Data Mining & Pattern Recognition

Versuch 3:
Recommender Systeme mit Collaborative Filtering

Datum:	
Namen:	
Unterschrift Betreuer:	

Prof. Dr. Johannes Maucher
Medieninformatik Bachelor (MIB)
WS 12/13

Inhaltsverzeichnis

1 Einführung

1.1 Lernziele

In diesem Versuch sollen Kenntnisse in folgenden Themen vermittelt werden:

- **Ähnlichkeit:** Verfahren zur Bestimmung der Ähnlichkeit zwischen Personen (Kunden) und Elementen (Produkten)
- **Empfehlungssysteme:** Collaborative Filtering
- **Collaborative Filtering:** Nutzerbezogener Ansatz und elementbasierter Ansatz

Sämtliche Verfahren und Algorithmen werden in Python implementiert.

1.2 Protokoll

Abzugeben sind pro Gruppe spätestens vor Beginn des nächsten Versuchs:

- Antworten auf alle in dieser Anleitung gestellten Fragen. Die Antworten sollten in einem elektronischen Dokument verfasst und als .pdf bereitgestellt werden. Die Antworten sind entsprechend der Kapitelnummerierung in dieser Anleitung zu ordnen.
- Die Antworten auf die Fragen im Kapitel 1.4 müssen **nicht** protokolliert und abgegeben werden. Diese Fragen müssen im Rahmen der Versuchsvorbereitung geklärt werden und werden im Kolloq zu Beginn des Versuchs abgefragt.
- Alle Dateien von denen in dieser Anleitung ein Abspeichern gefordert wird.

Die Durchführung und Dokumentation eigener Experimente, die über die Aufgabenstellungen in diesem Dokument hinausgehen, wird ausdrücklich unterstützt und in der Bewertung entsprechend berücksichtigt.

1.3 Theorie zur Vorbereitung

1.3.1 Recommender Systeme

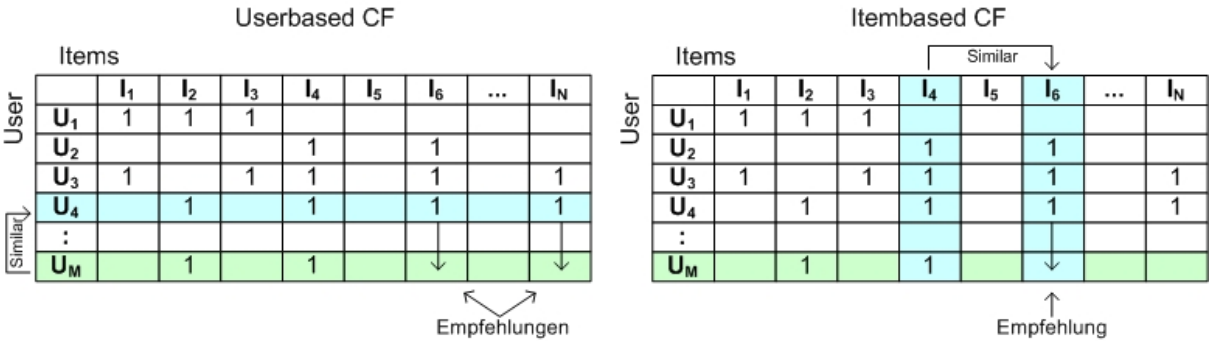
Recommender Systeme werden im E-Commerce eingesetzt um Werbung in Form von kundenspezifischen Empfehlungen zu verteilen. Weitläufig bekannt sind die Amazon-Empfehlungen, die entweder per e-mail geschickt oder nach dem Log-In in der Web-Page angezeigt werden. Diese Empfehlungen werden in Abhängigkeit von den bisher vom jeweiligen Kunden gekauften bzw. bewerteten Produkten erstellt. In diesem Versuch werden die derzeit wohl am weitest verbreiteteten Verfahren für die Erzeugung kundenspezifischer Empfehlungen vorgestellt, darunter das elementweise Collaborative Filtering, welches z.B. auch von Amazon eingesetzt wird.
Direkt-Marketing Methoden wie die kundenspezifische Erzeugung und Bereitstellung von Werbung erfordern detaillierte Kunden- und Warenkorbanalysen. Kunden mit ähnlichem Kaufverhalten werden in Kundengruppen zusammengefasst. Die Warenkorbanalyse untersucht u.a. welche Waren bevorzugt im Verbund von der gleichen Person gekauft werden. Damit kann ein Händler Werbung in Form von Empfehlungen individuell und gezielt an seine Kunden richten, abhängig davon welcher Kundengruppe er angehört und welche Produkte bevorzugt von dieser Kundengruppe nachgefragt werden.

