Hibridni sustav za preporučivanje glazbe zasnovan na suradničkom filtriranju i sadržajnim značajkama

Vuksanović, Ana

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1312

HIBRIDNI SUSTAV ZA PREPORUČIVANJE GLAZBE ZASNOVAN NA SURADNIČKOM FILTRIRANJU I SADRŽAJNIM ZNAČAJKAMA

Ana Vuksanović

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 1312

HIBRIDNI SUSTAV ZA PREPORUČIVANJE GLAZBE ZASNOVAN NA SURADNIČKOM FILTRIRANJU I SADRŽAJNIM ZNAČAJKAMA

Ana Vuksanović

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1312

Pristupnica:	**Ana Vuksanović (0036542208)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Marin Šilić

Zadatak: **Hibridni sustav za preporučivanje glazbe zasnovan na suradničkom filtriranju i sadržajnim značajkama**

Opis zadatka:

Proučiti i opisati sustave za preporučivanje zasnovane na suradničkom filtriranju te one zasnovane na sadržajnim značajkama. Prikupiti prikladne i javno dostupne skupove podataka za učenje i vrednovanje sustava za preporučivanje glazbe. Dodatno, proučiti metode izvlačenja sadržajnih značajki iz glazbe. Oblikovati i programski ostvariti sustav za preporučivanje glazbe zasnovan na suradničkom filtriranju. Nadalje, oblikovati i programski ostvariti sustav za preporučivanje glazbe zasnovan sadržajnim značajkama. Konačno, primjenom ostvarenih sustava za preporučivanje ostvariti hibridni sustav za preporučivanje glazbe koji će koristiti značajke suradničkog filtriranja i sadržajne značajke. Ispitati uspješnost ostvarenog sustava primjenom prikladno odabranih mjera te prikazati i opisati rezultate ispitivanja. Uz rad je potrebno predati i dokumentirati izvorni kod ostvarenog sustava, korištene skupove podataka te navesti korištenu literaturu.

Rok za predaju rada: 14. lipnja 2024.

Zahvaljujem se obitelji i prijateljima koji su mi pomogli po putu. Ovo je samo početak.

Sadržaj

1. Uvod

Ulazak u informacijsko doba, zajedno s razvojem i širenjem informacijskih tehnologija, omogućio je ljudima pristup prividno neograničenoj količini sadržaja. Ovaj sadržaj pruža priliku za edukaciju, istraživanja, web trgovinu, povezivanje s drugim ljudima te konzumaciju sadržaja zabavne prirode, uključujući i glazbu. Usprkos obilju koje sadrže digitalne glazbene platforme, bez prikladne metode upravljanja podacima, izvjesno je da se ne moći postići kvalitetno korisničko iskustvo. Zbog ogromne količine dostupne glazbe, nije dovoljno podržati isključivo mogućnost jednostavne pretrage, već je potrebno osmisliti način kako suziti prostor pretraživanja i usmjeriti korisnika prema glazbi koja odgovara njegovom ukusu. Tom se svrhom uvode sustavi za preporučivanje.

Cilj ovog rada je napraviti osnovnu taksonomiju sustava za preporučivanje, navesti pojedine prednosti i mane te opisati najčešće korištene algoritme pojedinih modela. Naglasak će biti stavljen na modele zasnovane na suradničkom filtriranju i one zasnovane na sadržajnim značajkama. Dodatno će biti proučeno ostvarenje hibridnog sustava koji koristi značajke suradničkog filtriranja i sadržajne značajke. Implementacija i evaluacija navedenih modela ostvarene su u programskom jeziku Python.

2. Sustavi za preporučivanje

Sustavi za preporučivanje (eng. recommendation systems) su sustavi bazirani na skupu metoda za analizu korisničkih podataka, kojima predviđaju koji će sadržaj biti zanimljiv korisniku.

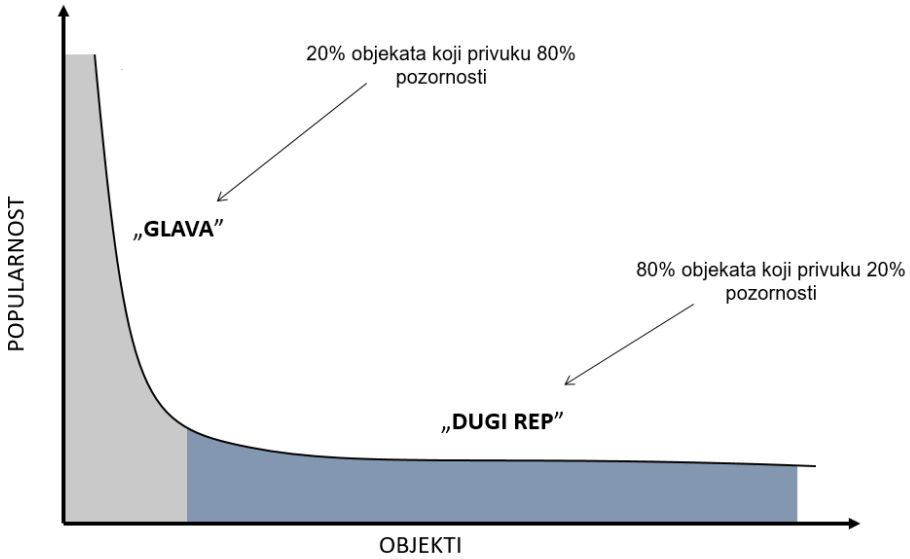
2.1. Problem preporučivanja

Problem preporučivanja postavljen je sredinom devedesetih godina prošlog stoljeća, a njegovo se područje primjene još uvijek uvelike proširuje. Sama tema dobiva na popularnosti 2006. godine kada američki filmski i televizijski distributer Netflix organizira otvoreno natjecanje "Netflix Prize" s ciljem optimizacije Netflixovog algoritma predikcije.

Povećana važnost problema preporučivanja uzrokovana je prijelazom iz razdoblja oskudice u razdoblje obilja (eng. *scarcity to abundance*). Ako uzmemo primjer glazbe, u prošlosti su ljudi odlazili u glazbene trgovine. Čak i u većim trgovinama, broj polica je ograničen što znači da mogu prodavati samo određen broj albuma. Slična situacija bila je i s radio postajama. Bilo je moguće puštati samo određen broj pjesama jer dan ima ograničen broj sati. Razvojem interneta, ova ograničenja nestaju. Samim time, ljudi dobivaju više izbora te su u mogućnosti pronaći više glazbe koja odgovara njihovom ukusu. Iz toga se rađa koncept „dugog repa“.

Koncept „dugog repa“ (eng. Long Tail) ukazuje na činjenicu da zapravo mali broj objekata privlači većinu pozornosti. Prikaz navedenog nalazi se na slici 2.1. Ako opisano primijenimo na naš primjer glazbe, onda „glava“ distribucije predstavlja one albume koji su se nekada pronalazili u trgovinama ili one pjesme koje su se puštale na radiju, a danas su to pjesme koje se nađu na top listama i imaju milijune slušanja. „Dugi rep“ predstav-

lja one pjesme, koje prije nisu bile lako dostupne, koje imaju manji broj slušanja, ali ih je toliko mnogo da je njihov ukupan broj slušanja značajan. Važno je da preporuke korisniku proizlaze iz objekata koji se nalaze u tom dijelu jer su najpopularniji objekti korisniku već najčešće poznati te nisu posebice značajni za svakog pojedinog korisnika. Sposobnost sustava za preporučivanje da riješi ovaj problem će nam biti mjerilo kvalitete sustava.



