Recommender Systeme mit Collaborative Filtering

Proseminar Data Mining

René Romen
Fakultät für Informatik
Technische Universität München
Email: rene.romen@tum.de

***Kurzfassung*— Recommender Systeme geben Benutzern personalisierte Empfehlungen, um ihnen so die Wahl aus einer Menge von Items zu erleichtern. Diese Technologie ist vor allem im Internet, wo Onlineshops Millionen von Artikeln anbieten, weit verbreitet. Collaborative Filtering ist ein Begriff der zahlreiche Verfahren, mit denen ein Recommender System umgesetzt werden kann vereinigt. Alle gemeinsam haben sie, dass direkte oder indirekte Bewertungen von Benutzern über Items verwendet werden, um einem aktiven Benutzer eine personalisierte Empfehlung zu geben. In anderen Worten, helfen sich die Benutzer des Systems gegenseitig, indem sie Bewertungen der Items die sie betrachtet haben hinterlassen. Die Idee dahinter ist das die Benutzer sich in Gruppen aufteilen lassen die ähnliche Präferenzen für Items haben. Dieser Ansatz hat sich als erfolgreich erwiesen und sich in der Praxis gegenüber Contentbased Methoden, die versuchen die Items Inhaltlich zu vergleichen, durchgesetzt. In dieser Arbeit werden eine Reihe von einfachen, aber effektiven und weit verbreiteten Collaborative Filtering Algorithmen vorgestellt. Es wird besprochen wie mit ihrer Hilfe einzelne Items oder eine Liste von interessanten Items für einen Benutzer gefunden werden. Insbesondere werden User-based und Item-based Collaborative Filtering Ansätze besprochen und verglichen, die auf k-nearest neighbor Methoden basieren.**

***Schlüsselworte*— Recommender Systeme Collaborative Filtering User-based Item-based**

I. Einleitung

Collaborative Filtering sind Algorithmen die aus Präferenzen einer Gruppe von Usern personalisierte Empfehlungen für Items generieren. Sie sind weit verbreitet im Internet. Webseiten wie Amazon, Netflix und Spotify verwenden Recommender Systeme mit Collaborative Filtering, um ihren Usern zu helfen aus Millionen Artikeln, Filmen, Serien und Songs auszuwählen. Recommender Systeme kommen häufig Hand in Hand mit einer Suchmaschine und sollen dabei helfen dem User Items vorzuschlagen, die er alleine nicht oder nur mit Aufwand gefunden, hätte. Wie wichtig diese Recommender Systeme für das Unternehmen sind, kann man gut am Wettbewerb den Netflix 2006 ausgeschrieben hat sehen. Sie boten eine Million USD für eine Verbesserung ihres Recommender Systems um 10%. Netflix veröffentlichte dafür einen Datensatz mit 100 Millionen Bewertungen, die eine halbe Million User zu über 17.000 Filmen abgegeben haben. Teilnehmer mussten einen Collaborative Filtering Algorithmus finden der die Top-10 Empfehlung des Netflix Recommender Systems übertritt. Gewonnen hat den Preis das Team BellKor's Pragmatic Chaos [1], mit einer Ansammlung an Algorithmen, deren Vorhersagen zu einer endgültigen Vorhersage vereinigt wurden. In dieser Arbeit werden Collaborative Filtering Methoden vorgestellt die auf k-nearest neighbor basieren. Sie sind einfach zu implementieren und sind in der Lage personalisierte Empfehlungen zu machen. Sie sind in der Praxis wegen ihrer Einfachheit und Effizienz weit verbreitet.

