Eksploracja danych

Piotr Lipiński

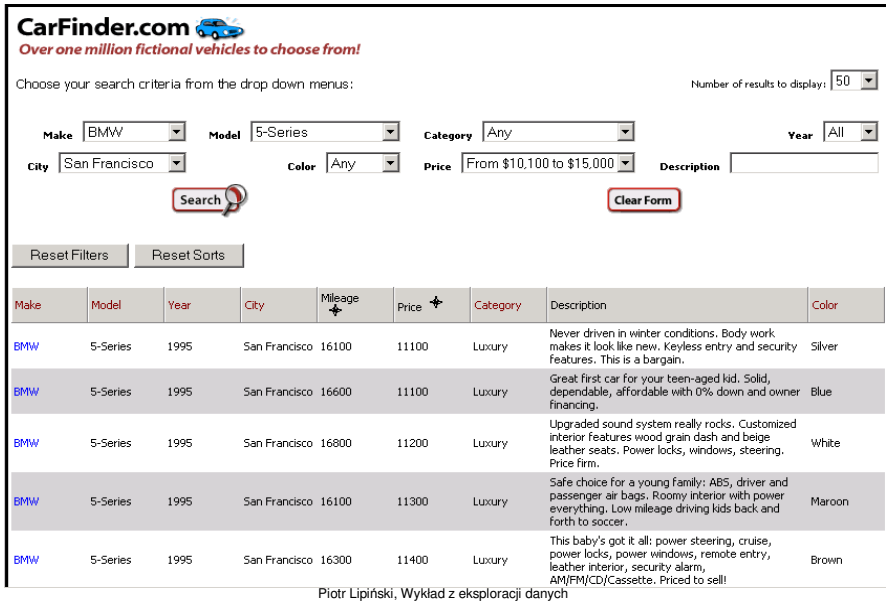
Systemy rekomendujące

- Dostępność coraz większej ilości informacji wymaga jej filtrowania. Użytkownik często potrzebuje rekomendacji – pomocy w wyborze interesującej go części informacji:
  - pomoc przy wyborze książki/filmu/muzyki
  - pomoc przy wyborze wiadomości prasowych do przeczytania
  - pomoc przy wyborze artykułów w sklepie internetowym
  - pomoc przy oznaczaniu niechcianych wiadomości pocztowych (trochę inny problem, ale podobny)
  - pomoc przy korzystaniu z bardziej złożonych programów komputerowych (inteligentny interfejs – też trochę inny problem)
- Problem rekomendowania jest rozszerzeniem problemu wyszukiwania informacji.
  - Użytkownik chce znaleźć pewną informację, ale nie potrafi sprecyzować kryteriów wyszukiwania.
  - W praktyce: to raczej nie użytkownik chce znaleźć informację, ale osoba trzecia (najczęściej właściciel portalu internetowego) chce mu dostarczyć taką potencjalnie ciekawą informację. Nie można więc liczyć na bezpośrednią współpracę z użytkownikiem.

Wyszukiwanie informacji

- Parametryczne wyszukiwanie informacji
  - wyszukiwanie zazwyczaj dotyczy danych ustrukturalizowanych bądź danych nieustrukturalizowanych, ale opisanych przez ustrukturalizowane meta-dane
    - Przykład: książki opisane przez meta-dane (autor, tytuł, wydawnictwo, słowa kluczowe)
  - sprowadza się do przeszukiwania baz danych
  - istnieją efektywne algorytmy, oparte na indeksowaniu tabel
  - wyzwania dotyczą spraw niezwiązanych z eksploracją danych
- Nieparametryczne wyszukiwanie informacji
  - wyszukiwanie może dotyczyć danych nieustrukturalizowanych, m.in. tekstów, obrazów, dźwięków, itp.
    - Przykład: treść książek, a nie tylko ich opis przez meta-dane
  - wymaga niestandardowego podejścia (jak reprezentować dane, jak określić zapytanie do bazy danych)
- Problemem ubocznym jest uporządkowanie znalezionych informacji (określenie kolejności w jakiej zostaną zwrócone, istotne zwłaszcza przy długiej liście wyników).

