Recommender system seminar

Žiga Milan Holc
FAMNIT
Koper, Slovenia

ABSTRACT

Python Recommendation system

1 UVOD

Pri predmetu Sistemi za podporo o odločanju sem naredil seminar, kjer sem implementiral preprost sistem za podporo o odločanju. Uporabljal sem movielens podatke. Zaradi olajšanja inštalacije paketov, ki jih potrebujemo za zagon Recommender seminarja sem ustvaril preprosto skripto *install.sh*. Skripta vsebuje pakete: *pandas, sklearn, scikit-learn, scipy* in *tqdm*. Paket *tqdm* nam je v pomoč pri for zankah, saj nam omogoči vizualizacijo ali for zanka teče ali ne, predvsem pa nam omogoči pogled v to kako hitro deluje.

2 VNOS PODATKOV

Za vnos podatkov sem ustvaril svoj datoteko, ki se imenuje *Data.py*. Na začetku sem implementiral razred *UserItemData*, podal sem mu potrebne parametre in začetni datum in končni datum. Pri uvozu podatkov sem naletel na težavo z datumom. Želel sem uporabiti funkcijo *datetime* zaradi tega sem moral najprej preimenovati stolpce, da jih lahko funkcija prepozna. Nato sem dodal stolpec timestamp in funkciji, ki bosta služil za filtriranje podatkov po datumu. Dodal sem nato še eno filter funkcijo, ki filtrira filme po tem koliko imajo ocen. Nato sem naredil še en razred *MovieData*, ki nam uvozi vse podatke o filmih. In vrne glede na id filma njegov naslov.

3 BASIC RECOMMENDATIONS

Random predictor

Random predictor sem definiral tako, da sem najprej vzel stolpec id-jov od filmov, pretvoril sem ga v array s pomočjo *numpy* funkcije. Nato sem naredil slovar *movieDict*. Skozi slovar sem se zapeljal s for zanko in vsakemu filmu dodal naključno oceno med najmanjšo možno oceno in najvišjo možno ceno (1-5).

Recommender

Razred *Recommender* sprejme metodo *predictorja*. V *Recommender* razredu najprej pridobimo ocene za filme, nato sem ustvaril pogoj, da če ni nobene napovedi se vrnemo iz programa. Nato pogledamo ali je uporabnik gledal filme tako se z for zanko zapeljemo skozi podatke, da najdemo ujemajoče. Nato sortiramo glede na ocene filmov in vrnemo array dobrih filmov. V primeru, da smo vrnili dovolj filmov, končamo z iskanjem.

Average predictor

Average predictor sem definirali tako, da sem izračunal najprej vsoto vseh ocen za film. Nato sem naredil nov slovar v katerega bomo shranjevali podatke. Zapeljemo se skozi idje filmov in vsakemu filmu izračunamo in dodamo njegovo povprečno oceno, ter ga vstavimo v slovar. Nato vrnemo narejeno v slovarju, kjer je ključ *movieID* in *vrednost* povprečna vrednost ocene tega filma.

Recommending the most watched movies

Za predlaganje najbolj gledanih filmov sem ustvaril *ViewsPredictor*. Razred deluje tako, da najprej seštejemo vse ocene, ki jih ima film. Nato podatke filtriram glede na dolžino ocen in vrnemo filme, ki imajo največ ocen, saj za oceno filma je potrebno film pogledati.

4 COLLABORATIVE FILTERING

Predicting scores with similarity between products

Pri razredu *ItemBasedPredictor* sem se najprej lotili tako, da sem shranili podatke od ocen. Nato sem naredili pivot tabelo in sicer id-ji od uporabnikov so v vrsticah, id-ji od filmov pa v stolpcih, kot podatki so pa podane ocene za ocenjen film v primeru, da ocene ni je podatek *NaN*. Nato sem uporabili funkcijo *unique()* zato, da se nam ne bodo uporabniki ponavljali. Nato se z zanko zapeljemo skozi *userIDs*, da pridobimo ocene za vse filme vključno z *NaN*. Vse *NaN* vrednosti nato nadomestim z povprečno oceno in potem odštejemo od vseh vrednosti povprečno oceno. Potem sem ustvaril tabelo, kjer so stolpci in vrstice *MovieID* podatki. V tabeli so nato podobnosti med filmi glede na ocene od uporabnikov. Nato se skozi novo tabelo zapeljemo s for zanko skozi vrstice in nato še z eno vgnezedno skozi stolpce, da pogledamo id filma na stolpcu in na vrstici. V primeru da imamo dva različna filma ju primerjamo in shranimo na tisto mesto oceno podobnosti, če pa imata filma enak id pa pripišemo tam ničlo. Podobnost izračunamo tako, da vzamemo oceno enega in drugega filma, če imamo dovolj ocen za izračun. Podobnost sem izračunal po naslednji formuli:

similarity = 1 - spatial.distance.cosine(p1Ratings, p2Ratings)

Dodal sem tudi pogoj, da v primeru, da je ocena manjša kot naš prag podamo oceno 0.

