Recommender Systems - Homework 01

Klappert Lukas, Scheffknecht Daniel, Sum Stephan

1. TASK 1

Beschreibung	schedl_icmr	Unser Datenset
Anzahl User ungefiltert	120322	2533
Anzahl User gefiltert	120175	2201
Anzahl Artists ungefiltert	3190371	40640
Anzahl Artists gefiltert	585095	4826
Altersdurchschnitt	25,4	0,0
Geschlechterverteilung	53.65 n/a, alle anderen:	71.66% m, 28.33% f, 100.0% n/a

Table 1: Vergleich der Datensets

Um unsere User-Basis zu generieren, haben wir den User.getfriends() - Call der LastFM API genutzt. Dabei wird ein Username als Parameter übergeben, zurückgeliefert wird eine Liste von allen FreundInnen dieses Users. Wir haben die Seed-User aus der Datei lastfm_users_100.csv genutzt, sowie die daraus resultierenden User, um deren FreundInnen zu bekommen. Insgesamt haben wir auf diese Weise 2533 NutzerInnen erhalten, von denen wir die "inaktiven" NutzerInnen wieder aussortiert haben. Als "inaktiv" bezeichnen wir NutzerInnen mit einem geringeren Playcount als 1000. Nach diesem ersten Cleansing-Schritt erhielten wir noch 2347 NutzerInnen.

Für jeden dieser NutzerInnen sammelten wir anschließend mittels User.getrecenttracks() die letzten 1000 Listening Events. Durch diese konnten wir nochmals NutzerInnen (sowie deren Listening Events) herausfiltern, die weniger als 10 Artists gehört haben. Durch diesen Cleansing-Schritt erhielten wir 1.726.505 Listening Events von 40.640 Artists. Nur 4.826 der 40.640 Artists wurden jedoch von mehr als 10 NutzerInnen gehört, weshalb wir die Listening Events dieser Artists wiederum herausgefiltert haben.

Insgesamt erhielten wir nach dem Cleansing unserer Daten 1.175.884 Listening Events, 2201 User und 4826 Artists.

Sowohl das Alter, als auch das Geschlecht für diese NutzerInnen ist unbekannt ("0" bzw. "n"), weshalb wir diese Charakteristiken nicht mit denen aus dem LFM-1b Datenset vergleichen können. Wir vermuten, dass diese Daten grundsätzlich aus der LastFM API entfernt wurden.

Jedoch können wir Aussagen über die Verteilung des Herkunftslandes machen. Wie auch im LFM-1b Datenset kommen die meisten NutzerInnen aus den USA (33,23%), gefolgt von Brasilien (13,00%), UK (12,52%) und Polen (5,94%) mit jeweils mehr als 100 NutzerInnen. All diese Länder tauchen auch in den Top 6 des LFM-1b Datenset auf. Die ähnliche Verteilung ist darauf zurückzuführen, dass wir einige Nutzer des LFM-1b Datensets als Seed-User genutzt haben. Wir gehen dabei davon aus, dass die meisten Freunde aus dem gleichen Land kommen.

Eine weitere, interessante Statistik: Der Track "Do I Wanna Know?" von den Arctic Monkeys ist der meistgehörte in unserem Datenset. Auch bei Spotify ist dieser Track der meistgehörte von diesem Artist, was wir als Indiz für die Validität unseres Datensets sehen. Insgesamt liegen die Arctic Monkeys jedoch nur auf Rang 3 unserer beliebtesten Artists, hinter den Beatles (Platz 2, 8596 Listening Events) und Radiohead (Platz 1, 11382), die fast 1% aller Listening Events ausmachen!

Item	Number
Users	2201
Artists	4826
Tracks	136917
Listening Events	1175884
Unique <user, artist> pairs	193703

Table 2: Statistik zu unserem Datenset.

Land	Anzahl User	Anzahl User in Prozent
United States	621	33,23%
Brazil	243	13,00%
United Kingdom	234	12,52%
Poland	111	5,94%
Mexico	70	3,75%
Germany	64	3,32%
Russian Federation	62	3,32%
Chile	50	3,68%
n/a	332	15,08%

Table 3: Länderverteilung unseres Datensets (nur Länder mit mindestens 50 NutzerInnen)

Da wir die Listening Events im gleichen Format gespeichert haben wie in den vorgegebenen Daten, konnten wir

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

auch den bereits vorhandenen UAM Converter nutzen. Dabei werden die User, sowie deren Playcounts pro Artist in eine Matrix umgewandelt. Anschließend wird diese Matrix pro User normalisiert, sodass bei der späteren Empfehlung NutzerInnen mit besonders hohem Playcount nicht höher gewichtet werden, als NutzerInnen mit niedrigem Playcount.

