Flexible Online-Recommender-Systeme durch die Integration in ein Datenstrommanagementsystem

Cornelius A. Ludmann
Universität Oldenburg
Escherweg 2
26121 Oldenburg, Germany
cornelius.ludmann@uni-oldenburg.de

Marco Grawunder
Universität Oldenburg
Escherweg 2
26121 Oldenburg, Germany
marco.grawunder@uni-oldenburg.de

H.-Jürgen Appelrath
Universität Oldenburg
Escherweg 2
26121 Oldenburg, Germany
appelrath@uni-oldenburg.de

ZUSAMMENFASSUNG

In dieser Arbeit wird eine flexible, erweiterbare und domänenunbhängige Integration von Online-RecSys-Funktionen in ein Datenstrommanagementsystem (DSMS) vorgestellt. Dazu werden neue logische Operatoren eingeführt, mit der Nutzer eines DSMS auf einer abstrakten Ebene RecSys-Funktionen nutzen können. Des Weiteren wird eine beispielhafte Realisierung durch physische Operatoren vorgestellt, wie sie aus den logischen Operatoren durch Transformationsregeln erzeugt werden kann. Durch dieses Konzept können Benutzer ein RecSys mit Hilfe einer Anfragesprache auf die domänenspezifischen Bedürfnisse anpassen und mit anderen Funktionen eines DSMS kombinieren. Des Weiteren bringt ein DSMS einige Eigenschaften (z. B. Anfrageplanoptimierung, Fragmentierung, Scheduling etc.) mit, von denen ein Online-RecSys profitieren kann. Die Flexibilität eines DSMS ermöglicht den Vergleich und die Evaluation verschiedener RecSys-Algorithmen durch den Benutzer.

Keywords

recommender system, collaborative filtering, data stream management system, stream processing

1. EINLEITUNG

Recommender-Systeme (RecSys) haben das Ziel, das Interesse eines Benutzers an einem bestimmten Objekt zu schätzen, um dem Benutzer aus einer großen Menge an Objekten die subjektiv interessantesten Objekte zu empfehlen. Eine gängige Methode ist das modellbasierte, kollaborative Filtern (CF), bei dem aufgrund von bekannten Objektbewertungen verschiedener Benutzer (z. B. Produktbewertungen) ein Modell angelernt wird, mit dem geschätzt werden kann, wie ein Benutzer unbekannte Objekte bewerten würde. Durch die Entwicklung von Online-Algorithmen für das CF können neue Bewertungen von Benutzern in die Modelle integriert werden, ohne dass das Modell von Grund auf neu gelernt werden muss. Das ermöglicht eine neue Betrachtung des RecSys-Problems: Während bisherige Methoden das RecSys-Problem als periodisches Anlernen eines Modells basierend auf einen statischen und endlichen Datensatz betrachten, kann durch Online-Algorithmen das Problem als Verarbeitung von Bewertungs-Ereignissen eines kontinuierlichen und potenziell unendlichen Datenstroms betrachtet werden. Das hat den Vorteil, dass durch die sofortige Integration neuer Bewertungsinformationen nicht nur das grundsätzliche Interesse, sondern auch das aktuelle Interesse des Benutzers berücksichtigt werden kann.

Zur Konzeption eines datenstrombasierten RecSys schlagen wir vor, auf ein anwendungsunabhängiges und generisches Datenstrommanagementsystem (DSMS) aufzubauen. Die Eingabedaten werden als kontinuierlich auftretende, zeitannotierte Ereignisse eines potenziell unendlichen Datenstroms betrachtet. Das DSMS berechnet mit Hilfe von Datenstrom-Operatoren und Anfrageplänen kontinuierlich ein RecSys-Modell. Dieser Ansatz hat folgende Vorteile:

(1) Die Modellierung der Daten als Datenströme kommt dem realen Einsatz eines RecSys näher als die Verwendung von statischen Datensätzen.

(2) Ein DSMS bringt als Grundlage für ein RecSys bereits Konzepte und Operatoren zur effizienten Verarbeitung von Datenströmen mit temporal kohärenten Ereignissen mit.

(3) In einem DSMS kann ein RecSys in logische Teile zerlegt werden, welche jeweils durch einen Operator repräsentiert werden. Das ermöglicht eine flexible und auf die konkrete Anwendung angepasste Komposition von RecSys-Operatoren mit Standardoperatoren, z. B. für die Datenvor- bzw. -nachbearbeitung, Kontextmodellierung etc.

(4) Eine Aufgabe, zum Beispiel das Modelllernen, kann durch austauschbare Operatoren mit gleicher Semantik aber unterschiedlicher Implementierung realisiert werden. Das ermöglicht den Vergleich sowie den bedarfsgerechten Einsatz von verschiedenen Lernalgorithmen.

Ziel der Arbeit ist die Einführung eines generischen Konzepts zur Integration von RecSys-Funktionen in ein DSMS. Dazu führen wir im folgenden Abschnitt zunächst die Grundlagen ein und stellen in Abschnitt 3 verwandte Arbeiten vor. Danach beschreiben wir die logischen Operatoren für die Umsetzung eines RecSys (Abschnitt 4) und stellen eine Transformation in physische Operatoren vor (Abschnitt 5). In Abschnitt 6 diskutieren wir anschließend einige Entscheidungen unseres Konzepts und stellen Alternativen vor. Zum Schluss präsentieren wir in Abschnitt 7 unsere prototypische Implementierung in ein Open-Source-DSMS und zeigen die

27^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 26.05.2015 - 29.05.2015, Magdeburg, Germany.

