

Systemy Rekomendacji

Systemy rekomendacji oraz przykłady użycia

Bartłomiej Twardowski, @btwardow

Instytut Informatyki, Politechnika Warszawska

2019.03.02

- ① Szybki wstęp do systemów rekomendacji
- ② Typy Systemów Rekomendacji
- ③ Rekomendacje Kontekstowe
- ④ Spersonalizowany Ranking
- ⑤ Ewaluacja systemów rekomendacji
- ⑥ Przykład: Rekomendacje w e-commerce

Systemy rekomendacji - po co?

- odpowiedź na zalew informacji
- w e-commerce:
 - zamiana przeglądarka w zakupoholika
 - cross-selling oraz up-selling
 - budowanie lojalności klienta
- personalizacja całych serwisów
- spersonalizowane wyszukiwanie. SR często uważane za następców wyszukiwarek[19].

Rekomendować możemy: filmy, utwory muzyczne, książki, produkty, usługi, leki, tagi, artykuły, restauracje, partnerów życiowych, lekarstwa...

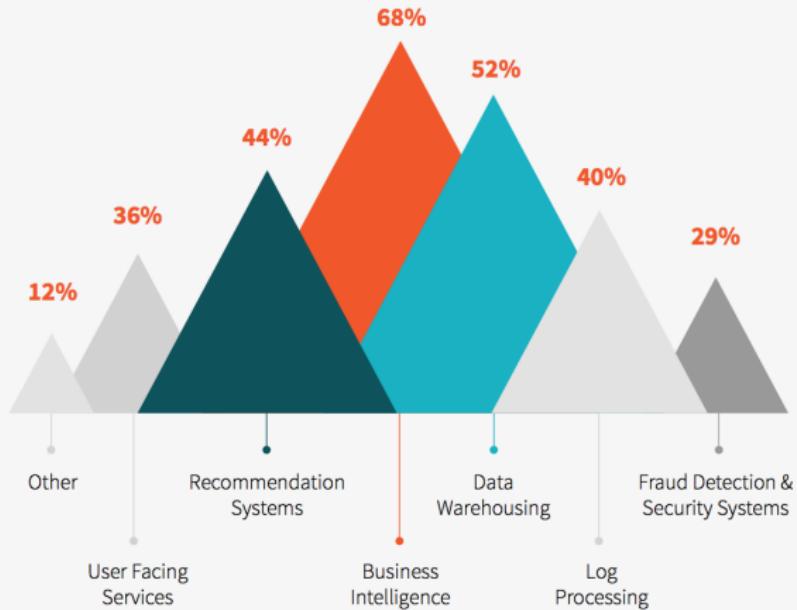
Jest to obszar w DM/ML aktywnie rozwijany. Konferencje: RecSys, SIGIR, KDD

- Netflix: 2/3 of the movies watched are recommended
- Google News: recommendations generate 38% more clickthrough
- Amazon: 35% sales from recommendations
- Choicestream: 28% of the people would buy more music if they found what they liked

Źródło: [46]

DataBricks Survey 2015

SPARK IS USED TO CREATE MANY TYPES OF PRODUCTS INSIDE OF DIFFERENT ORGANIZATIONS



[1] 1,417 respondents representing over 842 organizations

- łatwo jest nam stwierdzić co działa źle
- szybko możemy podać konkretny przykład/kontrzykład dobrej/złej rekomendacji
- nie myślimy o skali problemu, ale specyficznych przypadkach
- RS expert \neq Generator Super Inteligentnych Hipotez Rekomendacyjnych

Standardowo system rekomendacji rozwiązuje zadanie regresji. Dla każdego użytkownika $U = \{u_1, u_2, \dots\}$ oraz produktu $I = \{i_1, i_2, \dots\}$ należy znaleźć funkcję docelową $y : U \times I \rightarrow \mathbb{R}$. Funkcja docelowa to funkcja oceny produktu i przez użytkownika u - $y(u, i)$.

Mając podany zbiór obserwacji wejściowych, wartości $y(u, i)$ dla zbioru $S \subset U \times I$, zadaniem jest predykcja \hat{y} oceny dla każdego użytkownika oraz produktu.

Inne przedstawienie problemu: estymacja brakujących wartości w macierzy $R^{|U| \times |I|}$ z preferencjami użytkowników.

Dlaczego nie klasyczne podejście do regresji/klasyfikacji?

- zmienne w systemach rekomendacji to zmienne z domeną kategorycznej, gdzie przestrzeń jest bardzo duża i nie posiadamy o niej żadnej wiedzy a-priori
- dane wejściowe - obserwacje są bardzo rzadkie $|S| \ll |U \times I|$ (last.fm-99.998%, ML100K-93,7%, DBbook-99,86% niezaobserwowanych wartości)
- często problemem predykcji jest ranking obserwacji jednej zmiennej mając dany wektor obserwacji innej zmiennej (kontekst), nie tworzymy jednego globalnego rankingu
- oczekujemy zwrócenia N najlepiej dopasowanych przedmiotów (przykładowo AR zwróci współkupowane, ale ciężko uzyskać zawsze N rekomendacji)

Typy Systemów Rekomendacji

Podział systemów rekomendacji

Istnieje wiele taksonomii w zależności od aspektu na którym się koncentrujemy:

- dane: rating, ranking, location-aware, social-relationship
- memory based/model based
- użyta podejście: probabilistyczne, NN, BN, SVD, ...
- performance: real-time, offline

Jeżeli rozpatrujemy systemy rekomendacji jako podkласę systemów IR(filtering), możemy dokonać podziału na:

- Collaborative Filtering
- Content-Based Filtering
- Hybrid Systems
- Inne: *Knowledge-Based, Demographic, Social/Trust-Based*

- bazujemy na założeniu, że podobni do siebie użytkownicy będą dokonywać takie same decyzje[22]
- wykorzystujemy historię zachowań użytkowników w systemie
- Opinia użytkownika o ofercie może być wyrażona:
 - explicit (jawną) : ocena np. 1-5, "polubienie", sortowanie ofert
 - implicit (niejawną) : oglądanie/przesłuchanie oferty, czas spędzony na odwiedzanej stronie[47], zakup, komentarz, wyszukiwania,...
- w metodzie tej wierzymy w tz. „mądrość tłumu”