2. TASK 2

Nachdem die Datenakquise abgeschlossen ist, möchten wir den CF Recommender aus der Laborübung erweitern und zwar so, dass er anstelle des ähnlichsten Nachbarn nun k ähnliche Nachbarn bei der Vorhersage von Artists berücksichtigt.

Die Ähnlichkeiten des Users zu seinen Nachbarn werden wieder über das Innere Produkt berechnet. Die resultierenden Ähnlichkeitswerte werden aufsteigend nach Größe sortiert. Der letzte Eintrag in der Liste ist dabei der User selbst, da er zu sich selbst die höchste Ähnlichkeit aufweist. Wir selektieren also die letzten k Nachbarn unserer sortierten Liste und stellen sicher, dass der User selbst nicht Teil der Selektion ist.

Im nächsten Schritt ermitteln wir die für die Vorhersage in Frage kommenden Artists. Vorschlagen möchten wir nur Artists die von den Nachbarn auch wirklich gehört wurden. Gleichzeitig möchten wir keine Artists vorschlagen, die der User bereits kennt. Wir bilden hierzu eine Liste aller gehörten Artists der Nachbarn und bilden die Mengendifferenz mit der Liste aller gehörten Artists des Users. Anschließend werden Mehrfacheinträge entfernt, so dass wir eine distinkte Auflistung potentieller Artists erhalten.

Für alle Artists dieser Liste berechnen wir nun einen Artist-Score, der quasi das „Herzstück“ des Algorithmus darstellt. In die Berechnung fließen folgende Überlegungen mit ein:

1. Wie oft wurde der Artist von allen Nachbarn gehört?
2. Wie oft wurde der Artist von einem ähnlichen/weniger ähnlichen Nachbarn gehört?
3. Wie viele unterschiedliche Nachbarn haben diesen Artists gehört?

(1) ließe sich schnell verwirklichen, allerdings wollen wir nicht einfach nur alle Playcounts eines Artists über alle Nachbarn aufsummieren, sondern wir möchten zusätzlich berücksichtigen, ob der Playcount den wir gerade aufsummieren, nun von einem besonders ähnlichen Nachbarn stammt, oder von einem etwas weniger Ähnlichen (2). Daher gewichten wir die einzelnen Summanden der Artist Playcounts mit dem Ähnlichkeitswert des betrachteten Nachbarn zum User. Im letzten Schritt (3) ermitteln wir wie viele unterschiedliche Nachbarn diesen Artists gehört haben, normalisieren diesen Wert, indem wir durch die Gesamtanzahl der ähnlichen Nachbarn dividieren (also k) und rechnen den entstandenen Faktor ebenfalls mit in den Artist Score ein.

Die errechneten Artists Scores werden der Größe nach absteigend sortiert. Möchte man dem Nutzer p Artists empfehlen, gibt man die ersten p Artists der sortierten Liste an den Benutzer zurück.

Um die Performance unseres CF bei der Evaluation mit anderen Recommender Systems vergleichbar zu machen, implementierten wir einen zusätzlichen Random Baseline Algorithmus.

	A1	A2	A3	A4
N1 (0.2)	0	4	5	0
N2 (0.3)	2	8	0	0
N3 (0.7)	0	2	4	6

$$Playcount_{A3} = (0.2 * 5) + (0.3 * 0) + (0.7 * 4)$$

$$UserArtistCount_{A3} = \frac{2}{3} \quad \textit{(2 von 3 Nachbarn, haben A3 gehört)}$$

$$Score_{A3} = Playcount_{A3} * UserArtistCount_{A3}$$

Figure 1: Exemplarische Berechnung eines Artist-Scores

Zunächst wählen wir einen zufälligen User aus der Menge aller User (abzüglich des Nutzers für den die Empfehlung erfolgen soll). Wir ermitteln nun alle gehörten Artists des zufällig gewählten Users und bilden die Mengendifferenz zu den gehörten Artists unseres Users. Aus der entstandenen Artist Liste werden nun k zufällige Artists ausgewählt und dem Benutzer empfohlen.

3. TASK 3

Wir evaluieren die Algorithmen im Python-File `Evaluate_Recommender.py`. Dazu benutzen wir die k-fold cross validation Methode, wobei wir k = 10 setzen. Hierbei wird das Datenset in 10 Teile geteilt, wobei 9 zum Trainieren und 1 zum Testen dient. Das wird 10 mal gemacht, wobei immer ein anderer Teil zum Testen verwendet wird. Dies führt dazu, dass wir im Vergleich zu der traditionellen Aufteilung des Datensets (70% Trainings- und 30% Test-Daten) weniger vom Zufall abhängig sind, auf welchen Daten gelernt und auf welchen Daten getestet wird. Dies ist besonders praktisch, wenn man wenig Daten hat, da man sonst womöglich zu wenig Daten zum Lernen im Trainings-Set hätte.