Machbarkeit durch eine Evaluation, bevor wir mit einer Zusammenfassung und einem Ausblick die Arbeit beenden.

2. GRUNDLAGEN

Ein **RecSys** hat die Aufgabe einem aktiven Benutzer $u' \in U$ eine Menge an Objekten aus I (Menge aller Objekte) zu empfehlen, für die dieser sich interessiert (Empfehlungsmenge). Dazu schätzt das RecSys für jedes zur Empfehlung infrage kommende Objekt eine Bewertung $\hat{r} \in R$, welche das Interesse des Benutzers an dem Objekt quantifiziert. Die Menge der infrage kommenden Objekte wird als Menge der Empfehlungskandidaten bezeichnet. Die Empfehlungsmenge wird durch die K Objekte der Empfehlungskandidaten gebildet, die die höchsten Bewertungen haben (Top-K-Menge). Um die Bewertung eines Objekts für einen Benutzer bestimmen zu können, ermittelt das RecSys (z. B. mit der Matrix-Faktorisierung) eine Annäherung $\hat{f}_R$ an eine wahre aber unbekannte Bewertungsfunktion $f_R : U \times I \to R$.

Seit den 1970er/1980er Jahren sind relationale *Datenbankmanagementsysteme* (DBMS) die bedeutendste Technik, Daten dauerhaft zu speichern. Ein DBMS abstrahiert von der Komplexität der Datenspeicherung und sorgt für eine effiziente Verarbeitung von komplexen Anfragen, um Daten zu speichern, zu lesen, zu aktualisieren und zu löschen. DBMS sind allerdings nicht mit dem Ziel entwickelt worden, kontinuierlich erzeugte Daten zu verarbeiten. Diese Lücke schließen **DSMS**, die auf die fortlaufende Verarbeitung von Datenströmen ausgelegt sind.

Während ein DBMS wahlfreien Zugriff auf gespeicherte Daten hat, erhält ein DSMS die Daten von einem kontinuierlichen Datenstrom (potenziell unendliche Sequenz von Datenstromelementen). Die Datenstromelemente werden von einer aktiven Quelle erzeugt und an das DSMS übertragen. Ein DSMS hat keine Kontrolle über die Reihenfolge, in der die Datenstromelemente von der Quelle gesendet werden – weder innerhalb eines Datenstroms, noch zwischen verschiedenen Datenströmen. Sobald ein Element verarbeitet ist, wird es verworfen oder archiviert. Das DSMS kann nicht erneut auf bereits verarbeitete Elemente zugreifen, es sei denn, sie werden explizit im Arbeitsspeicher gehalten (one-pass paradigma). Dies kann allerdings nur für einen begrenzten Teil der Datenstromelemente geschehen, da der Speicher im Gegensatz zum Datenstrom in der Größe begrenzt ist [4].

Die Verarbeitung der Daten in einem DBMS erfolgt üblicherweise mit Hilfe von Einmal-Anfragen. Diese werden einmalig auf aktuellem Datenbestand ausgeführt und der anfragende DBMS-Benutzer erhält einmalig eine Antwort. Bei der Verarbeitung von kontinuierlichen Datenströmen stellt der Benutzer eine kontinuierliche Anfrage. Diese wird einmalig dem DSMS übergeben und produziert kontinuierlich Ergebnisse [4]. Zur Verarbeitung von relationalen Datenströmen kann ein DSMS eine datenstrombasierte Variante der relationalen Algebra einsetzen. Eine kontinuierliche Anfrage wird, wie bei Anfragen in DBMSs, in einer Anfragesprache formuliert. Beispiele für DSMS-Anfragesprachen sind die SQL-ähnliche Sprache CQL [3] und die PQL [2].

Eine Anfrage wird von einem DSMS in einen Anfrageplan übersetzt. Dieser besteht aus Operatoren, die durch Warteschlangen miteinander verbunden sind. Ein Anfrageplan kann als gerichteter Graph interpretiert werden, dessen Knoten die Operatoren und Kanten die Warteschlangen darstellen. Die Ausführung der Operatoren koordiniert ein Scheduler [4]. Zu einer Anfrage wird ein logischer Anfrageplan, bestehend aus logischen Operatoren, erzeugt. Auf einem logischen Anfrageplan können Optimierungen ausgeführt werden (z. B. Veränderung der Operatorreihenfolge), ehe er in einen physischen Anfrageplan mit physischen Operatoren überführt wird. Physische Operatoren enthalten konkrete Implementierungen für die Verarbeitung der Daten. Das DSMS wählt zu jedem logischen Operator ein oder mehrere physische Operatoren, die für die Operation am geeignetsten sind.