Collaborative Filtering - kNN

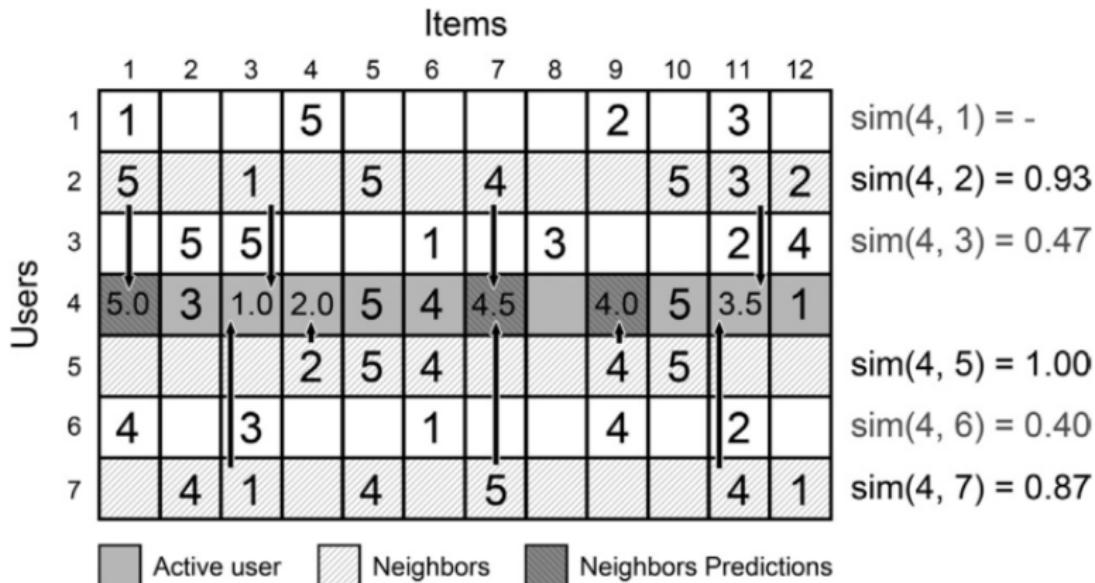


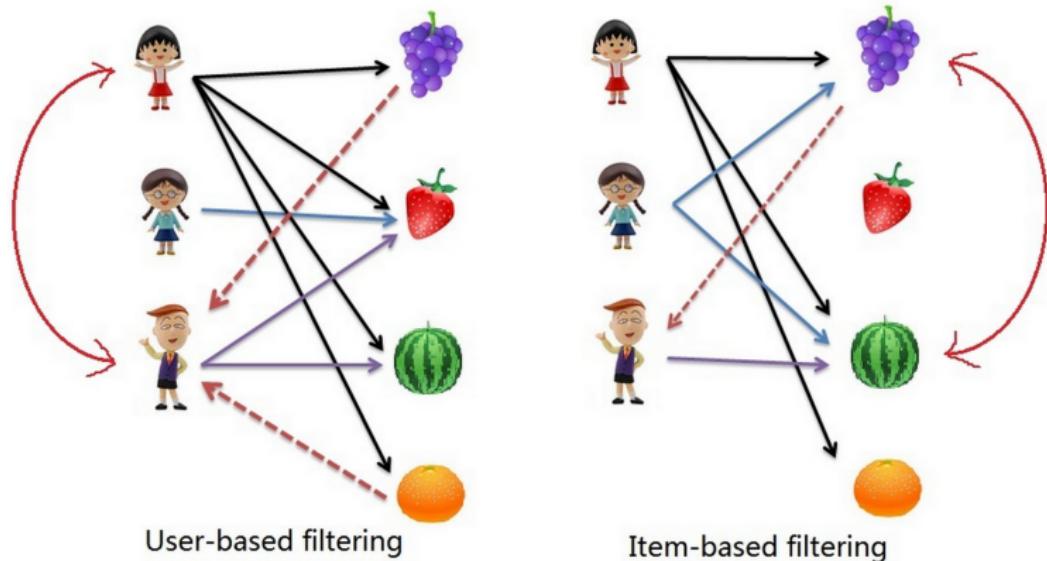
Fig. 4. User to user kNN algorithm example, $k = 3$. Similarity measure: 1 – (mean squared differences). Aggregation approach: average.

Źródło: [7]

- Log-Likelihood
- Tanimoto
- Cosine
- Pearson
- odmiany ważone
- ...

Collaborative Filtering - User2User vs Item2Item

Item2Item podejście wykorzystane przez Amazon od 2003[27].



Źródło: <http://thegeek.com/do-you-know-about-collaborative-filtering/>

Collaborative Filtering

Zalety

- metoda niezależna od domeny
- brak dodatkowych informacji o użytkownikach i tym co rekomendujemy (mogą być jedynie identyfikatory)
- łatwość użycia (popularność Mahout/Spark)

Wady

- cold-start problem
- preprocessing: outliers, usuwanie globalnych efektów (przykładowo: zawyżonej średniej pojedyńczego użytkownika)
- ciężko jest uzyskać sensowne rekomendacje dla użytkownika o unikalnych preferencjach gdy *podążamy za tłumem*
- rzadkość danych
- skalowalność - rośnie wraz z wzrostem liczby użytkowników oraz produktów

Wykorzystanie faktoryzacji macierzy

Cel

zmniejszenie wymiarowości problemu do najbardziej znaczących cech

Najprostsze sformułowanie problemu [41]: U - zbiór użytkowników, I - zbiór przedmiotów. Niech R będzie macierzą $|U| \times |I|$ zawierającą wszystkie preferencje użytkowników dla wszystkich przedmiotów. Chcemy odkryć K cech ukrytych takich, że mając dwie macierze P ($|U| \times K$) oraz Q ($|I| \times K$) otrzymujemy macierz \hat{R} :

$$\hat{R} = PQ^T$$

Aby dokonać predykcji dla użytkownika u_i oraz przedmiotu i_j obliczamy:

$$\hat{r}_{ij} = \mathbf{p}_i \mathbf{q}_j^T = \sum_{k=1}^K p_{ik} q_{jk}$$

Wykorzystanie faktoryzacji macierzy

Aby obliczyć macierze P i Q staramy się zminimalizować błąd na obserwacjach gdzie użytkownik określił preferencję względem przedmiotu - wcześniej zdefiniowane S .

$$E = \sum_{(u_i, i_j) \in S} (r_{ij} - \hat{r}_{ij})^2 = \sum_{(u_i, i_j) \in S} (r_{ij} - \mathbf{p}_i \mathbf{q}_j^T)^2$$

Dodając regularyzację [49]:

$$E = \sum_{(u_i, i_j) \in S} (r_{ij} - \mathbf{p}_i \mathbf{q}_j^T)^2 + \frac{\beta}{2} \sum_{k=1}^K (\|P\|^2 + \|Q\|^2)$$

Macierze P i Q zawierają odwzorowanie powiązania do ukrytych cech (możemy więc zrobić np. DIMSUM na Q).

Funk SVD[5]

Monday, December 11, 2006

Netflix Update: Try This at Home



Stochastic Gradient Descent:

$$q_{ik} \leftarrow q_{ik} + \gamma((r_{ui} - \sum_{k=1}^K p_{uk} q_{ik})p_{uk} - \lambda q_{ik})$$

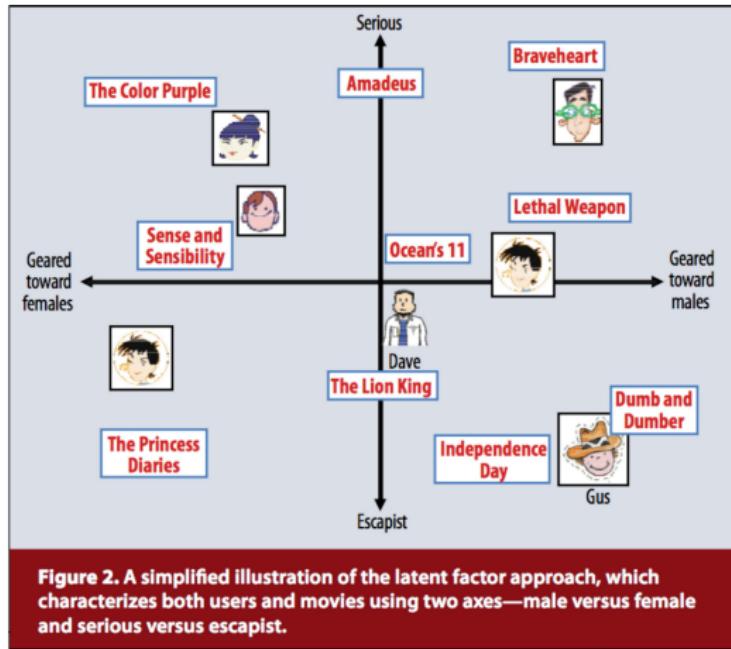
$$p_{uk} \leftarrow p_{uk} + \gamma((r_{ui} - \sum_{k=1}^K p_{uk} q_{ik})q_{ik} - \lambda p_{uk})$$

Przykładowy kod z użyciem GD:

http://btwardow.github.io/data_science/reco/2014/09/22/matrix-factorization.html

Netflix Prize

Metody faktoryzacji macierzy bazujące na SVD zyskały dużą popularność przy okazji konkursu Netflix Prize. Nagroda - 1M\$. Zwycięska drużyna: *BellKor's Pragmatic Chaos*.



