Item-based Collaborative Filtering

Final Presentation

Martin Krüger, Sebastian Kölle

07.07.2011

Seminar Collaborative Filtering

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Intermediate presentation

Vorhersage basierend auf der Hierarchie

- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

Final presentation

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Vorhersage basierend auf der Hierarchie

- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

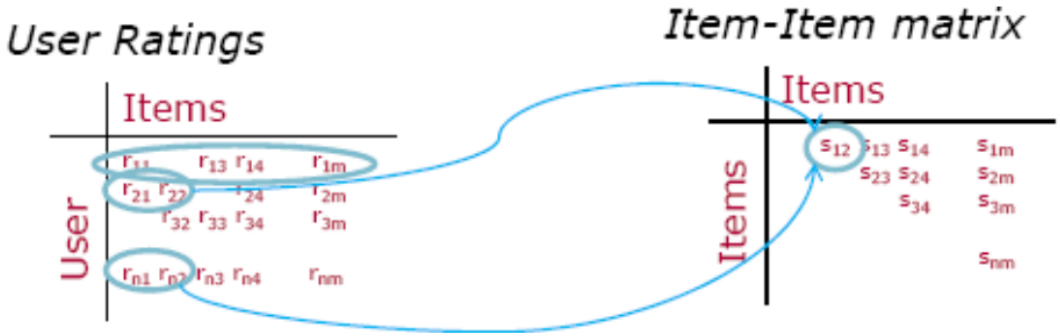
Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

Item-based Collaborative Filtering

Vorberechnung

Erstelle eine Item-Item Matrix, berechne dabei die Ähnlichkeit jedes Item-Paares unter Verwendung eines Ähnlichkeitsmaßes (*Cosinus-based*, *Correlation-based* oder *Adjusted Cosine similarity*).



Vorhersage

Gegeben: User **u**, Item **i**. Gesucht: Rating $\mathbf{r_{ui}}$

1. Finde die **K** zu **i** ähnlichsten Nachbarn **N(i;u)**, die von **u** bewertet wurden.
2. Berechne den gewichteten Mittelwert auf Basis der Ähnlichkeiten oder berechne das Rating mit einem Regressionsmodell.

[1, S. 6 ff]

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Vorhersage basierend auf der Hierarchie

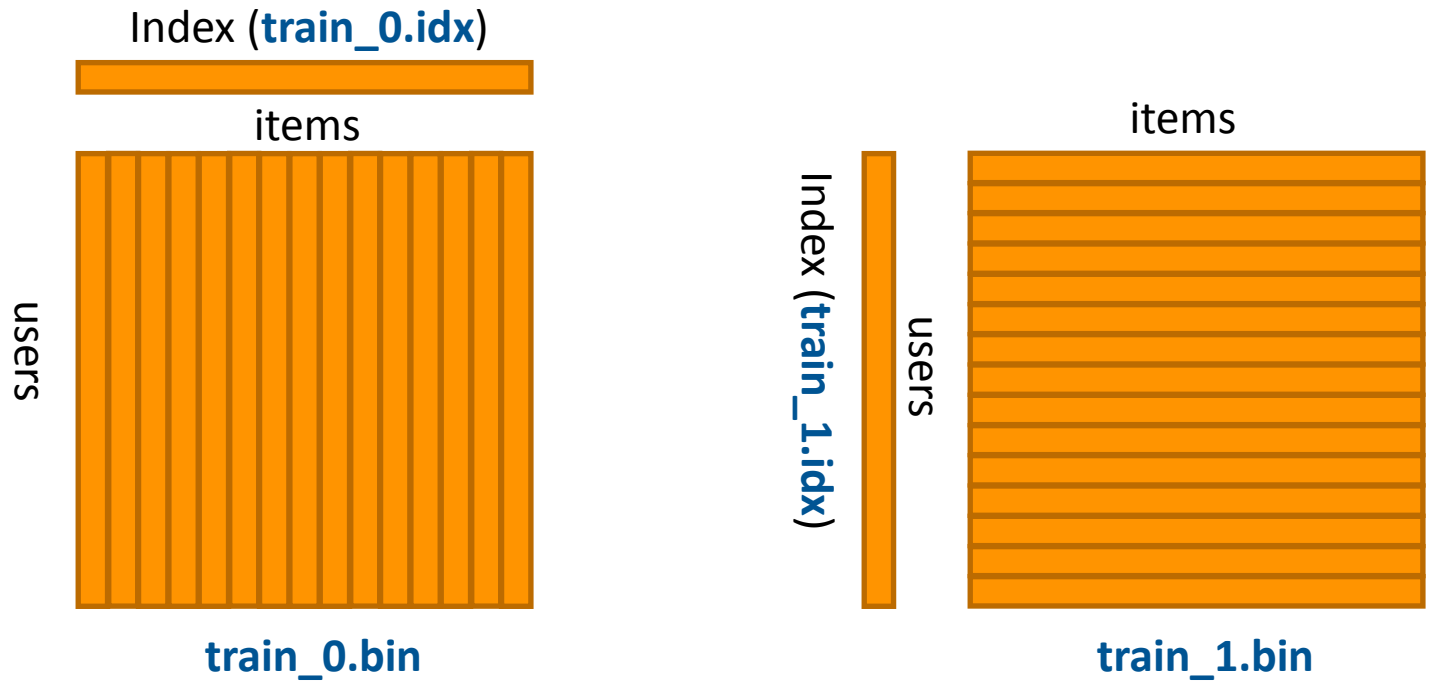
- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