A. Recommender Systeme

Recommender Systeme versuchen mittels Data Mining Items zu finden die einem bestimmten User gefallen. Diese Items werden dann dem User vorgeschlagen. Es gibt viele verschiedene Arten dies zu erreichen, aber sie lassen sich in zwei Ansätze aufteilen. Content based Recommender Systeme finden ähnliche Items, indem sie diese miteinander vergleichen. Dies kann zum Beispiel durch das Auftreten von gleichen Wörtern in Texten geschehen. Dieser naive Ansatz hat natürlich viele Probleme, weil man mit gleichen Worten verschiedenes ausdrücken und gleiches mit verschiedenen Worten sagen kann. Dies heißt nicht das Content based Ansätze ungeeignet sind, sie werden aber schnell sehr komplex. Die Alternative sind Recomender Systeme mit Collaborativen Filtering. Sie arbeiten nicht auf Items, sondern auf Präferenzen, die User Items direkt oder indirekt geben. Die Idee dabei ist das es Personengruppen gibt, die einen ähnlichen Geschmack, für die in Frage stehenden Items haben. Man kann also die User in Gruppen teilen und eine Empfehlung generieren, indem man einfach Items empfiehlt die anderen Usern aus derselben Gruppe gefallen haben. Einer der ersten Versuche User in ein Recommender System miteinzubeziehen war Tapestry [2]. User konnten ihre Reaktionen zu Dokumenten in Annotationen speichern. Das System nutzte diese, um bessere Informationsfilter zu bieten als Content based Filtering alleine.

B. Vorgestellte Methoden

In dieser Arbeit werden zwei Collaborative Filtering Methoden vorgestellt. Beide sind k-nearest neighbor Algorithmen, aber sie haben dennoch große Unterschiede. Die vorgestellte User-based Methode arbeitet direkt mit der User-Item Beziehung und liefert damit gute Ergebnisse, leidet aber unter Skalierbarkeitsproblemen. Diese sollen von Item-based Verfahren gelöst werden. Diese arbeiten auf einem Modell, so dass sie eine gute online Laufzeit erreichen. Der Item-basierte Algorithmus, der vorgestellt wird, ist aus der Arbeit von Sarwar [3].

C. Organisation

Kapitel 2 gibt eine Einführung in Collaborative Filtering und bespricht wie User Präferenzen aussehen und Vorschläge für Methoden, die man verwenden kann um Ähnlichkeiten zwischen Usern oder Items zu bestimmen. In Kapitel 3 wird User-based Collaboratives Filtering eingeführt und besprochen. Kapitel 2 und 3 stellen eine User-based Methode aus der Arbeit von Segaran [4] vor. Am Ende des Kapitels werden dessen Probleme besprochen. Der vorgestellte Im 4 Kapitel geht es um Item-based Collaborative Filtering, welches Probleme vom User-based Ansatz löst. Im letzten Kapitel werden die Ergebnisse aus der Arbeit von Sawar et al. [3] vorgestellt und besprochen.

II. Collaborative Filtering

Collaborative Filtering Algorithmen erhalten als Input eine User-Item Rating Matrix und geben als Output entweder eine Vorhersage, wie ein User ein bestimmtes Item bewerten würde oder eine Lister der besten n Items. Dabei sind Items die der User schon bewertet hat ausgeschlossen. Ein Collaborative Filtering System, wie in Abbildung 1, besteht aus einer Liste von m Usern $U = \{u_1, u_2, \ldots, u_m\}$ und n Items $I = \{i_1, i_2, \ldots, i_n\}$. Die Matrix welche im Element (p_{ij}) die Meinung vom i-ten User über das j-te Item hat nennen wir User-Item Matrix. Diese Matrix kann leere Einträge enthalten, ein User kann also auch keine Präferenz über ein Item abgeben. Des weiteren sei die Menge aller Items die der j-te User bewertet hat gegeben durch I_j. Es wird immer versucht einem bestimmten User Vorschläge zu machen, diesen nennt man aktiven User. Die Collaborative Filtering Algorithmen versuchen nun mit hilfe dieser Matrix als Input, eines der folgenden Probleme zu lösen:

1) Vorhersage: Der Algorithmus versucht eine Vorhersage eines leeren Matrixeintrages (p_{ij}) zu treffen. Der Wert ist eine Vorhersage wie sehr dem aktiven User u_i Item i_j gefallen wird.

2) Top-N Empfehlung: Der Algorithmus sucht für den aktiven User u_i, eine geordnete n-elementige Liste an Items die dem User am wahrscheinlichsten gefallen. Items für die der User bereits eine Meinung abgegeben hat dürfen nicht enthalten sein.