Im ersten Teil der Übung werden fiktive Daten in einer überschaubaren Menge verwendet. Es handelt sich hier um Filmbewertungen. Anhand dieses *Spiel-Beispiels* sollen die notwendigen Methoden und Abläufe implementiert und getestet werden. Diese werden im zweiten Teil der Übung auf *echte Daten* angewandt.

Hierzu werden über eine Python-API Daten vom Internet-Meta-Radio *last.fm* integriert. Auf der Basis dieser Daten sollen dann Musikempfehlungen für *last.fm* User berechnet werden.
Recommender Systeme lassen sich mit

- Clustering Verfahren
- Suchalgorithmen
- Collaborativen Filtering

realisieren. Am häufigsten wird hierbei das Collaborative Filtering eingesetzt. Für das Collaborative Filtering wird jeder der M User durch einen N-dimensionalen Vektor beschrieben, wobei N die Anzahl der Produkte im Angebot des Händlers ist. Jedes Element im Vektor gehört zu einem speziellen Produkt. Das Element hat den Wert 1, wenn der User dieses Produkt bereits gekauft hat, sonst 0 [1]. Alle M Zeilenvektoren können zur *User/Item* Matrix zusammengefasst werden (siehe Abbildung).



Das traditionelle *userbasierte Collaborative Filtering (UCF)*, benutzt die Ähnlichkeit zwischen Benutzern: Um für User U_i eine Empfehlung zu erzeugen wird zunächst der diesem User ähnlichste Kunde[2] ermittelt. Dann werden U_i die Produkte (Items) empfohlen, welche der ähnlichste Kunde gekauft hat, U_i selbst jedoch noch nicht.
Dieser Ansatz skaliert schlecht im Fall sehr großer *User/Item*-Matrizen. Ausserdem ist er für User, welche erst wenige Produkte gekauft haben unzuverlässig. Besser eignet sich in diesen Fällen das *itembasierte Collaborative Filtering (ICF)*. Es wird u.a. von Amazon.com eingesetzt. Diese Variante benutzt die Ähnlichkeit zwischen Produkten (Items). Dabei sind Produkte umso ähnlicher je mehr Kunden diese Produkte gemeinsam gekauft haben. Für die Produkte welche ein Referenzuser U_i bereits gekauft hat, werden die ähnlichsten Produkte ermittelt. Diese ähnlichsten Produkte werden U_i empfohlen, wenn er sie nicht schon selbst gekauft hat.

Im folgenden Abschnitt werden einige gebräuchliche Metriken für die Berechnung der Ähnlichkeit zwischen Benutzern oder Artikeln vorgestellt. Für Collaboratives Filtering wird sehr häufig das Cosinus - Ähnlichkeitsmaß eingesetzt.

1.3.2 Gebräuchliche Ähnlichkeitsmaße

Euklidische Distanz

[1] Andere Wertbelegungen sind möglich, z.B. wenn Produktbewertungen vorliegen
[2] oder eine Menge vom ähnlichsten Kunden

Die euklidische Distanz $d_E(\underline{a}, \underline{b})$ zwischen zwei n-dimensionalen Vektoren $\underline{a} = (a_1, \ldots, a_n)$ und $\underline{b} = (b_1, \ldots, b_n)$ berechnet sich zu

$$d_E(\underline{a}, \underline{b}) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$$

Zwei Vektoren können als umso ähnlicher erachtet werden, je kleiner deren euklidische Distanz ist. Ein auf der euklidischen Metrik basierendes Ähnlichkeitsmaß zwischen zwei Vektoren $\underline{a}$ und $\underline{b}$ kann durch

$$s_E(\underline{a}, \underline{b}) = \frac{1}{1 + d_E(\underline{a}, \underline{b})}$$

angegeben werden.