Parametryczne wyszukiwanie informacji



Nieparametryczne wyszukiwanie informacji

- Klasyczny przykład:
  - Która sztuka Szekspira zawiera słowa Brutus i Caesar ale nie zawiera słowa Calpurnia?
  - Można wypisać wszystkie sztuki Szekspira zawierające "Brutus" i "Caesar", a później wykreślić z listy te niezawierające "Calpurnia".
    - podejście bardzo nieefektywne (dla dużych danych)
  - Warunek niezawierania (NOT Calpurnia) jest trudny.
  - Bardziej złożone warunki, na przykład znalezienie słowa "Romans" w pobliżu "countrymen" też nie są łatwe.
- Potrzebna jest efektywna reprezentacja danych umożliwiająca łatwe przetwarzanie zapytań.
  - Skupimy się na przetwarzaniu dokumentów tekstowych. Inne dane, obrazy, dźwięki, multimedia, wymagają innych technik.

Term-Document Index (TDI)

- **Pierwsze podejście:** Niech T będzie zbiorem wszystkich słów, które mogą występować w analizowanych dokumentach. Niech d = |T|. Każdy dokument można przedstawić jako wektor binarny długości d, którego kolejne pozycje odpowiadają kolejnym słowom z T i mają wartość 1 jeśli dane słowo występuje w dokumencie lub 0 w przeciwnym przypadku.
  - Odpowiedź na zapytanie "Brutus AND Caesar BUT NOT Calpurnia" wymaga więc tylko porównania wektorów reprezentujących dokumenty z maską bitową odpowiadającą zapytaniu.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

Term-Document Index (TDI)

- Problemy:
  - Wielkość słownika spowoduje, że wektory będą bardzo długie.
  - W słowniku będzie wiele słów nieistotnych (spójniki, zaimki, itp.).
  - W słowniku będzie wiele form tego samego słowa (odmiana słów).
  - Przydatne techniki preprocessingu: stop words elimination, stemming.
- **Drugie podejście:** Zamiast słów używać termów (słów po obcięciu końcówek).
- Problemy:
  - Reprezentacja dokumentu nie uwzględnia liczby wystąpień danego termu.
- **Trzecie podejście:** Zamiast wartości binarnych wektor może zawierać liczbę wystąpień danego termu w dokumencie.
- Każdy dokument można więc przedstawić jako wektor w przestrzeni R^d.
- Zbiór N dokumentów można więc przedstawić jako macierz M rozmiaru d x N. Macierz M nazywa się Term-Document Matrix (TDM). Element M[i, j] takiej macierzy określa znaczenie i-tego termu dla j-tego dokumentu (liczbę wystąpień i-tego termu w j-tym dokumencie).
- Zapytanie w analogiczny sposób można przedstawić jako wektor w przestrzeni R^d.

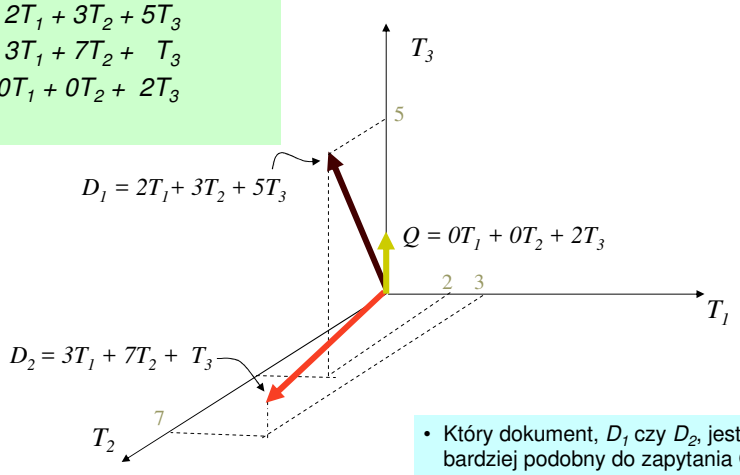
Reprezentacja graficzna

Przykład:

$D_1 = 2T_1 + 3T_2 + 5T_3$

$D_2 = 3T_1 + 7T_2 + T_3$

$Q = 0T_1 + 0T_2 + 2T_3$



- Który dokument, D_1 czy D_2, jest bardziej podobny do zapytania Q?
- Jak mierzyć podobieństwo?