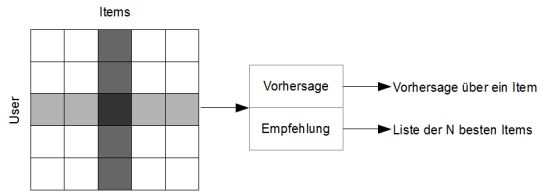

Abbildung 1. Collaborative Filtering prozess

Die Meinungen von Nutzern können direkt oder indirekt gesammelt werden und auf Nummern abgebildet werden. Wie diese genau aussehen hängt von der Anwendung ab. Beispiele für direkte Meinungen sind Bewertungen von Items mit 1 bis 5 Sterne oder ein Like/Dislike System. Im ersten Fall können die Sterne direkt auf die Nummern 1 bis 5 abgebildet werden. Im Zweiten kann man -1 für Dislike, 0 für keine abstimmung und 1 für Like wählen. Wichtig ist das 0 Werte nur eingetragen werden wenn User das Item angeschaut oder gekauft haben, aber keine Bewertung hinterlassen haben. Indirekte Meinungen können zum Beispiel durch Surf- oder Kaufverhalten von Usern gesammelt werden. So kann man eine 0 setzen, wenn ein User einen Artikel im Onlineshop betrachtet und eine 1, wenn er ihn kauft. Um die vorgestellten Algorithmen kurz vorzuzeigen wird die User-Item Matrix aus Tabelle I verwendet. In dieser sind die Bewertungen von 4 Usern für 5 Items mit Ratings von 1 bis 5 zu finden. Wir möchten User u_4's Bewertung für Item i_2 vorhersagen.

Tabelle I
Beispiel User-Item Matrix

/	i_1	i_2	i_3	i_4	i_5
u_1	1	5		5	2
u_2	4	3	4		1
u_3		2	4	5	5
u_4	2	?	3	4	3

Collaborative Filtering Algorithmen lassen sich in zwei Kategorien aufteilen. User-based oder auch Memory-based Collaborative Filtering arbeitet auf der ganzen User-Item Matrix um eine Vorhersage, b.z.w. Top-N Empfehlung zu machen. Dies bedeutet, dass die gesamte Matrix im Hauptspeicher sein muss, weil ständig auf ihr zugegriffen wird. Diese Algorithmen versuchen ähnliche User zu finden, um ihre Bewertungen von Items zu Vorhersagen zusammenzufassen. Item-based oder Modell-based Collaborative Filtering baut aus der User-Item Matrix ein Modell. Wenn jetzt eine Vorhersage oder Top-N Empfehlung getroffen werden soll, wird nur noch das Modell zur Hilfe gezogen.

A. Ähnlichkeitsfunktionen

Im Folgenden sollen einige Möglichkeiten vorgestellt werden um die Ähnlichkeit zwischen Usern oder Items zu messen. Gemeinsam haben alle Methoden, dass sie sowohl User, als auch Items, durch tauschen von Items und Usern in der User-Item Matrix, vergleichen können. Im Weiteren werden aus Gründen der Einfachheit immer User verglichen, aber das Vergleichen von Items läuft analog. Sind sich zwei User ähnlich, dann soll ihr Ähnlichkeitswert relativ zu den anderen groß sein und sind sie sich nicht ähnlich sollen sie einen niedrigen Ähnlichkeitswert erhalten. Die hier vorgestellten Funktionen trifft man in der Literatur häufig an, sie können aber durch für die Anwendung angemessenere Methoden ersetzt werden. Sei die Ähnlichkeit von User i und j bezeichnet als die Ähnlichkeitsfunktion $sim\,(i, j)$.

