Realisierung eines dezentralen Recommender Systems für PDAs

Henrik Mühe, Wolfgang Wörndl, Georg Groh

Lehrstuhl für Angewandte Informatik / Kooperative Systeme
Technische Universität München
Boltzmannstr. 3
85748 Garching
{muehe, woerndl, grohg}@in.tum.de

Abstract: Dezentrale Recommender Systeme erscheinen in mobilen Szenarien sinnvoll, wurden allerdings bisher noch kaum untersucht und erprobt. Wir haben daher in diesem Projekt einen Ansatz zur Empfehlung von Gegenständen (Items) auf Personal Digital Assistants (PDAs) realisiert, der ein gemeinsames Display zusätzlich zu den PDAs einbindet. Dabei werden Bewertungsvektoren zwischen den PDAs ausgetauscht, lokale Matrizen der Item-Ähnlichkeit errechnet und zur Empfehlung ausgewertet. Unsere Neuerungen gegenüber existierenden Ansätzen betreffen insbesondere die Erweiterbarkeit des Modells und die Optimierung des auf einem mobilen Gerät erforderlichen Speicherbedarfs. Die Tauglichkeit unseres Ansatzes wurde in einem kleinen Anwendertest evaluiert. Des Weiteren wurde die Skalierbarkeit mit einem Standard-Datensatz im Bereich Recommender Systeme nachgewiesen.

1 Einführung

Recommender Systeme sind ein mittlerweile etablierter Forschungsbereich der Informatik, der auch erfolgreiche Verwendung in kommerziellen Anwendungen wie etwa bei Amazon oder anderen Online-Shops gefunden hat. Das Ziel dabei ist es, Empfehlungen für Produkte wie Bücher oder andere Gegenstände (engl. *items*) – oft auf Basis von explizit abgegebenen Bewertungen – zu generieren. Zumeist laufen diese Recommender Systeme zentralisiert auf einem Server ab. Eine dezentrale Speicherung und Auswertung von Daten ohne zentralen Server verspricht jedoch einige Vorteile. Dezentrale Recommender können auch ohne Verbindung zu einem Server eingesetzt werden und verbessern somit die Portabilität des Systems [MKR04]. Empfehlungen können lokal abgegeben werden, indem Benutzer, die sich in der Nähe aufhalten, einbezogen werden. Des Weiteren wird eine Dezentralität von Recommender Systemen in der Literatur als Möglichkeit angesehen, die Kontrolle eines Benutzers über seine persönlichen Daten, wie etwa Bewertungen, und damit seine Privacy (Privatheit, Datenschutz), zu verbessern (siehe z.B. [Ko07]). Dezentrale Recommender Systeme wurde bisher jedoch vergleichsweise wenig untersucht oder in Anwendungsszenarien getestet.

Das hier betrachtete Szenario besteht aus Benutzern, die sich Informationen auf ihren privaten mobilen Endgeräten, wie z.B. einem Personal Digital Assistant (PDA), empfehlen lassen wollen. Items sollen dazu auf den PDAs personalisiert verwaltet und angezeigt werden. Ein Beispiel wären Messebesucher, die auf ihrem PDA Informationen zu einer Messe, wie z.B. verfügbare Ausstellungsstände, Produkte und Nachrichten bewerten und abfragen wollen. Zusätzlich sollen in unserem System als Neuerung gemeinsame Displays eingebunden werden die öffentlich für eine Gruppe von Personen – die zu einem Zeitpunkt anwesenden Benutzer – Informationen darstellen.

Das Ziel dieser Arbeit ist es, ein Konzept für die Umsetzung dieses Szenarios zu erarbeiten, zu implementieren und zu testen. Als konkretes Anwendungsbeispiel dient die Bewertung und Empfehlung von Bildern, wobei das Konzept auch auf andere Itemkategorien übertragen werden kann. Der Rest dieses Beitrags ist wie folgt organisiert: Zunächst erläutern wir einige Grundlagen zu Recommender Systemen und stellen einen existierenden Ansatz aus der Literatur im Bereich dezentrale Recommender vor. Im folgenden Kapitel 3 werden dann der Entwurf und die Implementierung unseres Systems mit Benutzerschnittstelle beschrieben. In Kapitel 4 diskutieren wir die Evaluierung des Systems, die in Form eines kleinen Anwendertests sowie einer Untersuchung zur Skalierbarkeit durchgeführt wurde. Dabei gehen wir auch auf verwandte Arbeiten ein. Unser Beitrag schließt mit einer kurzen Zusammenfassung und einem Ausblick.

2 Dezentrale Recommender Systeme

Wir werden in diesem 2. Kapitel zunächst eine Einführung in Recommender Systeme allgemein geben, und einen für die Realisierung auf PDAs geeignet erscheinenden Ansatz genauer betrachten. Dieser Ansatz dient dann als Ausgangspunkt des eigenen Systems.

2.1 Recommender Systeme und kollaboratives Filtern

Die Grundidee von Recommender Systemen ist es, Items wie z.B. Bücher oder CDs für einen aktiven Benutzer zu empfehlen. Dazu wird auf Basis von Informationen über den Benutzer, die Items und ggf. weiterer Daten berechnet, inwieweit ein Item dem Geschmack des Benutzers entspricht. Zu den Merkmalen von Empfehlungsalgorithmen gehören neben der Empfehlungsqualität u.a. Speicher- und Laufzeitkomplexität, sowie Anonymität und Erweiterbarkeit des Modells.