3. VERWANDTE ARBEITEN

Verschiedene Veröffentlichungen über *inkrementelle* oder *online* Algorithmen für CF (z. B. [10, 11]) zeigen wie neue Lerndaten in CF-Modelle integriert werden können, ohne dass das Modell komplett neu gelernt werden muss. Diese Algorithmen bieten die Basis für eine datenstrombasierte Verarbeitung von Bewertungsdaten für RecSys. Die Arbeit von Diaz-Aviles et al. [7] stellt eine Methode zur datenstrombasierten Verarbeitung von Bewertungsdaten zum Lernen eines Matrix-Faktorisierungs-Modells vor. Dazu wird das Modell mit Daten aus einem Reservoir aktualisiert, welches eine Stichprobe des Datenstroms vorhält. Im Gegensatz zu diesen Arbeiten stellen wir keinen neuen Algorithmus vor, sondern setzen den Fokus auf die Komposition eines flexiblen RecSys mithilfe von Datenstromoperatoren, in denen diese Algorithmen als Operatoren integriert werden können.

Einige Veröffentlichungen nutzen Frameworks zur Implementierung eines RecSys, die bei der Entwicklung von datenstrombasierten Systemen unterstützen. Zum Beispiel setzt Ali et al. [1] *Apache Storm* ein, um einen CF-Algorithmus zu realisieren. Das datenstrombasierte Data-Mining-Framework *Massive Online Analysis* (MOA) [5] ermöglicht die Evaluation unter anderem von datenstrombasierten RecSys-Algorithmen. Im Gegensatz zu unserem Konzept bieten diese Arbeiten keinen anwendungsunabhängigen, erweiterbaren und flexiblen Ansatz, der auf die Komposition von Datenstromoperatoren setzt. Unser Ansatz kann außerdem von diversen DSMS-Funktionen profitieren.

Mit *StreamRec* stellen Chandramouli et al. [6] einen Ansatz zur Nutzung eines DSMS für ein RecSys vor. Dort wird das DSMS *Microsoft StreamInsight* genutzt. Im Gegensatz zu unserem Konzept werden in StreamRec ausschließlich Basisoperatoren eines DSMS eingesetzt und es ist auf die Berechnung von Objektähnlichkeiten zur Bestimmung der Empfehlungsmenge beschränkt. Unser Konzept verfolgt einen generischeren Ansatz, der auch die Integration von modellbasierten Lernalgorithmen erlaubt und durch logische RecSys-Operatoren eine logische Abstraktionsebene für den Nutzer des DSMS bietet.

4. LOGISCHE OPERATORSICHT

Betrachtet man die Ein- und Ausgabedaten eines RecSys, so kann man diese als folgende Datenströme auffassen: Erstens überträgt die Benutzeranwendung Benutzer-Objekt-Bewertungs-Tripel (u, i, r) für jede Bewertung, die ein Benutzer vornimmt. Zweitens werden von einem Benutzer u Empfehlungen angefordert (z. B. dann, wenn ein Benutzer die Empfehlungsseite der Benutzeranwendung öffnet; im folgenden Empfehlungsanforderungen – engl. Request for Recommendations, RfR – genannt). Drittens sendet das RecSys eine Empfehlungsmenge zurück an die Benutzeranwendung. Des Weiteren können die Eingabedaten

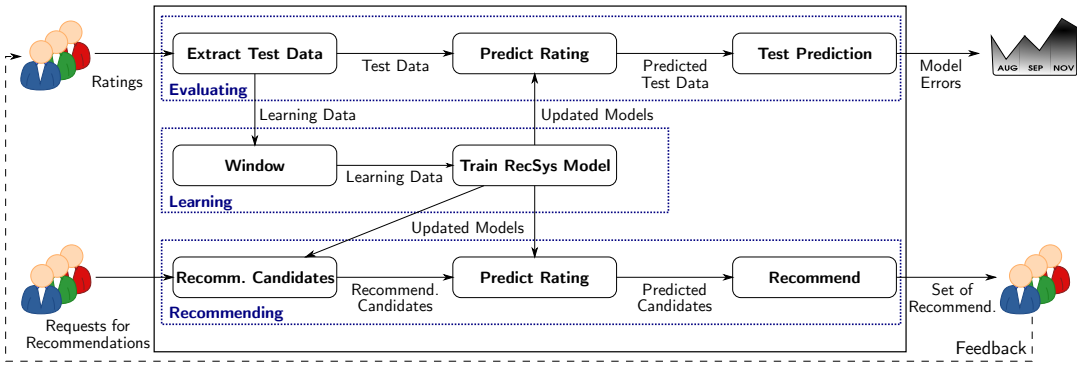

Abbildung 1: Logische Sicht auf die Operatoren eines DSMS-basierten RecSys

um ein Benutzerfeedback ergänzt werden. Im folgenden erwarten wir das Benutzerfeedback in der selben Form wie neue Bewertungen (u-i-r-Tripel). Möchte man das RecSys kontinuierlich evaluieren, so erhält man einen zusätzlichen Ausgabedatenstrom mit Fehlerwerten, der den Modellfehler angibt.

Zwischen den Ein- und Ausgabedatenströmen verarbeiten Datenstromoperatoren einer oder mehrerer Anfragen die Daten. Um ein RecSys mit Hilfe eines DSMS zu realisieren, muss mit Hilfe einer Anfrage aus den Eingabedatenströmen (u-i-r-Tripel und Empfehlungsanforderungen) als Antwort ein Datenstrom an Empfehlungsmengen erzeugt werden, der für jede Empfehlungsanforderung eine aktuelle und personalisierte Liste an Empfehlungen enthält.

