PERSONALIZOVANI SISTEM ZA PREPORUKU FILMOVA

Nevena Nikolić, 1021/2018

Luka Kalinić, 1058/2018

UVOD

- Sistemi za preporuku su sistemi čiji je glavni zadatak da pruže korisniku informaciju - preporuku o potencijalno zanimljivom sadržaju i predstavljaju jednu od najuspešnijih primena mašinskog učenja
- Popularnost ovih sistema naročito je porasla poslednjih godina, povećanjem količine informacija koje nas okružuju, te digitalizacijom naše svakodnevnice i pojavom internet servisa, kao što su *Facebook, Google, Amazon, eBay, Netflix*...

RAZLIČITI PRISTUPI

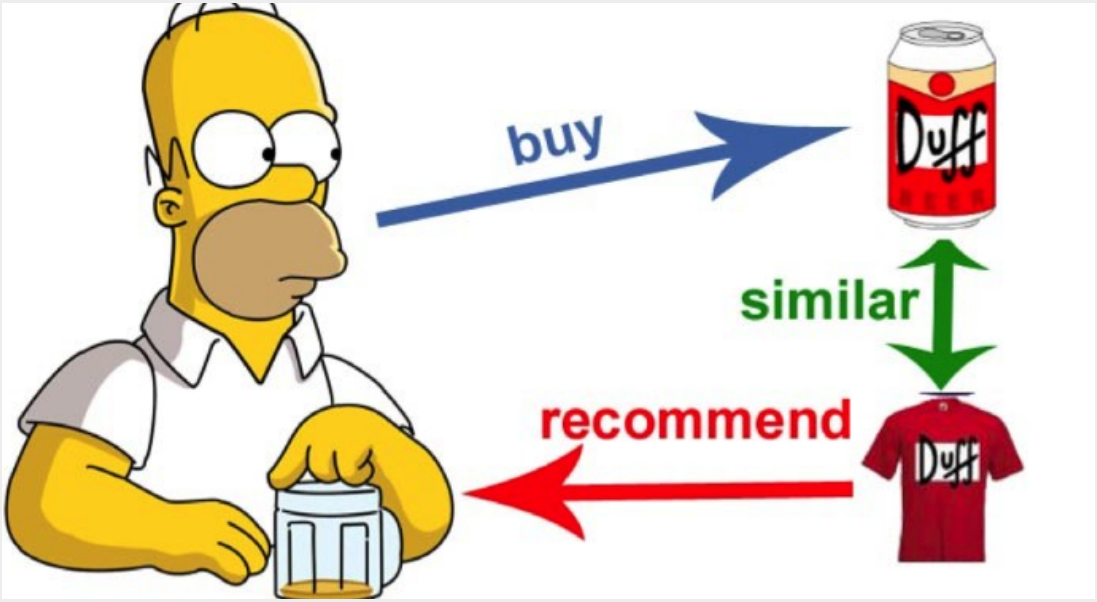
- Sistemi za preporuku zasnovani na sadržaju (eng. *content-based*)
- Sistemi za preporuku zasnovani na kolaborativnom filtriranju (eng. *collaborative filtering*)
- Hibridni sistemi (eng. hybrid)

RAZLIČITI PRISTUPI

- Sistemi za preporuku zasnovani na sadržaju (eng. *content-based*)
- Sistemi za preporuku zasnovani na kolaborativnom filtriranju (eng. *collaborative filtering*)
- Hibridni sistemi (eng. hybrid)

CONTENT-BASED

- Korisniku se preporučuju objekti slični objektima koji su mu bili zanimljivi u prošlosti, poput objekata koje je ranije pretraživao ili ocenio visokom ocenom