Der Einfachheit halber setzen wir alle Listening-Events des Test-Sets unserer UAM auf 0. Dies macht eine erneute Normalisierung der verbleibenden Daten notwendig, welche wir wie in Task 01 beschrieben, durchführen.

Beim Testen unseres CF-Recommenders schlagen wir maximal so viele Interpreten vor, wie groß das Testset ist (`len(test_aidx)`). Wir haben uns für diese Methode entschieden, da wenn wir für jeden User eine fixe Anzahl an Vorschlägen berechnen, der Recall-Wert stark schwankt: Wenn der User nur sehr wenige verschiedene Interpreten angehört hat, ist es wahrscheinlicher, dass wir alle relevanten Künstler vorschlagen, als wenn der User sehr viele unterschiedliche Interpreten gehört hat. Bsp.: User a: 10 unique artists in test set User b: 200 unique artists in test set Wenn wir stets 100 Interpreten vorschlagen, so ist ein Erreichen eines hohen Recall-Wertes bei a wahrscheinlicher als bei b; bei b ist es sogar nicht möglich, einen perfekten Recall-Wert zu erreichen, da dazu 200 Recommendations nötig wären. Aber auch der Precision-Wert ist davon abhängig. Wenn wir mehr Interpreten vorschlagen, als im Test-Set vorhanden sind, so ist ein perfekter Wert bei der Precision unmöglich - schließlich

können ja dadurch nicht alle unsere Vorhersagen richtig sein, sondern eben maximal so viele, wie im Test-Set vorhanden sind.

3.1 Random Baseline (komplett zufällige Interpreten auswählen)

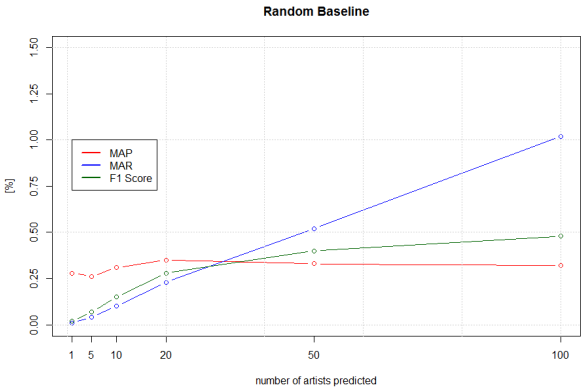

Figure 2: Random Baseline Performance

Anzahl der vorgesagten Interpreten	Mean Average Precision (MAP)	Mean Average Recall (MAR)	F1 Score
1	0,28	0,01	0,02
5	0,26	0,04	0,07
10	0,31	0,10	0,15
20	0,35	0,23	0,28
50	0,33	0,52	0,40
100	0,32	1,02	0,48

Table 4: Performance der Random Baseline

3.2 Random Baseline (random user picking)

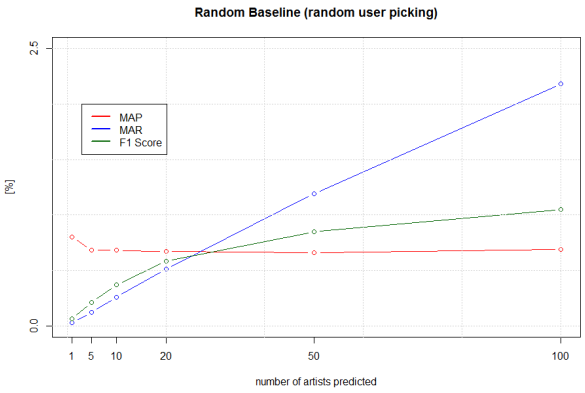

Figure 3: Random Baseline (user picking) Performance

Anzahl der vorgesagten Interpreten	Mean Average Precision (MAP)	Mean Average Recall (MAR)	F1 Score
1	0,8	0,03	0,06
5	0,68	0,12	0,21
10	0,68	0,26	0,37
20	0,67	0,51	0,58
50	0,66	1,19	0,85
100	0,69	2,18	1,05

Table 5: Performance der Random Baseline (user picking)