Slika 2.1. Koncept „dugog repa"

2.2. Formalna definicija problema

Neka je U konačan skup kojeg nazivamo skup korisnika (eng. users), neka je I konačan skup kojeg nazivamo skup objekata (eng. items) i neka je T potpuno uređen skup. Definiramo funkciju korisnosti (eng. utility function) $r : U \times I \to T$ koja pridružuje ocjenu nekog objekta pojedinom korisniku. Intuitivno, manja ocjena označava da se objekt manje svidio korisniku, veća ocjena označava da se objekt više svidio korisniku. Vrijednosti te funkcije zapisuju se u matricu korisnosti R. Primjer takve matrice nalazi se na slici 2.2. Problem se tada formalno definira kao pronalaženje $i_u^{'}$ za kojeg vrijedi:

$$(\forall u \in U)\ i_u^{'} := \underset{i \in I}{\operatorname{argmax}}\, r(u, i).$$

Objekti / Korisnici	Objekt 1	Objekt 2	Objekt 3	Objekt 4	...	Objekt N
Matea	2		5	4	...	
Lucija		5			...	1
Marta	4		3		...	4
Ana	5	5		2	...	

Slika 2.2. Matrica korisnosti *R*

Glavni izazovi koji se javljaju pri rješavanju problema su:

- prikupljanje "poznatih" ocjena u matrici
- korištenje poznatih ocjena za predviđanje nepoznatih ocjena
- evaluacija metoda predikcije

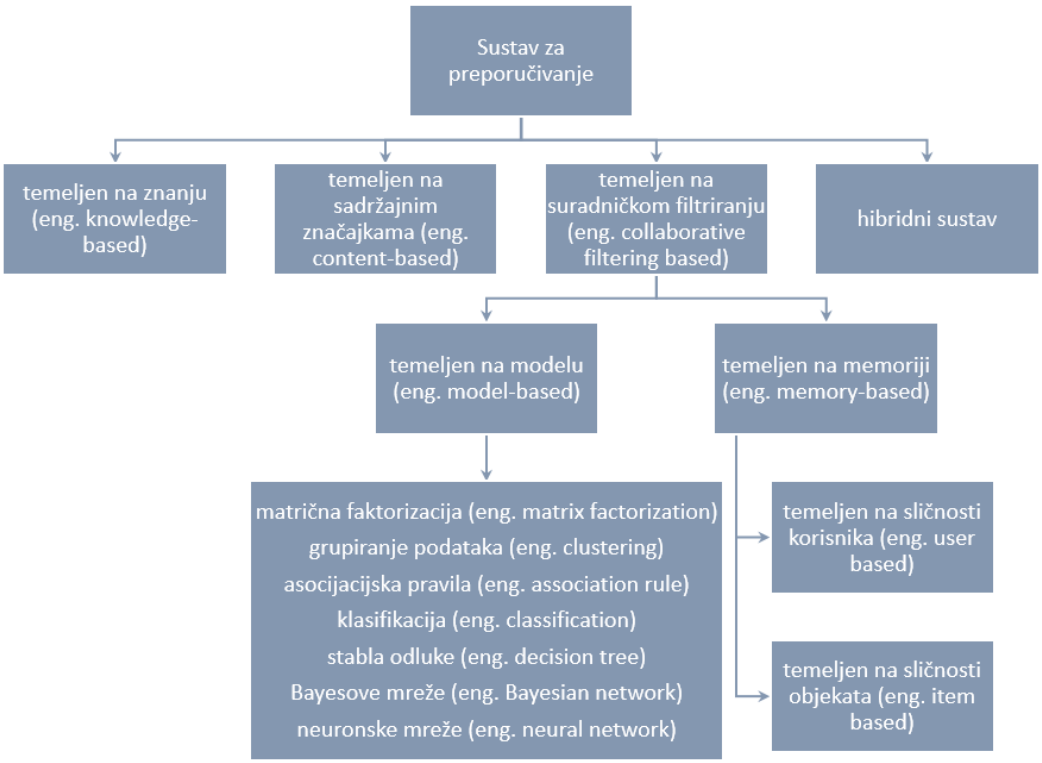
Izazovu prikupljanja početnih podataka pristupa se na jedan od dva načina: eksplicitnom ili implicitnom metodom. Eksplicitno prikupljanje podataka zahtjeva da korisnik ocijeni objekt. Prednost metode su njezina jednostavnost i izravnost, ali ono što ju čini neučinkovitom u praksi činjenica je da samo mali broj korisnika ocjenjuje objekte. Implicitno prikupljanje podataka bazira ocjene korisnika na temelju njegovih radnji. Na primjeru glazbe, može se ocijeniti da bi korisnik dao visoku ocjenu pjesmi koju često sluša od početka do kraja. Mana ove metode je ta da je vrlo teško implicitno detektirati niže ocjene. U praksi se koristi kombinacijom ovih dviju metoda pri čemu se eksplicitne ocjene nadopune implicitnima.

Izazov korištenja poznatih ocjena za predviđanje nepoznatih ocjena otežan je zbog dva razloga. Prvi je razlog vrlo rijetka popunjenost matrice korisnosti. Većina korisnika nije došla u kontakt, a samim time ni ocijenila, većinu proizvoda. Drugi je razlog premalo ocjena korisnika na početku rada (eng. cold start). Ako uvedemo novi objekt, on neće biti ocijenjen od ijednog korisnika, a ako uvedemo novog korisnika, on neće imati nikakvu prošlost.

Još neki od problema koji se mogu pojaviti su korisnik ili objekt koji nema sličnosti s niti jednim drugim korisnikom ili objektom (eng. black sheep), problem raznolikosti ili pretreniranja (eng. diversity ili overfitting) te pitanje privatnosti.

2.3. Podjela sustava za preporučivanje

Na slici 2.3. nalazi se detaljna podjela sustava za preporučivanje. U 3. poglavlju pobliže su obrađeni sustavi temeljeni na suradničkom filtriranju, u 4. poglavlju opisani su sustavi temeljeni na sadržajnim značajkama, a hibridni će sustav biti opisan pri pregledu implementacije programskog rješenja.



Slika 2.3. Podjela sustava za preporučivanje

3. Sustavi temeljeni na suradničkom filtriranju

Sustavi za preporučivanje temeljeni na suradničkom filtriranju dijele se na sustave temeljene na memoriji i one temeljene na modelu. Sustavi temeljeni na memoriji baziraju se na obradi podataka, dok oni temeljeni na modelu prvo treniraju model koji zatim koriste za davanje preporuka.

3.1. Sustavi temeljeni na memoriji

3.1.1. Sustavi temeljeni na sličnosti korisnika

Sustavi za preporučivanje temeljeni na sličnosti korisnika predviđaju koju bi ocjenu korisnik dao pojedinom objektu uzimajući u obzir k-najsličnijih korisnika koje nazivamo k-najbliži susjedi (eng. k-nearest neighbours - kNN).