Alternating Least Squares (ALS)

Zaproponowana przez Bell, Koren w 2007 [6].

Optymalizujemy oddzielnie macierze P i Q - zamrażając wartości w drugiej macierzy. Problem staje się wtedy o złożoności $O(|R|k^2 + |I|k^3)$.

Macierze inicjalizujemy małymi wartościami losowymi.

Jeżeli robimy to naprzemiennie zbliżamy się do globalnego optimum.

Alternating Least Squares (ALS)

Jeżeli zamrożymy wartości w macierzy cech przedmiotu Q , wtedy dla każdego użytkownika wektor wag dla cech ukrytych wynosi:

$$p_u = \left(QS^u Q^T + \lambda n_u \mathcal{I} \right)^{-1} Q r_u^T$$

gdzie $S^u \in \mathbb{R}^{|I| \times |I|}$ to macierz diagonalna taka w której:

$$S_{jj}^u = \begin{cases} 1 & \text{if } r_{uj} \neq 0 \\ 0 & \text{else} \end{cases}$$

a n_u to suma wszystkich obserwacji użytkownika u (w R napisalibyśmy: $\text{sum}(S^u)$).

W „naprzemiennym” kroku identycznie postępujemy dla q_i , a zamrażamy macierze cech użytkownika P .

Alternating Least Squares (ALS)

Cytat o rozpraszaniu obliczeń w ML [16]

We can think of two reasons for using distributed machine learning: because you have to (so much data), or because you want to (hoping it will be faster). Only the first reason is good.

ALS przeszedł ewolucje:

naive - bez optymalizacji wydajnościowych

broadcast - przesyłamy zamrożoną macierz do wszystkich węzłów

blocked - zarówno R jak i zamrożoną macierz dzielimy na bloki które przesyłamy, aktualnie w Spark

Polecam: [http://data-artisans.com/
how-to-factorize-a-700-gb-matrix-with-apache-flink/](http://data-artisans.com/how-to-factorize-a-700-gb-matrix-with-apache-flink/)
Inne których nie znajdziecie jeszcze w Spark :-): conjugate gradient(CG) and coordinate descent(CD) [20], ALS with incremental learning bias [50].

Zalety

- duża rzadkość danych przestaje być problemem
- wydajność - w *offline* obliczamy P i Q (*ALS*), w *online* predykcja polega na wymnożeniu wektora o długości $|K|$ i macierzy $|I| \times K$
- implementacje dostępne na większość platform/języków (np. SVD++, ALS lub podstawowe NMF)

Wady

- cold-start problem
- hyper-parameters: $K, \lambda, \beta, \text{liczba iteracji}$
- cechy których nie jesteśmy w stanie zinterpretować

Content-Based Filtering

- rekomendacje bazują na informacji o samym przedmiocie rekomendacji (*content*) nie biorąc pod uwagę opinii oraz interakcji innych użytkowników
- system rekomendacyjny stara się zwrócić przedmioty podobne (komplementarne lub w innej relacji - bazując na naszej wiedzy) do tych którymi użytkownik był zainteresowany
- w celu określenia podobieństwa dokonywana jest ekstrakcja cech przedmiotów rekomendowanych
- cechami w zależności od produktu mogą być: atrybuty przedmiotu, tagi, termy z tekstu opisującego, synsety z WordNet, sygnały z dźwięku, miary jakości obrazu, dostępna ontologia, genom utworu z Music Genome Project, trójki dbpedii,...
- Wykorzystanie *Linked Open Data* w systemach rekomendacji [30, 31, 8].

Content-Based Filtering - Przykład

Dwie funkcje:

$$\text{Content}(i) \rightarrow \mathbb{R}^d$$

zwraca reprezentacje przedmiotu i w przestrzeni d wymiarowej liczb rzeczywistych

$$\text{ContentBasedProfile}(u) \rightarrow \mathbb{R}^d$$

zwraca profil użytkownika u zbudowany na historii które przedmioty $i \in S_u$ oglądał. Agregacja Content na danych historycznych do tej samej przestrzeni \mathbb{R}^d .

Wartość dopasowania jest funkcja oceny podobieństwa $score$:

$$\hat{r}_{u,i} = score(\text{ContentBasedProfile}(u), \text{Content}(i))$$

Przykład 1 - Obie funkcje *ContentBasedProfile* oraz *Content* mogą zwracać *TF/IDF*. Funkcja *score* - cosinus kąta między wektorami.

Przykład 2 - Najbanalniejszy: Indeks Lucene i wykorzystanie MoreLikeThis :-)

Przykład 3 - Bardziej odważnie: wykorzystanie *word2vec* dla opisu naszych ofert. Ale dla specyficznych domen należy wytrenować *word2vec* na swoim korpusie danych.

- **Local Sensitivity Hashing** [23, 18], zachowujemy bliskość miary w obrębie "koszyka", np. dla Tanimoto/Jackard może być *MinHash*. Rekomendacje na Google News - *PLSI + LSH-MinHash* [15]
- **DIMSUM**[48] - Twitter, dla porównywania cosinusów wszystkich par wierszy w macierzy. Dostępne w Mllib na platformie Spark. Najlepiej kiedy macierz jest "wysoka".
- **Annoy** - memory efficient hash index, Spotify, <https://github.com/spotify/annoy>. Indeks przechowuje podobieństwa 20M utworów.

Content-Based Filtering

Zalety

- brak problemu cold-start lub niewystarczającej liczby zdarzeń o preferencjach użytkownika
- możemy już coś sensownego podsunąć użytkownikowi o niestandardowych preferencjach
- rekomendujemy przedmioty które nie koniecznie muszą być popularne, ale rekomendację w tym przypadku łatwo jest wytłumaczyć

Wady

- ekstrakcja cech - specyficzna dla domeny i nie do końca łatwa
- łatwo nadmiernie dopasować model, cierpi na tym element poznawczy rekomendacji (novelty)

Weighted The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.

Switching The system switches between recommendation techniques depending on the current situation.

Mixed Recommendations from several different recommenders are presented at the same time

Feature combination Features from different recommendation data sources are thrown together into a single recommendation algorithm.

Cascade One recommender refines the recommendations given by another.

Feature augmentation Output from one technique is used as an input feature to another.

Meta-level The model learned by one recommender is used as input to another.