Pearson Korrelation
Die Ähnlichkeit zwischen zwei Vektoren kann auch durch den Pearson-Korrelationskoeffizient $\rho_{\underline{a},\underline{b}}$ ausgedrückt werden. Er berechnet sich zu

$$\rho_{\underline{a},\underline{b}} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{(a_i - \overline{a})}{\sigma_a} \frac{(b_i - \overline{b})}{\sigma_b}$$

Dabei bezeichnet N die Länge der Vektoren, $\overline{a}$ den Mittelwert und σ_a die Standardabweichung des Vektors $\underline{a}$.
Der Pearson-Korrelationskoeffizient misst die lineare Abhängigkeit zwischen zwei Vektoren. Der maximale Wert von $+1$ wird erreicht, wenn die durch die beiden Vektoren definierten N Punkte im 2-dimensionalen Raum auf einer ansteigenden Geraden liegen. Der Minimalwert von -1 wird erreicht, wenn die Punkte auf einer abfallenden Geraden liegen. Der Betrag des Koeffizienten ist umso kleiner, je stärker die Punkte von einer fiktiven Geraden (kann durch lineare Regression berechnet werden) abweichen. Der Koeffizient ist 0 wenn keine lineare Abhängigkeit zwischen den Vektoren besteht.

Cosinus Ähnlichkeitsmaß
Die Ähnlichkeit zwischen zwei Vektoren kann auch durch den Cosinus $\cos(\underline{a}, \underline{b})$ ausgedrückt werden. Er berechnet sich zu

$$\cos(\underline{a}, \underline{b}) = \frac{\underline{a} \cdot \underline{b}}{\|\underline{a}\| \cdot \|\underline{b}\|}$$

wobei im Zähler das Skalarprodukt der beiden Vektoren steht und mit $\|\underline{x}\|$ der Betrag des Vektors $\underline{x}$ bezeichnet wird.

Falls die Vektoren $\underline{a}$ und $\underline{b}$ mittelwertfrei sind, ist der Cosinus-Ähnlichkeitswert gleich dem Pearson-Korrelationswert. In der Dokument- und Textanalyse wird vornehmlich das Cosinus-Ähnlichkeitsmaß verwendet.

Russel Rao Ähnlichkeitsmaß
Die Russel Rao-Ähnlichkeit zwischen zwei binären Vektoren $\underline{a}$ und $\underline{b}$ mißt das Verhältnis zwischen der Anzahl α der Stellen in denen beide Vektoren den Wert 1 haben und der Länge n der Vektoren. Z.B. ist für die Vektoren $\underline{a} = (1, 0, 1, 0, 0, 1)$ und $\underline{b} = (0, 1, 1, 1, 0, 1)$ die Russel-Rao-Ähnlichkeit $s_{RR}(\underline{a}, \underline{b}) = 2/6 = 0.333$.

Jaccard Ähnlichkeitsmaß
Die Jaccard-Ähnlichkeit zwischen zwei binären Vektoren $\underline{a}$ und $\underline{b}$ mißt das Verhältnis zwischen der Anzahl α der Stellen in denen beide Vektoren den Wert 1 haben und der Anzahl der Stellen in denen mindestens

einer der beiden Vektoren ungleich 1 ist. Z.B. ist für die Vektoren $\underline{a} = (1, 0, 1, 0, 0, 1)$ und $\underline{b} = (0, 1, 1, 1, 0, 1)$ die Jaccard-Ähnlichkeit $s_J(\underline{a}, \underline{b}) = 2/5 = 0.5$.
Eine detaillierte Beschreibung zu Distanz- und Ähnlichkeitsmaßen befindet sich z.B. in [MLM].

1.3.3 Externe Referenzen zur Vorbereitung

- Pylast: Python-Api für last.fm: `http://code.google.com/p/pylast/`

1.4 Vor dem Versuch zu klärende Fragen

Eine Untermenge der im Folgenden aufgeführten Fragen wird zu Beginn des Versuchs im Rahmen eines Gruppenkolloqs abgefragt. Auf jede Frage sollte von mindestens einem Gruppenmitglied eine Antwort geliefert werden und jedes Gruppenmitglied muss mindestens eine der gestellten Fragen beantworten können.