Za računanje sličnosti koristit će se matrica korisnosti slična onoj prikazanoj na slici 2.2. Ključ pri odabiru funkcije sličnosti biti će način na koji će se tretirati nedostajuće vrijednosti u matrici, odnosno objekti koje korisnik nije ocijenio. Neke od mogućih funkcija sličnosti su:

- L1 sličnost (eng. Manhattan similarity)
- Euklidska sličnost (eng. Euclidean similarity)
- Jaccardova sličnost (eng. Jaccard similarity)
- kosinusova sličnost (eng. cosine similarity)
- Pearsonov koeficijent korelacije (eng. Pearson correlation)

Većina navedenih sličnosti izvedena je iz istoimenih udaljenosti. Tako, L1 sličnost koristi L1 normu, poznatu kao i Manhattan udaljenost. Neka su r_u i r_v vektori koji predstavljaju korisnike u i v te neka oni sadrže samo ocjene objekata koje su ocijenili oba korisnika. Nad njima definiramo Manhattan udaljenost kao sumu apsolutnih razlika između odgovarajućih dijelova dvaju vektora:

$$d_{Manhattan}(r_u, r_v) = \sum_i |R(u,i) - R(v,i)|$$

L1 sličnost se računa kao recipročna vrijednost dobivene udaljenosti, odnosno što je udaljenost manja, to znači da je sličnost veća. Sličnost se preslikava u interval [0, 1].

$$sim_{Manhattan}(r_u, r_v) = \frac{1}{1 + d_{Manhattan}(r_u, r_v)}$$

Euklidska sličnost izvedena je iz euklidske udaljenosti, još poznate kao i L2 norma. Neka su r_u i r_v vektori koji predstavljaju korisnike u i v te neka oni sadrže samo ocjene objekata koje su ocijenili oba korisnika. Nad njima definiramo euklidsku udaljenost kao sumu najkraćih udaljenosti između odgovarajućih dijelova dvaju vektora:

$$d_{euklid}(r_u, r_v) = \sqrt{\sum_i |R(u,i) - R(v,i)|}$$

Slično kao i kod L1 sličnosti, euklidska sličnost se računa kao recipročna vrijednost euklidske udaljenosti te se preslikava u interval [0, 1].

$$sim_{euklid}(r_u, r_v) = \frac{1}{1 + d_{euklid}(r_u, r_v)}$$

Jaccardova sličnost računa se kao omjer broja objekata koje su ocijenili oba korisnika i broja objekata koje je ocijenio barem jedan od korisnika. Neka su r_u i r_v vektori koji predstavljaju korisnike u i v te neka oni sadrže ocjene objekata korisnika u i v. Tada definiramo Jaccardovu sličnost kao omjer presjeka vektora r_u i r_v i unije vektora r_u i r_v.

$$sim_{jaccard}(r_u, r_v) = \frac{|r_u \cap r_v|}{|r_u \cup r_v|}$$

Jaccardova sličnost preslikava se na interval [0, 1]. Nedostatak Jaccardove sličnosti je što ne uzima u obzir vrijednost ocjene dane pojedinom objektu.

Kosinusova sličnost, za razliku od dosad opisanih sličnosti, računa se nad vektorima koji sadržavaju sve ocjene objekata. Dakle, neka su r_u i r_v vektori koji predstavljaju korisnike u i v te neka oni sadrže sve ocjene. Na mjesto objekta kojeg korisnik nije ocijenio upisuje se nula. Definiramo kosinusovu sličnost kao kosinus kuta između ta dva vektora.

$$sim_{kosinus}(r_u, r_v) = \frac{r_u \bullet r_v}{||r_u||\ ||r_v||} = \frac{\sum_i R(u,i)\ R(v,i)}{\sqrt{\sum_i R^2(u,i)}\ \sqrt{\sum_i R^2(v,i)}}$$

Kosinusova sličnost preslikava se na interval [0, 1]. Mana ove sličnosti je da nedostajuće vrijednosti tretira kao negativne. Ovo se može popraviti tako da se ocjene korisnika normaliziraju koristeći njegovu prosječnu ocjenu.

$$sim_{centered-kosinus}(r_u, r_v) = \frac{(r_u - \overline{r_u}) \bullet (r_v - \overline{r_v})}{||r_u - \overline{r_u}||\ ||r_v - \overline{r_u}||} = \frac{\sum_i (R(u,i) - \overline{r_u})\ (R(v,i) - \overline{r_v})}{\sqrt{\sum_i (R(u,i) - \overline{r_u})^2}\ \sqrt{\sum_i (R(v,i) - \overline{r_v})^2}}$$

Centrirana kosinusova sličnost preslikava se u interval [-1, 1], ali može se prilagoditi te se preslikati u interval [0,1].

$$sim_{centered-kosinus}(r_u, r_v) = \frac{1}{2}(sim_{centered-kosinus}(r_u, r_v) + 1)$$

Ovaj pristup podupire intuitivnu ideju o sličnosti korisnika jer nedostajuće vrijednosti

tretira kao prosječnu ocjenu te se uspješnije prilagođava "strogim" i "darežljivim" ocjenjivačima.

Pearsonov koeficijent korelacije jako je sličan centriranom kosinusu, ali ključna je razlika činjenica da Pearsonov koeficijent u obzir ponovno uzima samo objekte koje su ocijenili oba korisnika. Znači, neka su u i v neki korisnici te neka je S skup koji sadrži objekte koje su ocijenili oba korisnika. Tada definiramo Pearsonov koeficijent korelacije formulom:

$$sim_{pearson}(u,v) = \frac{n \sum_{s \in S} R(u,s)\, R(v,s) - \sum_{s \in S} R(u,s) \sum_{s \in S} R(v,s)}{\sqrt{n \sum_{s \in S} R^2(u,s) - (\sum_{s \in S} R(u,s))^2}\, \sqrt{n \sum_{s \in S} R^2(v,s) - (\sum_{s \in S} R(v,s))^2}}$$

Pearsonov koeficijent korelacije preslikava se na interval [-1, 1]. Vrijednost 0 predstavlja da ne postoji linearna korelacija između dvije varijable, 1 predstavlja potpunu pozitivnu linearnu korelaciju, a -1 predstavlja potpunu negativnu linearnu korelaciju.

Općeniti algoritam za predviđanje ocjene korisnika u za objekt i sljedeće korake. Neka je N skup k-najbližih susjeda korisnika u koji su ujedno i ocijenili objekt i. Za određivanje susjeda može se koristiti jedna od gore navedenih funkcija sličnosti. Tada možemo procjenu napraviti kao prosječnu ocjenu koju su najbliži susjedi dali objektu i.

$$R(u,i) = \frac{\sum_{v \in N} R(v,i)}{k}$$

Ovaj pristup ignorira koliko su k-najbliži susjedi zapravo slični korisniku u što nije povoljno jer je moguće da je značajna razlika u sličnostima između prvog najbližeg susjeda i k-tog najbližeg susjeda. Zato se pristup prilagodi koristeći težinsku aritmetičku sredinu ocjena najbližih susjeda gdje je težina jednaka sličnosti između korisnika u i susjeda v.

$$R(u,i) = \frac{\sum_{v \in N} sim(v,i)\, R(v,i)}{k}$$

Neke manjkavosti sustava temeljenih na sličnosti objekata pronalaze se u svojstvima matrice korisnosti R i u otežanoj analizi ljudskog ponašanja. Kao što je prije navedeno, matrica R vrlo je rijetko popunjena i najčešće je velikih dimenzija. Što se tiče samih korisnika, oni imaju sklonost ocjenjivati samo objekte koji im se sviđaju i često ne daju gradaciju koliko im se nešto svidjelo, nego daju isključivo najvišu ocjenu. Također, ukus korisnika se mijenja, dok neke ocjene objekata ovise o kontekstu u kojem su dane.