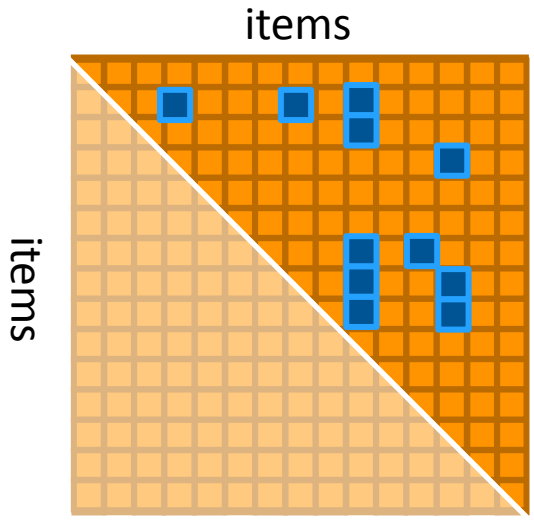
Optimierung: Speicherbedarf

Vollständige User-Item-Matrix wäre ca. **582 GB** groß



Lösung: speicher-effizientes Matrix-Format (**1,9 GB**)

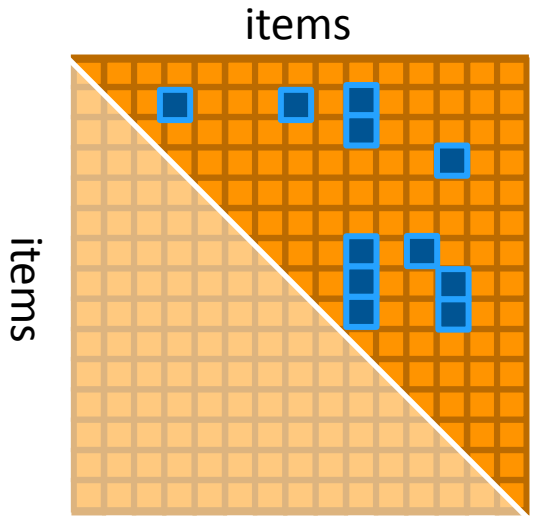
Optimierung: Hybrid-Ansatz



1) Vorberechnung

- Ähnlichkeit für die 89.273.180 (10%) „teuersten“ Item-Paare (683,5 MB)
- cost(pair) = count(pair) * (ratingCount(pair.item1) + ratingCount(pair.item2))

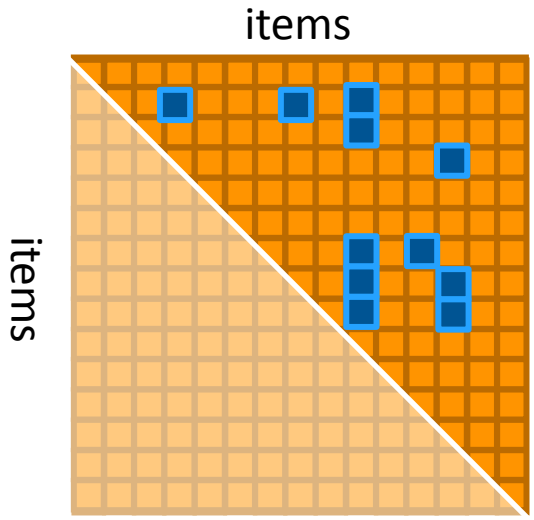
Optimierung: Hybrid-Ansatz



2) Vorhersage

- Wenn Ähnlichkeit nicht vorberechnet, berechne sie ad-hoc

Optimierung: Hybrid-Ansatz



Ergebnisse

Ermitteln der teuersten Paare	**8 Stunden**	
Vorberechnung	**4 Stunden**	
Prediction auf Validation	**2 Stunden**	(RMSE: **24,37**)
Prediction auf Test	**3 Stunden**	(RMSE: **26,48**)

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Vorhersage basierend auf der Hierarchie

- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

Hierarchie

Ermittlung eines besseren **Fallback-Ratings** mit Hilfe der Typ-Informationen

Ohne Typ-Informationen

- Durchschnittliche Item-Bewertung
- Durchschnittliche Nutzer-Bewertung
- Globaler Bewertungsdurchschnitt

Mit Typ-Informationen

- **Durchschnittliche Nutzer-Bewertung für Items des Typs**
- Durchschnittliche Item-Bewertung
- Durchschnittliche Nutzer-Bewertung
- Globaler Bewertungsdurchschnitt
- **Globaler Bewertungsdurchschnitt für Items des Typs**

Validation-RMSE 26,39 → 25,48

Hierarchie

Track:
- Bewertung des Albums
- **Bewertung des Künstlers**
- **Bewertung der Genres** **(Durchschnitt)**
- Bewertung aller Tracks des gleichen Albums (Durchschnitt)
- **Bewertung aller Tracks des gleichen Künstlers** **(Durchschnitt)**

Album:
- Bewertung der enthaltenen Items (Durchschnitt)
- Bewertung des Künstlers
- **Bewertung der Genres** **(Durchschnitt)**
- Bewertung von Alben des gleichen Künstlers (Durchschnitt)

Künstler:
- Bewertung aller Alben des Künstlers (Durchschnitt)
- Bewertung aller Tracks des Künstlers (Durchschnitt)

Genre:
- **Bewertung von Alben des Genres** **(Durchschnitt)**
- Bewertung von Tracks des Genres (Durchschnitt)