Term-Document Matrix (TDM)

- **Trzecie podejście:** Zamiast wartości binarnych wektor może zawierać liczbę wystąpień termu w dokumencie.
- Problemy:
  - Przyjęliśmy model "Bag of words", w którym niektóre dokumenty są utożsamiane:
    - "John is quicker than Mary" = "Mary is quicker than John"
    - (na razie ten problem będziemy ignorować)
  - Dokumenty mogą być różnej długości, więc liczba wystąpień termów jest nieobiektywna (3 wystąpienia w 10-termowym dokumencie, a 3 wystąpienia w 1000-termowym dokumencie to nie to samo).

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	157	73	0	0	0	0
Brutus	4	157	0	1	0	0
Caesar	232	227	0	2	1	1
Calpurnia	0	10	0	0	0	0
Cleopatra	57	0	0	0	0	0
mercy	2	0	3	5	5	1
worser	2	0	1	1	1	0

Term-Document Matrix (TDM)

- **Czwarte podejście:** Zamiast liczby wystąpień termu w dokumencie wektor może zawierać liczbę wystąpień danego termu podzieloną przez sumę liczb wystąpień wszystkich termów w dokumencie.
- Praktyka pokazuje, że lepiej jest jednak znormalizować frekwencję termów inaczej.

Term-Frequency Matrix (TF)

- **Piąte podejście:** Niech TF będzie macierzą rozmiaru d x N o elementach

  $$TF[i, j] = M[i, j] / \max(\{M[k, j] : k = 1, 2, \ldots, d\}).$$

  Macierz TF nazywa się Term-Frequency Matrix.
- Problemy:
  - Same termy mogą mieć różną siłę dyskryminacyjną dla dokumentów, więc traktowanie ich jednakowo jest nieobiektywne.
  - Term, który ma duże znaczenie dla większości dokumentów (występuje często w większości dokumentów) nie pozwoli zbyt dobrze charakteryzować dokumentów.
  - Term, który ma duże znaczenie jedynie dla niektórych dokumentów (występuje często w małej liczbie dokumentów) pozwoli dużo lepiej charakteryzować dokumenty.

TF-IDF Matrix

- **Szóste podejście:** Niech IDF będzie wektorem długości d o elementach

  $$IDF[i] = \log (N / |\{j : M[i, j] > 0\}|).$$

  Wartości IDF[i] nazywa się Inverse-Document-Frequency. Mierzą one jak wiele informacji dostarcza dany term. Niech TF-IDF będzie macierzą rozmiaru d x N o elementach

  $$TF\text{-}IDF[i, j] = TF[i, j]\ IDF[i].$$

- Podejście z TF-IDF jest dość popularne i często sprawdza się w praktyce.
- Istnieją jeszcze inne podejścia, modyfikacje i rozszerzenia przedstawionych tutaj, m.in. z innym normowaniem wystąpień termów w dokumencie pozwalającym unikać błędów numerycznych.

Miary podobieństwa dokumentów

- Do wyszukiwania dokumentów potrzebna jest jeszcze miara podobieństwa dokumentu do zapytania lub ogólniej miara podobieństwa dokumentów (bo zapytanie może być traktowane jako dokument).
- Popularne miary:
  - miara euklidesowa – nie sprawdzi się (dlaczego?)
  - iloczyn skalarny
  - miara kosinusów (iloczyn skalarny podzielony przez iloczyn długości wektorów)