3.1.2. Sustavi temeljeni na sličnosti objekata

Sustavi temeljeni na sličnosti objekata su vremenski učinkovitiji u odnosu na one temeljene na sličnosti korisnika. Razlog tome je što su sličnosti između objekata manje podložne promjenama te ih je zato moguće unaprijed izračunati. Stvara se matrica koja svakom objektu pridružuje sličnost sa svakim drugim objektom. Primjer takve matrice nalazi se na slici 3.1. Izrada ove matrice traje dugo zbog velikog broja objekata i njihovih kombinacija, ali se obavlja rijetko te je taj posao moguće paralelizirati. Za objekte pretpostavljamo da su slični ako ih je mnogo korisnika ocijenilo istom ocjenom.

Objekti / Objekti	Objekt 1	Objekt 2	Objekt 3	Objekt 4	...	Objekt N
Objekt 1	1	0.32	0.02	0.78	...	0.54
Objekt 2	0.32	1	0.89	0.17	...	0.43
Objekt 3	0.02	0.89	1	0.65	...	0.39
Objekt 4	0.78	0.17	0.65	1	...	0.56
...	...	...	...	...	...	...
Objekt N	0.54	0.43	0.39	0.56	...	1

Slika 3.1. Matrica objekt-objekt S

Neka je U skup korisnika, neka je I skup objekata te neka je T potpuno uređen skup. Neka je $r : U \times I \to T$ funkcija koja pridružuje ocjenu nekog objekta pojedinom korisniku. Vrijednost funkcije r za objekte koje korisnik nije ocijenio je 0. Vrijednosti te funkcije zapisuju se u matricu korisnosti R. Neka je N skup k-najsličnijih objekata objektu i kojeg je ocijenio korisnik u. Sličnost objekata iščitava se iz matrice S čije su vrijednosti određene funkcijom sličnosti $s : I \times I \to \mathbb{R}$. Tada predviđamo ocjenu objekta i koristeći težinsku aritmetičku sredinu.

$$R(u,i) = \frac{\sum_{j \in N} S(i,j) \bullet R(u,j)}{\sum_{j \in N} S(i,j)}$$

Funkcija sličnosti bira se analogno kao u sustavima temeljenim na sličnosti korisnika, a u praksi se najbolja pokazala centrirana kosinusova sličnost.

Osim što su vremenski efikasniji, sustavi temeljeni na sadržajnim značajkama u većini slučajeva daju kvalitetnije preporuke od onih temeljenih na sličnosti korisnika. Ta pojava je zapravo lagana za razumjeti jer su objekti inherentno "jednostavniji" nego ljudi. Na primjer, u glazbi, pjesme se mogu podijeliti u žanrove, dok se glazbeni ukus ljudi vrlo teško može kategorizirati. Samim time, određivanje sličnosti je preciznije kod objekata te sličnost objekata nosi veću težinu nego sličnost korisnika.

3.2. Sustavi temeljeni na modelu

Dosadašnji opisani modeli nudili su preporuke uzimajući u obzir sve podatke u bazi, dok će sustavi temeljeni na modelu istrenirati model koristeći se algoritmima rudarenja podataka (eng. data mining) i strojnog učenja (eng. machine learning). Ovi sustavi nude brže izvršavanje procesa davanja preporuka i bolju skalabilnost. Neki njihovi nedostaci su potreba za velikom količinom podataka za učinkovito treniranje modela, kompleksnost izrade te potreba za redovitim ažuriranjem modela.

U ovom radu ovi sustavi nisu detaljno opisani, ali je bitno naglasiti da se i te metode, dakako, mogu koristiti za izradu sustava za preporučivanje glazbe. Na primjer, SVD dekompozicija, metoda matrične faktorizacije, može se koristiti za otkrivanje latentnih faktora i dekompoziciju matrice korisnosti. Također, grupiranje podataka moglo bi se koristiti za grupiranje najsličnijih korisnika i onda preporuke temeljiti na najpopularnijim pjesmama unutar tog klastera.

4. Sustavi temeljeni na sadržajnim značajkama

Cilj je algoritma sustava za preporučivanje temeljenih na sadržajnim značajkama (eng. content-based recommender systems) preporučiti korisniku nekoliko objekata sličnih onima koje je taj korisnik već visoko ocijenio. Svaki objekt opisan je pomoću atributa i ključnih riječi. Kod glazbe, pjesme su određene svojim metapodacima poput izvođača, producenta, pisca, datuma izlaska, žanra i sl.

Navedene značajke objekta služe za izgradnju njegovog profila. Profil se stvara za svaki objekt, a predočava se u obliku vektora. Svaka ćelija označava koliko se određena značajka odnosi na objekt. Ta vrijednost može biti numerička ako postoji neki način da se pojedina značajka kvantificira, a može biti i binarna, ako je općenito bitno posjeduje li objekt neku značajku ili ne. Naravno svaka značajka nema istu važnost i ne daje nam jednako preciznu sliku o osobnosti objekta. Zato se primjenjuje standard TF-IDF (eng. term frequency-inverse document frequency). Standardom se primarno koristi u polju tekstualne analize, ali se može modificirati i za potrebe preporučivanja.

Neka je j objekt, a neka je i značajka tog objekta. Frekvencija $freq(i, j)$ označava broj pojavljivanja značajke i u objektu j. Mjera TF (eng. term frequency) je omjer frekvencije $freq(i, j)$ i maksimalne frekvencije pojavljivanja ostalih značajki u objektu, $maxOther(i, j)$.

$$TF(i,j) = \frac{freq(i,j)}{maxOther(i,j)}$$

Mjera TF osigurava da broj ukupnih značajki objekta nema utjecaj na relevantnost frekvencije pojedinačne značajke.

Neka je n ukupni broj objekata, a neka je n_i broj objekata u kojima se pojavljuje značajka i. Mjera IDF, inverzna frekvencija dokumenta (eng. inverse term frequency), pridaje važnost značajki tako da je važnost značajke manja što se pojavljuje u više dokumenata. IDF je logaritam omjera n i n_i.

$$IDF(i) = \log \frac{n}{n_i}$$

Kombinirana mjera TF-IDF je umnožak mjera TF i IDF.

$$TF - IDF(i,j) = TF(i,j) \bullet IDF(i)$$

Nakon izgradnje profila objekata moguće je izgraditi profil korisnika. Svaki korisnik je određen profilima objekata koje je ocijenio. Najjednostavniji način konstrukcije profila korisnika je računanje srednje vrijednosti. Međutim, to ne uzima u obzir da su se neki objekti korisniku više svidjeli nego drugi pa je prikladnije računati težinsku srednju vrijednost. Također je potrebno prepoznati da svaka ocjena nema istu vjerojatnost pojavljivanja kod svih korisnika. Neki korisnici su "stroži" od drugih tako da se pojedini objekt mogao jednako svidjeti dvjema korisnicima, ali ne dobiti istu ocjenu. Za iskorijeniti ovaj problem, potrebno je normalizirati težinske srednje vrijednosti koristeći prosječnu danu ocjenu korisnika.