1) Euklidische Distanz: Eine Ähnlichkeitsfunktion ist die Distanz der Vektoren von User i und j. Dabei werden nur Items berücksichtigt die beide User bewertet haben, also in $S_{ij} := I_i \cap I_j$ liegen. Damit das ganze den Anforderungen an eine Ähnlichkeitsfunktion entspricht, addieren wir 1 und

bilden den Kehrwert, wie in (1) zu sehen ist. Haben zwei User keine gemeinsamen Items geben wir ihnen die Ähnlichkeit 0.

$$sim(i,j) = \frac{1}{1 + \sum\limits_{x \in S_{ij}} \sqrt{p_{ix}^2 - p_{jx}^2}} \tag{1}$$

2) Pearson-Korrelation: Die Pearson-Korrelation (2) ist ein Maß dafür wie gut zwei Datensätze auf eine Linie passen. Er liefert bessere Ergebnisse als die Euklidische Distanz, wenn die Daten nicht gut normalisiert sind. Wenn zum Beispiel zwei User ähnliche Bewertungen haben, aber einer immer eine konstant niedrigere Bewertung vergibt, liefert die Pearson-Korrelation bessere Ergebnisse als Euklidische Distanz.

$$sim(i,j) = \frac{\sum\limits_{x \in S_{ij}} (p_{ix} - \overline{p}_x)(p_{jx} - \overline{p}_x)}{\sqrt{\sum\limits_{x \in S_{ij}} (p_{ix} - \overline{p}_{ix})^2} \sqrt{\sum\limits_{x \in S_{ij}} (p_x - \overline{p}_x)^2}} \tag{2}$$

Hier ist mit $\overline{p}_x$ das arithmetische Mittel aller Bewertungen für das Item x bezeichnet.

3) Cosinus Ähnlichkeit: Die Cosinus Ähnlichkeit ist der Winkel zwischen der Matrixzeile des i-ten und des j-ten Users. In der Formel 3 beschreibt $\overrightarrow{i}$ und $\overrightarrow{j}$ die User Vektoren und $\cdot$ das Skalarprodukt von zwei Vektoren.

$$sim(i,j) = cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \cdot \overrightarrow{j}}{||\overrightarrow{i}||_2 * ||\overrightarrow{j}||_2} \tag{3}$$

III. User-based Collaborative Filtering

In diesem Kapitel wird ein User-based Collaborative Filtering Algorithmus vorgestellt und anschließend werden die schwächen von User-based Ansätzen besprochen. Allgemein arbeiten User-Based verfahren auf der User-Item Matrix, um dort erst nach einer Gruppe von Usern zu suchen, die dem aktiven User ähnlich sind. Dann werden deren Bewertungen von Items herangezogen um eine Empfehlung zu berechnen. Der hier vorgestellte Algorithmus nutzt erst eine der Änlichkeitsfunktionen aus Kapitel 2, um k ähnliche User zu finden und ein Gewichtetes arithmetisches Mittel ihrer Meinungen zu bilden.

A. Änliche Nutzer finden

Im ersten Schritt müssen k ähnliche User gefunden werden. Dazu wird der aktive User u_k mit jedem Nutzer $u_i \in U$ mit $i \neq k$ verglichen. Dazu kann man eine der Ähnlichkeitsfunktionen aus Kapitel 2 verwenden. Es kann sich hier auszahlen eine reihe an Methoden zu probieren und die beste auszuwählen. Mithilfe der berechneten Ähnlichkeitswerte können die User nach ihrer Ähnlichkeit zum aktiven User angeordnet werden und aus dieser Rangliste die besten k als Ergebnis genommen werden. Die Tabelle II und III zeigen die Ähnlichkeitswerte zwischen u_4 und allen anderen Usern mit Euklidischer Distanz b.z.w. Pearson-Korrelation.

Tabelle II
Beispiel Ähnlichkeit zwischen u_4 und allen anderen Usern mit Euklidischer Distanz

/	u_1	u_2	u_3
u_4	0.25	0.14	0.1

Tabelle III
Beispiel Ähnlichkeit zwischen u_4 und allen anderen Usern mit Pearson-Korrelation