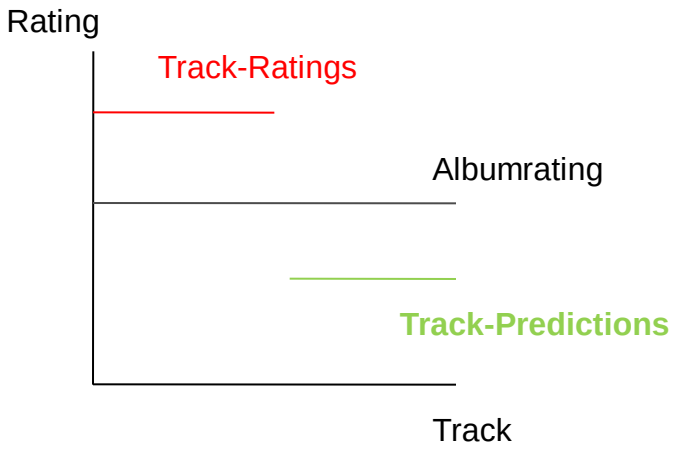
Hierarchie

Track

Kombination Album-Rating und Track-Ratings

- Album-Rating als Track-Durchschnitt angenommen
- Durchschnitt der fehlenden Ratings bestimmbar

Kombination Artist-Rating und Track-Ratings



Album

Kombination Artist-Rating und Album-Ratings

Track/Album

Vorhersagen für Vorhersagen

- Wenn Artist-Rating nicht gegeben, wird Vorhersage für den Artist verwendet.

Validation-RMSE	**25,48**	→	**24,74**

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Vorhersage basierend auf der Hierarchie

- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

Kombination der verschiedenen Verfahren

item-prediction	hierarchy-prediction	matrix-prediction	realrating
76,8627	54,9020	54,1176	70
81,9608	72,1569	92,1569	90
86,2745	85,0980	80,0000	70
...			

Lineare Regression mit Weka:

realrating =
0.4671 * itemprediction +
0.1259 * hierarchyprediction +
0.4537 * matrixprediction +
-4.7543

Kombination der verschiedenen Verfahren

item-prediction	hierarchy-prediction	matrix-prediction	realrating
81,9608	80,0000	80,0000	?
83,5294	87,8431	89,8039	?
78,0392	78,0392	70,1961	?
...			

Lineare Regression mit Weka:

realrating =
0.4671 * itemprediction +
0.1259 * hierarchyprediction +
0.4537 * matrixprediction +
-4.7543

Kombination der verschiedenen Verfahren

item-prediction	hierarchy-prediction	matrix-prediction	realrating
81,9608	80,0000	80,0000	**79,8976**
83,5294	87,8431	89,8039	**86,0658**
78,0392	78,0392	70,1961	**73,3709**
...			

Lineare Regression mit Weka:

realrating =
0.4671 * itemprediction +
0. diction +
0. tion +
-4.7543

KDD-RMSE

26,7070 → **25,6300**

Verbesserung: Clustering nach Items

Lineare Regression pro Item-Type

Track

realrating =
0.2768 * itemprediction +
0.3334 * hierarchyprediction +
0.4859 * matrixprediction +
-9.4909

Album

realrating =
0.4191 * itemprediction +
0.3868 * hierarchyprediction +
0.29 * matrixprediction +
-8.4196

Artist

realrating =
0.7295 * itemprediction +
-0.2627 * hierarchyprediction +
0.5064 * matrixprediction +
2.0478

Genre

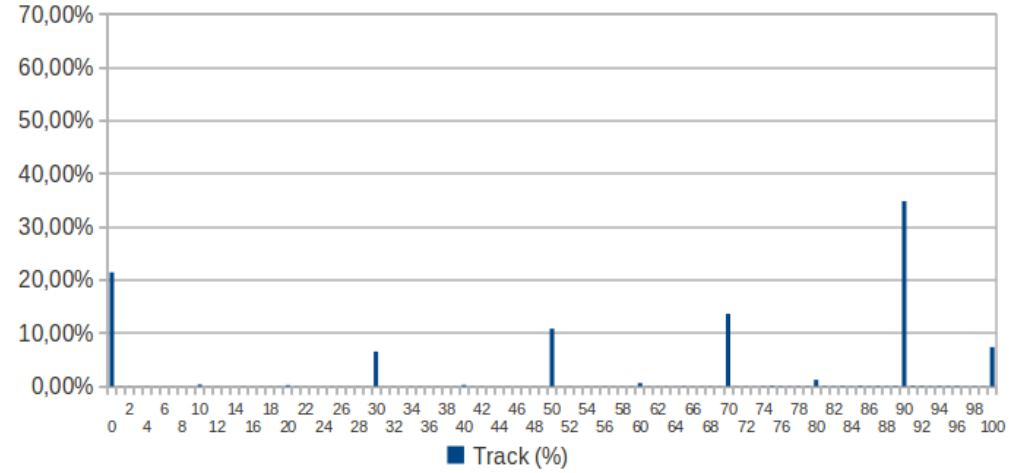
realrating =
0.7469 * itemprediction +
-0.0823 * hierarchyprediction +
0.3185 * matrixprediction +
-1.0081