Izgradnjom profila objekata i korisnika, moguće je napraviti predviđanje hoće li se korisniku u svidjeti objekt i. Potrebno je izračunati sličnost između profila korisnika u,r_u, i profila objekta i, r_i,. U tu svrhu koristi se kosinusova sličnost.

$$sim_{kosinus}(r_u, r_i) = \frac{r_u \bullet r_i}{||r_u||\ ||r_i||}$$

Prednost korištenja sustava za preporučivanje temeljenih na sadržajnim značajkama neovisnost je o podacima drugih korisnika što znači da je moguće dati relevantne preporuke čak i ako se mali broj korisnika koristi sustavom. Također, sustavi temeljeni na sadržajnim značajkama bolje su prilagođeni za davanje preporuka korisnicima s vrlo

specifičnim ukusom, odnosno rješavaju problem „crne ovce“. Ova metoda najbolje savladava preporučivanje novih ili manje popularnih objekata. Profile novih objekata moguće je izgraditi odmah pri njihovom uvođenju u bazu, što znači da se od početka mogu preporučivati, dok se nepopularni objekti kvalitetno preporučuju jer popularnost ne igra nikakvu ulogu u ovakvom sustavu. Naposljetku, moguće je dati objašnjenje zašto je sustav ponudio određenu preporuku navođenjem točnih sadržajnih značajki koje su imale ključan utjecaj.

Pronalazak prikladnih sadržajnih značajki na kojima će preporučivanje biti temeljeno mukotrpan je proces i znatno otežava izgradnju sustava te igra ulogu u manjku popularnosti ovog pristupa. Drugi je nedostatak prekomjerna specijalizacija. Budući da se cijeli algoritam bazira na preporučivanju objekata sličnim onima koje je korisnik već pozitivno ocijenio, ne postoji način da se korisnika upozna s novom grupom objekata koji su različiti, ali bi mu se također mogli svidjeti. Ovaj efekt naziva se efekt eho komore (eng. echo chamber). Posljednja se mana očituje u izgradnji profila novog korisnika. U praksi, ova mana se zaobilazi tako da se za novog korisnika koristi prosječni profil svih korisnika koji se s vremenom i sve većim korištenjem personalizira.

5. Implementacija sustava za preporučivanje glazbe

Inspiracija za implementaciju sustava za preporučivanje glazbe došla je kao odgovor na Spotifyev *Million Playlist Dataset Challenge* (skraćeno: *MPD Challenge*). Spotify je najpopularnija digitalna platforma za reprodukciju glazbe. Osnovana je u Švedskoj, a danas broji preko 615 milijuna korisnika. U 2018., Spotify je organizirao *RecSys Challenge 2018*, istraživački izazov u području znanosti o podacima (eng. data science) čiji je fokus bio preporučivanje glazbe, točnije zadatak automatskog nastavljanja popisa za reprodukciju. Implementacija opisana u ovom radu nije direktan odgovor na zahtjeve postavljene unutar tog izazova, ali je korišten skup podataka *The Million Playlist Dataset* koji je bio objavljen u sklopu izazova. Radi se o najvećem javno dostupnom skupu popisa za reprodukciju. Iako je izazov službeno zatvoren u srpnju 2018., zbog svoje popularnosti u istraživačkoj zajednici, Spotify ga je naknadno ponovno učinio javno dostupnim na platformi AICrowd.

5.1. Priprema podataka

The Million Playlist Dataset sačinjen je od 1000 JSON datoteka od kojih svaka sadrži po 1000 popisa za reprodukciju. Svaka datoteka formatirana je u obliku JSON rječnika (eng. JSON dictionary) sastavljenog od dva polja: "info" i "playlists". Polje "info" je rječnik koji se sastoji od osnovnih informacija o datoteci. Polje "playlist" je polje (eng. array) koje sadrži 1000 popisa za reprodukciju, a svaki popis je oblikovan kao rječnik sa sljedećim poljima:

- *pid* - jedinstveni identifikacijski broj popisa za reprodukciju u rasponu 0 - 999,999
- *name* - ime popisa za reprodukciju
- *description* - opcionalno; ako postoji, onda sadrži opis popisa za reprodukciju

- *modified_at* - vremenska oznaka trenutka kada je popis zadnji put mijenjan
- *num_artists* - ukupni broj jedinstvenih izvođača u popisu za reprodukciju
- *num_albums* - ukupni broj jedinstvenih albuma u popisu za reprodukciju
- *num_tracks* - ukupni broj jedinstvenih pjesama u popisu za reprodukciju
- *num_followers* - broj pratitelja popisa za reprodukciju
- *num_edits* - ukupan broj uređivačkih sesija
- *duration_ms* - ukupno trajanje cijelog popisa za reprodukciju
- *collaborative* - označava je li više korisnika izradilo popis za reprodukciju (istina ili neistina)
- *tracks* - polje koje sadrži informacije o svakoj pojedinoj pjesmi; svaki element polja je oblikovan u obliku rječnika sa sljedećim poljima:
  - *track_name* - ime pjesme
  - *track_uri* - Spotify uri pjesme, npr. *"spotify:track:56hioFjQ0DXrdn04hZcFgG"*
  - *album_name* - ime albuma
  - *album_uri* - Spotify uri albuma, npr. *"spotify:album:1o59UpKw81iHR0HPiSkJR0"*
  - *artist_name* - ime izvođača
  - *artist_uri* - Spotify uri izvođača, npr. *"spotify:artist:6M2wZ9GZgrQXHCFfjv46we"*
  - *duration_ms* - trajanje pjesme
  - *pos* - pozicija pjesme unutar popisa za reprodukciju

Primjer početka pohrane jednog popisa za reprodukciju prikazan je na slici 5.1.

Prije početka izrade sustava za preporučivanje, potrebno je prilagoditi format u kojem je skup podataka pohranjen. Podaci se preoblikuju za pohranu u CSV (eng. comma-seperated values) datoteke jer one omogućuju lako učitavanje podataka u podatkovne okvire (eng. data frames) i manipulaciju njima. U tu svrhu napisan je modul *generate_csv.py* koji funkcijama *mpd_to_csv* i *save_csv* generira i pohranjuje potrebne CSV datoteke. Kodom se generira pet datoteka:

- *playlists.csv*
- *tracks.csv*
- *albums.csv*
- *artists.csv*
- *tracks_playlists.csv*

```
{
    "info": {
        "generated_on": "2017-12-03 08:41:42.057563",
        "slice": "0-999",
        "version": "v1"
    },
    "playlists": [
        {
            "name": "Throwbacks",
            "collaborative": "false",
            "pid": 0,
            "modified_at": 1493424000,
            "num_tracks": 52,
            "num_albums": 47,
            "num_followers": 1,
            "tracks": [
                {
                    "pos": 0,
                    "artist_name": "Missy Elliott",
                    "track_uri": "spotify:track:0UaMYEvWZi0ZqiDOoHU3YI",
                    "artist_uri": "spotify:artist:2wIVse2owClT7go1WT98tk",
                    "track_name": "Lose Control (feat. Ciara & Fat Man Scoop)",
                    "album_uri": "spotify:album:6vV5UrXcfyQD1wu4Qo2I9K",
                    "duration_ms": 226863,
                    "album_name": "The Cookbook"
                },
```


Slika 5.1. Isječak iz datoteke *mpd.slice.0-999.json*

Datoteke su oblikovane tako da svaka igra ulogu jedne relacijske sheme koje zajedno čine jednu bazu podataka. ER model opisanog nalazi se na slici 5.2.