- **Knowledge-Based Recommenders** - wykorzystanie wiedzy domenowej o przedmiotach, użytkownikach lub samym fakcie rekomendacji. Przykład: conversational recommendations - critiquing-based [12]
- **Social (Trust-Based) Recommenders** - rekomendacje bazujące na zaufaniu w sieciach społecznościowych. Algorytmy używane w rekomendacjach: Advogato, Appleseed, MoleTrust, TidalTrust
- **Demographic** - kategoryzacja użytkownika na bazie jego atrybutów (profilu) i próba rekomendacji bazującej np. na geolokalizacji

Rekomendacje Kontekstowe

Context-Aware Recommender Systems (CARS)

Istnieje dodatkowa informacja która wpływa na preferencje użytkownika.

Zdefiniujmy kontekst jako $c \in \mathcal{C}$.

Aby mówić o rekomendacjach kontekstowych musimy być w stanie określić *kto* (u) oceniał *co* (i) w jakim *kontekście* (c).

Przykłady kontekstu: czas ($\mathcal{C} = R^+$), przeglądarka internetowa której korzysta użytkownik ($\mathcal{C} = \text{useragent}_1, \dots, \text{useragent}_2$), ostatnio przeglądana/zakupiona oferta ($\mathcal{C} = \mathcal{P}(\mathcal{I})$).

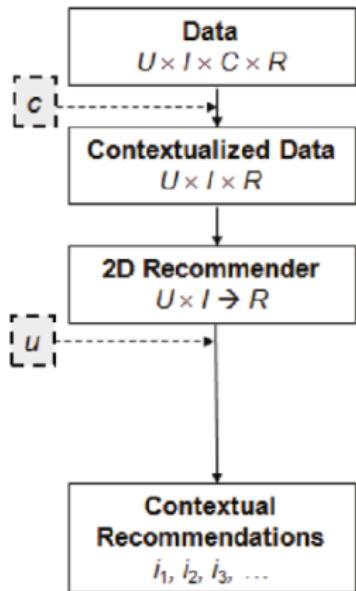
W takim modelu poszukiwana funkcja oceny y może być zdefiniowana jako:

$$y : U \times I \times \mathcal{C}_1 \times \dots \mathcal{C}_m \rightarrow \mathbb{R}$$

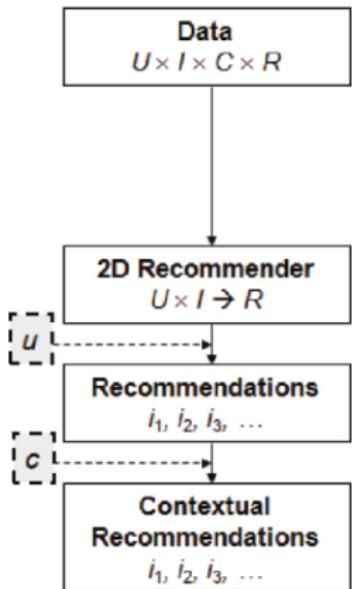
gdzie $\mathcal{C}_1, \dots, \mathcal{C}_m$ to zdefiniowane konteksty.

Context-Aware Recommender Systems - Podejścia

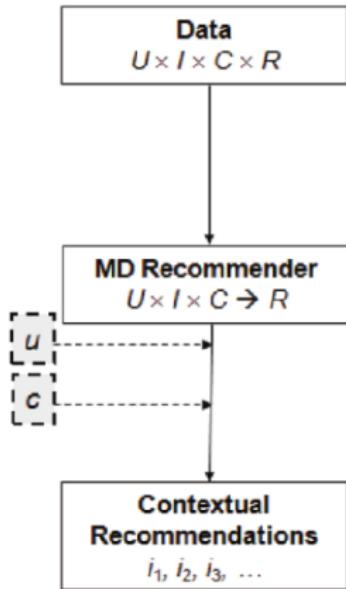
(a) Contextual Pre-Filtering



(b) Contextual Post-Filtering



(c) Contextual Modeling



Źródło: [36] - Chapter 7

- matrix factorization: dedykowane np. timeSVD++, General Factorization Framework[43], CARS2[39]
- tensor decomposition: HOSVD, TD/PARAFAC/PITF[32]
- Factorization Machines[33, 35]

Context-Aware Recommendation - Multiverse Recommendation/PARAFAC/PITF

Metody bazujące na dekompozycji tensorowej Tuckera (TD):

- *Multiverse Recommendation*[24]
- *Parallel Factor Analysis (PARAFAC)*[32]
- *Pairwise Interaction Tensor Factorization(PITF)*[32] - tensor wejściowy jest zbudowany z porównań obiektów rekomendowanych *I* parami w każdym kontekście. Tensor \mathcal{B} jest diagonalny, a odpowiednie (nieużywane w danym kontekście) kolumny w V_1, \dots, V_k wypełnione 1.

Wszystkie stosują RMSE przy optymalizacji.

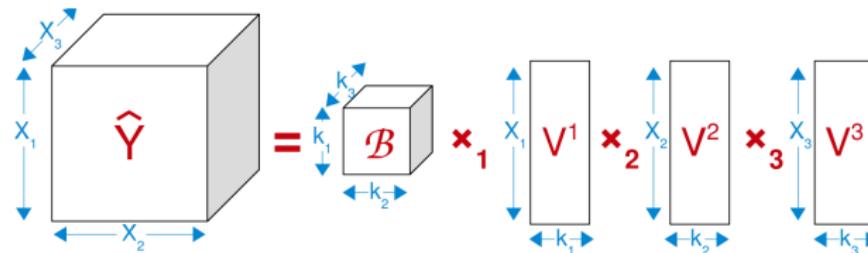


Fig. 5.1 Tucker decomposition (3-mode): The target \hat{Y} is approximated by a factorization \hat{Y}

- Factorization Machines - 2010, autor S.Randle [33, 35].
- próba połączenia zalet SVM z modelami bazującymi na faktoryzacji macierzy.
- W przeciwieństwie do standardowych SVM wszystkie zależności pomiędzy zmiennymi są modelowane z wykorzystaniem parametrów które są faktoryzowane - stąd FM są w stanie estymować zależności dla bardzo rzadkich danych (systemy rekomendacji).
- FM dokonują predykcji z dowolnych danych posiadających wartości rzeczywiste
- Ranking może być uczyony na parach obserwacji (i_a, i_b) , gdzie i_a ma większą ocenę od i_b .

Równanie modelu predykcji wartości oceny dla wymiaru $d = 2$ jest opisane:

$$\hat{y}(\mathbf{x}) = w_0 + \sum_{i=1}^n w_i x_i + \sum_{i=1}^n \sum_{j=i+1}^n \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

gdzie parametry do estymacji to:

$$w_0 \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^n, \mathbf{V} \in \mathbb{R}^{n \times k}$$

i $\langle ., . \rangle$ to iloczyn skalarny dwóch wektorów o rozmiarze k :

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^k \mathbf{v}_{i,f} \cdot \mathbf{v}_{j,f}$$

a wiersz $\mathbf{v}_i \in \mathbf{V}$ opisuje i-tą zmienną k -cechami.