Wyszukiwanie informacji tekstowej

- Podsumowanie:
  - modele oparte wektorowej reprezentacji dokumentów i na algebrze liniowej
  - model TF-IDF okazuje się dość efektywny w praktyce
  - implementacja może być dość efektywna
    - m.in. dzięki współczesnym procesorom potrafiącym szybko przetwarzać dane macierzowe
  - wiele szczegółów należy jeszcze doprecyzować
  - więcej informacji na wykładach z wyszukiwania informacji, przetwarzania tekstów, przetwarzania języka naturalnego, itp.
- Pozostałe problemy:
  - model "Bag of words"
  - brak informacji semantycznej (m.in. sensu słowa)
  - brak informacji syntaktycznej (m.in. struktury zdania, kolejności słów, itp.)
  - założenie niezależności słów (m.in. nie uwzględnia się synonimów)
  - logiczna niedoskonałość modelu (m.in. wymaganie występowania słowa w dokumencie, nie uwzględnianie synonimów, itp.)
  - dla zapytania złożonego z dwóch termów A i B, model może preferować dokument zawierający A z dużą częstością, ale bez B, bardziej niż dokument zawierający oba słowa A i B, ale z mniejszą częstością

Podejścia bardziej zaawansowane …

- Macierz TDM składa się z dwóch czynników:
  - Czynnik lokalny l_{ij}:
    - term frequncy (TF), f_{ij} = liczba wystąpień termu i w dokumencie j
    - binary local factor, $bool(f_{ij} <> 0)$
    - logarithmic local factor, $\log(1+f_{ij})$
    - alternate-log, $bool(f_{ij} <> 0)\ (1 + \log f_{ij})$
    - augmented normalized TF, $(bool(f_{ij} <> 0) + f_{ij} / \max_k f_{kj}) / 2$
  - Czynnik globalny g_i:
    - 1
    - entropia, $1 + \Sigma_j p_{ij} \log p_{ij} / \log n$, gdzie $p_{ij} = f_{ij} / \Sigma_k f_{ik}$
    - IDF, $\log (n / \Sigma_j bool(f_{ij} <> 0))$

Podejścia bardziej zaawansowane …

- Usprawnienie: ważenie termów – skalowanie przestrzeni R^d, każdy wymiar ma inną skalę.

  $$(a_{1j}, a_{2j}, \ldots, a_{dj}) \quad |-> \quad (\alpha_1 a_{1j}, \alpha_2 a_{2j}, \ldots, \alpha_d a_{dj})$$

  gdzie wagi termów – współczynniki $\alpha_1, \alpha_2, \ldots, \alpha_d$ są ustalane indywidualnie dla każdego zapytania.

- Problem: jak ustalać te wagi?
  - badanie statystyk występowania słów
  - badanie statystyk zapytań użytkownika
  - profile użytkowników (dla księgowych terminy łacińskie nie mają znaczenia, dla lekarzy owszem)
  - ręczne ustawianie (absolutnie niepraktyczne)

Podejścia bardziej zaawansowane …

- Modele ze sprzężeniem zwrotnym:
  - Wyszukiwanie informacji można postrzegać jako proces ciągły.
  - Użytkownik zadaje zapytanie, zaraz po tym otrzymuje listę rezultatów, ale na tym nie kończy się działanie systemu.
  - Przez zadany okres użytkownik będzie otrzymywał kolejne rezultaty dotyczące zadanego zapytania (np. nowe dokumenty pojawiające się w bazie wiedzy).
  - Każdy z rezultatów jest oceniany przez użytkownika pod względem przydatności i stopnia dopasowania do zapytania (ukryte intencje).
  - Oceny użytkownika są używane do dostosowywania działania mechanizmu wyszukującego (ustawianie wag).
  - Prowadzi to do problemu optymalizacji

Podejścia bardziej zaawansowane …

- Modele ze sprzężeniem zwrotnym:
  - Prowadzi to do problemu optymalizacji
  - Niech $D^*_1, D^*_2, \ldots, D^*_N$ oznaczają dokumenty zwrócone dotychczas użytkownikowi.
  - Niech $r_1, r_2, \ldots, r_N \in [0, 1]$ oznaczają oceny dopasowania rezultatu do zapytania nadane tym dokumentom przez użytkownika (0 – źle, 1 – dobrze).
  - Niech $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ będzie wektorem wag termów.
  - Niech $D^{(\alpha)}_1, D^{(\alpha)}_2, \ldots, D^{(\alpha)}_{N'}$ oznaczają dokumenty zwrócone przy zastosowaniu wektora wag termów α ($N' > N$ jest ustaloną liczbą).