Datoteka *playlist.csv* u svakom retku sadrži sve osnovne podatke o pojedinom popisu za reprodukciju. Primarni ključ u relaciji je *pid*, odnosno jedinstveni identifikacijski broj popisa za reprodukciju.

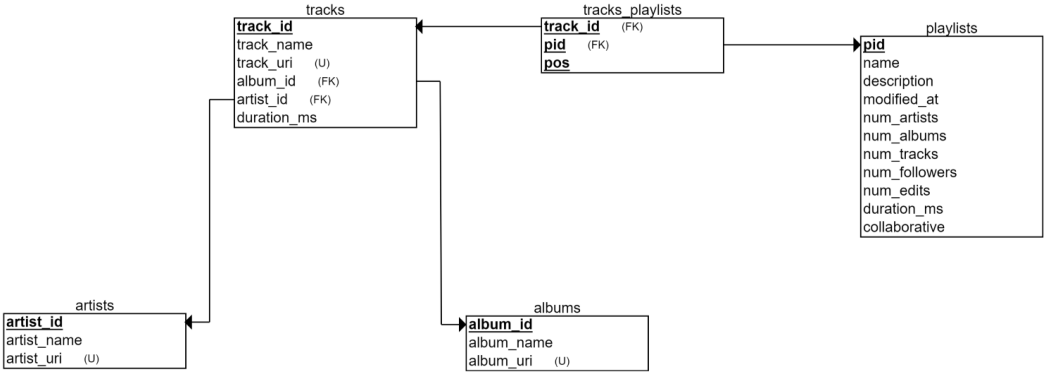
Datoteka *tracks.csv* u svakom retku sadrži sve osnovne podatke o pojedinoj pjesmi. Primarni ključ u relaciji je *track_id*, odnosno jedinstveni identifikacijski broj pjesme. Relacija sadrži i dva strana ključa, *album_id* i *artist_id*, koji povezuju pjesmu s informacijama o albumu kojem pripada, odnosno izvođaču koji ju izvodi.

Datoteka *albums.csv* u svakom retku sadrži osnovne podatke o pojedinom albumu. Primarni ključ u relaciji je *album_id*, odnosno jedinstveni identifikacijski broj albuma. Alternativni ključ u relaciji je *album_uri*.

Datoteka *artists.csv* u svakom retku sadrži osnovne podatke o pojedinom izvođaču. Primarni ključ u relaciji je *artist_id*, odnosno jedinstveni identifikacijski broj izvođača.

Alternativni ključ u relaciji je *artist_uri*.

Datoteka *tracks_playlists.csv* je relacija koja služi kao poveznica između *playlists.csv* i *tracks.csv*. Ona povezuje koje se pjesme nalaze na kojem popisu za reprodukciju i to na kojoj poziciji. Sadrži strane ključeve *pid* i *track_id*, a oni zajedno s atributom *pos* čine primarni ključ relacije.



Slika 5.2. ER model baze podataka

Implementirano je više sustava temeljeno na različitim sadržajnim značajkama, pa se tako jedna implementacija služi podacima o akustičnim značajkama (eng. acoustic features) pojedine pjesme. Kako originalni MPD skup podataka ne sadrži te podatke, njih je bilo potrebno dodati. U tu svrhu korištena je Python biblioteka *spotipy* koja podupire sve funkcionalnosti koje nudi Spotify Web API.

Spotify Web API je RESTful API (eng. Representational State Transfer Application Programing Interface) koji omogućuje izradu aplikacija koje imaju interakciju s Spotify digitalnom platformom. To uključuje dohvat metapodataka o sadržaju koji se nalazi na Spotifyu. Spotify Web API sadržan je u sklopu platforme *Spotify For Developers*. Za pristup API-ju potrebno se ulogirati Spotify korisničkim računom na *Spotify Developer Dashboard*. Zatim je potrebno kreirati aplikaciju da bi dobili pristup autorizacijskim podacima (eng. credentials). Za svaki API zahtjev (eng. API requests) koristi se OAuth 2.0 autorizacija.

Unutar modula *spotify_api.py* implementirane su funkcije koje služe za ekstrakciju potrebnih akustičnih značajki. U njemu je uvezena biblioteka *spotipy*. Prvo je inicijaliziran pristup Web API-ju. Kod funkcije prikazan je na slici 5.3. Klijentski podaci dobiveni stvaranjem aplikacije na *Spotify Dashboard-u* pohranjeni su u datoteci *credentials.json*.

Datoteka je u formatu JSON rječnika s poljima *client_id* i *client_secret*.

```
def init_spotify():
    credentials = json.load(open('credentials.json'))
    client_id = credentials['client_id']
    client_secret = credentials['client_secret']

    auth_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
    sp = spotipy.Spotify(auth_manager=auth_manager)

    return sp
```

Slika 5.3. Inicijalizija spotipy

Ekstrakciju i spremanje podataka implementira funkcija *save_features*. Funkcija učitava u Pandas podatkovni okvir datoteku *tracks.csv*. Zatim iterira po svim vrijednostima atributa *track_uri*. Za interakciju s Web API-jem pozivamo funkciju *audio_features* koja kao argument prima polje *track_uri-ja* s maksimalno 100 elemenata. Funkcija vraća akustične značajke u obliku rječnika. Opisan postupak i spremanje dobivenih značajki prikazani su na slici 5.4.

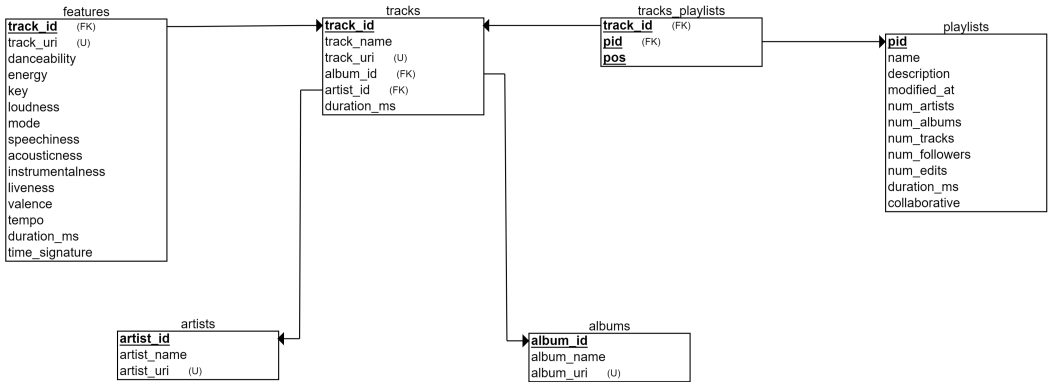
```
for i in tqdm(range(saved, len(tracks_uri), 100)):
    jump = 100
    if i + jump >= len(tracks_uri):
        jump = len(tracks_uri) - i
    while True:
        try:
            features = sp.audio_features(tracks_uri[i:i + jump])
            features_df = pd.DataFrame(features)
            features_df.drop(columns=['type', 'id', 'uri', 'track_href', 'analysis_url'], inplace=True)
            features_df.insert( loc: 0,  column: 'track_uri', tracks_uri[i:i + jump])
            features_df.insert( loc: 0,  column: 'track_id', tracks_id[i:i + jump])

            if i == 0:
                with open(save_path, 'w', encoding='utf-8', newline='') as f:
                    writer = csv.writer(f, delimiter=',')
                    writer.writerow(list(features_df.columns))

            with open(save_path, 'a', encoding='utf-8', newline='') as f:
                writer = csv.writer(f, delimiter=',')
                writer.writerows(features_df.values)

            break
        except Exception as error:
            r = open("features_error.txt", "a")
            r.write(datetime.datetime.now().strftime("%d.%b %Y %H:%M:%S") + ": " + str(error) + '\n')
            r.close()
            time.sleep(3)
            continue
```