Die Anfrage für ein RecSys haben wir in logische Operatoren zerlegt, die jeweils einen Teil der RecSys-Funktionalität übernehmen. Diese Operatoren bilden eine Abstraktionsebene, sodass der DSMS-Benutzer diese in der Anfrage nutzen kann, ohne die genaue Umsetzung durch physische Operatoren kennen zu müssen. Wo dies möglich ist, werden diese Operatoren bei der Transformation zu physischen Operatoren durch vorhandene Operatoren ersetzt.

Abbildung 1 zeigt eine Übersicht über die logischen Operatoren und wie diese in einer Beispielanfrage zusammenhängen. Auf der linken Seite sehen wir die Eingabedatenströme und auf der rechten Seite die Ausgabedatenströme. Die dazwischenliegenden Operatoren können in drei Kategorien eingeteilt werden: Operatoren zum Lernen des RecSys-Modells (mittig), zum Empfehlen von Objekten (unten) und zum Evaluieren (oben).

Die Anfrage besteht aus folgenden logischen Operatoren:

Window: Ein zeitbasiertes Fenster (umgesetzt durch einen WINDOW-Operator) ermöglicht, die Gültigkeit der Bewertungsdaten zu beschränken. Das kann sinnvoll sein, um ein Modell anzulernen, welches sich auf die neuesten Daten beschränkt und ältere Daten verwirft (z. B. um Concept Drifts zu berücksichtigen). Zusätzlich beschränkt es die Menge der Bewertungsdaten, die im Speicher gehalten werden müssen. Sollen alle Daten für immer behalten werden, so kann dieser Operator entfernt werden.

Train RecSys Model: Dieser Operator lernt aus den Bewertungsdaten ein Modell an. Wenn neue Daten hinzugefügt werden gibt es ein aktualisiertes Modell aus. Das Modell ist für eine bestimmte Zeitspanne gültig, sodass zu jedem Zeitpunkt genau ein Modell für die Vorhersage der Bewertung zuständig ist.

Recomm. Candidates: Für jede Empfehlungsanforderung bestimmt dieser Operator die Empfehlungskandidaten. Diese sind in der Regel alle Objekte, die von dem anfordernden Benutzer noch nicht bewertet wurden.

Predict Rating: Dieser Operator wird sowohl für die Bestimmung der Empfehlungsmenge als auch für die Evaluation genutzt. Er schätzt die Bewertung des Benutzers für jeden Empfehlungskandidaten bzw. für jedes Testtupel ab. Mit diesem Operator wird sichergestellt, dass genau das Model genutzt wird, welches zum gleichen Zeitpunkt wie die Empfehlungsanforderung bzw. das Testtupel gültig ist. Das stellt eine deterministische Verarbeitung der Daten sicher, was insbesondere für die Evaluation von Vorteil ist.

Recommend: Aus den bewerteten Empfehlungskandidaten werden die Objekte ausgewählt, die dem Benutzer empfohlen werden sollen. Das sind in der Regel die K Objekte mit den höchsten, geschätzten Bewertungen (Top-K-Menge). Eine weitere Möglichkeit besteht darin, nur Elemente mit einer minimalen Bewertung zu berücksichtigen.

Extract Test Data: Um das RecSys zu evaluieren, gibt dieser Operator neben den Lerndaten auch Testdaten aus. Eine mögliche Implementierung filtert 10 % der Bewertungsdaten als Testdaten aus und leitet die restlichen Daten als Lerndaten an den Modelllerner weiter.

Test Prediction: Dieser Operator implementiert eine Evaluationsmetrik, z. B. Root Mean Square Error (RMSE). Dabei vergleicht er die wahren Bewertungen aus den Testdaten mit den geschätzten Bewertungen zu den Testdaten oder die wahre Position in einem Ranking mit der Position aufgrund der geschätzten Bewertungen. Der daraus resultierende Modellfehler wird zu einem gleitenden Durchschnittswert oder einem gesamten Durchschnittswert aggregiert.

5. PHYSISCHE OPERATORSICHT

Die logischen Operatoren werden durch Transformationsregeln in physische Operatoren überführt. Abbildung 2 zeigt ein Beispiel für einen physischen Anfrageplan, der aus einer Anfrage generiert wird, der die logischen Operatoren nutzt. Die linke Spalte enthält die Operatoren für die Evaluation (entspricht dem oberen Teil von Abbildung 1), die mittlere Spalte die Operatoren für das Lernen des Modells (mittlerer Teil von Abbildung 1) und die rechte Spalte die Operatoren für die Berechnung der Empfehlungsmenge (unterer Teil von Abbildung 1). Dabei sind die Operatoren, die zu *einem* logischen Operator aus Abbildung 1 gehören, durch ein gestricheltes Kästchen zusammengefasst. Wo es möglich ist,