/	u_1	u_2	u_3
u_4	0.96	-0.5	0.5

B. Items empfehlen mit Gewichteten arithmetischen Mittel

Die Bewertungen der ähnlichen Nutzer werden nun zu Vorschlägen für den aktiven User kombiniert. Dies soll mit dem Gewichteten arithmetischen Mittel geschehen. Dazu summiert man für jedes Item das der aktive Nutzer noch nicht kennt, die Bewertungen aller ähnlichen Nutzer gewichtet mit ihrer Ähnlichkeit auf und dividiert durch die summe aller Ähnlichkeiten, die mit eingeflossen sind. Um eine Empfehlung für den i-ten User über das j-te Item zu berechnen, gegeben S die Liste von ähnlichen Usern und s_i die Ähnlichkeit vom i-ten User mit dem aktiven User, reicht Formel (4).

$$p_{ij} = \frac{\sum\limits_{x \in S} s_x p_{xj}}{\sum\limits_{x \in S} s_x} \tag{4}$$

Für eine Top-N Empfehlung können wir nun mit (4) Vorhersagen für alle Items machen, diese in eine Rangliste bringen und die besten n auswählen.

Im Beispiel ergibt sich, als vorhersage für das Item i_2 von u_4, der Wert 3.97 mit Pearson-Korrelation und 3.72 mit Euklidischer Distanz. Die Rechnungen dafür sind in den Tabellen IV und V gezeigt. Bei Tabelle V ist zu beachten, dass der User u_3 nicht ins Rating mit einfließt, weil negative Ähnlichkeiten mit Gewichteten arithmetischen Mittel keinen Sinn ergeben.

Tabelle IV
Beispiel Vorhersage für Item i_2 mit Euklidischer Distanz

User	Ähnlichkeit zu u_4	Wertung	Ähnlichkeit * Wertung
u_1	0.25	5	1.25
u_2	0.14	3	0.42
u_3	0.1	2	0.2
Summe	0.49		1.87
Ergebnis			3.82

Tabelle V
Beispiel Vorhersage für Item i_2 mit Pearson-Korrelation.

User	Ähnlichkeit zu u_4	Wertung	Ähnlichkeit * Wertung
u_1	0.96	5	4.8
u_2	-	3	-
u_3	0.5	2	1
Summe	1.46		5.8
Ergebnis			3.97

C. Probleme von User-based Methoden

User-based Methoden teilen sich eine Reihe von Problemen, die damit zusammenhängen, dass sie auf der ganzen User-Item Matrix arbeiten. Zum einen haben sie Probleme mit dünnbesetzten Matrizen. Es ist realistisch davon auszugehen das die User-Item Matrix in der Praxis dünnbesetzt ist. Viele große Systeme, man denke an online Dienste wie Amazon, Spotify oder Netflix. Diese haben sowohl Millionen von Usern sowie Items. Auch sehr aktive User werden nur eine sehr kleine Teilmenge aller Items bewerten. Dadurch kann es vorkommen, dass für einen User gar keine Empfehlungen gefunden werden kann. Darunter leidet die Präzision des Recommender Systems. Das zweite große Problem ist die Skalierbarkeit eines User-Based Systems. Die Laufzeit des Algorithmus wächst mit der Anzahl der User und Items. Große Recommender Systeme können Empfehlungen nicht mehr in annehmbarer Zeit berechnen.

IV. Item-based Collaborative Filtering

Item-Based Collaborative Filtering versucht die Probleme von User-Based verfahren zu lösen, indem nicht direkt mit der User-Item Matrix gearbeitet wird, sondern aus ihr ein Model erstellt wird. Dies kann natürlich auf sehr viele verschiedene Arten geschehen. In diesem Kapitel wird eine Variante vorgestellt, bei der aus der User-Item Matrix eine Item-Item Beziehung aufgebaut wird. Dieses Modell hält für jedes Item k ähnliche Items. Das Erstellen dieser Item-Item Beziehung ist natürlich rechenaufwendig, aber sobald sie fertig ist, kann sie immer wieder herangezogen werden, um Empfehlungen zu generieren. Dies hat zur Folge, dass die Item-Item Matrix vor dem Einsatz generiert werden kann und der Algorithmus eine gute online Performance hat. Während der Laufzeit einer Anfrage für eine Empfehlung, wird die User-Item Matrix nicht mehr benötigt. Die Item-Item Beziehung muss regelmäßig erneuert werden, um das Modell aktuell zu halten. Dies kann aber auf einer eigenen Maschine geschehen und stört so nicht das laufende Empfehlungssystem. Dadurch sind Item-based Ansätze deutlich skalierungsfähiger als User-based Methoden. Auch Item-based Ansätze können Probleme haben mit dünnbesetzten User-Item Matrizen, aber meist liefern sie bessere Ergebnisse als User-based Methoden.