Slika 5.4. Petlja unutar funkcije *save_features*

Podaci su spremani u datoteku *features.csv*. Datoteka u svakom retku sadrži vrijednosti sljedećih atributa:

- *track_id* - jedinstveni identifikacijski broj pjesme
- *track_uri* - Spotify uri pjesme, npr. *"spotify:track:56hioFjQ0DXrdn04hZcFgG"*
- *danceability* - opisuje koliko je pjesma prikladna za plesenje na skali 0.0 - 1.0
- *energy* - perceptivna mjera intenziteta i aktivnosti u pjesmi na skali 0.0 - 1.0
- *key* - tonalitet pjesme naveden kao broj -1 - 11 koji odgovara standardnoj notaciji visine tona; -1 označava da podatak o tonu ne postoji
- *loudness* - generalna glasnoća pjesme izražena u decibelima; vrijednosti pretežito variraju na skali -60.0 - 0.0
- *mode* - modus; 0 označava mol, a 1 označava dur
- *speechiness* - detektira prisutnost govora u pjesmi na skali 0.0 - 1.0
- *acousticness* - mjera pouzdanosti da je pjesma akustična na skali 0.0 - 1.0
- *instrumentalness* - detektira sadrži li pjesma vokal na skali 0.0 - 1.0
- *liveness* - vjerojatnost da je pjesma snimka nastupa uživo na skali 0.0 - 1.0
- *valence* - opisuje generalnu pozitivu pjesme na skali 0.0 - 1.0
- *tempo* - procijenjen tempo pjesme izražen u BPM (eng. beats per minute)
- *duration_ms* - trajanje pjesme
- *time_signature* - takt dan brojem u rasponu 3 - 7; npr. broj 3 predstavlja 3/4, broj 7 predstavlja 7/4

Datoteka je dodana među ostale CSV datoteke te je također tretirana kao relacija u bazi podataka čiji je primarni ključ *track_id*, a alternativni ključ *track_uri*. Prisutnost atributa *track_uri* i *duration_ms* krši normalne forme, ali su zadržani u svrhu lakše provjere jesu li ekstraktirani ispravni podaci. Izgled ER modela nakon dodane datoteke *features.csv* prikazan je na slici 5.5.



Slika 5.5. ER model baze podataka nakon dodavanja datoteke *features.csv*

5.2. Implementacija sustava temeljenog na suradničkom filtriranju

Pri teorijskom pregledu sustava temeljenih na suradničkom filtriranju, podijelili smo ih na dvije vrste: one temeljene na sličnosti korisnika i one temeljene na sličnosti objekata. Kako u MPD skupu podataka ne postoje podaci vezani uz konkretnog korisnika, ulogu korisnika u skupu igraju popisi za reprodukciju. Tako se pjesme koje oni sadržavaju mogu promatrati jednako kao što bi se promatrali pozitivno ocijenjeni objekti nekog korisnika u nekom drugom kontekstu. Implementacija koja će biti detaljnije opisana zato spada u sustave temeljene na sličnosti korisnika.

Jedan od najvažnijih dijelova implementacije je izgradnja matrice korisnosti. S tim ciljem korištena je Python biblioteka *scipy.sparse*. Ona nudi klasu *csr_matrix* (eng. compressed sparse row) koja služi za pohranu vrlo rijetko popunjenih matrica. Alternativna klasa koju nudi biblioteka *scipy.sparse* je *csc_matrix* (eng. compressed sparse column), ali je u ovom slučaju prikladnije koristiti *csr_matrix* jer su zadaci unutar sustava pretežito usmjereni na popise za reprodukciju, a oni će biti pohranjeni kao redovi u matrici. Brzina pretraživanja i memorijska učinkovitost *csr_matrix* je posljedica toga što matrica pohranjuje samo podatke o tome gdje se nalaze ne-nula vrijednosti.

Generiranje matrice korisnosti prepušteno je funkciji *generate_csr_matrix* koja se nalazi u modulu *data_utils.py*. Ista funkcija će kasnije, između ostalog, biti korištena za generiranje matrica potrebnih za preporučivanje temeljenog na sadržajnim značajkama, pa je kao argument *item* pri pozivu funkcije potrebno predati string *'tracks'*. Funkcija vraća csr matricu te dva rječnika: *row_to_pid* i *pid_to_row*. Rječnici služe kako bi mogli povezati koji red matrice predstavlja koji popis za reprodukciju pri čemu su popisi određeni identifikatorom *pid*. Što se tiče stupaca, broj stupca jednak je identifikatoru *track_id* pojedine pjesme. Vrijednosti ćelija matrice mogu biti 0 ili 1, ovisno o tome nalazi li se pjesma u popisu za reprodukciju ili ne. Relevantan dio logike funkcije *generate_csr_matrix* prikazan je na slici 5.6. O ulozi varijable *holdout* biti će riječi pri evaluaciji sustava, ali zasad nije potrebno o tome govoriti jer će ona pri generiranju matrice korisnosti imati vrijednost *None*.

```
if item == 'tracks':
    unique_tracks_playlists = tracks_playlists.drop_duplicates(subset=['track_id', 'pid'])

    if holdout is not None:
        holdout_set = set(zip(holdout['pid'], holdout['track_id']))
        unique_tracks_playlists = unique_tracks_playlists[
            ~unique_tracks_playlists.apply(lambda x: (x['pid'], x['track_id']) in holdout_set, axis=1)
        ]

    for row, (pid, playlist) in enumerate(tqdm(unique_tracks_playlists.groupby('pid'), desc: 'Generating CSR matrix')):
        row_to_pid[row] = pid
        pid_to_row[pid] = row
        playlist_tracks = playlist['track_id']
        for track in playlist_tracks:
            rows.append(row)
            cols.append(track)
            data.append(1)

    matrix = sp.csr_matrix( arg1: (data, (rows, cols)), shape=(len(unique_tracks_playlists['pid'].unique()),
                                                    len(tracks)), dtype=np.int32)
```

Slika 5.6. Isječak koda funkcije *generate_csr_matrix*

Sustav temeljen na suradničkom filtriranju implementiran je u klasi *KNNRecommender*. Glavna logika sustava sadržana je u atributu *model* koji je instanca klase *NearestNeighbors*. *NearestNeighbors* dio je Python biblioteke *sklearn.neighbors* te nudi funkcionalnosti za nenadzirano i nadzirano učenje. U sustavu za preporučivanje služit će za dobivanje k-najbližih susjeda pojedinog popisa za reprodukciju. U konstruktoru klase *NearestNeighbors* bira se funkcija sličnosti za korištenje, algoritam za treniranje modela te broj susjeda *k*. Ponuđeni su algoritmi za treniranje: *brute*, *kd_tree* i *ball_tree*, dok ponuđene funkcije sličnosti ovise o odabiru algoritma. U konstruktoru klase *KNNRecommender* prosljeđuju se argumenti *algorithm*, *metric* i *k* koji se predaju u konstruktor atributa *model*. U konstruktor sustava se također prosljeđuje podatkovni okvir *tracks_playlists* koji sadrži sve pjesme koje se mogu pojaviti u popisu za reprodukciju. Opisano je prikazano na slici 5.7.