Grundsätzlich unterscheidet man individuelle und kollaborative Recommender Systeme. Individuelle Systeme ermitteln auf Basis des Profils des aktiven Benutzers empfehlenswerte Items. Oftmals geschieht dies mit Hilfe von Metadaten, die die Items beschreiben – daher werden diese Systeme auch inhaltsbasierte Recommender genannt. Zur Realisierung kann z.B. ein Regel-basiertes System eingesetzt werden, das auf Basis von explizit ausgewählten oder implizit erlernten Benutzerprofilattributen Items empfiehlt.

Die zweite Recommender Kategorie ist kollaboratives Filtern (engl. *collaborative filtering*, CF). Dazu werden bei der Empfehlung auch andere Benutzer berücksichtigt. Dies erfolgt mit Hilfe von Bewertungen (engl. *rating*, R) der Benutzer für Items, z.B. auf einer Skala von 1 bis 5. Als Bewertungsvektor wird dabei der Vektor aller Bewertungen eines Benutzers bezeichnet. Bei kollaborativen Filtern gibt es wieder zwei Varianten, nämlich speicherbasiertes (engl. auch *User-based CF* genannt) oder modellbasiertes (engl. auch als *Item-based CF* bezeichnet) kollaboratives Filtern.

Der Empfehlungsprozess bei speicherbasiertem kollaborativen Filtern besteht grundsätzlich aus zwei Schritten: 1. Berechnung einer Menge von k Benutzern, die in der Vergangenheit ähnlich wie der aktive Benutzer bewertet haben (Nachbarschaft), 2. Empfehlung von (neuen) Items für den aktiven Benutzer. Zur Nachbarschaftsberechnung im 1. Schritt wird der Bewertungsvektor des aktiven Benutzers mit denen der anderen verglichen. Dazu wurden in der Literatur verschiedenen Metriken vorgeschlagen, z.B. Euklidische Distanz oder Pearson-Spearman Korrelation [AT05]. Dieses Verfahren wertet somit die User-Item Matrix der Bewertungen aus (vgl. Abbildung 1, links). Im 2. Schritt werden dann Items ausgewählt, die der aktive Benutzer noch nicht beurteilt hat, die aber in seiner Nachbarschaft schon als empfehlenswert bewertet wurden.

	$item_1$	$item_2$	⋯	$item_i$
$user_1$	R	R	⋯	R
$user_2$	R	-	⋯	-
⋮	⋮	⋮	⋱	⋮
$user_{u-1}$	-	-	⋯	R
$user_u$	-	R	⋯	R

	$item_1$	$item_2$	⋯	$item_i$
$item_1$	1		⋯	
$item_2$	S	1	⋯	
⋮	⋮	⋮	⋱	
$item_i$	S	S	⋯	1

Abbildung 1: User-Item und Item-Item Matrix

Bei modellbasiertem CF wird nicht die Ähnlichkeit der Benutzer betrachtet, sondern die der Items [SKKR01]. Die User-Item Matrix wird also nicht zeilen-, sondern spaltenweise ausgewertet. Ein prinzipieller Unterschied ist, dass es zunächst unerheblich ist, für welchen Benutzer eine Empfehlung erfolgen soll, und somit die Berechnung im Voraus erfolgen kann. Als „Modell“ dieses Verfahrens ergibt sich eine Item-Item Matrix, wobei ein Element $S_{i,j}$ dieser Matrix die aus den Bewertungen berechnete Ähnlichkeit zwischen Item i und Item j ausdrückt (vgl. Abbildung 1, rechts). Für die eigentliche Empfehlung wird dann schließlich der Bewertungsvektor des aktiven Benutzers herangezogen, und Items vorgeschlagen, die ähnlich zu in der Vergangenheit positiv bewerteten Items sind.

Zu beachten ist, dass modellbasiertes kollaboratives Filtern wenig mit individuellem, inhaltsbasiertem Filtern gemein hat, da die Item-Ähnlichkeit nur auf Basis der Bewertungen der Benutzer errechnet wird – Metadaten von Items spielen keine Rolle. Modellbasiertes CF hat gegenüber Speicher-basiertem CF den Vorteil, dass die Berechnung der Empfehlungen weitaus weniger Rechenkapazität erfordert, weil die Item-Item Matrix als „Zwischenspeicher" der Item-Ähnlichkeit unabhängig vom aktiven Benutzer berechnet werden kann. Außerdem muss zum Zeitpunkt der Empfehlung nur der Bewertungsvektor des aktiven Benutzers vorliegen, was Vorteile hinsichtlich des Datenschutzes der persönlichen Informationen wie Bewertungen verspricht. Deshalb ist modellbasiertes CF für eine dezentrale Realisierung auf PDAs besonders gut geeignet.

2.2 PocketLens

PocketLens [MKR04] ist ein Ansatz eines dezentralen Recommender Systems aus der Literatur, der als Basis unseres Systems dient. Die grundsätzliche Motivation der Entwicklung von PocketLens liegt in der Tatsache begründet, dass sich die meisten Charakteristika eines modellbasierten Recommenders zwar bereits gut für den dezentralen Einsatz eigenen, sich jedoch auch einige Probleme aus der Nutzung einer Item-Item Matrix ergeben. Hierbei ist es insbesondere erforderlich, die Matrix, welche in ihren Feldern die Ähnlichkeit jeweils zweier Items enthält, neu zu generieren, sobald eine neue Bewertung hinzugefügt wird.