CONTENT-BASED

PREDNOSTI

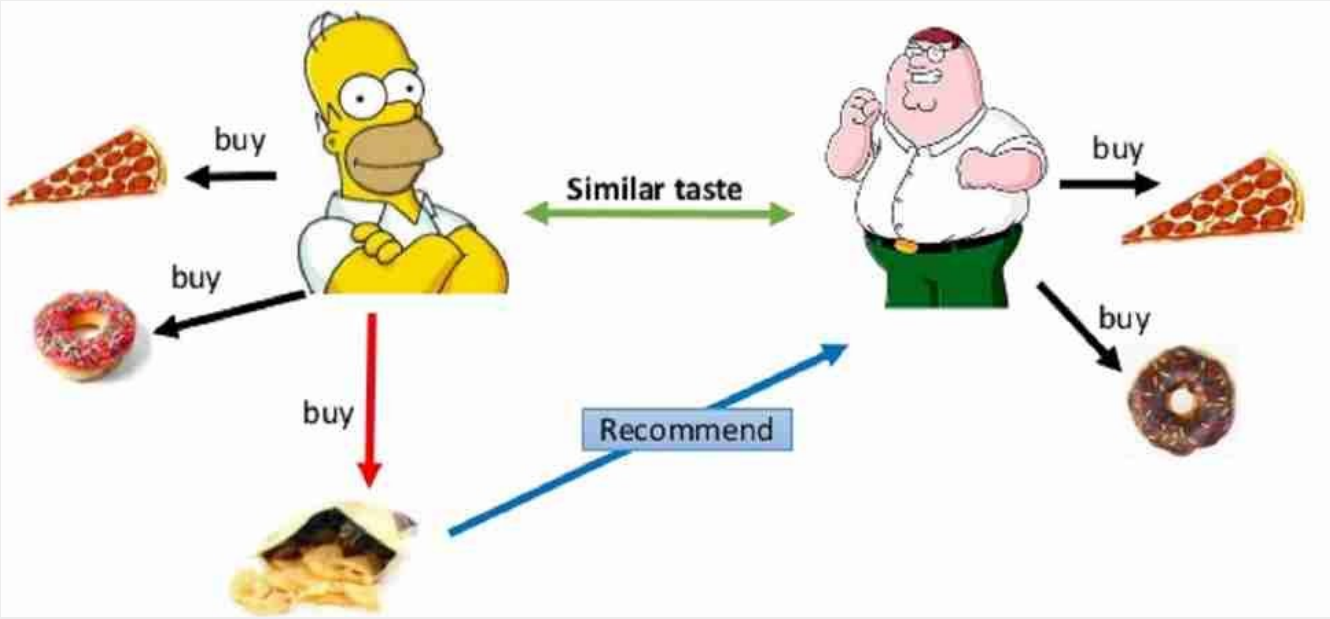
- nezavisnost od ostalih korisnika, nema formiranja zajednice
- mogućnost poređenja objekata
- mogućnost preporučivanja novih objekata koji prethodno nisu bili ocenjeni
- mogućnost preporučivanja korisnicima sa jedinstvenim ukusom
- transparentnost – moguće je jasno odrediti zašto je objekat preporučen

MANE

- generisanje atributa može biti komplikovano
- tendencija ka preteranoj specijalizaciji (eng. *overspecialization)* - preporučuju se jedino objekti koji su slični postojećim zanimljivim objektima
- problem novog korisnika - u slučaju novog korisnika, sistem ne može dati preciznu preporuku

COLLABORATIVE FILTERING

- Korisniku se preporučuju objekti koji su bili zanimljivi korisnicima sa sličnim interesima (sa sličnim ukusom)



COLLABORATIVE FILTERING

PREDNOSTI

- radi za proizvoljne objekte – generisanje atributa nije potrebno
- ignoriše kontektst
- jednostavnost implementacije
- bolje performanse u odnosu na content-based pristup (u smislu greške predviđanja i vremena izršavanja)

MANE

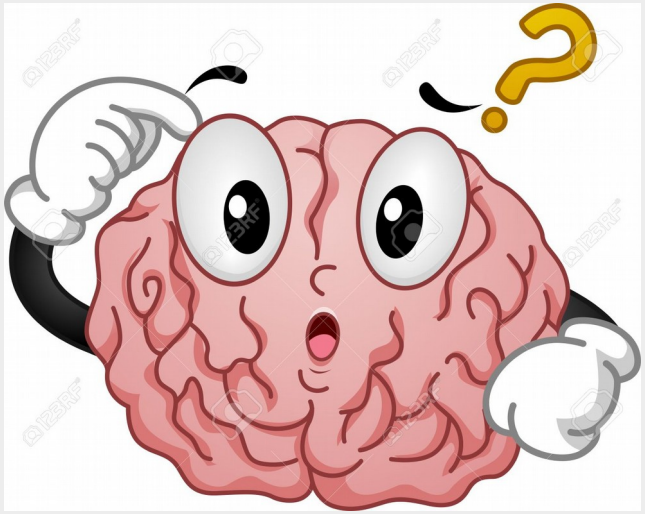
- problem novog objekta – ako nije dovoljno puta ocenjen, objekat neće biti preporučivan
- problem novog korisnika - potrebno je imati dovoljan broj korisnika u sistemu za nalaženje sličnog
- user/ratings matrica je retka, teško je pronaći korisnike koji su ocenili iste objekte
- tendencija ka preporučivanju popularnih objekta

PREPORUKA FILMOVA

- Naš cilj je izgradnja personalizovanog sistema za ocenjivanje i preporuku filmova
- Različiti ljudi imaju različit ukus za filmove; naš sistem pomaže korisnicima da odmah pronađu filmove po svom ukusu, bez obzira na to koliko njihovi ukusi mogu biti raznoliki

METODE

- Eksperimentišemo sa oba pristupa
- Content-based:
  - TF-IDF, doc2vec
- Collaborative filtering:
  - K-najbližih suseda, dekompozicija matrice



PODACI

- Koristimo **The Movies** skup podataka
  - preko 45,000 filmova, 26 miliona ocena od preko 270,000 korisnika
- Podaci su podeljeni u dva skupa

  (1) prvi skup sadrži listu filmova, njihove ukupne ocene i atribute kao što su budžet, prihod, uloge itd. (*movies_metadata.csv*)