- Beschreiben Sie das Prinzip des userbasierten Collaborativen Filtering (UCF).
- Welche Nachteile hat das UCF?
- Worin besteht der Unterschied zwischen UCF und itembasierten Collaborativen Filtering (ICF)?
- Gegeben seien die Vektoren

$$\begin{aligned} \underline{a} &= [1, 2, 3, 4, 5, 6] \\ \underline{b} &= [3, 3, 5, 6, 7, 8] \end{aligned}$$

  Zeigen Sie am Beispiel des Vektors $\underline{a}$ wie Mittelwert und Varianz eines Vektors berechnet werden.
- Wie werden Mittelwert und Varianz mit *numpy* berechnet?
- Wie groß ist die
  - Euklidische Ähnlichkeit
  - Pearson Ähnlichkeit
  - Cosinus Ähnlichkeit

  zwischen den Vektoren $\underline{a}$ und $\underline{b}$?
- In welchen Fällen sind Cosinus- und Pearsonähnlichkeit der euklidischen Ähnlichkeit vorzuziehen?
- Wie wird in Python ein doppelt verschachteltes Dictionary angelegt und wie greift man auf dessen Elemente zu?
- Wie können mit Hilfe der *last.fm*-Api `pylast.py` alle Alben einer Band bestimmt werden?

2 Durchführung Teil 1: Fiktive Filmbewertung

Laden Sie die Datei *recommendationsTemplate.py* vom Skripteserver, speichern Sie diese unter dem Namen *recommendations.py* ab, und implementieren Sie alle im folgenden spezifizierten Funktionen in dieser Datei. Das Hauptprogramm, welches die Funktionen benutzt ist in einer eigenen Datei mit dem Namen *recommtest1.py* zu implementieren.

2.1 Daten: Filmbewertung

Die vorgegebene Datei *recommendationsTemplate.py* enthält ein verschachteltes Dictionary mit folgendem Inhalt.

	Lady in the Water	Snakes on a Plane	Just My Luck	Superman Returns	You, Me and Dupree	The Night Listener
Lisa Rose	2.5	3.5	3	3.5	2.5	3
Gene Seymour	3	3.5	1.5	5	3.5	3
Michael Phillips	2.5	3		3.5		4
Claudia Puig		3.5	3	4	2.5	4.5
Mick LaSalle	3	4	2	3	2	3
Jack Matthews	3	4		5	3.5	3
Toby Segaran		4.5		4	1	

Die Keys des Python-Dictionary definieren die Namen von Personen (Zeilen in der Matrix), die Filme bewertet haben. Die Values sind selbst wieder Dictionarys, welche als Keys die Filmnamen (Spalten in der Matrix) und als Values die jeweilige Filmbewertung (Matrixelment) enthalten.

2.2 Ähnlichkeitsbestimmung

Für die Bestimmung der Ähnlichkeit zwischen Personen und Produkten werden in diesem Versuch ein auf der euklidischen Distanz basierendes Ähnlichkeitsmaß und die Pearson-Korrelation verwendet. Für beide Ähnlichkeitsmaße sind die entsprechenden Python-Funktionen schon in der Datei *recommendationsTemplate.py* implementiert. **Zu beachten** ist, dass in beiden Funktionen für die Berechnung der Ähnlichkeit zwischen zwei Personen nur die Produkte berücksichtigt werden, welche von beiden Personen schon bewertet wurden. Es handelt sich hier also um modifizierte Ähnlichkeitsfunktionen.

1. Welche Bedeutung hat der Übergabeparameter `normed` in der Funktion `sim_euclid`?
2. Schreiben Sie eine Funktion `topMatches(prefs,person,similarity)`, welche für eine beliebige in `critics` (wird an `prefs` übergeben) enthaltene Person die Ähnlichkeitswerte zu allen anderen Personen berechnet und in einer geordneten Liste zurück gibt. Der Funktion soll als Übergabeparameter auch die anzuwendende Ähnlichkeitsfunktion (`sim_euclid` oder `sim_pearson`) übergeben werden können. Berechnen Sie mit dieser Funktion für jede Person die *top matches*, zunächst unter Verwendung der euklidischen- dann unter Verwendung der Pearson-Ähnlichkeit.
3. Vergleichen Sie die beiden Ähnlichkeitsmaße. Welches Ähnlichkeitsmaß erscheint Ihnen für diesen Anwendungsfall sinnvoller und warum?