Bei diesem Vorgang ist, sofern lediglich der errechnete Ähnlichkeitswert in einem Feld gespeichert wird, eine erneute Betrachtung aller für die Item-Item Ähnlichkeit entscheidender Bewertungen notwendig. Dies verhindert jedoch einen umfassenden Datenschutz, da alle Bewertungsvektoren für den Fall der Neugenerierung des Datenmodells gespeichert werden müssten. Hinzu kommt, dass ein vollständiges Verwerfen der bislang berechneten Distanz bei einer Erweiterung des Modells um lediglich eine Bewertung, die Anforderungen an die genutzte Hardware erhöht.

PocketLens [MKR04] verwendet nun einen modifizierten Algorithmus zur Ermittlung der Ähnlichkeit zweier Items, bei dem Bewertungsvektoren hinzugefügt werden können, ohne dass alle betroffenen Felder der Item-Item Matrix vollständig neu berechnet werden müssen. Hierzu wird in jedem Matrixfeld nicht mehr nur die ermittelte Ähnlichkeit zweier Items gespeichert, sondern es werden die zur Berechnung notwendigen Zwischenergebnisse abgelegt. Aufgrund der Verwendung des Cosinus-Abstandsmaßes speichert PocketLens für jede Kombination zweier Items das Kreuzprodukt ihrer Bewertungsvektoren sowie deren Länge separat. Hierdurch ist es nun möglich, die Matrix um Bewertungen neuer Nutzer zu erweitern, indem ihre Bewertungen lediglich zum zwischengespeicherten Kreuzprodukt und deren Vektorlängen addiert werden, ohne dass das gesamte Datenmodell unter Nutzung aller bisher verwendeten Bewertungsvektoren neu generiert werden muss.

Zusätzlich wird bei PocketLens keine Item-Item Matrix generiert, die eine Empfehlung für beliebige Nutzer erlaubt, sondern lediglich ein Ausschnitt dieser Matrix, der auf einen Anwender fokussiert ist. Da der Algorithmus für die Nutzung auf mobilen Geräten mit zumeist nur einem festen Nutzer konzipiert ist, werden Zeilen und Spalten, die für die Empfehlung eines Items an diesen einzelnen Nutzer unerheblich sind, aus der Matrix entfernt, was die Speicherkomplexität verbessert.

Neben dem geringeren Aufwand bei der Erweiterung der Matrix erlaubt das skizzierte Verfahren außerdem, dass Bewertungsvektoren neuer Nutzer nach der Modellerweiterung verworfen werden können, wodurch eine Verbesserung der Anonymität des Recommender-Algorithmus möglich ist.

3 Entwurf und Implementierung

Auf Basis des im vorherigen Kapitel vorgestellten PocketLens Algorithmus, soll nun auf den Entwurf und die Erweiterungen des in diesem Projekt erarbeiteten Recommenders eingegangen werden [Mu08]. Neben dem verteilten Empfehlungsalgorithmus wird auch erläutert, wie eine Empfehlung an eine Gruppe von Nutzern ausgesprochen werden kann und ein Überblick über die konkrete Implementierung im Projektszenario mit Benutzerschnittstelle geboten.

3.1 Entwurf des Recommenders auf Basis von PocketLens

Der zuvor vorgestellte PocketLens Algorithmus ist bereits zur Nutzung in verteilten Systemen und mit weniger leistungsfähigen Endgeräten vorgesehen. Dennoch zeigt sich bei der Konzeption eines verteilten Recommenders, dass in vielen Szenarien Verbesserungen notwendig sind, um den sinnvollen Einsatz auf PDAs zu ermöglichen.

Zum einen lässt sich leicht erkennen, dass durch die Verwendung eines Modells erheblich mehr Daten gespeichert werden müssen, als dies bei einem speicherbasierten Recommender der Fall wäre. Zum anderen realisiert PocketLens zwar das einfache Hinzufügen neuer Nutzer und ihrer Bewertungsvektoren, erlaubt es aber nicht, eine neue Version eines Bewertungsvektors mit neuen Bewertungen oder gar Änderungen hinzuzufügen. In einem verteilten Szenario jedoch, bei dem Benutzer regelmäßig Daten austauschen, kann dies dazu führen, dass dem Modell ein Benutzer hinzugefügt wird, der erst sehr wenige Items bewertet hat und alle von diesem Nutzer in der Zukunft getätigten Bewertungen nicht mehr in das Modell einfließen.

Um den Algorithmus um die genannten Fähigkeiten zu erweitern, wurde zunächst untersucht, in wieweit eine Optimierung des Datenmodells möglich ist und ein Konzept entwickelt, wie veränderte Bewertungsvektoren eines bekannten Nutzers in das Modell integriert werden können.

3.1.1 Speicherung des Recommender-Modells

Um die Matrix von Ähnlichkeitswerten, die von PocketLens genutzt wird, möglichst kompakt speichern zu können, wurden deren Merkmale untersucht. Hierzu wurde PocketLens implementiert und der Aufbau der Speichermatrix beim Hinzufügen von Daten auf die Eigenschaften Größe im Vergleich zum speicherbasierten Recommender, Besetztheit und Doppeleinträge untersucht.