Factorization Machines - Przykład modelowania z kontekstem dla rekomendacji

Przykład [33]:

Mamy zapis który użytkownik $u \in U$ oceniał film (item) $i \in I$ w określonym czasie $t \in \mathbb{R}$ w skali $r \in 1, 2, 3, 4, 5$. $U = \text{Alice (A), Bob (B), Charlie (C)}$ $I = \text{Titanic (TI), Notting Hill (NH), Star Wars (SW), Star Trek (ST)}$ $S = (\text{A, TI}, 2010-1, 5), (\text{A, NH}, 2010-2, 3), (\text{A, SW}, 2010-4, 1), (\text{B, SW}, 2009-5, 4), (\text{B, ST}, 2009-8, 5), (\text{C, TI}, 2009-9, 1), (\text{C, SW}, 2009-12, 5)$

Feature vector \mathbf{x}										Target y										
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...
A User	B	C	...		TI	NH	SW	ST	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Last Movie rated
Movie										Other Movies rated	Time									

- Różne podejścia do optymalizacji w FM: *MCMC, SGD, ALS.*
- Drugie miejsce na KDD Cup 2012
(<http://www.kddcup2012.org/>) [34].
- Implementacja autora w C++: libfm.org.
- W Julia: <https://github.com/btwardow/FactorizationMachines.jl>
- Rozszerzenie: Field Aware Factorization Machines

Spersonalizowany Ranking

Dlaczego algorytmy rankingu Top-N rekomendacji?

- W praktyce często musimy wybrać małą liczbę N -najlepszych ofert wybranych dla użytkownika z względu na umiejscowienie rekomendacji - karuzela z ofertami na stronie internetowej, urządzenie mobilne
- rezultat jest przedstawiony na **posortowanej** liście
- stąd zamiast przewidywać ocenę jaką użytkownik wystawiłby wszystkim przedmiotom powinniśmy optymalizować całą listę N pokazanych użytkownikowi ofert
- generowanie Top-N najbardziej relevantnych rekomendacji dla indywidualnego użytkownika to tak naprawdę utworzenie **spersonalizowanego rankingu**
- badanie jakości rekomendacji przy użyciu miar dla rankingu: *NDCG, MRR, MAP, ERR*

Dziedziną ML zajmującą się rankingowaniem jest *learning to rank*.

Przykład

Mamy produkt i z oceną 4 oraz produkt j z oceną 3 która zostanie użyta do weryfikacji poprawności predykcji. Mamy dane dwa różne podejścia do rekomendacji gdzie wynikami dla $[i, j]$ są wartości: [3, 4] oraz [5, 2].

Predykcja oceny

Z punktu widzenia MAE lub RMSE oba systemy są jednakowo dobre (kiepskie).

Predykcja rankingu

Z perspektywy rankingu, w drugim przypadku przewidywana ocena bardziej dokładnie oddaje relatywne preferencje użytkownika gdzie przedmiot i został wyżej oceniony niż j .

Learning to Rank - Podejścia

Point-wise	Pair-wise	List-wise
\rightarrow Score(C)	$\rightarrow f(A) > f(B)$	$\rightarrow P_{A,B,C}$
\rightarrow Score(B)	$\rightarrow f(A) > f(C)$	$\rightarrow P_{B,A,C}$
\rightarrow Score(A)	$\rightarrow f(B) > f(C)$	$\rightarrow P_{B,C,A}$
		\vdots
S(A)>S(B)>S(C)	$f(A) > f(B), f(A) > f(C), f(B) > f(C)$	$P_{A,B,C} > P_{B,A,C} > P_{B,C,A} \dots$

- **Pointwise** - funkcja rankingu minimalizuje funkcję celu dla pojedyńczych obserwacji, problem: regresja, klasyfikacja
- **Pairwise** - funkcja celu zdefiniowana na parach lepszy-gorszy, minimalizujemy odwrócenia par w rankingu, problem: binarna klasyfikacja
- **Listwise** - funkcja celu zdefiniowana na całych listach (np. maksymalizacja NDCG), problem: funkcje rankingu są nieróżniczkowalne - Direct optimization of ranking measures[25]

- **Pointwise** - GLM, SVM, GBDT. RS: MF
- **Pairwise** - RankSVM, RankBoost, FRank, RS: MF-BPR
- **Listwise** - SVM-MAP,... RS: ListRank-MF[40], AdaRank (boosting-NDCG), CLiMF/xCLiMF[38], RankALS[42], SLIM[29]

Evaluacja systemów rekomendacji

Predykcja oceny

RMSE, MAE, Precision/Recall/F-Measure

Rankingowe[14]

AUC, Precision@N, Recall@N, MRR, MAP, NDCG, ERR[11]

Inne[14]

Novelty, Diversity, Serendipity

Predykcja oceny

$$MAE = \frac{1}{|\mathcal{S}|} \sum_{(u,i) \in \mathcal{S}} |\hat{r}_{ui} - r_{ui}| \quad (1)$$

$$MSE = \frac{1}{|\mathcal{S}|} \sum_{(u,i) \in \mathcal{S}} (\hat{r}_{ui} - r_{ui})^2 \quad (2)$$

$$RMSE = \sqrt{MSE} \quad (3)$$

Ranking - Area Under the Curve

Area Under the ROC Curve (AUC) jest oceną jakości rankingu, gdzie wartość jest z przedziału 0.0 – 1.0. Rezultaty nielosowe - $AUC > 0.5$. Wartość obliczamy parami wg. wzoru:

$$AUC = \frac{1}{|\mathcal{I}_S| |\mathcal{I} \setminus \mathcal{I}_S|} \sum_{i \in \mathcal{I}_S} \sum_{j \in \mathcal{I} \setminus \mathcal{I}_S} \delta(\hat{r}(i) < \hat{r}(j))$$

Intuicyjne wyjaśnienie: *Wartość AUC określa prawdopodobieństwo wylosowania dwóch przykładów, takich, że ich relatywny ranking jest właściwy.*

Ranking - Mean Reciprocal Rank

Dla odpowiedzi z systemu \mathcal{S} wartość MRR jest zdefiniowana jako:

$$\text{MRR} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{\mathcal{S}} \frac{1}{1 + \text{rank}_i}$$

Jest kilka podejść do wyliczania tych miar, tu będziemy korzystać z [7, ?], gdzie jest wykorzystywana funkcja podobieństwa przedmiotów:

$$diversity_u = \frac{1}{|\mathcal{R}_u|(|\mathcal{R}_u| - 1)} \sum_{i \in \mathcal{R}_u} \sum_{j \in \mathcal{R}_u, j \neq i} (1 - s_{i,j})$$

$$novelty_i = \frac{1}{|\mathcal{R}_u| - 1} \sum_{j \in \mathcal{R}_u, j \neq i} (1 - s_{i,j}), i \in \mathcal{R}_u$$

$s_{i,j}$ jest miarą podobieństwa pomiędzy przedmiotami i i j . Można tu użyć znane miary podobieństwa z CF.

Dla tak zdefiniowanych miar, Diversity jest średnią wartością Novelty dla przedmiotów w zbiorze odpowiedzi.

Jeżeli \mathcal{R}_u definiuje rezultat dla użytkownika u a \mathcal{R}'_u jest rezultatem z rekomendera określonego jako *podstawowy*, wtedy:

$$serendipity_u = \frac{1}{|\mathcal{R}_u \setminus \mathcal{R}'_u|} \sum_{i \in \mathcal{R}_u \setminus \mathcal{R}'_u} rel_{u,i} \quad (4)$$

gdzie su, i jest funkcją relevantności przedmiotu i dla użytkownika u . Może tu być wykorzystane podobieństwo przedmiotu i do profilu użytkownika u z CBF, np. cosine lub Tanimoto.