3.3 Collaborative Filtering (CF) Recommender

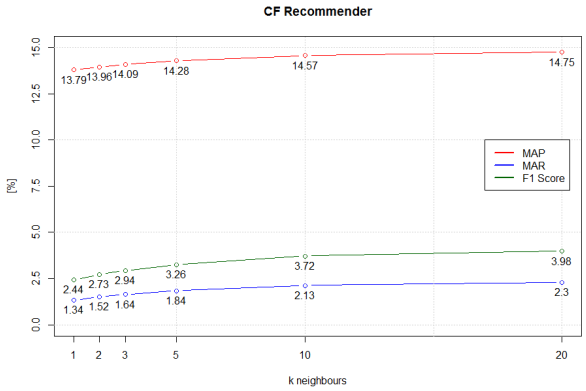

Figure 4: Collaborative Filtering Performance

Anzahl der betrachteten Nachbarn	Mean Average Precision (MAP)	Mean Average Recall (MAR)	F1 Score
1	13,79	1,34	2,44
2	13,96	1,52	2,73
3	14,09	1,64	2,94
5	14,28	1,84	3,26
10	14,57	2,13	3,72
20	14,75	2,3	3,98

Table 6: Collaborative Filtering (CF) Recommender)

3.4 Interpretation & Vergleich

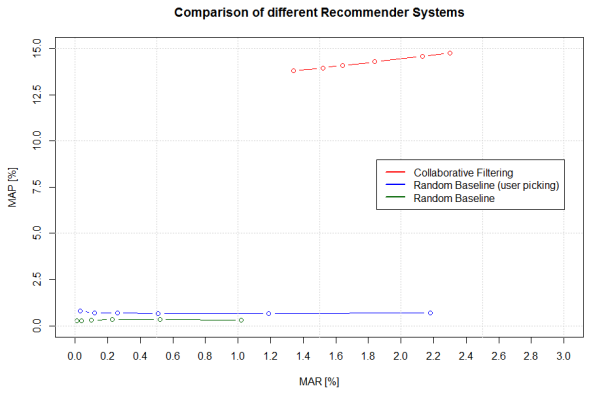

Figure 5: Vergleich der drei Recommender Systeme

Wie erwartet ist die selbst implementierte Random Baseline, die die Interpreten eines zufälligen Nutzers nimmt, besser, als komplett zufällig irgendwelche Interpreten vorzuschlagen. Der Grund ist der, dass sehr viele Interpreten Teil des "Long Tails" sind; also nur von sehr wenigen Nutzern gehört werden. Wenn wir aber einen zufälligen Nutzer nehmen und dessen Interpreten vorschlagen, so werden viele dieser Interpreten eher im "Short Head" sein und somit wahrscheinlicher auch von dem Nutzer, dem wir die Vorschläge machen, gehört worden sein.

Weiters ist zu Beobachten, dass zwar die Precision nicht mit Anzahl der Vorhersagen steigt, aber der Recall zunimmt. Der Grund dafür ist, dass die Chance höher ist, dass relevante Interpreten vorgeschlagen wurden. Collaborative Filtering schneidet deutlich besser ab als die Random Baseline. Dabei kann man erkennen, dass dessen Performance zunimmt, je mehr ähnliche Nutzer für die Vorhersagen in Betracht genommen werden. Zwischen 1 und 20 Nutzern ist so ein Unterschied von ca. 1 Prozentpunkt bei MAP und MAR. Dies ist darauf zurückzuführen, dass wir bei unserer Empfehlung in Betracht ziehen, wie populär ein Interpret über mehrere Nutzer hinweg gesehen ist; und wir bei mehreren Nutzern eher populärere Interpreten vorschlagen.

3.5 Alternative Möglichkeiten zur Evaluation

Einerseits könnten zusätzliche Kriterien, wie P@k zur Evaluation der Reihenfolge, oder Reciprocal Rank (RR), da ein hoher Recall meist nicht nötig für die Nutzer ist. Andererseits könnte man auch die Performance unter Klassifikations-Aspekten, also Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) und Rank Correlation, evaluieren.

Was aber wohl ganz wichtig für die Performance-Messung ist, ist eine nutzerzentrierte Evaluation durchzuführen; d.h. Wir fragen die Nutzer über Fragebögen, Interviews etc., wie zufrieden sie mit den Vorschlägen sind. Beispielsweise lässt sich mit den berechneten Kriterien schlecht messen, wie neu, wie eintönig und langweilig, wie ähnlich die Lieder für den jeweiligen Nutzer waren; zum Beispiel ist es vielleicht eine gute Idee, Nutzern mehr Interpreten des Long Tails zu empfehlen, auch wenn uns eigentlich die Performance-Messung per MAP/MAR/F1-Score dafür abstrafen würden. Wir messen schließlich, ob der Nutzer die vorgeschlagenen Interpreten schon einmal gehört hat (d.h. im Test-Set vorkommen), und nicht, ob neue vorher unbekannte vorgeschlagene Interpreten dem Nutzer gefallen.

Neben dem klassischen direkten Fragen könnte man die Nutzer auch einfach beobachten und deren Verhalten analysieren.