2.3 Berechnung von Empfehlungen mit User basiertem Collaborative Filtering

Für die Produkte, die von einer Person noch nicht gekauft wurden, sollen Empfehlungen berechnet werden. Die Empfehlungen können ebenfalls Werte zwischen 1 (wird nicht empfohlen) und 5 (wird stark

empfohlen) annehmen. Für die Berechnung der Empfehlung werden die Bewertungen des jeweiligen Produkts durch die anderen Personen herangezogen. Dabei werden die Bewertungen der ähnlichen Personen (d.h. hoher Pearson-Korrelationswert) stärker mit einbezogen als die Bewertungen durch Personen mit einem niedrigen Korrelationswert.

Beispiel:
Toby hat die Filme *The Night Listener*, *Lady in the Water* und *Just My Luck* noch nicht gekauft. Für diese Filme soll für Toby eine Empfehlung berechnet werden.

Person	Korrelation (K)	Night (N)	$K \cdot N$	Lady (La)	$K \cdot La$	Luck (Lu)	$K \cdot Lu$
Lisa	0.99	3	2.97	2.5	2.48	3	2.97
Gene	0.38	3	1.14	3	1.14	1.5	0.57
Claudia	0.89	4.5	4.02			3	2.68
Mick	0.92	3	2.77	3	2.77	2	1.85
Jack	0.66	3	1.99	3	1.99		
Sum			12.89		8.38		8.07
KSum			3.84		2.95		3.18
$\frac{Sum}{KSum}$			3.35		2.83		2.53

In der Tabelle enthält die zweite Spalte die Pearson-Ähnlichkeitswerte zwischen Toby und den anderen Personen. Die Spalten 3, 5 und 7 enthalten die Bewertungen der Filme *The Night Listener*, *Lady in the Water* und *Just My Luck* durch die anderen Personen. Die Spalten 4, 6 und 8 enthalten die jeweilige Filmbewertung gewichtet mit den Ähnlichkeitswerten der jeweiligen Person. Es fällt auf, dass in der Tabelle Michael nicht enthalten ist. Das liegt daran, dass Michael und Toby einen negativen Ähnlichkeitswert aufweisen, d.h. deren Interessen sind gegenläufig. Personen mit negativem Ähnlichkeitswert sollten für Empfehlungen nicht berücksichtigt werden.
Die Zeile *Sum* enthält die Summe aller gewichteten Bewertungen. Aus diesem Wert allein kann die Empfehlung noch nicht abgeleitet werden, da Filme die nur von wenigen Personen bewertet wurden, eine relativ kleine Summe ergeben. Deshalb sollte *Sum* noch durch die Anzahl der Bewertungen für diesen Film geteilt werden. Oder besser: Nicht durch die Summe der Bewertungen, sondern durch die Summe der relevanten Ähnlichkeitswerte ($KSum$). Der resultierende Empfehlungswert ist in der letzten Zeile eingetragen.

2.3.1 Aufgaben

Schreiben Sie eine Funktion `getRecommendations(prefs,person,similarity)`, mit der die Empfehlungswerte berechnet werden können und bestimmen Sie die Empfehlungswerte für Toby. Der Funktion wird

- das Dictionary (`critics`) mit den Filmbewertungen,
- der Name der Person, für welche Empfehlungen berechnet werden sollen
- die Methode für die Berechnung der Ähnlichkeit (`sim_euclid` oder `sim_pearson`)

übergeben. Die Methode soll eine geordnete Liste zurück geben. Jedes Listenelement enthält an erster Stelle den berechneten Empfehlungswert und an zweiter Stelle den Namen des Films. Die Liste soll nach Empfehlungswerten absteigend geordnet sein.

2.4 Berechnung von Empfehlungen mit Item basiertem Collaborative Filtering

In den vorigen Aufgaben wurden Ähnlichkeiten zwischen Personen bestimmt und für Produktempfehlungen benutzt (User basiertes Collaborative Filtering). Jetzt soll die Ähnlichkeit zwischen Produkten berechnet werden und auf der Basis dieser Produktähnlichkeit Empfehlungen berechnet werden (Item basiertes Collaborative Filtering).

Dabei sollen die bereits implementierten Ähnlichkeitsfunktion `sim_euclid` und `sim_pearson` sowie die Sortierfunktion `topMatches` unverändert eingesetzt werden.