A. Die Item-Item Beziehung

Die Ähnlichkeiten zwischen Items berechnet man gleich wie die Ähnlichkeit zwischen Usern. Wieder kann man sich Methoden aus Kapitel 2 bedienen. Die Anzahl an ähnlichen Items die wir für jedes Item suchen, die Nachbarschaftsgröße, hat Einfluss auf die Qualität der Ergebnisse. Für das kleine Beispiel kann man die Anzahl der betrachteten Items gleich der Anzahl an Items setzen und jedes mit jedem vergleichen, wie es in der Tabelle VI gezeigt ist.

B. Items empfehlen

Das Modell wird nun herangezogen um Vorhersagen und Top-N Empfehlungen zu berechnen. Dafür werden hier zwei Methoden vorgestellt.

Tabelle VI
Beispiel Item-Item Modell mit Pearson-Korrelation

/	1.		2.		3.		4.	
i_1	i_3	0.5	i_5	0.08	i_2	0.06	i_4	0.05
i_2	i_3	0.17	i_4	0.1	i_1	0.05	i_5	0.04
i_3	i_1	0.5	i_4	0.33	i_2	0.17	i_5	0.09
i_4	i_3	0.33	i_2	0.1	i_5	0.09	i_1	0.05
i_5	i_4	0.09	i_3	0.09	i_1	0.08	i_2	0.04

1) Gewichtete Summe: Man kann auch hier wieder ein Gewichtetes arithmetisches Mittel bilden, um eine Empfehlung zu berechnen. Die Vorhersage (5) wie gut dem aktiven User u das Item i gefällt wird berechnet mithilfe der i ähnlichen Items R_i. Es werden alle Bewertungen p_{ui} die u einem ähnlichen Item x gegeben hat multipliziert mit der Ähnlichkeit s_{ix} zu diesem, aufsummiert und durch die Summe aller Ähnlichkeiten geteilt. In Tabelle VII ist die Rechnung für das Beispiel, es wird 3.13 als Bewertung für i_2 von u_4 vorhergesagt.

$$p_{ji} = \frac{\sum\limits_{x \in R} p_{jx} s_{ix}}{\sum\limits_{x \in R} s_{ix}} \tag{5}$$

Tabelle VII
Beispiel Vorhersage für Item i_2

Item	Ähnlichkeit zu i_2	Wertung von u_4	Ähnlichkeit * Wertung
i_1	0.05	2	0.1
i_3	0.17	3	0.51
i_4	0.1	4	0.4
i_5	0.04	3	0.12
Summe	0.36		1.13
	Ergebnis		3.13

2) Regression: Dieser Ansatz ist ähnlich dem Gewichteten arithmetischen Mittel. Der einzige Unterschied besteht darin nicht direkt mit den Bewertungen änlicher Items zu arbeiten, sondern sie mit einem Regression Modell zu approximieren. Ein Beispiel wie dieses Modell aussehen kann ist (6). Dabei ist mit R_i der Rating Vektor der i ähnlichen Items gemeint und mit R_i' der neue approximierte Vektor. Die Idee dahinter ist, dass diese Bewertungsvektoren im Euklidischen Sinn weit voneinander entfernt sein können, wodurch sie eine niedere Ähnlichkeit bekommen, aber eigentlich sehr ähnlich sind. Daher können diese Werte große Fehler in die Berechnung bringen. Die lineare Regression versucht dies zu verhindern. Die Parameter α und β sind die Regression Parameter und ϵ ist der entstandene Fehler.

$$R_i' = \alpha R_i + \beta + \epsilon \tag{6}$$

V. Vergleich von User-based und Item-based Collaborative Filtering

Die Arbeit von Sarwar [3] untersucht insbesondere die Leistungsunterschiede zwischen User-based und Item-based Ansätzen. Ihre Ergebnisse werden in diesem Kapitel vorgestellt. Der Flaschenhals bei User-based verfahren ist das