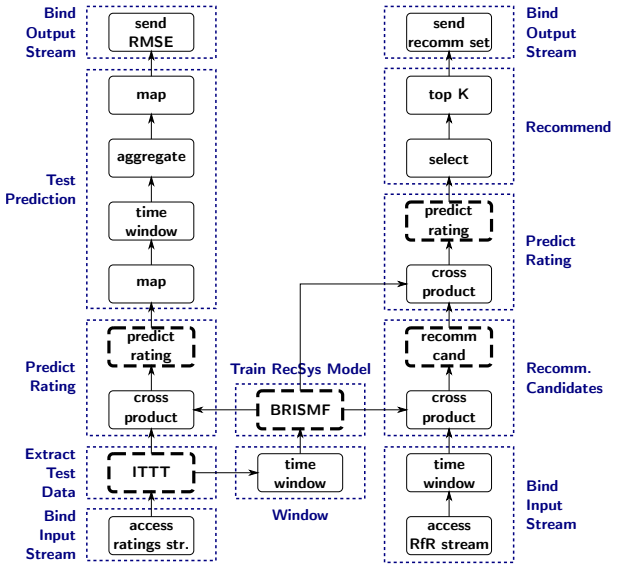

Abbildung 2: Physischer Operatorplan

werden die logischen Operatoren durch vorhandene physische Basisoperatoren (Operatoren mit durchgezogener Linie) implementiert. Neue, speziell für die RecSys-Funktion implementierte physische Operatoren (mit gestrichelter, dicker Linie) sind:

– **ITTT:** Impl. der Evaluationsmethode ITTT [8].
– **BRISMF:** Impl. des Lernalgorithmus' BRISMF [11].
– **RECOMM CAND:** Empfehlungskandidaten.
– **PREDICT RATING:** Schätzung von Bewertungen.

Im folgenden wird die Verarbeitung der Datenstromelemente anhand des physischen Anfrageplans aus Abbildung 2 näher erläutert. Dazu werden die drei Teilgraphen Lernen des Modells, Bestimmung der Empfehlungsmenge und Berechnung des Modellfehlers in jeweils einem Abschnitt behandelt.

Von Bewertungsdaten zu Modellen

Der ACCESS-Operator ist für die Anbindung von Datenströmen zuständig. Mit ihm werden das Transportprotokoll sowie das Datenformat festgelegt. Aus jedem ankommenden Datenstromelement wird ein Tupel in der Form (u, i, r) erzeugt, das die Benutzer-ID u, die Objekt-ID i sowie die Bewertung r des Benutzers u für das Objekt i enthält. Außerdem wird der Gültigkeitszeitraum des Datenstromelements festgelegt. Gibt die Datenquelle einen Zeitstempel mit, so kann dieser als Startzeitpunkt des Gültigkeitsintervalls genutzt werden. Andernfalls wird der aktuelle Zeitpunkt als Startzeitpunkt genutzt. Bewertungsdaten sind initial unendlich gültig und erhalten somit als Endzeitpunkt des Gültigkeitsintervals den Wert ∞.

Der Operator ITTT implementiert die Evaluationsmethode *Interleaved Test-Than-Train* (ITTT), die aus den Bewertungsdaten Tupel als Lern- und Testdaten erzeugt. Als Lerndaten werden alle Bewertungsdaten ohne Änderung weitergeleitet. Auf die Erzeugung der Testdaten wird später bei der Beschreibung der Evaluation näher eingegangen.

Ein TIME-WINDOW-Operator beschränkt die Gültigkeit der Lerndaten. So kann zum Beispiel festgelegt werden, dass die Lerndaten in dem BRISMF-Operator nur für 30 Tage lang berücksichtigt werden sollen. Der BRISMF-Operator implementiert den Lernalgorithmus BRISMF [11], der ein Modell mit den neuen Daten aktualisiert. Er legt das Gültigkeitsintervall des Modells fest, sodass zu jedem Zeitpunkt genau ein Modell gültig ist. Der Operator muss sicherstellen, dass alle Lerndaten berücksichtigt wurden, die im Gültigkeitsintervall des Modells gültig sind.

Von Empf.-Anforderungen zu Empf.-Mengen

Zu jeder Empfehlungsanforderung eines Benutzers u soll eine Menge an Empfehlungen ermittelt werden. Dazu bindet ein ACCESS-Operator den Eingabedatenstrom mit den Anforderungen und ein TIME-WINDOW-Operator setzt deren Gültigkeit. Das Gültigkeitsintervall wird so gesetzt, dass diese nur zum Zeitpunkt der Empfehlungsanforderung gültig sind und nach Verarbeitung direkt verworfen werden.

Zur Bestimmung der Empfehlungskandidaten werden der Datenstrom der Empfehlungsanforderungen und der Datenstrom der Modelle, der von dem BRISMF-Operator erzeugt wird, mit einem Kreuzprodukt-Operator zusammengeführt. Dieser Operator berücksichtigt die Gültigkeitsintervalle, sodass nur Datenstromelemente mit überlappenden Intervallen zusammengeführt werden. Da der BRISMF-Operator Modelle in der Art erzeugt, so dass zu jedem Zeitpunkt genau ein Modell gültig ist, wird jeder Empfehlungsanforderung genau ein Modell zugeordnet. Der RECOMM-CAND-Operator bekommt anschließend die Anforderungen mit den passenden Modellen und erzeugt einen Datenstrom, der für jedes in dem Modell nicht bewertete Objekt i des Benutzers u ein Tupel (u, i) mit dem anfordernden Benutzer und dem nicht bewerteten Objekt enthält.

Im nächsten Schritt wird für jeden Empfehlungskandidaten eine Bewertung geschätzt. Dazu wird mit einem Kreuzprodukt-Operator zu jedem Empfehlungskandidaten das temporal passende Modell zugeordnet. Der PREDICT-RATING-Operator nutzt nun das jeweilige Modell zu Bestimmung der Schätzung und gibt zu jedem Empfehlungskandidaten ein Tupel $(u, i, \hat{r})$ aus, welches den Benutzer u, den Empfehlungskandidaten i und die geschätzte Bewertung $\hat{r}$ enthält.

