statische Daten. Trotzdem könnte man eine künstliche Abschwächung über die Zeit einbauen (Rafailidis u. Nanopoulos, 2014). Dies ist eine weitere Stellgröße für zukünftige Experimente, die jedoch nicht im Rahmen dieser Projektarbeit behandelt wird, weil nur ein Ausschnitt in der Zeit betrachtet wird.

Fazit Wie man sehen konnte, kann man MF flexibel um neue Aspekte in Form von Featuren erweitern. Die Berechnung der Matrizen für jeden einzelnen Nutzer ist jedoch von der Anzahl an zu suchenden Features abhängig und begünstigt eher eine offline Berechnung. Dazu muss man sagen, dass sich die Berechnung der einzelnen Vektoren verteilt berechnen lässt, je nachdem wie man die Matrixzerlegung ausführt. Als eine Überlegung sollte man wegen des Kalt-Start Problems eine alternative Berechnung von Empfehlungen einsetzen, welche für die Anfangszeit eines Nutzers präsentiert wird. Die Übertragbarkeit auf andere Domänen ist ohne Probleme möglich, wenn eine ausreichende Anzahl an Bewertungen existieren.

4 Experiment

4.1 Vorgehen und Aufbau

Die Untersuchung von den gegebenen Frameworks und Verfahren wurde in mehreren Stufen durchgeführt. In der ersten Stufe wurde versucht die bekannten Ergebnisse nachzustellen. Danach wurde eine naive Implementierung von MF, anhand der Beschreibungen aus Kapitel 3 zum Vergleich umgesetzt. Zuletzt wurde die Ausführung auf einen neuen Datensatz mit Bewertungen von Büchern ausgeweitet. Der Grund dafür ist das Testen der Übertragbarkeit auf andere Datensätze. Während jeder Stufe wurde darauf geachtete, welche Stellgrößen die Erstellung von Empfehlung beeinflussen. Teilweise wurden bestimmte Erkenntnisse in den vorherigen Kapiteln unter den passenden Kategorien schon erläutert.

[3] Alle außer $q_i^T(p_{u'} + \sum_{a \in I(u)} \gamma_a)$

4.1.1 Einstellungen

MyMediaLite wurde im Experiment aus dem Fokus genommen, da Java für weitere Entwicklungen gewählt wurde. Sowohl Lenskit als auch Mahout sind in Java geschrieben und im Gegensatz zu MyMediaLite werden diese aktiv weiter entwickelt. Für die Bewertung der Genauigkeit von Empfehlungen wurde *MAE* und *RMSE* eingesetzt, um erste Erfahrungen mit statistischen Auswertungen zu sammeln.
Bei der Untersuchung der Umgebung wurden drei Datensätze verwendet die eine sehr geringe Dichte aufweisen. In Abbildung 4.1, 4.2 und 4.3 sieht man die jeweiligen Verteilungen der Bewertungen.

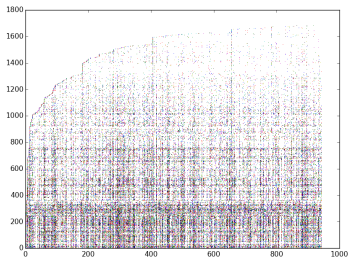

Abbildung 4.1: MovieLens

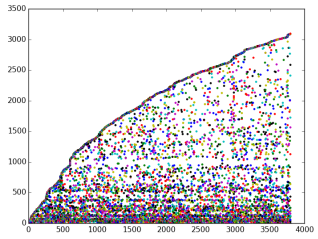

Abbildung 4.2: MovieTweeting

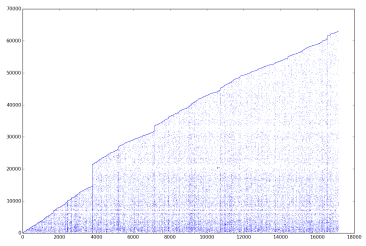

Abbildung 4.3: BookCrossing

Datenverteilung (x-Achse = Nutzer, y-Achse = Filme)

Datenformat Das Format von MovieLens ist simpel und ähnelt einer CSV Datei, wie man in Tabelle 4.1 sehen kann. Alle weiteren Datenquellen haben jeweils leicht abgewandelte Variationen von diesem. Die wesentlichen Unterschiede gegenüber MovieLens sind die neueren Bewer-

Benutzer ID	Artikel ID	Bewertung	Zeitstempel
78	345	5	803506521
12	301	1	860506923
45	23	2	870406913

Tabelle 4.1: MovieLens Datenformat

tungen von Filmen aus dem Jahr 2008. Zudem enthalten BookCrossing und MovieTweeting die Bewertungen im Wertebereich von 0 bis 10. Zudem wurde nur ein Bruchteil vom BookCrossing Datensatz verwendet.