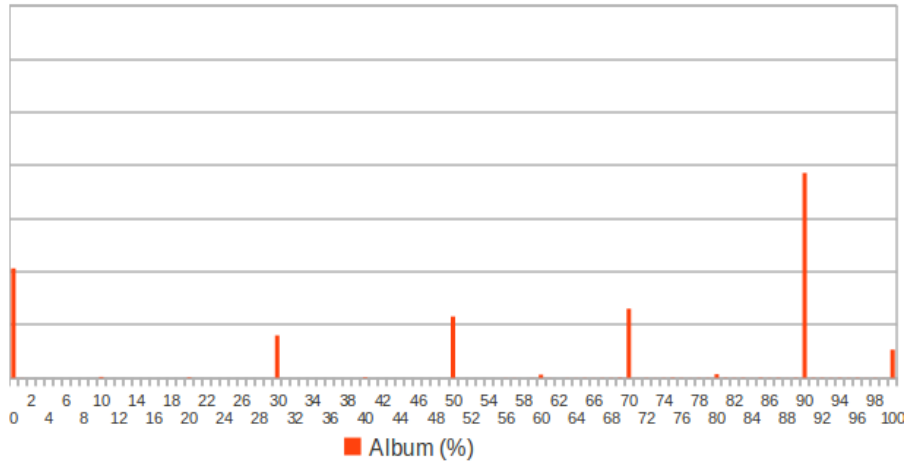
Clustering nach Items: Analyse

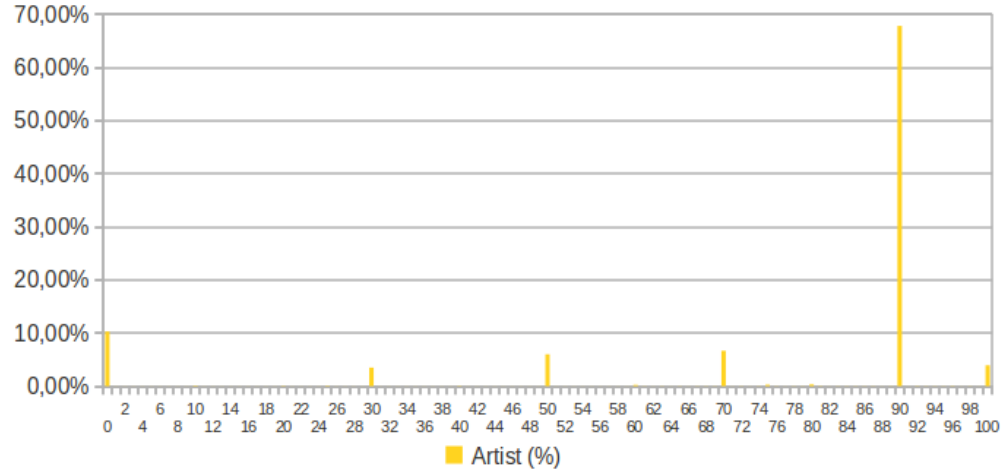
KDD-RMSE 25,6300 → 25,3884

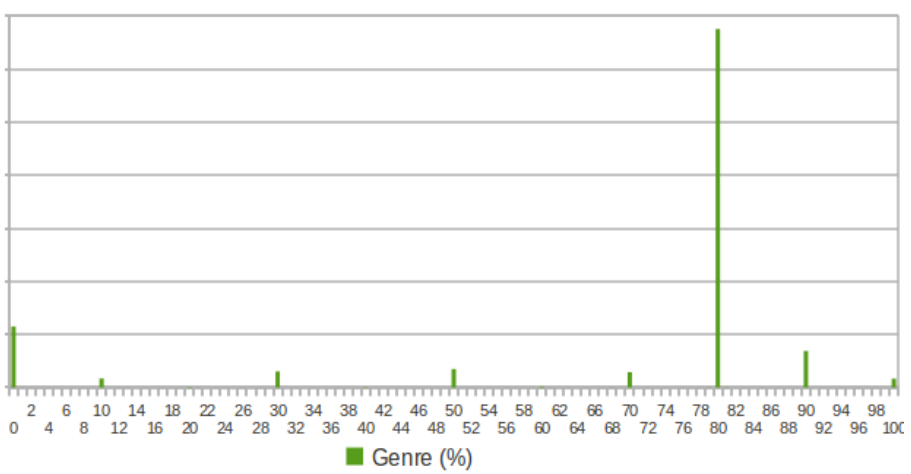
Item-Type	Cross-Validation RMSE	#Predictions Validation-Set (%)	#Predictions Test-Set (%)
Track	28,1786	26,58 %	28,70 %
Album	26,1160	10,62 %	11,01 %
Artist	20,6185	52,14 %	51,61 %
Genre	16,1906	10,66 %	8,68 %

Clustering nach Items: Analyse









Gute und schlechte Vorhersagen

	Gute Vorhersagen		Schlechte Vorhersagen	
	mean	stddev	mean	stddev
User Rating Count	212,8022	1040,7410	294,9270	1217,4103
User Rating Average	78,4742	20,4830	63,7396	21,3815
User Rating Std Dev	10,6213	11,2341	28,3794	11,6013
User Rating Distinct Value Count	3,8864	5,8530	6,9751	7,9889
User Rating Distinct Value Std Dev	51,7074	271,6368	61,8750	262,1981
Item Rating Count	99744,0391	86259,6328	58176,7031	87726,0234
Item Rating Average	61,8197	14,8202	55,3634	14,1593
Item Rating Std Dev	32,6378	3,6662	34,5511	3,7697
Item Rating Distinct Value Count	76,1745	29,9853	64,8470	32,5852
Item Rating Distinct Value Std Dev	6685,4473	6164,1992	3855,4983	6313,4126
Neighborhood Weight Sum	9,8049	9,2214	8,7827	9,0673
Neighborhood Size	33,1987	14,9796	41,7768	12,6120
Neighborhood Weight Std Dev	0,1428	0,0616	0,1307	0,0574
User 0-Rating Percentage	0,0748	0,1969	0,2142	0,2304
User 90-Rating Percentage	0,6503	0,3149	0,4331	0,2813
Item 0-Rating Percentage	0,2009	0,1534	0,2508	0,1448
Item 90-Rating Percentage	0,4439	0,2193	0,3391	0,1911

Automatisches Clustering nach Usern

- **Simple expectation maximisation clusterer** (EM)
- Training-Set: **50049 Instanzen** (5% Validation-Set)
- Attribute
  - userratingstddev
  - userratingdistinctvaluecount
  - user0ratingpercentage
  - user90ratingpercentage
- Ergebnis: **5 Cluster**

Clustering nach Usern: Analyse

KDD-RMSE 25,3884 → 25,3578

Cluster	Cross-Validation RMSE	#Predictions Validation-Set (%)	#Predictions Test-Set (%)
0	27,0928	5,47%	5,47%
1	27,7619	30,36%	30,36%
2	10,2791	26,34%	26,34%
3	31,0736	17,04%	17,04%
4	18,1050	20,79%	20,79%

Zusammenfassung

Item-Based Verfahren nach Sarwar et al.

