Slovenská technická univerzita
Fakulta informatiky a informačných technológii
Ilkovičova 3, 812 19 Bratislava

Exponea odporučač
Zadanie 2.

Vyhľadávanie informácií

Peter Válka
13/12/2018

Cvičiaci: Ing. Jakub Mačina
Študijný odbor: Inteligentné Softvérové Systémy
Ročník: 2. Ing
Akademický rok: 2018/2019

1. Úvod

Cieľom zadania bolo vytvoriť odporúčač na základe informácií o zákazníkových akcií. Následne počas trénovania sú uvedený testovací zákazníci, pričom odporúčač vygeneruje zoznam 10 predmetov, ktoré odporúča zákazníkovi. Dané predmety následné môžu byť napr. znázornené v sekcií odporúčané na webovej stránke.

2. Odporúčač

Odporúčač sa skladá z 3 častí. Pri testovaní boli použité rôzne kombinácie odporúčania pričom sa bral dôraz na počet predmetov s ktorými zákazník už interagoval. Pred vytváraním potrebných štruktúr bol stĺpec "type", ktorý určuje typ zákazníkovej akcie pretransformovaný na číselné hodnoty, pričom akcia zakúpenie predmetu mala najväčšiu hodnotu. Ako hlavnú myšlienku odporúčania bolo zvolené kolaboratívne odporúčanie. Kvôli veľkosti matice nebolo možné vytvoriť štandardnú maticu, ale bola vytvorená riedka "sparse" verzia, ktorá sa už vedela vložiť do pamäte. Kosínová podobnosť bola zvolená ako metrika vypočítavania podobnosti 2 prvkov.

- Odporúčanie založené na zákazníkoch
- Odporúčanie založené na predmetoch
- Odporúčanie podľa populárnych predmetov

a. Odporúčanie založené na zákazníkoch

Odporúčanie vyhľadá 50 najbližších zákazníkov, ktorí sa najviac podobajú cielu. Následne sa spočítajú akcie na jednotlivých predmetoch a vyberie sa 10 predmetov, ktoré majú najväčší súčet. Odporúčanie som použil pokiaľ mal zákazník aspon 10 rôznych predmetov, s ktorými interagoval.

b. Odporúčanie založené na predmetoch

Vytvorí sa inverzná matica ku matici so zákaznikmi a pri odporúčaní sa nájde najbližších 10 predmetov ku spomínanému predmetu. Pokiaľ má zákazník viacero predmetov o ktorých vieme, vyberie sa predmet, ktorý má súčet akcií najväčší. Odporúčanie som použil pokiaľ mal zákazník interakciu iba do 10 predmetov pri hybridnom odporúčaní.

c. Populárne predmety

Odporúčanie sa aplikuje, pokiaľ nevieme nič o zákazníkovi, pričom sa vyberie 10 predmetov, ktoré majú najväčší súčet akcií nad nimi.

3. Testovanie

Dataset sa rozdelil na trénovaciu časť a testovaciu časť, podľa časovej známky záznamu, pričom trénovacia časť obsahovala 80% všetkých záznamov. Z trénovacej časti sa vytvorili riedke matice pre odporúčanie podľa zákazníka a podľa predmetu. Následne sa získali unikátni používatelia a vykonalo sa odporúčanie pre každého. Počet unikátnych používateľov bolo 23001. Úspešnosť sa počítala spôsobom, že pokiaľ sa v zozname odporúčaných predmetov nachádzal predmet s ktorým zákazník v trénovacej časti interagoval, tak sa vyhlásil úspech a pripočítal sa do úspešných pokusoch. Nakonci sa vydelil počet úspešných s počtom celkových.

Odporúčanie	**Úspešnosť**
Iba populárne predmety	7%
Kombinácia všetkých odporúčaní	10.93%
Odporúčanie založené na zákazníkoch	21.9%

4. Zhodnotenie

Z odporúčania dosiahli lepšie výsledky ako iba populárne predmety, čím dokázali uspieť. Použitie riedkych matíc je šetrné pre pamäť. Nečakal som že kombinácia odporúčaní dosiahne slabší výsledok ako čisté odporúčanie podľa zákazníkoch. Ako zlepšenie by som navrhoval vylepšenie kombinácie odporúčaní kde sa vylepšia hodnoty predmetov, kedy sa aplikuje konkrétne odporúčanie.

5. Príloha: Zdrojový kód

- Vytváranie tabuľky

```
file = trainDF
user_u = list(sorted(file.customer_id.unique()))
item_u = list(sorted(file.product_id.unique()))

col = file['customer_id'].tolist()
row = file['product_id'].tolist()
data = file['type'].tolist()
sparse_matrix_user = csc_matrix((data, (row, col)))
sparse_matrix_item = csc_matrix((data, (col, row)))

similarities_user = cosine_similarities(sparse_matrix_user)
similarities_item = cosine_similarities(sparse_matrix_item)
```

- Testovanie na kombinácií odporúčania

```
hits = 0
misses = 0
print("Total Users: {}".format(len(testUsers)))
for i in range(len(testUsers)):
    user = testUsers[i]
    userItems = getCustomerItemsList(user, trainDF)
    if len(userItems) > 10:
        recommendeditems = recommendItems(similarities_user, user,
trainDF, 50)
    elif len(userItems) > 0:
        targetItem = getUsersBestItem(trainDF, user)
        recommendeditems = recommendItemBased(similarities_item,
targetItem, 10)
    else:
        recommendeditems = getPopularItems(trainDF, 10)
    actualItems = getCustomerItemsList(user, testDF)
    if len(actualItems) == 0:
        correctPredictions = "No Data in Future"
    else:
        correctPredictions = calculatePrediction(recommendeditems,
actualItems)
    if correctPredictions > 0:
        hits += 1
    else:
        misses += 1

print("Item Based and User Based Recommender had {} hits and {}
```

```
misses and overall ratio {}".format(hits,misses,
(hits/len(testUsers))))
```

- Testovanie čisto na odporúčaní pre zákazníka

```
hits = 0
misses = 0

for i in range(len(testUsers)):
    user = testUsers[i]
    recommendeditems = recommendItems(similarities_user, user,
trainDF, 50)
    actualItems = getCustomerItemsList(user, testDF)
    if len(actualItems) == 0 and len(getCustomerItemsList(user,
trainDF)):
        correctPredictions = "No Data in Future and Past"
    else:
        correctPredictions = calculatePrediction(recommendeditems,
actualItems)
    if correctPredictions > 0:
        hits += 1
    else:
        misses += 1
print("User Based only Recommender had {} hits and {} misses and
overall ratio {}".format(hits,misses, (hits/len(testUsers))))
```