4.2 Auswertung

Pipeline Die in Kapitel 2 genannten Aspekte für eine bessere Umgebung lassen sich alle wiederfinden, jedoch durch eine recht umständliche Recherche im Beispielcode. Die Zwischenergebnisse eines Schrittes in der Pipeline werden in temporäre Dateien abgespeichert, welche danach vom nächsten Schritt eingelesen werden. Das Nachstellen der Ergebnisse war nicht ohne Probleme möglich, da die Struktur der Pipeline erst zu diesem Zweck bereinigt und erweitert werden musste. Es existieren viele Methoden und Abläufe, die an eine Empfehlungs-Bibliothek angepasst sind und nicht generisch funktionieren. Dazu kommt, dass die Implementierung wegen der genannten Punkte schwerer wartbar ist.

	MovieLens			*MovieTweeting*			*BookCrossing*		
	$F = 1$	$F = 20$	$F = 50$	$F = 1$	$F = 20$	$F = 50$	$F = 1$	$F = 20$	$F = 50$
Mahout	T=0, R=61	T=3, R=62	T=7, R=61	T=0, R=0	T=0, R=0	T=1, R=0	T=0, R=0	T=0, R=0	T=1, R=0
$RMSE$	0.9461	0.9234	0.9438	1.7270	1.7841	1.8278	1.7303	1.7845	1.8273
MAE	0.7491	0.7272	0.7406	1.2937	1.3388	1.3539	1.2984	1.3378	1.3534
Lenskit	T=0, R=0	T=0, R=0	T=0, R=1	T=0, R=1	T=0, R=1	T=0, R=1	T=0, R=1	T=0, R=1	T=0, R=4
$RMSE$	0.9454	0.9438	0.9467	1.7407	1.7481	1.7722	1.7407	1.3123	1.3278
MAE	0.7468	0.7436	0.7461	1.3103	1.3123	1.3278	1.3103	1.7481	1.7722

Legende: F=Featureanzahl, T=Trainingszeit (Sekunden), R=Empfehlungszeit (Sekunden)

Tabelle 4.2: Auswertung der Experimente anhand der Testdaten. (Größere Werte sind besser)

Berechnungen Für die Aufteilung von Trainings- und Testdaten wurde ein zufälliger 80/20 Split genommen. Die MF wurde insgesamt mit drei verschiedenen Werten für die zu lernenden Features (1, 20, 50) gewählt, um die Bedeutung dieser Variable festzustellen. Das Anlernen wurde dabei auf 50 Iterationen eingeschränkt. Die Ergebnisse dafür sieht man in Tabelle 4.2. Bei beiden Frameworks sieht man einen Trend zu einer höheren Genauigkeit bei einer größeren Anzahl an Features, welche eine bessere Anpassung an die Daten verspricht. Für weitere Experimente könnte man noch den Einflussfaktor der Lerniterationen untersuchen. Bei der eigens programmierten naiven Implementierung von MF konnten keine Empfehlungen innerhalb einer Stunde produziert werden. Das Anlernen des Algorithmus mit einer hohen Anzahl an Features und Iterationen in Verbindung mit einer sehr großen Matrix, die viele 0 Werte hat, bremste den ganzen Prozess aus. Lenskit war von allen Läufen am schnellsten sowohl beim Lernen als auch beim Empfehlen. Mahout war vergleichbar schnell, jedoch mit einem Ausreißer beim MovieLens Datensatz, wo die Berechnung der Empfehlungen eine Minute gedauert hat. Die berechneten Metriken konnten Werte liefern mit dem man beide Bibliotheken untersuchen konnte, jedoch ohne eine Erklärung für das bessere oder schlechtere Abschneiden.

5 Ausblick

Entwicklung In dieser Projektarbeit wurden die Erkenntnisse beim Aufbau und der Validierung einer Umgebung erläutert, die auf Matrix Factorization basierend Empfehlungen von Artikel erzeugen kann. Dafür wurden die Ergebnisse von anderen Veröffentlichungen auf bestimmte Gesichtspunkte untersucht. Die eingesetzte Pipeline ist zu diesem Zweck geeignet, stellt jedoch langfristige gesehen in der aktuellen Form keine wartbare Lösung dar. Dafür müsste man den derzeitigen Programmierstand weiter bereinigen und neue Schnittstellen für Empfehlungs-Systeme und Datenquellen einpflegen. Es bleiben darüber hinaus noch viele offene Fragen, so z.B welche Arten eignen sich am besten um weitere Algorithmen oder Systeme miteinander zu Verknüpfen. Die verwendeten Bibliotheken bieten keine vor-implementierten Konzepte dafür. Das Einbeziehen von weiteren Kontextelementen konnte nicht umgesetzt werden, weil die benötigten Daten fehlten oder die Frameworks das nicht unterstützten.