Mit zunehmender Zahl an hinzufügten Bewertungsvektoren steigt der Speicherbedarf des Modells stark an. Die Matrix ist bereits nach dem Hinzufügen weniger Vektoren dicht besetzt, so dass die Nutzung eines Algorithmus zur Speicherung dünn-besetzter Matrizen keine Speicherersparnis erzielen würde. Unter den ausgefüllten Feldern befinden sich jedoch viele Duplikate, d.h. Felder, die den exakt gleichen Inhalt haben, so dass an dieser Stelle Speicher gespart werden kann. Ausgehend von einer dicht-besetzten Zeigermatrix ist jeder Feldinhalt bei PocketLens zumindest 20 Byte, jeder Zeiger jedoch lediglich 4 Byte groß, so dass für jedes doppelte Element in der Matrix eine Einsparung von 16 Byte möglich ist, wenn lediglich Zeiger auf einzigartige Feldinhalte abgelegt werden.

Neben der so erzielten Einsparung, kann die Definition gleicher Feldinhalte verändert werden. PocketLens speichert Fließkommawerte für Ähnlichkeit und Vektorlänge als Inhalt der Matrixfelder. Definiert man gleiche Felder als genau die Felder, die sich bis zu einer bestimmten Nachkommastelle und nicht bitgenau nicht unterscheiden, ergibt sich ein geringer Einfluss auf die Empfehlungsfindung, jedoch erhöht sich die Anzahl der nunmehr „ungenauen“ Duplikate erheblich, so dass eine höhere Speicherersparnis möglich ist. Kombiniert man dieses Verfahren mit einer dynamischen Anpassung der Vergleichsgenauigkeit, können unter Verringerung der Empfehlungsqualität erheblich größere Modelle verarbeitet werden, als dies bei einer direkten Speicherung der Fall wäre. Die Beschreibung der Evaluierung unseres Ansatzes mit einem Standard Datensatz im Bereich Recommender Systeme erfolgt in Kapitel 4.2. dieses Beitrags.

3.1.2 Versionierte Bewertungsvektoren

Zur Verbesserung der Erweiterbarkeit von PocketLens ist zunächst festzuhalten, dass diese nicht auf Kosten der Anonymität erfolgen soll, d.h. es soll nicht notwendig sein, die Bewertungsvektoren aller Nutzer vorzuhalten. Außerdem ist darauf zu achten, den Mehrverbrauch an Speicher so gering wie möglich zu halten.

Um die genannten Kriterien erfüllen zu können, wurde ein Algorithmus entwickelt, der es erlaubt, den Einfluss eines Bewertungsvektors auf das Modell zurückzusetzen, ohne dafür sämtliche in das Modell eingeflossenen Bewertungsvektoren zu kennen. Hierzu ist es zunächst notwendig, das Konzept des Bewertungsvektors von einer Menge an Zwei-Tupeln bestehend aus Item-Identifikator und Bewertung so zu erweitern, dass alte Zustände des Bewertungsvektors nachvollzogen werden können. Das daraus entstandene Konzept wird mit dem Namen „Versionierter Bewertungsvektor" bezeichnet und ist nunmehr eine Menge an Zwei-Tupeln, welche neben dem Item-Identifikator wiederum eine Menge an Versionsnummern und dazugehörigen Bewertungen enthält. Die für jedes Item enthaltenen Versions- und Bewertungspaare werden als rückwärts-verkettete Liste realisiert, um einen schnellen Zugriff auf die aktuellste Bewertung zu ermöglichen (vgl. Abbildung 2).

Bewertungsvektor

Subjekt	Wertung
A	4
C	1

Versionierter Bewertungsvektor

Subjekt	Versionierte Wertungen
A	(1,5) ← (4,4)
B	(2,4) ← (5,0)
C	(3,1)

Abbildung 2: Vergleich der konkreten Ausprägung eines Bewertungsvektors und eines versionierten Bewertungsvektors

Bei der Änderung einer Bewertung wird beim versionierten Bewertungsvektor nun lediglich ein weiterer Eintrag in die Liste des entsprechenden Items eingefügt, so dass alte Zustände rekonstruierbar bleiben. Fügt ein Endgerät den Bewertungsvektor eines Benutzers zu seinem Datenmodell hinzu, so speichert es die Version des hinzugefügten Vektors, die als Maximum alle einzelnen Versionsnummern alle enthaltener Bewertungen definiert ist. Soll eine neue Version mit geänderten oder zusätzlichen Bewertungen in das Modell einfließen, kann der Einfluss des zuvor in PocketLens eingefügten, alten Vektors rückgängig gemacht werden, indem aus dem hinzuzufügenden Vektor zunächst die alte, bereits genutzt Version errechnet, deren Einfluss auf das Modell rückgängig gemacht und dann die aktuelle Version hinzugefügt wird (vgl. [Mu08]).

Insgesamt ist es durch die Einführung versionierter Bewertungsvektoren möglich, PocketLens um neue Bewertungen bekannter Nutzer zu erweitern und Änderungen an bekannten Bewertungen vorzunehmen. Hierbei wird der erhöhte Speicheraufwand für geänderte Bewertungen und für die Speicherung des komplexeren, modifizierten Bewertungsvektors jeweils vom verursachenden Nutzer getragen. Insbesondere wird hierdurch der Aufwand zur Speicherung der Ähnlichkeitsmatrix nicht erhöht und es muss pro hinzugefügten Bewertungsvektor lediglich die genutzte Version, jedoch nicht etwa der gesamte Bewertungsvektor oder pro modifizierten Matrix Feld ein Wert gespeichert werden.

Schließlich gewährleistet der skizzierte Algorithmus immer noch den gleichen Grad an Anonymität, wie ihn auch PocketLens selbst bietet, so dass vor allem keine Speicherung der konkreten Bewertungsvektoren jedes Nutzers notwendig ist und sich diese auch nicht aus dem gespeicherten Modell errechnen lassen.