Suchen von ähnlichen Usern, weil die Berechnungen der User-User Ähnlichkeiten zu viel Zeit in Anspruch nehmen. Der Modellbasierte Ansatz von Item-based Methoden soll dies umgehen, indem die aufwendigen Berechnungen von ähnlichen Usern und die Berechnung der Empfehlungen getrennt werden. Dadurch kann eine gute online Performance erreicht werden. Des Weiteren werden die Item-Item Beziehungen für das Modell berechnet anstatt der User-User Beziehungen, weil diese sich seltener oder langsamer ändern. Um eine komplette Item-Item Beziehungsmatrix zu erstellen, muss jedes Item mit jedem verglichen werden, dadurch braucht das Ergebnis $\mathcal{O}(n^2)$ an Speicherplatz für n Items. Online können die Ähnlichkeiten dadurch dann aber einfach aus der Matrix abgelesen werden. Weil es aber nur einen Teil der ähnlichen Items braucht, um gute Empfehlungen zu generieren, wählt man einen Parameter k, der sehr viel kleiner sein kann als n und nimmt für jedes Item nur die k ähnlichsten Items in das Modell auf. Die Wahl von k entscheidet über die Geschwindigkeit des Algorithmus und über die Qualität der Empfehlungen. Je mehr Items miteinbezogen werden, umso genauer sollten die Empfehlungen sein, aber umso länger braucht die Berechnung dieser.

A. Die User-Item Matrix

Die Experimente wurden auf 100.000 Ratings aus dem MovieLens Recommender System durchgeführt. Dabei wurden 943 User die alle mindestens 20 der 1682 Filme bewertet haben ausgewählt. Die Matrix ist dünnbesetzt da 93,7% der Einträge 0 sind. Dieser Datensatz wurde dann nochmal aufgeteilt in Trainingsdaten und Testdaten. Dafür wurde eine variable x genommen, die aussagt, wie viele Prozent der Daten im Trainingsset sind. Es wurden Werte zwischen $x = 0.9$, also 90% und $x = 0.2$ in Betracht gezogen.

B. Bewertungsmetrik

Um die Qualität der Bewertungen zu unterscheiden wurde der mittlere absolute Fehler (MAE) verwendet. Dieser berechnet den Durchschnitt aus dem absoluten Unterschied zwischen dem vorhergesagten Wert des Systems und der tatsächlichen Bewertung des Users. Die N Paare (p_i, q_i) die diese Werte für die i-te Bewertung in den Testdaten beschreiben werden von (7) verwendet, um den MAE zu berechnen.

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \tag{7}$$

C. Ergebnisse

Die Wahl der Größe von x also der Größe des Trainingssets fällt auf 0.8. Das Regression Modell ist dabei besser mit niederen x Werten, während das Gewichtete arithmetische Mittel mit steigendem x besser wird. Als Nachbarschaftsgröße k stellt sich 30 als Optimum heraus. Das Regression Modell wurde mit steigender Nachbarschaftsgröße schlechter und das Gewichtete arithmetische Mittel wurde nach 30 nur noch sehr langsam besser. In Bild 2 kann man nun die Ergebnisse für die Qualität sehen für $x = 0.8$ und in 3 für eine

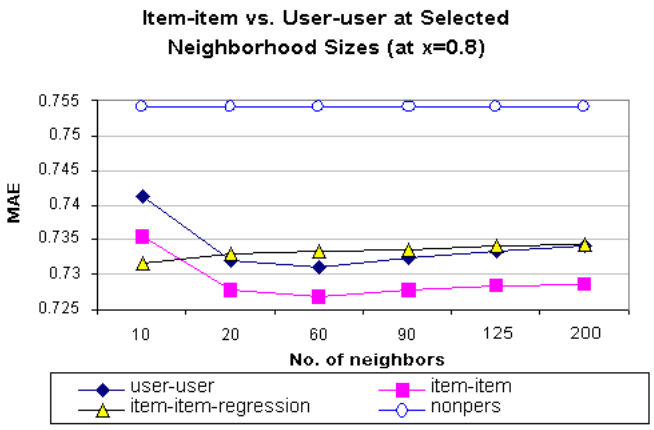

Abbildung 2. Vergleich von User-based und Item-based Collaborative Filtering in den Experiment von Sawar [3]