Podejścia bardziej zaawansowane …

- Idea: dobrać wagi termów tak, aby wśród dokumentów $D^{(\alpha)}_1$, $D^{(\alpha)}_2$, … , $D^{(\alpha)}_N$, znalazło się jak najwięcej dokumentów D^*_1, D^*_2, … , D^*_N o wysokich ocenach, a jak najmniej dokumentów o niskich ocenach.
- Funkcja celu (przykład):

$$F(\alpha) = \beta_{0.8} / (1 + \gamma_{0.2})$$

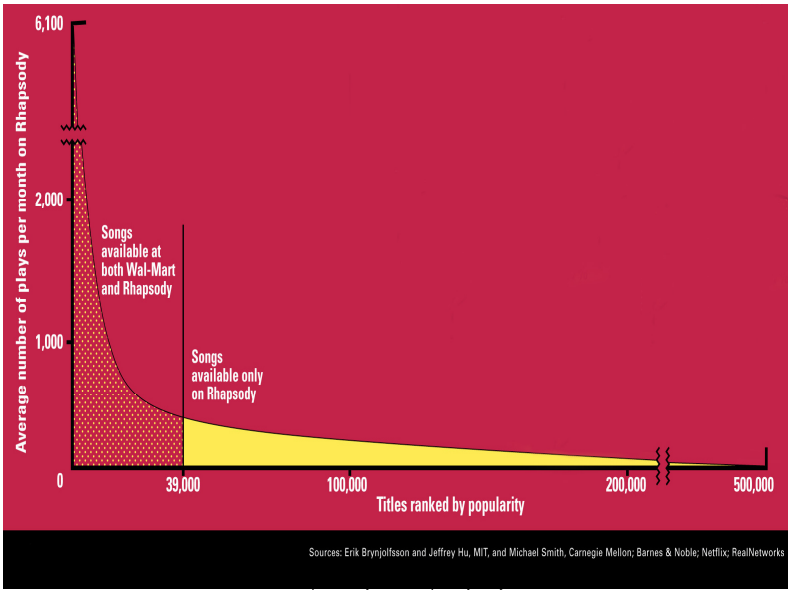
gdzie β_x oznacza liczbę dokumentów o $r > x$, zaś γ_x oznacza liczbę dokumentów o $r < x$.

- Funkcję celu F można maksymalizować przy użyciu algorytmu ewolucyjnego.
- Dwa algorytmy:
  - ES1 - strategie ewolucyjne ES
  - ES2 - strategie ewolucyjne z wbudowaną redukcją wymiarowości
- Problem:
  - długość chromosomu = liczba termów (ok. 5000)
    - ustawianie tylko wybranego podzbioru wszystkich wag (termy występujące w dokumentach D^*_1, D^*_2, … , D^*_N)

Systemy rekomendujące

- Dostępność coraz większej ilości informacji wymaga jej filtrowania. Użytkownik często potrzebuje rekomendacji – pomocy w wyborze interesującej go części informacji:
  - pomoc przy wyborze książki/filmu/muzyki
  - pomoc przy wyborze wiadomości prasowych do przeczytania
  - pomoc przy wyborze artykułów w sklepie internetowym
  - pomoc przy oznaczaniu niechcianych wiadomości pocztowych (trochę inny problem, ale podobny)
  - pomoc przy korzystaniu z bardziej złożonych programów komputerowych (inteligentny interfejs – też trochę inny problem)
- Problem rekomendowania jest rozszerzeniem problemu wyszukiwania informacji.
  - Użytkownik chce znaleźć pewną informację, ale nie potrafi sprecyzować kryteriów wyszukiwania.
  - W praktyce: to raczej nie użytkownik chce znaleźć informację, ale osoba trzecia (najczęściej właściciel portalu internetowego) chce mu dostarczyć taką potencjalnie ciekawą informację. Nie można więc liczyć na bezpośrednią współpracę z użytkownikiem.