2.4.1 Aufgaben

1. Implementieren Sie eine Funktion, welche das Bewertungsdictionary `critics` derart transformiert, dass die Funktionen `sim_euclid`, `sim_pearson` und `topMatches` für das Item basierte CF unverändert eingesetzt werden können. Die transformierte Matrix soll unter dem Namen `transCritics` abgespeichert werden.

2. Schreiben Sie eine Funktion `calculateSimilarItems` die aus der transformierten Matrix `transCritics` ein Dictionary berechnet, welches die Ähnlichkeit zwischen allen Filmen beschreibt. Die Keys des Dictionary sind die Filmnamen. Die Values sind geordnete Listen, welche die Funktion `topMatches` zurückgibt, wenn sie für die Filme (nicht für die User) aufgerufen wird. Dieses Dictionary wird an das aufrufende Programm zurück geben.

3. Schreiben Sie eine Funktion `getRecommendedItems`, welche basierend auf dem unten dargestellten Verfahren unter Vorgabe der Bewertungsmatrix und der zu verwendenden Ähnlichkeitsfunktion Produktempfehlungen berechnet.

2.4.2 Beispiel für die Berechnung von ICF Produktempfehlungen

Toby hat die Filme *The Night Listener*, *Lady in the Water* und *Just My Luck* noch nicht gekauft. Für diese Filme soll für Toby eine Empfehlung berechnet werden. Gekauft und bewertet hat Toby die Filme *Snakes on a plane*, *Superman Returns* und *You and me and Dupree*. Diese bereits vorhandenen Filme bilden die erste Spalte der unten dargestellten Matrix. In der zweiten Spalte befinden sich Tobys Bewertungen dieser Filme. Die Spalten 3,5 und 7 enthalten die Ähnlichkeitswerte (mit `calculateSimilarItems` unter Verwendung des euklidischen Ähnlichkeitsmaßes berechnet) zwischen den drei von Toby noch nicht gekauften Filmen und den drei von Toby bewerteten Filmen. Diese Ähnlichkeitswerte werden jeweils mit Tobys Bewertungen multipliziert. Das Resultat dieser Multiplikation befindet sich in den Spalten 4,6 und 8. Der finale Empfehlungswert für die von Toby noch nicht gekauften Filme wird berechnet in dem in den Spalten 4,6 und 8 zunächst die Summe über die Werte dieser Spalte in den drei oberen Zeilen berechnet wird und durch die Summe über die Werte der Spalten 3,5 und 7 geteilt wird. Im Fall, dass die Pearson-Korrelation zwischen den Filmen als Ähnlichkeitswert herangezogen wird, können negative Ähnlichkeitswerte auftreten. Dann soll in die Berechnung eines Empfehlungswert für Film A nur dann die Bewertung von Film B einfließen, wenn der Korrelationswert zwischen beiden > 0 ist.

Film	Bewertung (B)	Night (N)	$B \cdot N$	Lady (La)	$B \cdot La$	Luck (Lu)	$B \cdot Lu$
Snakes	4.5	0.73	3.29	0.73	3.29	0.58	2.61
Superman	4.0	0.67	2.68	0.61	2.44	0.51	2.04
Dupree	1.0	0.68	0.68	0.76	0.76	0.65	0.65
Sum		2.08	6.65	2.1	6.49	1.74	5.3
Normalisiert			3.19		3.09		3.05

3 last.fm Musikempfehlungen

3.1 Durchführung

Laden Sie die Datei `pylast.py` vom Skripteserver. In dieser Datei sind alle Zugriffsfunktionen auf *last.fm* Dienste implementiert. Die notwendigen Anmelde- und Authentifizierungsdaten für den User `pythonlab` sind ebenfalls schon in diesem Modul eingetragen.