Inne podejścia do rekomendacji

- **Explore-Exploit** - Multi Armed/Contextual Bandit [9, 26]
- **Deep Learning** - Spotify[2], wykorzystanie RNN do predykcji utworów na playliście, próby w Netflix[3], Collaborative DL[45], Wide&Deep Learning for RS[13], RNN for Sessions[44]
- **Restricted Boltzman Machines** - próba w Netflix Prize[37], [17]
- **Hierarchical HMM** *Adapting Recommendations to Contextual Changes*[21]

Przykład: Rekomendacje w e-commerce

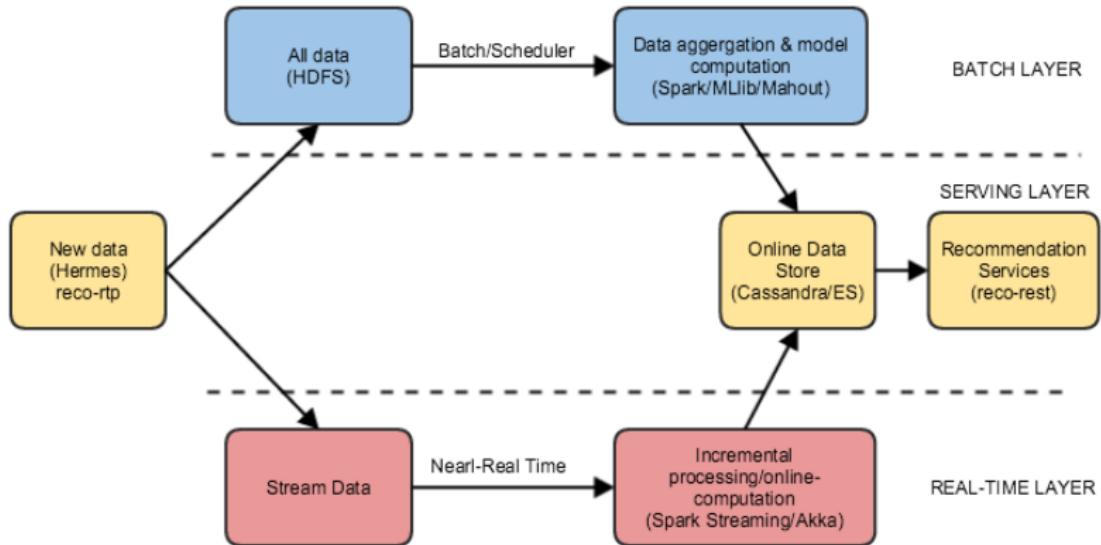
Cele:

- dostarczenie usługi Recommendation-as-a-Service (*RaaS*)
- rekomendacje dostępne dla wielu scenariuszy (np. Main Page WWW, mobilny koszyk, push notification, email itp.)
- możliwość łatwiej i szybkiej implementacji nowych algorytmów
- integracja z zewnętrznymi rozwiązaniami
- możliwość wykonywania eksperymentów i testów A/B (a nawet uruchomienie *Bayesian Bandit*)

Trochę o Platformie Rekomendacyjnej w liczbach:

- $\sim 16,9M$ Real Users (Gemius/PBI 8.2015 report)
- $\sim 40M$ aktywnych ofert
- $\sim 50M$ różnych zdarzeń o użytkownikach dla PR dziennie
- $\sim 40M$ wygenerowanych spersonalizowanych rekomendacji
- ~ 1300 req/s w szczytce

Platforma Rekomendacyjna - Architektura Lambda[28]



Trochę o tym z czego korzystamy

scala, spark, sbt, cassandra, elasticsearch, mllib, mahout, python, luigi, akka, kafka, hive, spray.io, GPU/CUDA, AWS, redis, hermes, mesos, R, Kibana, Gatling, Specs2, $\geq CDH5.3$ (YARN, Hive, Tez, Hue,...)

Podstawowym odnośnikiem dla rezultatów systemu rekomendacji często stanowią niespersonalizowane:

- przedmioty z najlepszą średnią oceną
- najpopularniejsze oferty (np. w danej kategorii)
- najlepiej sprzedające się produkty w e-commerce

Jeżeli system rekomendacji nie jest w stanie zaoferować nic sensownego, zawsze możemy zwrócić jedną z wyżej wymienionych opcji jako „koło ratunkowe”.

Bestsellers - co przecież może pójść źle?

W kategorii "Dom i Ogród":



✓ GARDENA Lanca
zraszająca Premium 90cm
8155

kup teraz 129,00 zł



✓ 47324 PATIO FOTELE
FOTEL OGRODOWY NA
TARAS BALKON

kup teraz 249,00 zł



KOSIARKA SPALINOWA
HECHT 546SXW 55L 46CM
+GRATIS

kup teraz 1 059,00 zł

4,5M+

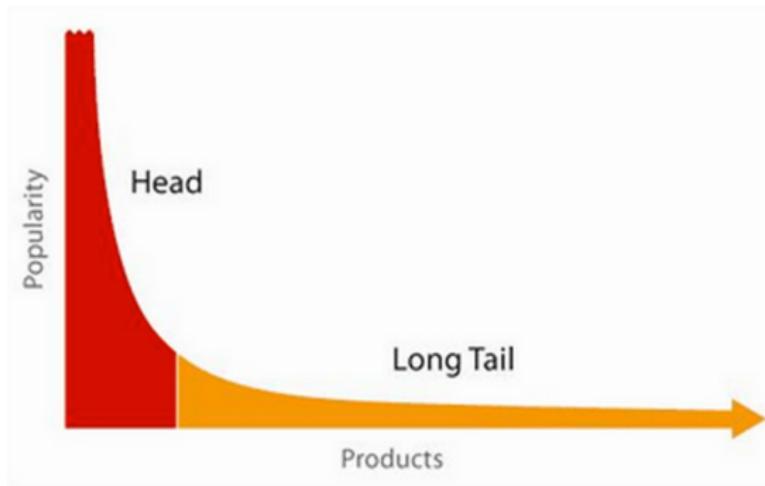


Niebawem otrzymaliśmy email:

...z ciekawości :) to był najpopularniejszy produkt na całym allegro?...

Long-tail Problem

Ale przez takie działania system rekomendacji nie spełnia swojej misji i nadal wydłużamy nasz już i tak długi ogon.



Źródło: http://www.slate.com/articles/arts/books/2006/07/the_wrong_tail.html

Oferty na serwisach aukcyjnych

- brak bytu produktu oraz usługi katalogującej
- ofertę opisuje wystawiający: nazwa, cena, kategoria, atrybuty oraz opis. Prawdziwym ograniczeniem aktualnie jest tylko ludzka wyobraźnia.
- drzewo kategorii nie do końca oddaje rzeczywistość - występuje problem klasyfikacji gdzie oferta powinna trafić
- sprzedawcy stosują różne obejścia, np. wystawianie po kilka przedmiotów raz dziennie, stosowanie w nazwach **NIE Zara, HM, Sephora**, cena przedmiotu - koszt dostawy,...

Drzewo kategorii - czy faktycznie to drzewo?

- Ilość liści w drzewie kategorii - 48757.
- Ilość liści w drzewie o różnych nazwach - 14275.
- Często występują liście typu: *Pozostałe, Inne miejscowości, Akcesoria, Zestawy, Inne, Uszkodzone* itp.