Systemy rekomendujące



Systemy rekomendujące

- Typy rekomendacji:
  - redakcyjne (rekomendacje ręczne)
  - proste agregaty
    - Top 10, Most Popular, Recent Uploads
  - dopasowane do konkretnych użytkowników
    - Amazon, Netflix, …

Model formalny

- Niech *C* oznacza zbiór klientów.
- Niech *S* oznacza zbiór produktów.
- Niech *u*: *C* x *S* → *R* będzie funkcją użyteczności.
  - *R* to zbiór możliwych ocen. Musi być zbiorem uporządkowanym.
    - na przykład: liczba gwiazdek, od 0 do 5
    - na przykład: liczba rzeczywista z przedziału [0, 1]
  - u(c, s) to użyteczność produktu s dla klienta c

Macierz użyteczności

	King Kong	LOTR	Matrix	National Treasure
Alice	1		0.2	
Bob		0.5		0.3
Carol	0.2		1	
David				0.4

Systemy rekomendujące

- **Pierwsze podejście:** Mając funkcję użyteczności, klientowi c rekomendujemy produkty s, dla których u(c, s) ma wysokie wartości.
- Problemy:
  - zazwyczaj nie znamy funkcji użyteczności
  - dla ustalonego zbioru klientów C i zbioru produktów S możemy gromadzić informacje o wartościach funkcji użyteczności w macierzy użyteczności U rozmiaru |C| x |S| (m.in. na podstawie opinii klientów po zakupie), jednak ... w ten sposób nie otrzymamy informacji o użyteczności produktów, których klienci jeszcze nie kupili
  - gromadzenie informacji nie jest łatwe:
    - explicite – proszenie użytkownika o ocenę – większość użytkowników zignoruje prośbę
    - implicite – obliczać ocenę na podstawie działań użytkownika (na przykład wielokrotny zakup produktu) – problem w jaki sposób przyznawać niskie oceny
- **Drugie podejście:** nieznane wartości funkcji użyteczności można próbować estymować na podstawie zgromadzonych danych w macierzy użyteczności.

Estymacja macierzy użyteczności

- **Drugie podejście:** nieznane wartości funkcji użyteczności można próbować estymować na podstawie zgromadzonych danych w macierzy użyteczności.
- Kluczowy problem: macierz U jest rzadka, bo każda osoba ocenia jedynie niewielką część produktów.
- Popularne są trzy podejścia do estymacji macierzy użyteczności:
  - Content-based recommendations
  - Model-based recommendations
  - Collaborative filtering
  - Hybrid approach

Content-based recommendations

- **Idea:** Rekomendować klientowi produkty podobne do tych, które wysoko ocenił.
- Każdy produkt jest opisywany przez wartości pewnych cech. Jest więc reprezentowany przez wektor w przestrzeni cech. Wektor ten nazywa się profilem produktu.
  - Przykład: Książka może być opisywana przez cechy: autor, tytuł, wydawnictwo, rok wydania. Profil książki to wektor w przestrzeni AUTORZY x TYTUŁY x WYDAWNICTWA x LATA.
  - Przykład: Artykuł prasowy może być opisywany przez TF-IDF. Profil artykułu prasowego to wektor w przestrzeni R^d, gdzie d to liczba używanych termów.
- Dla każdego klienta tworzy się jego profil, który też jest wektorem w przestrzeni cech produktów.
  - Przykład: Profil klienta może być określony jako średnia (lub mediana) profili produktów wysoko przez niego ocenionych.
  - Przykład: Profil klienta może być określony jako średnia ważona profili produktów przez niego ocenionych z wagami będącymi różnicą między oceną danego klienta a średnią oceną produktu przez wszystkich klienta.
- Rekomenduje się klientowi te produkty, których odległość od profilu jest niewielka. Popularna miara odległości to miara kosinusów.