Anschließend werden aus den bewerteten Empfehlungskandidaten die Objekte für die Empfehlungsmenge ausgewählt. In der Beispielanfrage in Abbildung 2 werden dazu mit einem SELECT-Operator die Kandidaten entfernt, deren geschätzte Bewertung kleiner als ein bestimmter Schwellwert ist (z. B. 3,5). Von den verbleibenden Kandidaten werden mit dem TOP-K-Operator die K Objekte mit den größten Bewertungen ausgewählt (z. B. $K = 8$) und an den Ausgabedatenstrom übergeben.

Von Bewertungsdaten zu Modellfehlerwerten

Für die Evaluation werden vom ITTT-Operator Testdaten erzeugt. Ein Testdatum (u, i, r) wird als ein Empfehlungskandidat i für einen anfragenden Benutzer u betrachtet und vom PREDICT-RATING-Operator um eine geschätzte Bewertung $\hat{r}$ erweitert. Die PREDICT-RATING-Operatoren für die Berechnung der Empfehlungen und für die Evaluation haben die gleiche Implementierung: Sie erhalten vom vorgelagerten CROSS-PRODUCT-Operator ein Tupel, welches einen Benutzer u, ein Objekt i, ein Modell $\hat{f}$ und evtl. beliebig weitere Attribute besitzt. Die Ausgabe ist in beiden Fällen ein Tupel mit dem Benutzer u, dem Objekt i, der geschätzten Bewertung $\hat{r}$ durch Anwendung des Modells $\hat{r} = \hat{f}(u, i)$ und alle weiteren Attribute des Eingabetupels

(ohne das Modell $\hat{f}$, welches nach der Berechnung von $\hat{r}$ im folgenden Verlauf nicht mehr benötigt wird). Für die Evaluation zählt zu den weiteren Attributen des Eingabetupels insbesondere die wahre Bewertung r, die auch im Ausgabetupel enthalten ist. Somit gibt der PREDICT-RATING-Operator für die Evaluation ein Tupel $(u, i, r, \hat{r})$ aus.

Im nachfolgenden Verlauf wird nun ein Fehlerwert berechnet. In unserer Beispielanfrage wird dazu ein gleitender RMSE berechnet. Dazu berechnet zuerst ein MAP-Operator den quadratischen Fehler $se = (r - \hat{r})^2$ der Schätzung. Anschließend wird mit einem TIME-WINDOW-Operator die gleitende Zeitspanne festgelegt, über die der Fehlerwert aggregiert werden soll (z. B. die letzten 24 Stunden). Der nachfolgende AGGREGATE-Operator berechnet das arithmetische Mittel der quadratischen Fehler, die sich in einem Aggregationsfenster befinden (z. B. alle Werte der letzten 24 Stunden). Abschließend berechnet der letzte MAP-Operator die Quadratwurzel aus dem aggregierten Fehlerwert, welches dem RMSE über das Aggregationsfenster entspricht. Diese Werte werden an den Ausgabedatenstrom übergeben.

Für die Evaluation ist es wichtig, dass zum Testen des Modells das zu testende Datum nicht im Modell enthalten ist. Eine einfache Möglichkeit dies sicherzustellen ist die Trennung von Test- und Lerndaten. Dies kann durch einen ROUTE-Operator realisiert werden, der zufällig 10 % der Daten als Testdaten und die restlichen Daten als Lerndaten weiterleitet. Der hier vorgestellte Operator implementiert stattdessen die Evaluationsmethode *Interleaved Test-Than-Train* [8]. Mit dieser werden alle Daten sowohl zum Lernen als auch zum Testen genutzt. Dabei wird ein Testdatum zuerst zum Testen des Modells genutzt und anschließend in das Modell integriert. Das hat die Vorteile, dass mehr Testdaten genutzt werden und keine Daten für das Lernen des Modells verloren gehen. Letzteres ist insbesondere wichtig, wenn ein RecSys im laufenden Betrieb evaluiert werden soll (in dem Fall würde das aussortieren von Daten als Testdaten die Schätzungen für die realen Benutzer beeinflussen) und wenn temporale Zusammenhänge in den Daten beim Modelllernen berücksichtigt werden sollen (in dem Fall würden fehlende Daten ggf. erschweren, die temporalen Zusammenhänge zu erkennen).