Implementieren Sie den im folgenden beschriebenen Ablauf in einem Modul mit Namen `pylastRecTest.py`

1. Stellen Sie durch Aufruf der Funktion
   `network=pylast.get_lastfm_network()`
   eine Verbindung zu *last.fm* her. Beim Aufruf der Funktion wird die Anmeldung und Authentifizierung durchgeführt. Die Funktion gibt ein Objekt der Klasse `Network` zurück. Über dieses Objekt können Methoden wie

   - `get_artist("kuenstlerName")` (liefert Objekt der Klasse `Artist`)
   - `get_album("albumName")` (liefert Objekt der Klasse `Album`)
   - `get_track("songName")` (liefert Objekt der Klasse `Track`)
   - `get_user("userName")` (liefert Objekt der Klasse `User`)
   - `get_tag("tagName")` (liefert Objekt der Klasse `Tag`)
   - usw.

   aufgerufen werden. Die Menge aller verfügbaren Klassen und deren Attribute und Methoden können dem Modul `pylast.py` entnommen werden.

2. Rufen Sie über das oben instanziierte `Network`-Objekt die Methode `get_artist("BandIhrerWahl")` auf.

3. Rufen Sie über das oben instanziierte `Artist`-Objekt die Methode `topfans=get_top_fans(10)` auf. Die Methode gibt eine Liste von `User`-Objekt/Gewichtung-Paaren zurück. Die Gewichtungen von Objekten werden in diesem Versuch nicht benötigt. Legen Sie deshalb mit
   `group=[a.item for a in topfan]`
   eine Liste an, die nur noch die User Objekte enthält.

4. Implementieren Sie im Python-Modul `recommendations.py` eine Funktion `createLastfmUserDict()`. Dieser Funktion soll, die oben angelegte Liste von `User`-Objekten (`group`) übergeben werden. Für jeden User in `group` sollen die 20 beliebtesten Bands mit der Methode
   `topartists=get_top_artists()[0:20]`
   bestimmt werden. Die Methode gibt eine Liste von `Artist`-Objekt/Gewichtung-Paaren zurück. Die

Gewichtungen von Objekten werden in diesem Versuch nicht benötigt. Auf das *i.te* `Artist`-Objekt selbst kann mit `topartists[i].item` zugegriffen werden. Die Menge aller Bands, die auf diese Weise gesammelt werden, wird im folgenden mit `AllBands` bezeichnet. D.h. in `AllBands` befinden sich alle Bands, die für mindestens einen User in `group` zu den Top-20 gehören. Nun soll ein verschachteltes Dictionary mit Namen `userDict` wie folgt angelegt werden:

- Die Keys sind die Namen der `User`-Objekte in `group`. Auf den Namen eines Objekts kann mit `get_name()` zugegriffen werden.
- Die Values sind selbst wieder Dictionaries, deren Keys die Namen der Bands in `AllBands` sind. Achten Sie auch hier darauf, dass Sie nicht das `Artist`-Objekt selbst, sondern dessen Namen als Key verwenden.
- Für den User `a` und die Band `b` ist der Value `userDict[a][b]` $= 1$, falls `b` zu den Top-20 des Users `a` gehört. Andernfalls ist `userDict[a][b]` $= 0$.

Das derart angelegte Dictionary soll von der Funktion zurückgegeben werden.

5. Wählen Sie jetzt einen beliebigen User aus `group`. Bestimmen Sie zu diesem User die ähnlichsten User in `group` durch Anwendung der im ersten Teilversuch implementierten Funktion `topMatches()`. Der Funktion wird das angelegte `userDict` und der Name des gewählten Users übergeben. Als Ähnlichkeitsmaß soll die euklidische Metrik angewandt werden.

6. Bestimmen Sie dann für den gewählten User Band-Empfehlungen durch Anwendung der im ersten Teilversuch implementierten Funktion `getRecommendations()`. Der Funktion wird das angelegte `userDict` und der Name des gewählten Users übergeben. Als Ähnlichkeitsmaß soll die euklidische Metrik angewandt werden.

Literatur

[AR] G. Linden, B. Smith, J. York; *Amazon.com Recommendations*; IEEE Internet Computing; January 2003

[AK] Andre Klahold; *Empfehlungssysteme; Grundlagen, Konzepte und Lösungen*; Vieweg-Teubner Verlag, 2009

[TS] Toby Segaran; *Kollektive Intelligenz*; O'Reilly Verlag; 2008

[MLM] Johannes Maucher; Vorlesung Machine Learning; V3 Ähnlichkeitsmaße;
`https://www.mi.hdm-stuttgart.de/Downloads/Vorlesungsskripte/skripte/...`
`...MASTER_Computer_Science_and_Media/MachineLearning/SS10/V3AehnlichkeitHO.pdf`