Metriken Die vorhandenen Metriken geben die Möglichkeit verschiedene Systeme untereinander zu Vergleichen. Bieten jedoch keine wirkliche Erklärung wieso eins bessere oder schlechtere Empfehlungen berechnet. Darüber hinaus fehlen diesen Metriken der Bezug zu einem realen System, wo viele andere Faktoren in eine gute Empfehlung mit rein spielen. Die Präsentation und der Kontext eines Nutzers sollten mit in die Berechnung einfließen. Diese weiteren Informationen lassen sich jedoch schlecht erfassen und werden in der Regel aus gesetzlichen Gründen[1] nicht weiter gegeben. Darüber hinaus kann man nur bedingt Online-Untersuchungen ohne eine Kooperation mit einem Unternehmen umsetzen.

Übertragbarkeit Es werden oft Empfehlungs-Systeme speziell für eine Plattform oder Domäne entwickelt. Es wäre von Interesse eine Untersuchung durchzuführen, inwieweit sich die Ergebnisse auf andere Domänen übertragen lassen. Dabei ist die leitende Fragen, ob man generell Empfehlungs-System für Filme, auch z.B. zur Erstellung von Vorschlägen für Rezepte, Musik oder Artikel wieder verwenden?

[1] Zum Beispiel zum Schutz der Anonymität.

Literaturverzeichnis

[Dooms u. a. 2013] Dooms, Simon ; De Pessemier, Toon ; Martens, Luc: Movietweetings: a movie rating dataset collected from twitter. In: *Workshop on Crowdsourcing and human computation for recommender systems, CrowdRec at RecSys* Bd. 2013, 2013, S. 43 3

[Ekstrand u. Riedl 2012] Ekstrand, Michael ; Riedl, John: When Recommenders Fail: Predicting Recommender Failure for Algorithm Selection and Combination. In: *Proceedings of the Sixth ACM Conference on Recommender Systems*. New York, NY, USA : ACM, 2012 (RecSys '12). – ISBN 978–1–4503–1270–7, 233–236 1

[Garcin u. a. 2014] Garcin, Florent ; Faltings, Boi ; Donatsch, Olivier ; Alazzawi, Ayar ; Bruttin, Christophe ; Huber, Amr: Offline and online evaluation of news recommender systems at swissinfo. ch. In: *Proceedings of the 8th ACM Conference on Recommender systems* ACM, 2014, S. 169–176 3

[Herlocker u. a. 2004] Herlocker, Jonathan L. ; Konstan, Joseph A. ; Terveen, Loren G. ; Riedl, John T.: Evaluating collaborative filtering recommender systems. In: *ACM Transactions on Information Systems (TOIS)* 22 (2004), Nr. 1, S. 5–53 4

[Koren u. a. 2009] Koren, Yehuda ; Bell, Robert ; Volinsky, Chris: Matrix factorization techniques for recommender systems. In: *Computer* (2009), Nr. 8, S. 30–37 1, 3, 6

[McNee u. a. 2006] McNee, Sean M. ; Riedl, John ; Konstan, Joseph A.: Being Accurate is Not Enough: How Accuracy Metrics Have Hurt Recommender Systems. In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2006 (CHI EA '06). – ISBN 1–59593–298–4, 1097–1101 4

[Paterek 2007] Paterek, Arkadiusz: Improving regularized singular value decomposition for collaborative filtering. In: *Proceedings of KDD cup and workshop* Bd. 2007, 2007, S. 5–8 6

[Rafailidis u. Nanopoulos 2014] Rafailidis, Dimitrios ; Nanopoulos, Alexandros: Modeling the Dynamics of User Preferences in Coupled Tensor Factorization. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. New York, NY, USA : ACM, 2014 (RecSys '14). – ISBN 978–1–4503–2668–1, 321–324 8

[Said u. Bellogín 2014] Said, Alan ; Bellogín, Alejandro: Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. New York, NY, USA : ACM, 2014 (RecSys '14). – ISBN 978–1–4503–2668–1, 129–136 2, 3

[Said u. Bellogín 2014] Said, Alan ; Bellogín, Alejandro: Rival: A toolkit to foster reproducibility in recommender system evaluation. In: *Proceedings of the 8th ACM Conference on Recommender systems* ACM, 2014, S. 371–372 5

[Said u. a. 2012] Said, Alan ; Tikk, Domonkos ; Shi, Yue ; Larson, Martha ; Stumpf, Klara ; Cremonesi, Paolo: Recommender systems evaluation: A 3d benchmark. In: *ACM RecSys 2012 Workshop on Recommendation Utility Evaluation: Beyond RMSE, Dublin, Ireland*, 2012, S. 21–23 4

[Weigandt 2015] Weigandt, Eduard: Visual Analytics: Personalisierung im E-Commerce. In: *Master Seminar* (2015) 1

[Ziegler u. a. 2005] Ziegler, Cai-Nicolas ; McNee, Sean M. ; Konstan, Joseph A. ; Lausen, Georg: Improving recommendation lists through topic diversification. In: *Proceedings of the 14th international conference on World Wide Web* ACM, 2005, S. 22–32 3