3.2 Vorgehensweise zur Gruppenempfehlung

In unserem Szenario soll das dezentrale Recommender System nun auch genutzt werden, um für eine zu einem bestimmten Zeitpunkt vor einem gemeinsamen Display anwesende Gruppe von Personen, Items empfehlen zu können. Die Gruppenbildung kommt dadurch zustande, dass der Rechner, der das gemeinsame Display steuert, auch die Bewertungsvektoren der anwesenden PDAs empfängt. Eine Gruppe – und damit der gemeinsame Kontext – besteht dann aus der Menge der PDAs (beziehungsweise deren Benutzer), die in einem Zeitfenster ihren Vektor geschickt haben und damit als aktiv vor den Display erkannt wurden.

Für eine Empfehlung für eine Gruppe von Nutzern muss das vorgestellte, auf einen einzelnen Nutzer bezogene Verfahren erweitert werden. Es ist insbesondere nötig, die Repräsentation einer Gruppe mit mehreren Benutzern, die jeweils verschiedene Items verschieden bewertet haben, derart zu modifizieren, dass diese Gruppe für den Recommender-Algorithmus als ein kombinierter „Pseudouser“ wirkt [JS07].

Um einen kombinierten Bewertungsvektor für eine Gruppe zu erhalten, verwenden wir die Bildung eines Mittelwertes der Bewertungsvektoren aller beteiligten Gruppenmitglieder. Dazu wird zunächst eine Menge G aller von mindestens einem Gruppenmitglied bewerteten Items ermittelt. Der Bewertungsvektor der Gruppe kann dann gebildet werden, indem für jedes Element des Vektors, welches ein von mindestens einem Gruppenmitglied bewertetes Item repräsentiert, die Bewertungen der Nutzer, die diesen tatsächlich bewertet haben, aufsummiert und durch die Anzahl dieser Benutzer geteilt werden. Der so erhaltene Vektor repräsentiert nun den Meinungsdurchschnitt der Gruppe. Dieser Bewertungsvektor kann dann im vorgestellten Algorithmus genutzt werden, um Empfehlungen zu generieren. Der Vorteil der Berechnung eines kombinierten Bewertungsvektors für eine Nutzergruppe liegt somit darin, dass der eigentliche Recommender-Algorithmus im Vergleich zur Empfehlung für einen einzelnen Benutzer unverändert verwendet werden kann.

Neben der Anzeige von Empfehlungen an eine Gruppe, können auch am gemeinsamen Display Bewertungen abgegeben werden. Dazu wird für eine anwesende Gruppe ein neuer, anonymer, Benutzer beziehungsweise Bewertungsvektor angelegt, und abgegebene Bewertungen diesem Benutzer zugeordnet. Erfolgen mehrere Bewertungen in geringem zeitlichem Abstand, werden diese gemeinsam dem anonymen Bewertungsvektor hinzugefügt. Der Bewertungsvektor wird dann – analog zum Verhalten der mobilen Endgeräte – an alle anderen in der Nähe befindlichen Geräte übertragen, damit auf den PDAs die Item-Item Matrizen entsprechend aktualisiert werden können. Hierdurch ist es sowohl möglich, ohne eigenes Endgerät an der Empfehlungsbildung der anderen, mobilen Nutzer teilzunehmen, als auch, die Gruppenempfehlung für eine festgelegte Zeit mit zu beeinflussen.

3.3 Implementierung

Der erweiterte Algorithmus wurde zur Generierung von Bildempfehlungen für PDAs mit dem Betriebssystem Windows Mobile implementiert [Mu08]. Hierzu erfolgte zunächst eine Umsetzung des in diesem Beitrag beschriebenen Recommender-Algorithmus, welcher dann für die Empfehlung von Bildern spezialisiert wurde. Die Matrix, in der Item-Item Ähnlichkeiten gespeichert werden, ist hierbei als Referenzmatrix auf konkrete Feldinhalte implementiert, wobei eine Duplikatsuche und Elimination aus Gründen der Laufzeit nicht beim Einfügen in die Matrix sondern asynchron beim Erreichen der Speichergrenze des mobilen Endgeräts erfolgt.

Da ein möglichst realistisches Szenario abgedeckt werden sollte, wurde ein Protokoll für den Austausch von Bewertungsvektoren entwickelt, welches lediglich eine bestehende Verbindung in einem gemeinsamen Netzwerk voraussetzt. Konkret wurde ein auf UDP Multicast basierendes Netzwerkprotokoll entwickelt, welches es ermöglicht auch Datenmengen jenseits der Limitierungen für einzelne UDP Pakete auszutauschen, ohne dabei jedoch eine Aussage über den Erfolg der Übermittelung treffen zu können.

Die Nutzung eines Multicast Protokolls bietet sich insbesondere deshalb an, weil die Konstruktion des Algorithmus keine bidirektionale Kommunikation erfordert, und der eigene versionierte Bewertungsvektor lediglich anderen Teilnehmern zur Verfügung gestellt werden muss. Hat ein anderer Nutzer einen solchen Vektor empfangen, kann dieser unabhängig von der Frage, ob eine ältere Version bereits in das Modell eingeflossen ist oder nicht in das Empfehlungsmodell integriert werden. Für die Speicherung des aktuellen Modellzustands sowie für den Austausch zwischen den mobilen Geräten untereinander sowie mit dem Touchscreen Display werden die Datenstrukturen in eine XML Repräsentation umgewandelt und mittels „bzip2“ komprimiert.