- Vorberechnung der Item-Item-Matrix
- Adjusted Cosine Measure

Optimierung für große Datenmengen

- Datenstruktur
- Hybrid-Verfahren

Vorhersage basierend auf der Hierarchie

- Umgesetzte Ideen (Intermediate)
- Hinzugekommene Ansätze

Kombination der verschiedenen Verfahren

- Lineare Regression
- Clustering

Auswertung

Erfahrungen

- Item-based Collaborative Filtering auf großer Datenmenge
- Kombination mehrerer Verfahren bringt deutliche Vorteile
- Wichtig: Analyse der Daten

Verbesserungen

- Clustering
- User-based Collaborative Filtering
- Machine Learning
- Hierarchie optimieren


HPI

Anhang

Meilensteine

12. Mai	loader completely implemented and tested initial implementation presentation
15. Mai	basic item-item algorithm implementation finished first submission with complete dataset
26. Mai	global effects identified & removed *maybe:* split by item type *maybe:* neighborhood relationship model implemented
9. Juni	implemented hierarchy-based algorithm integration of different algorithms intermediate presentation
23. Juni	implementation finished parameters tweaked for optimal result final presentation
30. Juni	submission deadline

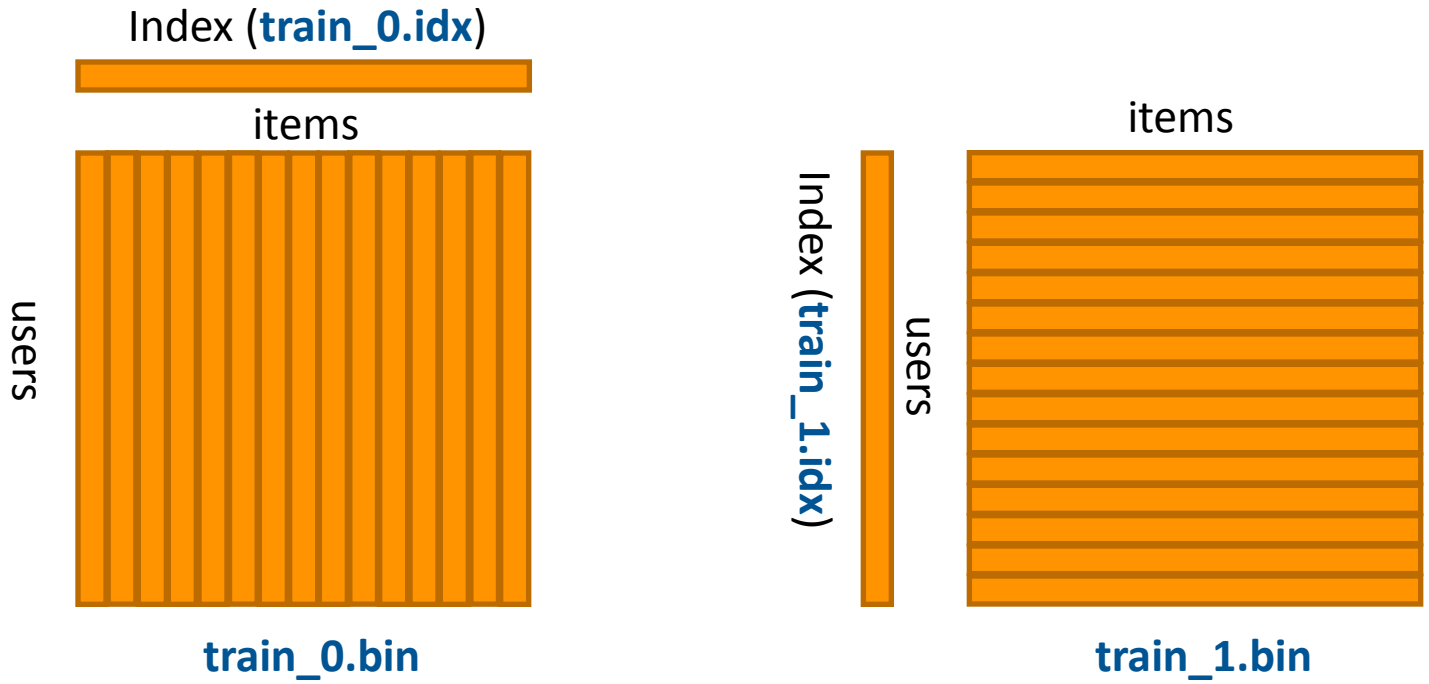
Effiziente Speicherung der User-Item Matrix

- **Problem**: Größe der User-Item-Matrix
  - 1.000.990 Users x 624.961 Items x 1 Byte ≈ 582,62 GB
- Benötigte **Funktionalität**:
  - Alle Ratings eines Users ermitteln
  - Alle Ratings eines Items ermitteln
  - Gezielt ein einzelnes Rating ermitteln

  } O(1)