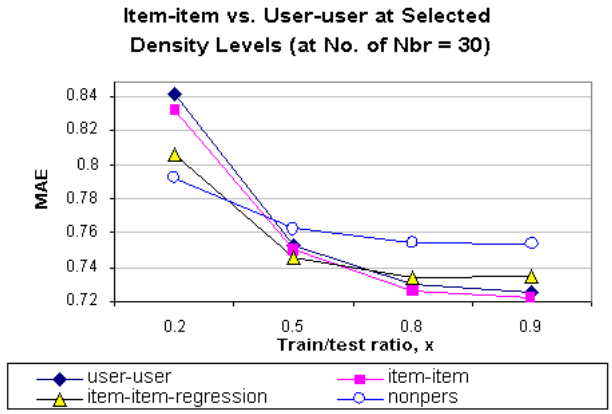

Abbildung 3. Vergleich von User-based und Item-based Collaborative Filtering in den Experiment von Sawar [3]

Nachtbarschaftsgröße von 30. Der Item-Item Algorithmus ist immer besser als der User-based Algorithmus. Das Regression Modell ist besser als die anderen bei niedrigen x und Nachbarschaftsgrößen, wird dann aber schlechter. Nonpers ist ein naiver nicht personalisierter Algorithmus [5]. Zusammenfassend sind Item-based Algorithmen von besserer Qualität als User-based Algorithmen, allerdings ist der Vorsprung nur gering. Des Weiteren ist das Regression Modell besser mit sehr dünn besetzten Matrizen und mehr Daten machen das Modell schlechter. Wahrscheinlich da das Modell wegen der zusätzlichen Daten zu stark angepasst wird. Die Item-Item Beziehung hat sich als relativ statisch herausgestellt und kann vorberechnet werden. Dadurch ergeben sich sehr gute online Geschwindigkeiten des Item-based Algorithmus. Des weiteren reichen relativ wenige ähnliche Items pro Item aus um gute Empfehlungen zu geben.

VI. Zusammenfassung und Ausblick

Die Entscheidung Item-based gegen User-based hängt von dem Anwendungsbereich ab. Wenn die User-Item Matrix nicht

dünn besetzt und nicht zu groß ist, kann man zu User-based Methoden greifen, weil sie einfacher sind und nicht gepflegt werden müssen. Auch wenn sich Item-Item Beziehungen häufig ändern ist User-based die bessere Wahl, wenn zum Beispiel im Extremfall die Item-Item Beziehungen jedes Mal neu berechnet werden muss. Ansonsten bringt Item-based viele Vorteile. Von Geschwindigkeit, Skalierbarkeit bis zur höheren Qualität der Ergebnisse. Modernere Forschung auf dem Gebiet von Recommender Systemen geht häufig auf Bedürfnisse der User ein. Dabei geht es um Privacy, Originalität der Vorschläge und Kontrolle des Users über den Vorgang. In großen Systemen in der Praxis wie man am Beispiel von Netflix [1] sieht werden keine Mühen gescheut, um ein möglichst präzises System zu bauen, welches Usern Inhalte vermittelt die sie auch wirklich sehen wollen.

LITERATUR

[1] A. Töscher, M. Jahrer, and R. M. Bell, "The bigchaos solution to the netflix grand prize," 2009.

[2] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, pp. 61–70, 1992.

[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *PROC. 10TH INTERNATIONAL CONFERENCE ON THE WORLD WIDE WEB*, 2001, pp. 285–295.

[4] T. Segaran, *Programming collective intelligence: building smart web 2.0 applications*. O'Reilly Media, 2007, ch. 2, pp. p. 7–28.

[5] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '99. New York, NY, USA: ACM, 1999, pp. 230–237. [Online]. Available: http://doi.acm.org/10.1145/312624.312682