Problem efemeryczności

- cykl życia ofert może być bardzo krótki (np. godziny od wystawienia)
- takie oferty cięźko zarekomendować z kilu powodów:
 - brak historii użytkowników dla nowych ofert
 - w momencie budowania modelu rekomendacji danej oferty nie było w systemie
 - kiedy zarekomendujemy dany przedmiot oferta może być już zakończona lub prezentowana cena nieaktualna
- podobny problem stanowią np. newsy informacyjne,
- jednym z sposobów radzenia sobie z problemem jest wprowadzenie grupowania - bardziej stałych odpowiedników

- próba przybliżenia bytu "produkту"
- uzyskanie bytów bardziej trwałych do których możemy odnosić nowo powstałe oferty oraz zachowania użytkowników z historii
- rekomendowanie odbywa się na poziomie MI, np. MI-to-MI
- (ciekawostka) nazwy typów MI: *ARS Clusters, CAP, mojito, kamikaze, ... kolejne drinki ;-)*

Zalety

- rozwiązuje w dużej mierze problem efemeryczności ofert
- wykorzystujemy treść oferty (*content-based*) oraz źródła zewnętrzne (*knowledge-based*) przy rekomendacjach

Wady

- główna wada - ranking ofert wewnątrz MI nie jest spersonalizowany!
- utrzymywanie wielu modeli per typ MI
- interakcje użytkownik-oferta agregowane na poziomie MI, możemy zatracić cenne szczegóły

Model preferencji użytkownika - Przykład

Model preferencji użytkownika bazuje na jego niejawnej opinii (*implicit feedback*).

Przykładowe zdarzenia użytkownika w serwisie:

Login, Logout, Bid, Buy, Item View, Search, {Add to, Remove from} x {Cart, Watchlist, Favourites}, Recommendation Click

Przykład modelu preferencji użytkownika:

- do każdego typu zdarzenia przypisujemy wagę
- zbieramy zdarzenia z X ostatnich dni
- zdarzenia starsze mają mniejsze znaczenie niż nowsze - starzejemy wagę zdarzenia np. liniowo od czasu

Blender - oszustwo użytkownika?

Recoblender

wewnątrz platformy może mieszać rekomendacje z innych rekomenderów w dowolnych proporcjach.

Przykład konfiguracji:

```
MAIN_PAGE_BLENDER_LOGGED {  
    type = BLENDER  
    filterSameSeller = true  
    filterAlreadySeen = true  
    ingredients = [{  
        recommender = RECENTLY_VISITED_RECOMMENDER  
        amount = 1.0 f  
    }, {  
        recommender = cf_mp_ars_tanimoto  
        amount = 1.0 f  
    }]  
}
```



Sam fakt mieszania rekomendacji z wielu modeli nie jest łatwy i może zaburzyć mierzenie miar rankingowych dla pojedynczego rekomendera.

Box z rekomendacjami ma np. tytuł *Oferty wybrane na bazie Twoich zainteresowań*. Czy to jest prawda?

Offline

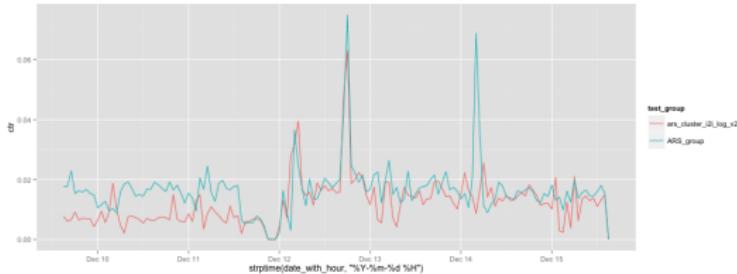
- standardowo jak w ML: train/validation/test
- w zależności od metody rekommendacji i modelu należy wykorzystać różne metryki (rating vs ranking)

Online

- badamy rezultaty jak dany scenariusz/algorytm działa na produkcyjnym środowisku
- oczywiście podstawowe marketingowe miary: *CTR, GMV*
- miary obliczane dla modelu offline warto obliczyć dla danych które spłyną z faktycznego wykorzystania online na produkcji i je porównać

Testy A/B

- testy A/B to podstawowa forma weryfikacji nowego podejścia/algorytmu z użytkownikami
- non-stop działają testy A/B weryfikujące nowe pomysły
- testy jesteśmy w stanie wykonać dla każdego scenariusza mówiąc jaki podział ruchu ma nastąpić
- testujemy odpowiednio długo aby uniknąć efektu nowości
- badamy istotność naszych testów



Fotelik czy siekierka? Oto jest pytanie...

16 lipca 2015 · 22:22 przez zakap (PW) | Skomentuj (11)

WTF!?

użytkownicy, szukający tego co Ty, oglądali

Produkt	Cena	Opis
Fotelik samochodowy ENCORE 9-36kg SKÓRA POCZYŁYNY	279,90 zł	do wyciągnięcia przedniem, 48 dostępnych ofert, 3 osoby klienci
FOTELIK SAMOCHODOWY PWBABY 9-36 kg COMFORT 2015	139,00 zł	nowy, 12 dni do końca, 299 dostępnych ofert, 1 osoba klienci
Deklers siekarka FREUDN 15 LAT GWAŁTAŃCA 2500 G	299,00 zł	nowy, 32/330, kup teraz
Siekarka z młotem 1kg VORTEX 32/330	39,70 zł	nowy, 32/330, kup teraz
MACZETKA KARCIŻOWA 35 CM HAMOTOWANE GOLCONA	16,99 zł	nowy, kup teraz
ZESTAW 3 SIEKAR 2500 1250 1000, PREMIUM SIEKARKA	121,50 zł	nowy, kup teraz

dobre 513

Źródło: <http://allegro.pl/listing/listing.p...>

f Udostępnij na Facebooku

Wyjaśnienia "Mistrzów" [4]:

- *Po prostu czasami musisz zabrać dziecko na ustawkę, a dzieci przewozi się w foteliku*
- *Proste - tańsza siekiera niż niechciane dziecko.*
- *Pomyśl lepiej czy dzisiejsza noc z małżonkiem/małżonką w jednym łóżku nie będzie Twoją ostatnią... .*
- *Pewnie użytkownik z Krakowa*
- *Po prostu - najpierw zakupy z myślą o dziecku, później z myślą o żonie.*

Q&A
Bartłomiej Twardowski
B.Twardowski@ii.pw.edu.pl, @btwardow

References I

-  Databricks spark spark survey 2015.
-  <http://erikbern.com/2014/06/28/recurrent-neural-networks-for-collaborative-filtering/>.
-  <http://techblog.netflix.com/2014/02/distributed-neural-networks-with-gpus.html>.
-  Mistrzowie - fotelik czy siekierka? oto jest pytanie...
-  Netflix update: Try this at home.
-  BELL, R. M., AND KOREN, Y.
Scalable collaborative filtering with jointly derived neighborhood interpolation weights.
In *Proceedings - IEEE International Conference on Data Mining, ICDM* (2007), pp. 43–52.
-  BOBADILLA, J., ORTEGA, F., HERNANDO, A., AND GUTIÉRREZ, A.
Recommender systems survey.
Knowledge-Based Systems 46 (July 2013), 109–132.