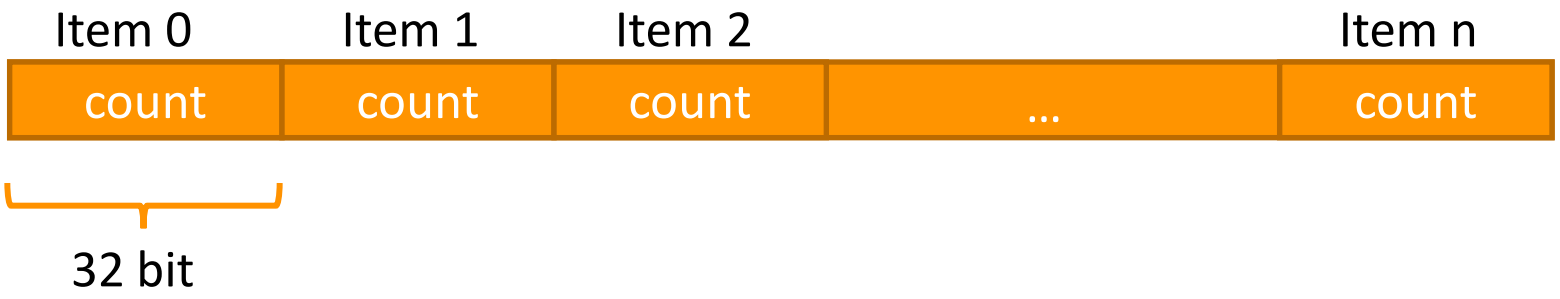
  } O(log (N/U))
- Aber: größtenteils gefüllt mit **Nullwerten**
  - Nur 0,04% der Matrixelemente haben einen Wert
- **Idee**:
  - Speichere nur die tatsächlich ‚gefüllten' Elemente
  - Verwende Indizes für effizienten Zugriff

Datenformat

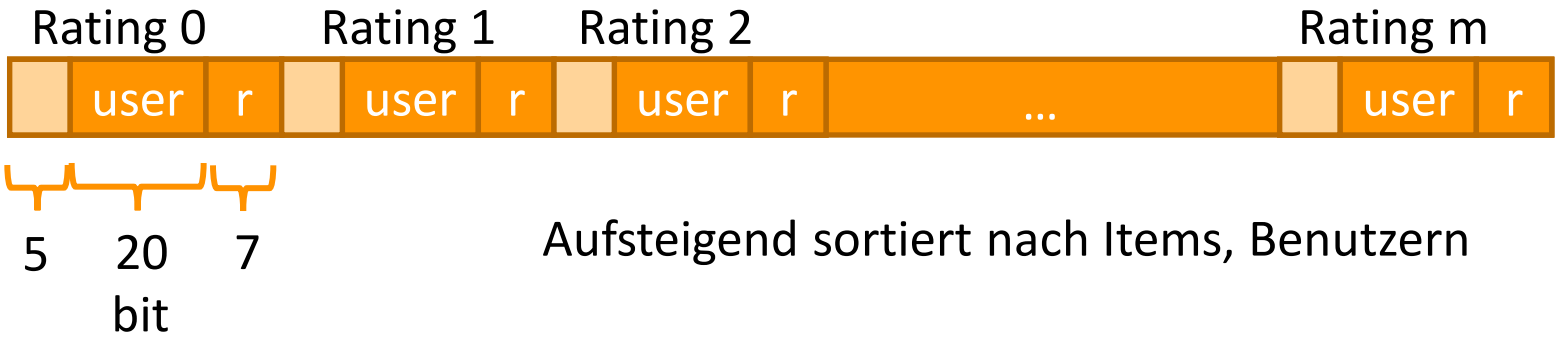


Matrix train_0 auf der Festplatte

train_0.idx
(2,4 MB)