Um sicherzustellen, dass zur Schätzung der Bewertung eines jeden Testdatums ein Modell genutzt wird, dass nicht das Testdatum, aber ansonsten so viele Lerndaten wie möglich enthält, wird das Gültigkeitsintervall des Testdatums vom ITTT-Operator angepasst. Ist ein Lerndatum im Gültigkeitsintervall $[t_s, t_e)$ mit dem Startzeitpunkt t_s und dem Endzeitpunkt t_e gültig, so wird die Gültigkeit des Testdatums so gesetzt, dass dieses ungültig wird, gerade bevor das äquivalente Lerndatum gültig wird. Bei einem Lerndatum von einem Gültigkeitsintervall $[45, \infty)$ erhält das Testdatum das Gültigkeitsintervall $[44, 45)$. Es ist genau wie die Empfehlungsanforderungen nur ein Zeitpunkt gültig: zum Zeitpunkt $t_1 = 44$ ist es gültig und zum Zeitpunkt $t_2 = 45$ bereits ungültig (wegen des halboffenes Intervalls). Wie bereits weiter oben gefordert, soll ein Modell genau alle Lerndaten enthalten, welche im Gültigkeitsintervall des Modells ebenfalls gültig sind. Der BRISMF-Operator erzeugt also ein Modell mit dem Lerndatum, welches ab $t_2 = 45$ gültig ist und das vorherige Modell wird somit zum Zeitpunkt $t_2 = 45$ ungültig. Der CROSS-PRODUCT-Operator vor dem PREDICT-RATING-Operator der Evaluation führt nun unter Berücksichtigung der Gültigkeitsintervalle Testdatum und Modell zusammen. Da sich die Gültigkeitsintervalle des Testdatums und des Modells, welches das zum Testdatum äquivalente Lerndatum enthält, nicht überschneiden, werden diese nicht zusammengeführt. Dafür überschneidet sich das vorherige Modell mit dem Testdatum und wird somit, wie bei der ITTT-Methode gefordert, zur Bewertungsschätzung genutzt.

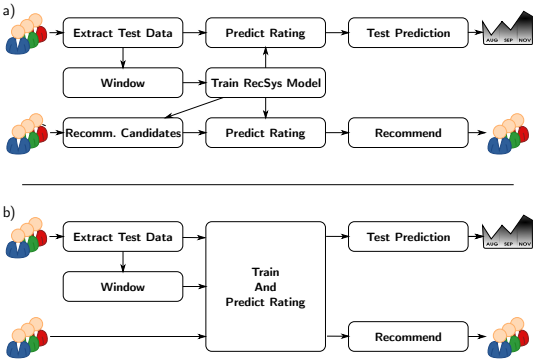

Abbildung 3: Gegenüberstellung: Drei Operatoren vs. ein Operator für die Bewertungsschätzung

6. DISKUSSION

Die Aufteilung der Bewertungsschätzung in drei Operatoren

Bei dem hier vorgestellten Konzept haben wir das RecSys-Problem der Schätzung von Bewertungen im Wesentlichen auf drei Operatoren aufgeteilt: TRAIN RECSYS MODEL, RECOMM. CANDIDATES und PREDICT RATING. Das hat den Nachteil, das eine Kopie des gelerntes Modells vom TRAIN-RECSYS-MODEL-Operator an die anderen beiden Operatoren als Datenstromelement übertragen werden muss. Das führt, insbesondere bei großen Modellen, zu einem zusätzlichen Aufwand. Eine Alternative wäre, alle drei Operatoren zu einem Operator zu vereinen (vgl. Abbildung 3). Dies hätte folgende Nachteile:

Durch die Aufteilung in drei Operatoren kann jeder Operator für sich verschiedene Implementierungen (physische Operatoren) haben. Zum Beispiel kann man sich ein RecSys für Filme vorstellen, bei dem der Benutzer angeben kann, dass er z. B. eine Komödie und keinen Actionfilm sehen möchte. Dies würde die Empfehlungsanforderung um eine Genre-Angabe erweitern. Ein für diese Anwendung angepasster RECOMM-CAND-Operator kann von vornherein als Empfehlungskandidaten nur die Objekte berücksichtigen, die unter die Kategorie *Komödie* fallen. Alternativ könnte man zwischen RECOMM CAND und PREDICT RATING einen Selektionsoperator einfügen, der alle Kandidaten, die nicht unter *Komödie* fallen, herausfiltert. Die anderen Operatoren bleiben davon unberührt. Die Aufteilung sorgt somit für klare Zuständigkeiten der Operatoren und ermöglicht eine einfachere Erweiterung der Anfrage.

Der TRAIN-RECSYS-MODEL-Operator bestimmt die Gültigkeit eines Modells, so dass zu jeden Zeitpunkt genau ein Modell gültig ist. Durch die Zusammenführung von Modell und Empfehlungsanforderung durch einen Kreuzprodukt-Operator findet eine deterministische, temporale Zuordnung statt. Diese Zuordnung basiert alleine auf den Gül-

tigkeitsintervallen und ist nicht davon abhängig, ob aufgrund der Steuerung des Schedulers oder anderer Latenzen zuerst das Lerndatum oder die Empfehlungsanforderung den Operator erreicht. Das Zeitmodell des DSMS sorgt dafür, dass genau das zuständige Modell reproduzierbar für die Vorhersage genutzt wird. Des Weiteren kann der TRAIN-RECSYS-MODEL-Operator das Modell für den Gültigkeitszeitraum anpassen, z. B. um temporale Verzerrungen aufgrund von Concept Drift zu berücksichtigen.

Die Ausgabe der Empfehlungskandidaten

Der RECOMM-CAND-Operator gibt für jeden Empfehlungskandidaten einer Empfehlungsanforderung ein Datenstromelement aus. Eine Alternative wäre die Ausgabe eines Datenstromelements je Empfehlungsanforderung, welches eine Liste von Empfehlungskandidaten enthält. Das hätte den Nachteil, dass der PREDICT-RATING-Operator für die Bestimmung der Empfehlungen und für die Evaluation andere Eingabedaten verarbeiten können müsste: im ersten Fall eine Liste von Benutzer-Objekt-Paaren, im zweiten Fall einzelne Benutzer-Objekt-Paare.