Recommender for user 78. Po zgrajeni funkciji za izračun podobnosti med filmom predlagamo nato nekatere filme uporabniku 78. To storimo tako, da vzamemo id-je vseh filmov kar jih je uporabnik ocenil. Vzamemo tudi id-je vseh ocenjenih filmov in jo uredimo. Filme za uporabnika nato vrnemo v slovarju.

Most similar movies

Pri najbolj podobnih filmih sem uporabljali enako metodo kot pri prejšnjem razredu. Na enak način smo pridobili pivotno tabelo in na enak način sem izračunal podobnost med filmoma. Podatke vrnemo na takšen način, da se s for zanko zapeljemo skozi filme in v slovar vrnemo film, ki ima array. V arrayu smo pa shranili id najbolj podobnega filma in oceno podobnosti.

Recommendation based on the currently viewed content

Za predlaganje vsebine glede na gledano vsebino uporabnika sem dodal funkcijo *similarItems* v naš razred *ItemBasedPredictor*. Funkcija deluje tako, da najprej ustvari slovar za shranjevanje ocen uporabnika. Nato pogleda vse podobnosti z drugimi filmi. V slovar shrani filme, ki so podobni glede na naše filme in jih vrne. Uredi jih tudi od najbolj podobnega do najmanj. Ko dosežemo željeno število filmov se funkcija ustavi.

Recommendation for yourself

Filme, sem si predlagal na tak način, da sem v datoteko *user-ratedmovies.dat* vstavil svoje podatke. Poiskal sem par filmov, ki so mi všeč in jih ocenil. Nato sem si predlagal filme s pomočjo *ItemBasedPredictor* funkcijo.

Prediction with Slope One method

Pri *SlopeOnePredictor* razredu sem najprej začel tako, da sem vzel pivot tabelo, ki smo jo zgradili pri *ItemBasedPredictor*-ju. Slope one deluje tako, da vzame ocene vseh uporabnikov. Poiščemo nato vse uporabnike, ki so uporabniku za katerega želimo film predlagati najbolj podobni. To sem naredil tako, da sem napisal funkcijo *similarity*, ki podobnost med uporabniki izračuna s pomočjo kosinusne podobnosti. Slovar z najbolj podobnimi uporabniki sem nato še uredil po vrsti tako, da je nam najbolj podoben uporabnik na začetku slovarja. Zaradi tega, ker mora Slope one zapolniti prazna mesta in nam predlagati predvidene ocene naših neocenjenih filmov, moramo izvedeti katerih filmov uporabnik ni gledal. Ko smo izvedeli katere filme moramo pridobiti pokličemo funkcijo *slopeOne*, ki najprej vzame ocene od uporabnika za katerega želimo film predlagati in nato še ocene uporabnika, ki nam je najbolj podoben po ocenjevanju filmov. Funkcija najprej pogleda, da ocene niso *NaN*, nato izračunamo vsoto tako:

sum = sum + ur[i] - r[i].

Potem preverimo ali imamo v tabeli še kaj vrstic za pregledati, če je odgovor ja izračunamo vsoto tako, da jo delimo z vsemi stolpci, ki so še na voljo. V primeru, da velja pogoj, da nimamo v tabeli ocene za film jo pa izračunamo tako, da uporabnikovi oceni prištejemo vsoto. Ko smo izračunali oceno za film jo vstavimo v tabelo od našega uporabnika za katerega smo računali. Nato nadaljujemo tako, da spet pogledamo ali obstaja drug nam podoben uporabnik, ki je ocenil kakšen film, ki ga potrebujemo za izračun naše ocene. V primeru, da smo izračunali vse možne prazne ocene vrnemo podatke.

5 EVALUATION OF THE RECOMMENDER SYSTEM

Evalvacije našega sistema za podporo o odločanju sem se lotil tako, da sem najprej poiskal uporabnika, ki je ocenil največ filmov. Nato sem naredil tako, da sem definiral dva slovarja. Prvi slovar so *prediction-values* drugi pa ima shranjene *test-set-values*. To nam pomaga nato izračunati napake, zaradi tega, ker imamo en slovar, ki ima prave vrednosti in drugega v katerega predlagamo ocene za uporabnika. Tako lahko potem na teh slovarjih izračunamo MAE, RMSE, recall, accuracy in F1. MAE in RMSE sta lahka izračuna, saj načeloma izračunamo samo povprečno absolutno napako in odklon povprečnega kvadrata. Za izračun recall in accuracy pa moremo narediti tabelo, ki ima vrednosti: True positive (TP), False positive (FP), True negative (TN), False negative (FN). Glede nato, da se ukvarjamo s predlaganjem filmov uporabnikom, ki imajo lahko ocene od 1 do 5 sem naredil naslednjo tabelo:

Dejansko/Napovedano	1-3	3.5-5
1-3	TN	FN
3.5-5	FP	TP

S pomočjo te tabele sem nato lahko izračunal recall, accuracy in F1, saj mi je pomagala pri implementaciji izračuna. Po pregledu rezultatov različnih predictorjev sem ugotovil, da je Slope one predictor najboljši predictor, saj ima vedno prav in praktično nima napake. Najslabša dva predictorja pa sta Item based predictor in pa Random predictor.