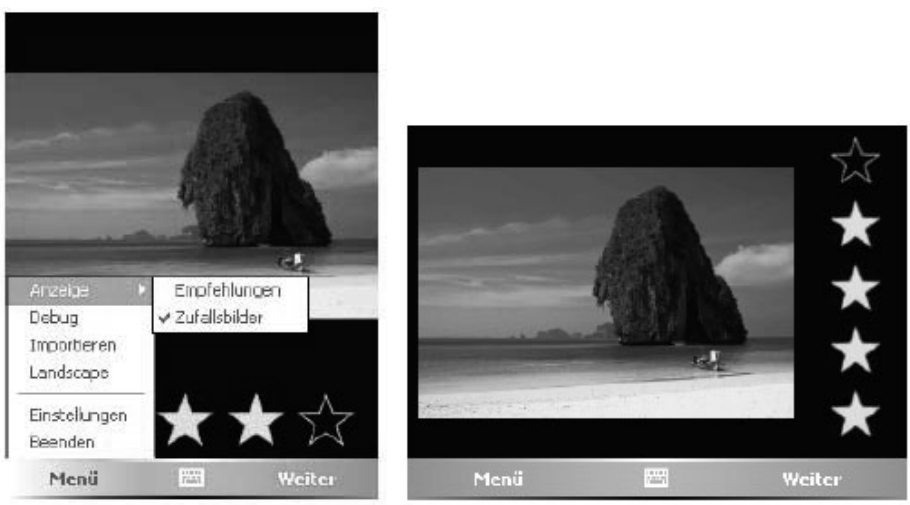

Abbildung 3: Mobile Benutzerschnittstelle

3.4 Benutzerschnittstelle

Abbildung 3 zeigt die Benutzerschnittstelle am Mobilgerät. Angezeigt wird ein Bild mit der Möglichkeit, dieses von 1 bis 5 (Sternen) zu bewerten, oder gegebenenfalls eine bereits abgegebene Bewertung anzuzeigen und zu ändern. Es ist möglich in eine „Landscape“ Ansicht zu wechseln (vgl. Abbildung 3, rechts), um die Bilder besser betrachten zu können.

In einem Menü kann man einstellen, ob man zufällige oder empfohlene Bilder anzeigen lassen will. Zufallsbilder sind sinnvoll, um Anfang einige Bewertungen abgeben zu können, auf deren Basis dann neue empfohlene Items ermittelt werden können, wie oben erläutert. Nach Auswahl der Funktion „Weiter“ wird ein weiteres Bild angezeigt. Die Benutzerschnittstelle wurde im Hinblick auf eine möglichst einfache Bedienbarkeit entworfen.

Als gemeinsames Display wird ein Monitor mit Touchscreen-Funktion verwendet. Dabei wird zunächst ein für die aktuell anwesende Gruppe von Personen empfohlenes Bild ohne Bewertungsleiste angezeigt. Berührt ein Benutzer den Touchscreen, findet ein Moduswechsel statt und ein Benutzer kann mittels einer Bewertungsleiste – analog zu der im mobilen Endgerät verwendeten Leiste – ein Bewertung für das Bild abgeben (vgl. Abbildung 4). Zusätzlich werden in unserem Prototypen rechts Informationen über aktuell als anwesende erfasste Benutzer bzw. PDAs, sowie Debug-Informationen, angezeigt. Dies würde bei einem realen Einsatz ausgeblendet werden. Zu bemerken ist, dass das Touchscreen Interface in unserem System keinen Server darstellt, sondern im Wesentlichen auch nur ein Peer mit angepasster Benutzerschnittstelle ist.

Abbildung 4: Szenario mit Touchscreen Monitor

4 Evaluierung

Das vorgestellte System wurde in einem (kleinen) Anwendertest in einem realistischen Szenario evaluiert [Mu08]. Dieser Test soll nicht eine formale Evaluierung des Systems darstellen, sondern die praktische Einsetzbarkeit zeigen und durch einen Fragebogen einige Tendenzen bezüglich interessanter Fragestellungen ermitteln. Des Weiteren wurden die Skalierbarkeit und Performance anhand eines Standard-Datensatzes für Recommender Systeme getestet. Schließlich diskutieren wir verwandte Arbeiten.

4.1 Anwendertest

Der Anwendertest wurde mit 13 Personen und Windows Mobile PDAs von HTC und HP durchgeführt, z.B. einem HTC P3600 PDA Phone. Als gemeinsames Display kam ein 32“ Monitor mit Touchscreen Funktionalität zum Einsatz (vgl. Abbildung 4). Die Itemmenge bestand aus 63 vorgegebenen Bildern. Zunächst sollten die Benutzer, Bewertungen für etwa 15-20 zufällig gewählte Bilder auf ihren PDAs abgeben. Anschließend wurden die so generierten Bewertungsvektoren ausgetauscht, wodurch vom Touchscreen Display ein Modell für die Gruppenempfehlung und von jedem einzelnen Endgerät ein Modell für die persönliche Empfehlung ermittelt werden konnte. Die so erstellten Empfehlungsmodelle wurden anschließend getestet. Hierzu betrachteten die Benutzer zum einen jeder für sich die ihnen persönlich vorgeschlagenen, neuen Bilder, zum anderen die in verschiedenen Gruppenkonstellationen der jeweiligen Gruppe vorgeschlagenen Bilder. Es befand sich immer nur eine Teilmenge der Benutzer vor dem gemeinsamen Display, die maximale gleichzeitige Gruppengröße in unserem Test war 6 Personen.