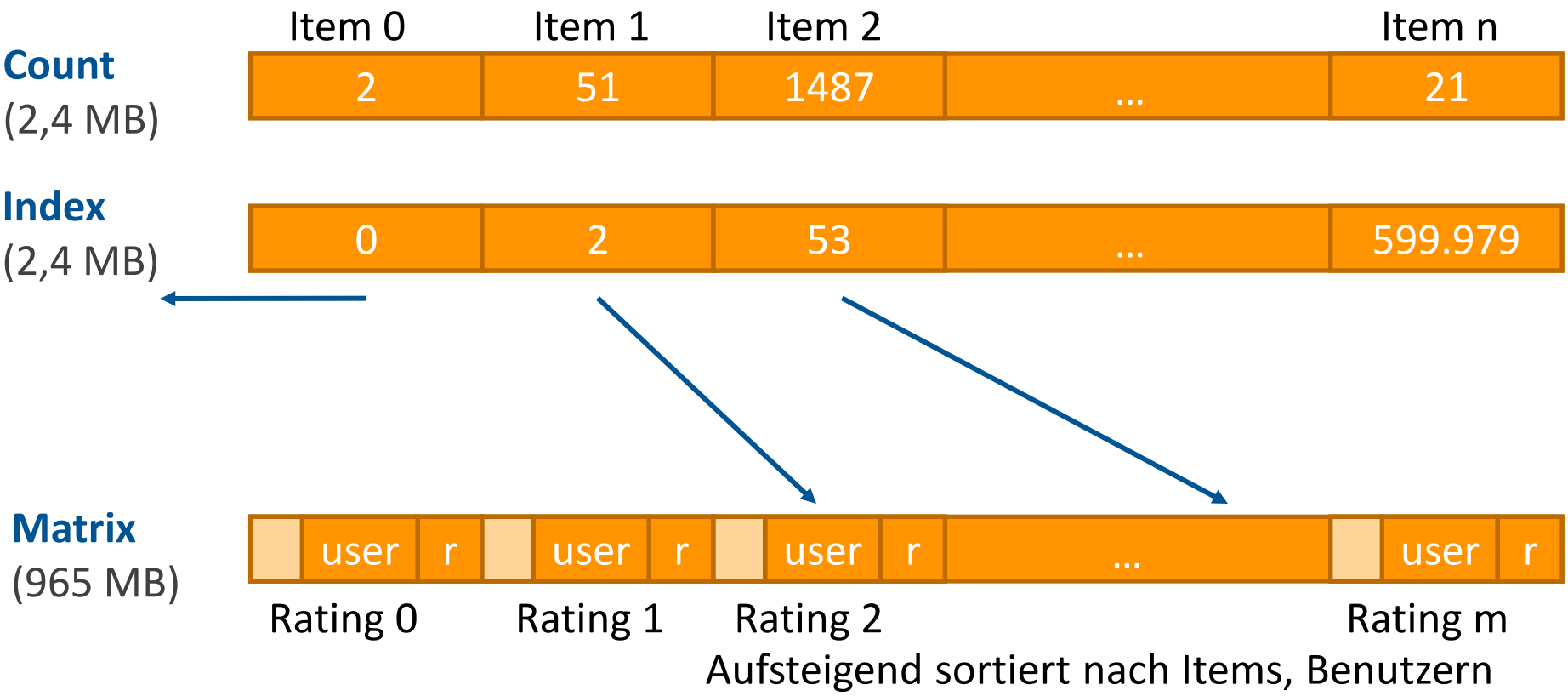


train_0.bin
(965 MB)



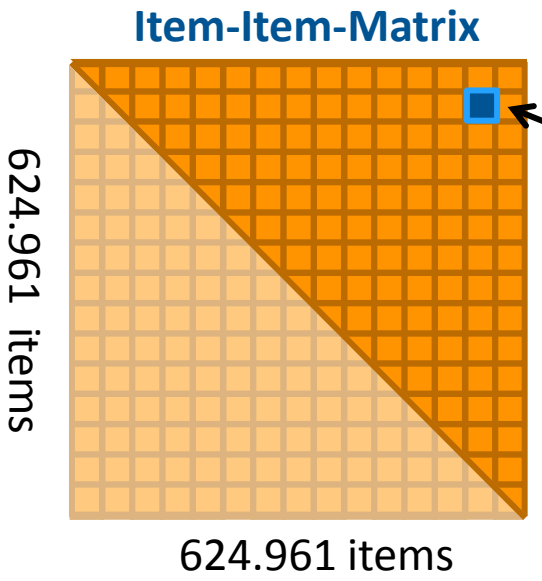
Aufsteigend sortiert nach Items, Benutzern

Matrix train_0 im Hauptspeicher



Herausforderung: Laufzeit (I)

1. Vorberechnung der Item-Item-Matrix



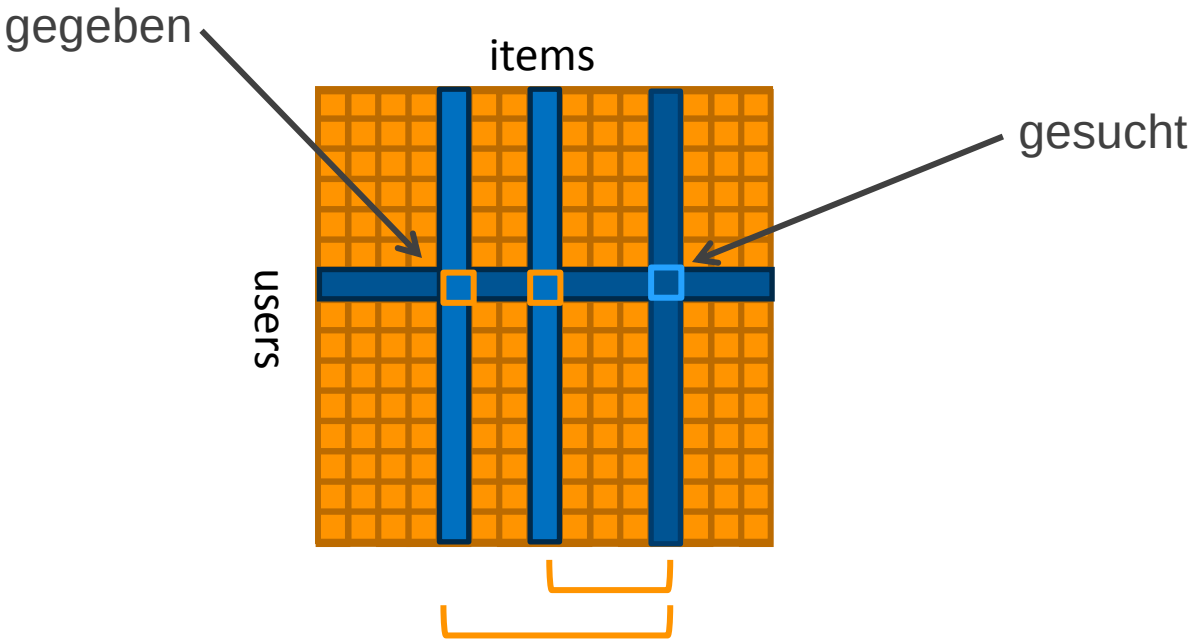
4 Byte Float
0,45 ms Berechnung

Insgesamt

Speicherbedarf	**727,5 GB**
Laufzeit	**1017,1 Tage**

Herausforderung: Laufzeit (II)