  (2) drugi skup sadrži korisnik-film ocene - ID korisnika, ID filma i ocenu između 1 i 5 koju je taj korisnik dao tom filmu (*ratings.csv*)
    - zbog računarskih organičenja, koristimo nasumičan podskup od oko 100,000 podataka

ANALIZA PODATAKA

- Top 10 filmova prema težinskim ocenama (eng. *weighted score*), izračunatih koristeći *IMDB weighting*

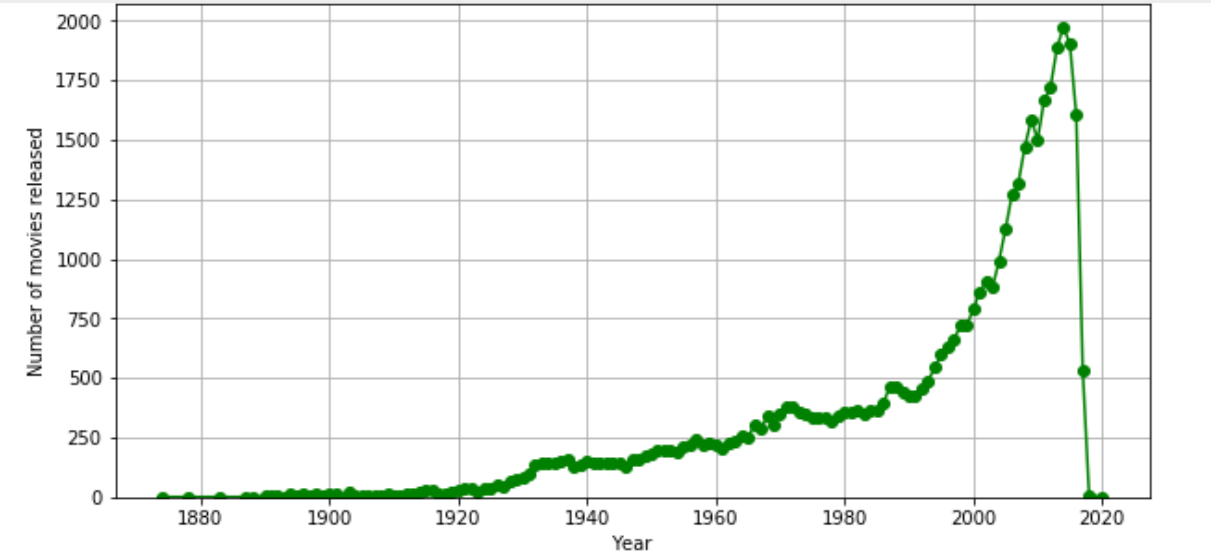
```
m = movies.vote_count.quantile(0.9)
C = movies.vote_average.mean()
qualified = movies[(movies['vote_count'] >= m) & (movies['vote_count'].notnull())
                   & (movies['vote_average'].notnull())]

v = qualified.vote_count
R = qualified.vote_average
weighted_score = (v/(v+m)*R) + (m/(v+m)*C)
weighted_score = round(weighted_score, 6)

movie_title = qualified.title

top_rated = np.vstack([movie_title,R,v,weighted_score]).T
top_rated = pd.DataFrame(top_rated,columns = ['Movie Title','Avg Votes','Num Votes','Weighted Score'])
top_rated = top_rated.sort_values('Weighted Score',ascending=False)
top_rated.head(10)
```

Movie Title	Avg Votes	Num Votes	Weighted Score
The Shawshank Redemption	8.5	8358	8.44587
The Godfather	8.5	6024	8.42544
Dilwale Dulhania Le Jayenge	9.1	661	8.42145
The Dark Knight	8.3	12269	8.26548
Fight Club	8.3	9678	8.25638
Pulp Fiction	8.3	8670	8.25141
Schindler's List	8.3	4436	8.20664
Whiplash	8.3	4376	8.2054
Spirited Away	8.3	3968	8.19605
Life Is Beautiful	8.3	3643	8.18717

ANALIZA PODATAKA

- Broj filmova po godini

```
movies['year'] = movies.release_date.str.extract("(\d{4})", expand = True)
movies.year = pd.to_datetime(movies.year, format='%Y')
movies.year = movies.year.dt.year
dftmp = movies[['id', 'year']].groupby('year')
fig, ax1 = plt.subplots(figsize=(10,5))
ax1.plot(dftmp.year.first(), dftmp.id.nunique(), "g-o")
ax1.grid(None)
ax1.set_ylim(0,)
ax1.set_xlabel('Year')
ax1.set_ylabel('Number of movies released')
```



ANALIZA PODATAKA

- Broj filmova po žanru

```
def counting_values(df, column):
    value_count = {}
    for row in df[column].dropna():
        if len(row) > 0:
            for key in row:
                if key in value_count:
                    value_count[key] += 1
                else:
                    value_count[key] = 1
        else:
            pass
    return value_count

genres_count = pd.Series(counting_values(movies, 'genres'))
colors = ['magenta','blue','green','yellow','orange','pink','aqua','gray','purple','red']
genres_count.sort_values(ascending = False).head(10).plot(kind = 'bar',color = colors)
plt.show()
```