Algorithms for the Social Web Task 6

Richard Fussenegger und Markus Deutschl

Experimentelle Offline-Evaluation von zwei algorithmischen Verfahren auf einem verfügbaren Datensatz (z. B. BX).

Vorbereitung

Um die Ausführungszeit einzugrenzen haben wir uns bei unserer Offline-Evaluierung auf die 50 Benutzer mit den meisten Bewertungen beschränkt. Die Bewertungen dienten uns als Basis um unsere Algorithmen zu testen. Als Referenzbenutzer wählen wir den Benutzer mit den meisten Bewertungen (in unserem Fall war das der Benutzer mit der ID 11676), für den alle Prognosen berechnet werden.

Im nächsten Schritt wählen wir mit dem *k*-Nearest Neighbor Algorithmus – unter Zuhilfenahme der Euklidischen Distanz – die 20 ähnlichsten Benutzer. Die Distanz vom ähnlichsten Benutzer war dabei 0,16139047779641 und des 20sten war 0,056678684699613. Die extrem niedrigen Werte ergeben sich aus dem verwendeten Testdatensatz – dem „Book-Crossing Dataset“ – da hier nur wenige valide Bewertungen von gleichen Büchern vorliegen.

Prognose-Algorithmus 1: Durchschnitt mit *k*-Nearest Neighbor

Unser erster Algorithmus zur Berechnung der Prognosen ist ein naiver Ansatz. Der erste Teil des Algorithmus wird bereits während den Vorbereitungen durchgeführt, durch Aufruf der Funktion *k_nearest_neighbor()*. In dieser Funktion werden anhand der Euklidischen Distanz die 20 Benutzer mit den ähnlichsten Bewertungen zu einem gegebenen Benutzer (der anhand seiner ID identifiziert wird) ermittelt. Die Funktion *average_from_neighbors()* erwartet sich den Ergebnis-Array von *k_nearest_neighbor()* als zweiten Parameter, der erste Parameter enthält das komplette Datenset (in unserem Fall eingeschränkt auf die „Top 50“-Benutzer) und als dritter Parameter wird der Identifikator für das Item übergeben (in unserem Fall sind das die ISBNs der Bücher). In der Funktion iterieren wir über alle Nachbarn und insofern eine Bewertung vorliegt, wird diese aufsummiert. Aus der Summe und der Anzahl an vorhandenen Bewertungen wird der Durchschnitt gebildet der der Vorhersage entspricht.

Prognose-Algorithmus 2: Slope One

Unser zweiter Algorithmus zur Berechnung der Prognosen ist einer der wenigen verfügbaren, quelloffenen Algorithmen, „Slope One“. Dieser wurde 2005 von Daniel Lemire und Anna Maclachlan

im Paper „Slope One Predictors for Online Rating-Based Collaborative Filtering" beschrieben[1], unsere Implementierung basiert auf der Referenzimplementierung von Daniel Lemire und Sean McGrath[2]. Die Funktion *slope_one_predict()* erwartet sich zwei Arrays und die ISBN für die eine Prognose erstellt werden soll als Übergabeparameter. Wir löschen an dieser Stelle die Bewertung des Items für das eine Prognose erstellt werden soll, dazu später mehr im Abschnitt „Vorgehensweise". Dann wird über alle Bewertungen vom Benutzer für den eine Prognose ermittelt werden soll iteriert. Insofern von beiden Benutzern eine Bewertung für das soeben betrachtete Item vorliegt, wird die Bewertungsdifferenz berechnet und akkumuliert. Als Ergebnis wird der Bewertung vom Nachbarn der Durchschnitt der Bewertungsdifferenz hinzuaddiert.

Vorgehensweise

Um die zwei verwendeten Algorithmen vergleichen zu können greifen wir auf die „Root Mean Square Error"-Methodik (RMSE) zurück. Um den Fehler in den Prognosen zu berechnen benötigen wir Referenzwerte. Dazu verwenden wir die Originalbewertung des Referenzbenutzers und vergleichen diese mit der prognostizierten Bewertung. Deshalb müssen wir bei der Bildung der „Slope One"-Prognose zu Beginn die Bewertung vom Referenzbenutzer aus dessen Bewertungs-Array löschen. Für das Experiment wird über alle Nachbarn iteriert und die Schnittmenge über die Items des Referenzbenutzers gebildet. In der zuvor genannten Iteration wird über die Schnittmengenitems iteriert und pro Item eine Prognose mit beiden Algorithmen getroffen. An dieser Stelle wird auch die Summe für die Berechnung des RMSE erstellt.

Interpretation

Ein grundlegendes Problem unseres Experiments ist das zugrundeliegende Datenset, da es nur sehr wenige Bewertungen gleicher Items (die Bewertungen mit 0 wurden von uns während des gesamten Experiments von Beginn an herausgefiltert, da unklar ist was diese bedeuten sollen) besitzt. Da beide Algorithmen von Benutzerähnlichkeiten profitieren, die in diesem Datensatz praktisch nicht vorhanden sind (Euklidische Distanz des ähnlichsten Benutzers von unter 0,2), konnten sie auch keine hinreichend zufriedenstellenden Ergebnisse liefern. Für unseren naiven Ansatz erhielten wir einen RMSE von 2,1517074525224, für Slope One 2,144047816781. Slope One hat hier zwar besser abgeschnitten als der einfache Durchschnitt, hat jedoch auch eine hohe Fehlerrate. Abschließend bleibt zu sagen, dass Sparsity ein sehr großes Problem bei Recommender Systemen darstellt und mit Sicherheit auch ausgefeilteren Algorithmen zu schaffen macht.

[1] http://lemire.me/fr/abstracts/SDM2005.html
[2] http://lemire.me/fr/abstracts/TRD01.html

Implementierung

Die Quelltexte für unser Experiment befinden sich auf GitHub und sind über folgenden Link abrufbar:

https://github.com/Ravenlord/AlgorithmsForTheSocialWeb