2. Ad-Hoc Berechnung



Sample mit 1000 Usern (von 1.000.990):

- **4 - 10 min** C, Single-Threaded
- **15 - 35 min** Java, 4 Threads

Gesamter Datensatz (Hochrechnung):

- **> 67 Stunden** Validation-Set
- **> 100h** Test-Set

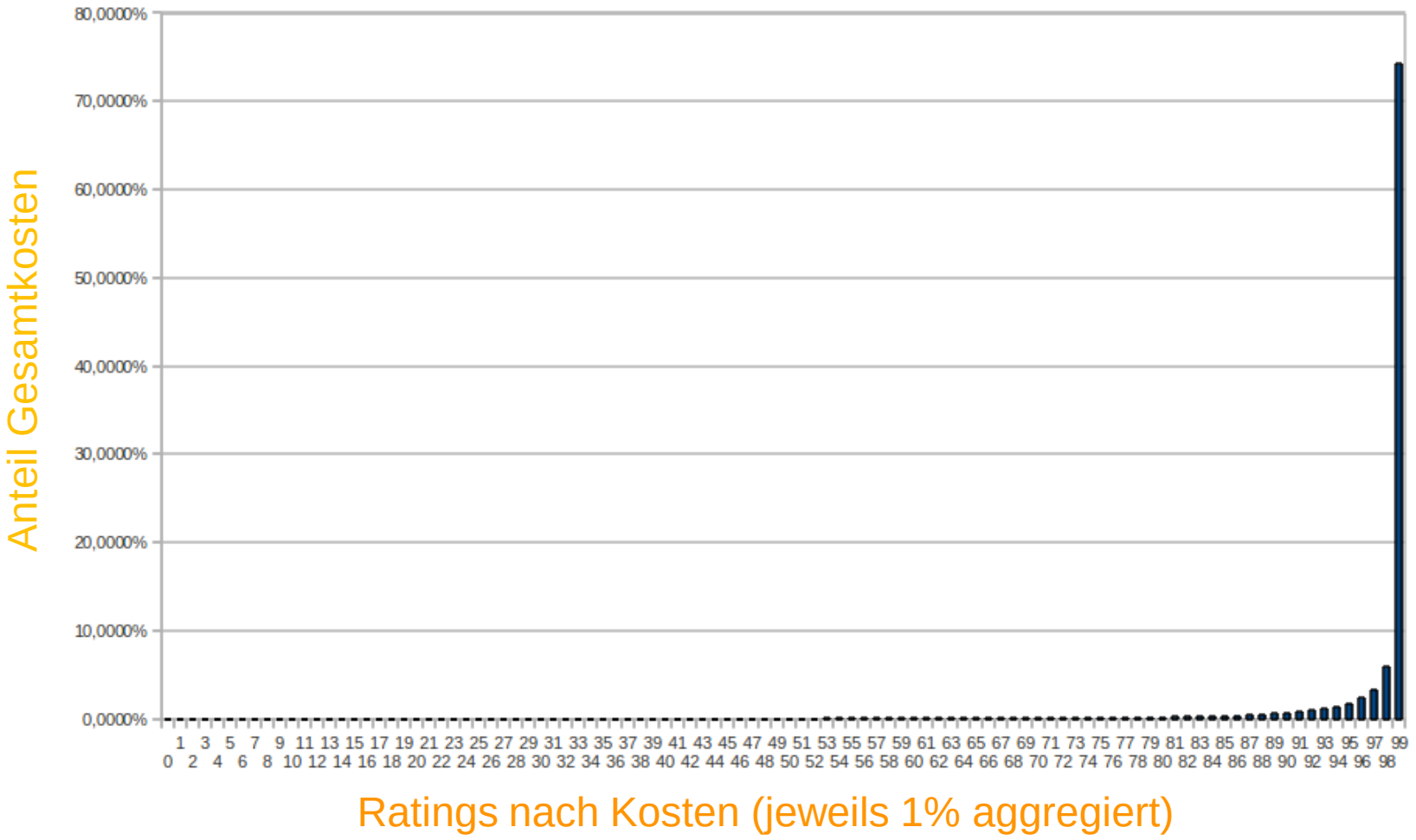
Beobachtungen

1. Bottleneck: Ähnlichkeitsberechnung (**> 90%**)

2. **Pareto-Prinzip**: Ähnlichkeit weniger Item-Paare wird sehr häufig berechnet

3. Aufwand für die Ähnlichkeitsberechnung: **O(n + m)**

cost(pair) = count(pair) * (ratingCount(pair.item1) + ratingCount(pair.item2))



Herausforderung: Laufzeit (III)

3. Hybrid-Ansatz

1) Vorberechnung:

Ähnlichkeit für die 89.273.180 (10%) „teuersten“ Item-Paare (683,5 MB)

2) Prediction:

Wenn Ähnlichkeit nicht vorberechnet, berechne sie ad-hoc

Ergebnisse

Ermitteln der teuersten Paare	**8 Stunden**	
Vorberechnung	**4 Stunden**	
Prediction auf Validation	**2 Stunden**	(RMSE: **24,37**)
Prediction auf Test	**3 Stunden**	(RMSE: **26,48**)

Geplant

Hierarchy-Based

- Ermittlung verschiedener simpler Vorhersagen anhand der Hierarchie
- Gewichteter Mittelwert als Hierarchy-Based Prediction

	Album	Artist	Genre	Track
Items	88909	27888	992	507172
Items in validation	44546	18623	749	194776
Items in test	50690	20398	790	238031
Ratings in validation	425278	2087772	426761	1064149
Ratings in test	661317	3099881	521122	1723620