References II

-  BOGERS, T., AND KOOLEN, M.
Workshop on New Trends in Content-based Recommender Systems (CBRecSys 2014).
379–380.
-  BOUNEFFOUF, D., BOUZEGHOUB, A., AND GANÇARSKI, A. L.
A contextual-bandit algorithm for mobile context-aware recommender system.
In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2012), vol. 7665 LNCS, pp. 324–331.
-  BURKE, R.
Hybrid web recommender systems.
The adaptive web (2007), 377–408.
-  CHAPELLE, O., AND METLZER, D.
Expected reciprocal rank for graded relevance.
Proceedings of the 18th ... (2009).
-  CHEN, L., AND PU, P.
Critiquing-based recommenders: survey and emerging trends.
User Modeling and User-Adapted Interaction 22, 1-2 (2012), 125–150.

References III

-  CHENG, H.-T., KOC, L., HARMSEN, J., SHAKED, T., CHANDRA, T., ARADHYE, H., ANDERSON, G., CORRADO, G., CHAI, W., ISPIR, M., ET AL. Wide & deep learning for recommender systems.
In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (2016), ACM, pp. 7–10.
-  CREMONESI, P., KOREN, Y., AND TURRIN, R. Performance of recommender algorithms on top-n recommendation tasks.
In *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10* (2010), p. 39.
-  DAS, A. Google News Personalization : Scalable Online. 271–280.
-  FASTML. <http://fastml.com/the-emperors-new-clothes-distributed-machine-learning/>.
-  GEORGIEV, K., AND NAKOV, P. A non-iid framework for collaborative filtering with restricted boltzmann machines.
In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (2013), pp. 1148–1156.

References IV

-  GIONIS, A., INDYK, P., AND MOTWANI, R.
Similarity Search in High Dimensions via Hashing.
Search 99, 1 (1999), 518–529.
-  GUY, I., JAIMES, A., AGULLÓ, P., MOORE, P., NANDY, P., NASTAR, C., AND SCHINZEL, H.
The RecSys 2010 Industry Panel Will Recommenders Kill Search ? Recommender Systems – An Industry Perspective.
Search (2010), 7–12.
-  HIDASI, B., AND TIKK, D.
Speeding up ALS learning via approximate methods for context-aware recommendations.
Knowledge and Information Systems (2015).
-  HOSSEINZADEH AGHDAM, M., HARIRI, N., MOBASHER, B., AND BURKE, R.
Adapting recommendations to contextual changes using hierarchical hidden markov models.
In *Proceedings of the 9th ACM Conference on Recommender Systems* (New York, NY, USA, 2015), RecSys '15, ACM, pp. 241–244.

References V

-  HUANG, Z. H. Z., ZENG, D., AND CHEN, H. C. H.
A Comparison of Collaborative-Filtering Recommendation Algorithms for E-commerce.
IEEE Intelligent Systems 22 (2007).
-  INDYK, P., AND MOTWANI, R.
Approximate nearest neighbors: towards removing the curse of dimensionality.
Proceedings of the thirtieth annual ACM symposium on Theory of computing 126 (1998), 604–613.
-  KARATZOGLOU, A., AMATRIAIN, X., AND OLIVER, N.
Multiverse Recommendation : N-dimensional Tensor Factorization for Context-aware Collaborative Filtering.
Context (2010), 79–86.
-  LE, Q., AND SMOLA, A.
Direct optimization of ranking measures.
arXiv preprint arXiv:0704.3359 1, 2999 (2007), 1–29.
-  LI, L., CHU, W., LANGFORD, J., AND SCHAPIRE, R. E.
A Contextual-Bandit Approach to Personalized News Article Recommendation.
Www 2010 (2010), 10.

References VI

-  LINDEN, G., SMITH, B., AND YORK, J.
Amazon.com recommendations: item-to-item collaborative filtering.
IEEE Internet Computing 7 (2003).
-  NATHAN MARZ AND JAMES WARREN.
Big data - Principles and best practices of scalable realtime data systems
(Chapter 1).
-  NING, X., AND KARYPIS, G.
SLIM: Sparse Linear Methods for top-N recommender systems.
Proceedings - IEEE International Conference on Data Mining, ICDM (2011),
497–506.
-  OSTUNI, V. C., NOIA, T. D., SCIASCIO, E. D., AND MIRIZZI, R.
Top-N Recommendations from Implicit Feedback leveraging Linked Open Data.
85–92.
-  PESKA, L., AND VOJTAŠ, P.
Using Linked Open Data to Improve Recommending on.
-  RENDLE, S.
Context-aware ranking with factorization models.
2010.

References VII

-  RENDLE, S.
Factorization machines.
In *Proceedings - IEEE International Conference on Data Mining, ICDM* (2010), pp. 995–1000.
-  RENDLE, S.
Social network and click-through prediction with factorization machines.
KDD-Cup Workshop (2012).
-  RENDLE, S., GANTNER, Z., FREUDENTHALER, C., AND SCHMIDT-THIEME, L.
Fast context-aware recommendations with factorization machines.
In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information* (2011), pp. 635–644.
-  RICCI, F., ROKACH, L., SHAPIRA, B., AND KANTOR, P. B., Eds.
Recommender Systems Handbook.
Springer US, Boston, MA, 2011.
-  SALAKHUTDINOV, R., MNIH, A., AND HINTON, G.
Restricted Boltzmann Machines for Collaborative Filtering.
791–798.

References VIII

-  SHI, Y., KARATZOGLOU, A., AND BALTRUNAS, L.
xCLiMF: optimizing expected reciprocal rank for data with multiple levels of relevance.
Proceedings of the 7th ... (2013), 0–3.
-  SHI, Y., KARATZOGLOU, A., BALTRUNAS, L., AND LARSON, M.
CARS 2 : Learning Context-aware Representations for Context-aware Recommendations.
-  SHI, Y., AND LARSON, M.
List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering.
RecSys '10 Proceedings of the fourth ACM conference on Recommender systems (2010), 269–272.
-  TAKÁCS, G.
Matrix Factorization and Neighbor Based Algorithms for the Netflix Prize Problem Categories and Subject Descriptors.
267–274.
-  TAKÁCS, G., AND TIKK, D.
Alternating least squares for personalized ranking.
Proceedings of the 6th ACM conference on Recommender systems - RecSys '12 (2012), 83.

References IX

-  TIKK, D.
General factorization framework for context-aware recommendations.
Data Mining and Knowledge Discovery (2015).
-  TWARDOWSKI, B.
Modelling contextual information in session-aware recommender systems with neural networks.
In *Proceedings of the 10th ACM Conference on Recommender Systems* (2016), ACM, pp. 273–276.
-  WANG, H., WANG, N., AND YEUNG, D.-Y.
Collaborative Deep Learning for Recommender Systems.
In *Proc. 18th International Conf. on Machine Learning* (2014), 314–321.
-  XAVIER AMATRIAIN, B. M.
Kdd2014 tutorial - the recommender problem revisited.
-  YI, X., HONG, L., ZHONG, E., NAN, N., AND SUJU, L.
Beyond Clicks : Dwell Time for Personalization.
113–120.
-  ZADEH, R.
All-pairs similarity via dimsum.
<https://blog.twitter.com/2014/all-pairs-similarity-via-dimsum>.

References X



ZHOU, Y., WILKINSON, D., SCHREIBER, R., AND PAN, R.

Large-scale parallel collaborative filtering for the netflix prize.

In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2008), vol. 5034 LNCS, pp. 337–348.



ZLWK, O. H., SDSHU, Q. W., LQWURGXFH, Z. H., HDVW, O., FODVVLILHG, E. H., WZR, L., PHPRU, J., DQG, E., AND EDVHG, P.

Alternating Least Squares with Incremental Learning Bias.
297–302.