Der Test zeigte zunächst, dass die Anwendung praxistauglich ist und im dargestellten Szenario funktionierte. Sowohl die Berechnung der Item-Item Matrix als auch die Ermittlung der Empfehlungen lief ohne Performanceprobleme. Die Teilnehmer waren im Anschluss an den Test aufgefordert, einen Fragebogen auszufüllen. Bei den meisten Fragen sollte ein Teilnehmer Aussagen auf einer Skala der Übereinstimmung von −2 für „trifft nicht zu“ bis +2 für „trifft zu“ mit Zwischenstufen beurteilen. Der verwendete Fragebogen sowie die detaillierteren Ergebnisse sind in [Mu08] zu finden.

Zunächst wurde abgefragt, inwieweit den Testteilnehmern der Schutz ihrer persönlichen Daten allgemein wichtig ist. Dies war mit einer mittleren Übereinstimmung von +1,38 der Fall. Außerdem wurde eine Aussage zur Beurteilung vorgelegt, ob die Nutzung eines Bewertungsprofils zur Generierung von Empfehlungen prinzipiell einen Eingriff in die Privatsphäre darstellt. Dabei ergab sich kein einheitliches Bild, die Tester nutzen die ganze Bandbreite der Skala relativ gleichmäßig aus. Eine weitere Frage, ob ein Recommender System, welches garantiert, dass die konkreten Bewertungen und Daten nicht offengelegt werden, die Bereitschaft zur Nutzung erhöhen würde, wurde mit vielen positiven Einschätzungen beurteilt (Mittelwert +0,92).

Bei der Evaluierung des getesteten Systems an sich, wurde zunächst eine Beurteilung der Dezentralität des Systems untersucht. Dabei erkannten fast alle Benutzer die Vorteile der Unabhängigkeit von einem Server im getesteten Szenario. Zur Beurteilung der Empfehlungsqualität könnten die Tester zunächst beurteilen, ob ihnen die vorgeschlagenen Bilder angemessen und sinnvoll – in Bezug auf ihre Bewertungen – erschien. Dies wurde mit einem Mittelwert von +0,54 (schwach) positiv beurteilt. Als weitere Frage sollten die Tester dann die Empfehlungsqualität der persönlichen Empfehlungen nach Austausch der Bewertungsvektoren begutachten. Dabei ergab sich mit einem Mittelwert von +0,69 auch eine positive Tendenz. Im Nachhinein betrachtet war für eine noch differenzierte Analyse der Empfehlungsqualität aus Nutzersicht unsere Itemmenge von 63 Fotos wahrscheinlich zu homogen.

Im letzten Teil des Fragebogens konnten die Tester die Einsetzbarkeit der Anwendung begutachten. Dabei wurden sowohl die Frage „Ein System dieser Art für die Bewertung und Empfehlung von Bildern/Musik/Videos halte ich für sinnvoll“ wie auch „Ich könnte mir vorstellen, ein solches System aktiv zu nutzen.“ mit einem Mittelwert von +0,92 bzw. +0,69 positiv beantwortet. Abschließend konnten die Teilnehmer noch konkrete Anwendungsgebiete für das System angeben, dabei wurde neben „Messen“ und „Konferenzen“ auch „Einkauf“, „Dezentrale Filesharing Netzwerke“, „Parties“ und „Museen“ genannt.

4.2 Test der Skalierbarkeit

Neben dem praktischen Test haben wir auch die Skalierbarkeit des implementierten und in Abschnitt 3.1. genauer erläuterten Algorithmus‘ näher untersucht. Dazu haben wir den oft für die Evaluierung von Recommender System herangezogenen „MovieLens“ Datensatz des GroupLens Projektes (siehe http://www.grouplens.de) der University of Minnesota verwendet [HKBR99]. Der Datensatz umfasst in der von uns benutzten Variante 100.000 Bewertungen, die von 943 Benutzern für 1682 Filme abgegeben wurden. Jeder Benutzer hat dabei mindestens 20 Bewertungen durchgeführt.

Das Datenmodell von PocketLens kann durch Elimination von echten Duplikaten von einer Größe von etwa 45 Megabyte auf eine Größe von 32 Megabyte reduziert werden. Vergleicht man die einzelnen Feldinhalte mit einer Genauigkeit von lediglich 0,05, so reduziert sich der erforderliche Speicher auf unter 20 Megabyte, so dass der gesamte Datensatz – verlustbehaftet – in den Speicher des PDAs geladen werden konnte.

Um eine möglichst geringe Laufzeit zu erreichen, wird eine Duplikatsuche lediglich dann ausgeführt, wenn eine bestimmte Speicherschranke überschritten wird. Im Mittel benötigten die Testgeräte etwa 1,5 Sekunden zum Einfügen eines Bewertungsvektors, was das Verfahren für den skizzierten Anwendungsfall tauglich macht. Der Großteil der Laufzeit wird hierbei für die nebenläufige Duplikatelimination benötigt, während nur ein Bruchteil der Laufzeit auf das Einfügen eines einzelnen Bewertungsvektors entfällt.