Der RECOMM-CAND-Operator gibt nur die Empfehlungskandidaten ohne Modell aus. Dieser Operator könnte auch das Modell den Empfehlungskandidaten beifügen, so könnte man auf den CROSS-PRODUCT-Operator vor PREDICT RATING verzichten. Das hier vorgestellte Konzept hat den Vorteil, dass BRISMF an RECOMM CAND und PREDICT RATING unterschiedliche Modelle übergeben kann. Wenn zum Beispiel aus dem Modell für die Schätzung der Bewertung gar nicht hervorgeht, welche Objekte ein Benutzer nicht bewertet hat (und somit zu den Empfehlungskandidaten gehört), so kann an RECOMM CAND ein Modell übergeben werden, dass die Zuordnung von Benutzer zu unbewerteten Objekten ermöglicht, dafür aber keine Schätzung der Bewertung vornehmen kann (z. B. ein einfaches Mapping $u \mapsto$ unbewertete Objekte).

7. IMPLEMENTIERUNG UND EVALUATION

Das vorgestellte Konzept haben wir mit dem erweiterbaren Open-Source-DSMS *Odysseus*[1] [2] umgesetzt. Dazu haben wir Odysseus um neue Operatoren und Transformationsregeln erweitert.

Die Machbarkeit des Konzepts und die korrekte Implementierung konnten wir durch den Vergleich der Evaluationsergebnisse mit MOA und dem MovieLens-Datensatz nachweisen. Dazu haben wir als BRISMF-Operator die MOA-Implementierung des BRISMF-Algorithmus' [11] integriert, der eine inkrementelle Matrix-Faktorisierung implementiert. Da MOA die Gültigkeit von Lerndaten nicht begrenzt, haben wir den WINDOW-Operator entfernt und die RMSE über den gesamten Datensatz berechnet (kein *gleitender* RMSE). Den MovieLens-Datensatz haben wir, wie MOA, zeilenweise eingelesen um einen Datenstrom zu simulieren und haben nach jedem getesteten Tupel den RMSE ausgegeben. Die RMSE-Werte stimmen dabei mit denen von MOA exakt überein. Das zeigt, dass die temporale Zuordnung von Lern- und Testdaten mit den Modellen sowie die Umsetzung der Evaluationsmethode korrekt funktionieren.

[1] `http://odysseus.informatik.uni-oldenburg.de/`

8. ZUSAMMENFASSUNG UND AUSBLICK

In dieser Arbeit haben wir ein Konzept für die Umsetzung eines modellbasierten und kollaborativen RecSys mit einem auf Operatoren basierten DSMS vorgestellt. Dazu haben wir logische Operatoren definiert, die für Teilaufgaben eines RecSys zuständig sind. Zu einem Beispiel eines logischen Anfrageplans für ein RecSys haben wir eine beispielhafte Umsetzung durch einen physischen Anfrageplan gezeigt und die Umsetzung und deren Alternativen diskutiert. Unser Konzept haben wir durch eine Implementierung in dem DSMS Odysseus und dem Vergleich der Evaluationsergebnisse mit MOA validiert.

Um die Praxistauglichkeit nachzuweisen, soll das Konzept in Zukunft unter realen Bedingungen, z. B. in einem Living Lab wie Newsreel[2] [9], evaluiert werden. Dazu stehen insbesondere die für den Datenstromkontext wichtige Parameter Latenz, Durchsatz und Speicheranforderungen im Vordergrund. Des Weiteren sollen weitere Algorithmen für die Empfehlungen und die Evaluation (z. B. Ranking-basierte Methoden) implementiert werden.

9. REFERENCES

[1] M. Ali, C. C. Johnson, and A. K. Tang. Parallel collaborative filtering for streaming data. *University of Texas Austin, Tech. Rep*, 2011.

[2] H.-J. Appelrath, D. Geesen, M. Grawunder, T. Michelsen, and D. Nicklas. Odysseus: A highly customizable framework for creating efficient event stream management systems. In *DEBS'12*, pages 367–368. ACM, 2012.

[3] A. Arasu, S. Babu, and J. Widom. The CQL continuous query language: semantic foundations and query execution. *VLDB Journal*, 15(2):121–142, 2006.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS 2002*, pages 1–16. ACM, 2002.

[5] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. *The Journal of Machine Learning Research*, 11:1601–1604, 2010.

[6] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. StreamRec: A Real-time Recommender System. In *SIGMOD'11*, pages 1243–1246. ACM, 2011.

[7] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-Time Top-N Recommendation in Social Streams. In *ACM RecSys*, pages 59–66, 2012.

[8] J. Gama, I. Zliobaite, A. Biefet, M. Pechenizkiy, and A. Bouchachia. A Survey on Concept Drift Adaptation. *ACM Comp. Surveys*, 1(1), 2013.

[9] F. Hopfgartner, B. Kille, A. Lommatzsch, T. Plumbaum, T. Brodt, and T. Heintz. Benchmarking news recommendations in a living lab. In *CLEF'14*, LNCS, pages 250–267. Springer Verlag, 09 2014.

[10] X. Luo, Y. Xia, and Q. Zhu. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems*, 27:271–280, 2012.

[11] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[2] `http://www.clef-newsreel.org/`