Content-based recommendations

- Problemy:
  - trudności w definiowaniu właściwych cech produktów
  - małe prawdopodobieństwo rekomendowania produktów odległych od profilu klienta (a klienta może mieć różne zainteresowania – system rekomendujący je "uśredni")
  - problem nowego klienta – klient, który wcześniej ocenił zbyt mało produktów, będzie miał mało wiarygodny profil

Model-based recommendations

- **Idea:** Dla każdego klienta stworzyć system klasyfikujący produkty (na interesujące i nieinteresujące) na podstawie ocen, które klient dotychczas wystawił niektórym produktom. Następnie użyć systemu do klasyfikacji nieocenionych jeszcze produktów.
- System klasyfikujący można tworzyć na rozmaite sposoby: drzewa decyzyjne, sieci neuronowe, SVM, modele bayesowskie.
- Problemy:
  - duża złożoność obliczeniowa
  - konieczność ponownego tworzenia klasyfikatora po zarejestrowaniu nowych ocen
  - słaba skalowalność rozwiązania

Collaborative filtering

- **Idea:** Dla każdego klienta wyznaczyć zbiór klientów do niego podobnych, a następnie estymować oceny nieocenionych jeszcze produktów na podstawie ocen tych produktów wystawionych przez klientów podobnych.
- Miara podobieństwa klientów:
  - klient jest reprezentowany przez wektor swoich ocen (wiersz macierzy użyteczności), wektor liczb rzeczywistych długości |S|
  - przykład: popularna miara odległości klientów to miara kosinusów
  - przykład: popularna miara podobieństwa klientów to współczynnik Pearsona

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \bar{r_x})(r_{ys} - \bar{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \bar{r_x})^2 (r_{ys} - \bar{r_y})^2}}$$

  (S_{xy} to produkty ocenione przez obu klientów, r_{xs} to przyznane im oceny)
- Dla danego klienta c i danego produktu s, system rekomendujący wyznacza n najbliższych mu innych klientów, którzy ocenili produkt s.
- Prognozowana ocena produktu s przez klienta c, to średnia arytmetyczna ocen tego produktu przez wyznaczonych n klientów.
- Zamiast średniej arytmetycznej można rozpatrywać średnią ważoną z wagami proporcjonalnymi do odległości między klientem a rozważanym klientem c.
- Powyższe podejście nazywa się user-user collaborative filtering.

Collaborative filtering

- Analogicznie można opracować item-item collaborative filtering.
- Miara podobieństwa produktów:
  - produkt jest reprezentowany przez wektor swoich ocen (kolumnę macierzy użyteczności), wektor liczb rzeczywistych długości |C|
  - miary podobieństwa produktów są analogiczne jak w user-user collaborative filtering
- Dla danego klienta c i danego produktu s, system rekomendujący wyznacza n najbliższych mu innych produktów, które ocenił klient c.
- Prognozowana ocena produktu s przez klienta c, to średnia arytmetyczna ocen produktów podobnych wystawionych przez klienta c.
- Zamiast średniej arytmetycznej można rozpatrywać średnią ważoną z wagami proporcjonalnymi do odległości między produktem a rozważanym produktem s.

Collaborative filtering

- Podsumowanie collaborative filtering:
  - podejście działa dla produktów dowolnego typu (niewymagane jest opisywanie produktu przez cechy)
  - problem nowego użytkownika
  - problem nowego produktu
  - problem "Cold Start" – do działania systemu wymagane jest zgromadzenie pewnej liczby klientów i ich ocen pewnej liczby produktów
  - problem "First Rater" – system nie zarekomenduje produktu, który nie został jeszcze nigdy oceniony
  - problem "Popularity Bias" – system częściej będzie rekomendować popularne produkty, system nie trafi w gusta osób o wyjątkowych wymaganiach
  - Content-Based Recommendations nie miało problemów "Cold Start" ani "First Rater".

Collaborative filtering

- Rozszerzenia collaborative filtering:
  - klientów i produkty warto pogrupować, aby zredukować ich liczbę
  - klientów można pogrupować wcześniej i aktualizować to grupowanie w wolnym czasie (nie za każdym razem kiedy liczona jest rekomendacja, bo przydział do grup powinien być stabilny i długotrwały)
  - można połączyć collaborative filtering z analizą meta-danych (o klientach i o produktach)
  - można użyć reguł asocjacyjnych do wykrywania grup produktów kupowanych razem i wykorzystać to do rekomendowania zakupu następników reguł

Slope One

- Rozszerzenia Collaborative Filtering wprowadzone przez Daniel Lemire i Anna Maclachlan w 2005 roku.
- propozycja referatu na następne zajęcia