4.3 Verwandte Arbeiten

Neben dem ausführlichen besprochenen PocketLens gibt es einige andere Ansätze in eine ähnliche Richtung. [BKR07] stellt ein verteiltes Recommender System vor, das eine Partitionierung der User-Item Matrix auf Basis einer Domänen-spezifischen Aufteilung der Items vornimmt. Es sind dazu allerdings zusätzliche Informationen über die Anwendungsdomäne nötig. Einige Beiträge wie [WPLR06] diskutieren dezentrales kollaboratives Filtern in Peer-to-Peer (P2P) Netzwerken, wobei allerdings keine mobilen Peers wie PDAs berücksichtigt werden. [JRH06] bespricht einen CF Ansatz für mobile „Media Agents“. Dabei werden den Items, wie z.B. Musik Dateien, Informationen und Regeln in einem Profil mitgegeben. Die Items sollen dann selbstständig als Software-Agenten interagieren und ein CF-ähnliches Verhalten simulieren.

Bezüglich der Empfehlung für Gruppen von Personen gibt [JS07] einen Überblick über den aktuellen Stand der Forschung. Ein frühes und bekanntes Beispiel ist das „PolyLens“ System [OCKR01], was zur Unterstützung einer Gruppe von Personen beim gemeinsamen Kinobesuch entwickelt wurde. Dabei wird kollaboratives Filtern verwendet. Die empirischen Untersuchungen in diesem Projekt haben gezeigt, dass Benutzer nicht nur die Empfehlungen des Systems als wertvoll eingeschätzt haben, sondern auch bereit waren, einen Teil ihrer Privacy für die Gruppenempfehlungen aufzugeben [OCKR01]. PolyLens ist allerdings kein dezentrales Recommender System. Ein mit unserem System direkt vergleichbarer und auf PDAs realisierter Ansatz ist uns nicht bekannt.

5 Fazit

In diesen Beitrag haben wir eine Realisierung eines dezentralen Recommender Systems für PDAs vorgestellt. Dabei haben wir insbesondere eine private Empfehlung und Anzeige auf einem PDA mit einem gemeinsamen Display für eine Empfehlung für Gruppen von Personen integriert. Darüberhinaus haben wir gegenüber dem existierenden PocketLens Ansatz eine verbesserte Erweiterbarkeit durch versionierte Bewertungsvektoren eingeführt, sowie eine Optimierung des auf einem mobilen Gerät erforderlichen Speicherbedarfs erzielt.

Neben einem weitergehenden Praxistest unseres Systems, auch mit einer anderen Itemmenge, sind verschiedene Erweiterungen für zukünftige Arbeiten denkbar. Zum einen könnte die Effizienz beim Prozess der Generierung von Empfehlungen noch weiter erhöht werden. Dazu könnten eine Speicherung relevanter Zwischenergebnisse und Änderungsmarkierungen eingesetzt werden, um auch bei einem großen Datenmodell oder einem sehr schwachen Mobilgerät Empfehlungen ableiten zu können. Auch kann das Verfahren zur Ermittlung von Empfehlungen für eine Gruppe von Personen insbesondere durch eine Prüfung weiterer Methoden zur Generierung des repräsentativen Bewertungsvektors einer Gruppe verbessert werden.

Schließlich wäre es von Interesse, den Datenaustausch zwischen einzelnen Benutzern weniger explizit zu gestalten und mit geeigneten Methoden, wie etwa asymmetrischen Verschlüsselungsverfahren oder einem impliziten Austausch von Bewertungen und Modellteilen, einen höheren Grad an Anonymität zu erreichen.

Literaturverzeichnis

[AT05] Adomavicius, G.; Tuzhilin, A.: Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, 2005.

[BKR07] Berkovsky, S.; Kuflik, T.; Ricci, F.: Distributed collaborative filtering with domain specialization. In Proceedings of the 2007 ACM Conference on Recommender Systems, Minneapolis, MN, 2007.

[HKBR99] Herlocker, J.L.; Konstan, J.A.; Borchers, A.; Riedl, J.: An algorithmic framework for performing collaborative filtering. Proceedings of the 22nd Annual international ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, CA, 1999.

[JRH06] Jacobsson, M.; Rost, M.; Holmquist, L. E.: When Media Gets Wise: collaborative filtering with mobile media agents. In Proceedings of the 11th International Conference on Intelligent User Interfaces (IUI '06), Sydney, Australia, 2006.

[JS07] Jameson, A.; Smyth, B.: Recommendation to Groups. In (Brusilovsky, P.; Kobsa, A.; Nejdl, W., Eds.): The Adaptive Web, LNCS 4321, Springer Verlag, Berlin/Heidelberg, 2007; S. 596-627.

[Ko07] Kobsa, A.: Privacy-Enhanced Personalization. Communications of the ACM, Vol. 50, No. 8, 2007; S. 24-33.

[MKR04] Miller, B.N.; Konstan, J.A.; Riedl, J.T.: PocketLens: Toward a Personal Recommender System. ACM Transactions on Information Systems, Vol. 22, No. 3, 2004; S. 437–476.

[Mu08] Mühe, H.: Entwurf und Realisierung eines dezentralen Recommender Systems für die Empfehlung von Items für Gruppen. Bachelorarbeit in Informatik, Technische Universität München, 2008.

[OCKR01] O'Connor, M.; Cosley, D.; Konstan, J. A.; Riedl, J.: PolyLens: a recommender system for groups of users. In Proceedings of the Seventh Conference on European Conference on Computer Supported Cooperative Work (ECSCW), Bonn, 2001.

[SKKR01] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J.: Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on the World Wide Web, Hong Kong, China, 2001.

[WPLR06] Wang, J.; Pouwelse, J.; Lagendijk, R. L.; Reinders, M. J.: 2006. Distributed collaborative filtering for peer-to-peer file sharing systems. In Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 06), Dijon, France, 2006.