Sustav se gradi u metodi *build_model* kojoj se predaje matrica korisnosti i pomoćni rječnik *row_to_pid*. Matrica korisnosti se predaje kao argument metodi *fit* klase *NearestNeighbors* koja će je pohraniti i na temelju nje određivati k-najbliže susjede.

Preporuke sustava dobivaju se pozivom metode *predict*. U funkciju se predaje popis za reprodukciju koji želimo nadopuniti, koliko pjesama se treba preporučiti te set koji sadrži identifikatore *track_id* pjesama koje se već nalaze na popisu. Izlaz funkcije je lista koja sadrži preporučene pjesme poredane silazno po stupnju preporučljivosti.

```python
class KNNRecommender:
    def __init__(self, tracks_playlists: pd.DataFrame, metric: str, algorithm: str, k: int):
        self.tracks_playlists = tracks_playlists
        self.metric = metric
        self.algorithm = algorithm
        self.k = k
        self.model = NearestNeighbors(metric=self.metric, algorithm=self.algorithm, n_neighbors=k, n_jobs=-1)
        self.sparse_matrix = None
        self.row_to_pid = None

    7 usages
    def build_model(self, data: sp.csr_matrix, row_to_pid: dict):
        self.sparse_matrix = data
        self.row_to_pid = row_to_pid
        self.model.fit(data)

    8 usages (2 dynamic)
    def predict(self, playlist: sp.csr_matrix, num_of_predictions: int, playlist_tracks: set):
        neighbors = self._get_neighbors(playlist)
        neighbor_pids = self._get_playlists_from_neighbors(neighbors)
        tracks = [self.tracks_playlists[self.tracks_playlists['pid'] == n]['track_id'].values for n in neighbor_pids]
        predictions = self._get_predictions_from_neighbor_tracks(playlist_tracks, tracks, num_of_predictions)
        return predictions
```

Slika 5.7. Prikaz konstruktora i glavnih metoda klase *KNNRecommender*

5.3. Implementacija sustava temeljenog na sadržajnim značajkama

U poglavlju 4. navedeno je da je jedan od glavnih izazova izgradnje sustava temeljenih na sadržajnim značajkama upravo sami odabir značajki koje će se koristiti za određivanje sličnosti. Da bi predočili koliko je bitan ispravan odabir značajki izrađena su dva sustava: jedan sustav određuje sličnost na temelju izvođača ili albuma koji se pojavljuju na popisu za reprodukciju, a drugi određuje sličnost na temelju akustičnih značajki.

5.3.1. Preporučivanje temeljeno na izvođačima i albumima

Implementacija ovog sustava ostvarena je malo drugačije od klasičnog pristupa opisanog u teorijskom dijelu ovog rada. Ponovno je korištena klasa *KNNRecommender*, a temeljna razlika u odnosu na sustav temeljen na suradničkom filtriranju biti će matrica predana za izgradnju modela.

Matrica korisnosti u ovom sustavu i dalje će odražavati popise za reprodukciju u redcima, a stupci će predstavljati odabranu sadržajnu značajku, izvođače ili albume. Vrijednosti u matrici predstavljat će koliko je određena značajka relevantna za pojedini popis. U tu svrhu se ponovno koristi funkcijom *generate_csr_matrix*, ali se ovdje kao argument *item* predaje string *'artists'*, odnosno string *'albums'*. Postupak stvaranja csr matrice analogan je postupku prikazanom na slici 5.6. Razlika je da u ovoj matrici vrijednosti nisu

samo 0 ili 1, već je potrebno odrediti frekvenciju pojavljivanja određene značajke na popisu za reprodukciju, pa se izostavlja stvaranje varijable *unique_tracks_playlist* te se na njenom mjestu svugdje koristi podatkovni okvir *tracks_playlist*. Prije povratka iz funkcije, izrađena se matrica normalizira, a za to se koristi funkcija *normalize_matrix* koja se također nalazi u modulu *data_utils*. Kod funkcije prikazan je na slici 5.8.

```
def normalize_matrix(matrix: sp.csr_matrix):
    tfidf_transformer = TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True)
    tfidf_matrix = tfidf_transformer.fit_transform(matrix)
    return tfidf_matrix
```

Slika 5.8. Funkcija *normalize_matrix*

Normalizacija matrice postiže se korištenjem TF-IDF transformacije, a ta funkcionalnost implementirana je u klasi *TfidfTransformer* koja je dio Python biblioteke *sklearn.feature_extraction.text*. Pozivom metode *fit_transform* nad matricom se primjenjuje TF-IDF transformacija koju slijedi normalizacija L2 normom.

Dobivena matrica predaje se kao argument metodi *build_model* klase *KNNRecommender*. Činjenica da se preporuke baziraju na k-najsličnijih popisa za reprodukciju odstupanje je od klasičnog pristupa. Međutim, budući da je sličnost popisa za reprodukciju određena isključivo njihovim sadržajnim značajkama, ovaj sustav se može svrstati u one temeljene na sadržajnim značajkama.

5.3.2. Preporučivanje temeljeno na akustičnim značajkama

Za implementaciju ovog sustava korištena je klasa *CBFRecommender*. U konstruktor klase predaje se podatkovni okvir *features* koji je učitan iz datoteke *features.csv*. Konstruktor klase prikazan je na slici 5.9. Nakon popunjavanja nedostajućih vrijednosti, nad podatkovnim okvirom se primjenjuje normalizacija korištenjem klase *MinMaxScaler* koja je sadržana u Python biblioteci *sklearn.preprocessing*.

U sklopu sustava omogućene su dvije vrste preporuka: preporuke bazirane na prosječnim vrijednostima akustičnih značajki predanog popisa za reprodukciju ili preporuke bazirane na akustičnim značajkama svake pojedinačne pjesme na popisu. Implementacija prve vrste preporučivanja prikazana je na slici 5.10., a druge na slici 5.11. Obje metode se služe funkcijom sličnosti *cosine_similarity* koja je uvezena iz Python biblioteke *sklearn.metrics.pairwise*.

```python
class CBFRecommender:
    def __init__(self, features: pd.DataFrame):
        self.features = features
        self.scaler = MinMaxScaler()

        self.continuous_columns = ['danceability', 'energy', 'loudness', 'speechiness', 'acousticness',
                                   'instrumentalness', 'liveness', 'valence', 'tempo']
        self.binary_columns = ['key', 'mode', 'time_signature']
        self.drop_columns = ['track_id', 'track_uri', 'duration_ms']

        self.features[self.continuous_columns] = self.features[self.continuous_columns].fillna(
            self.features[self.continuous_columns].mean())

        for col in self.binary_columns:
            self.features[col] = self.features[col].fillna(self.features[col].mode()[0])

        self.scaled_tracks = self.features.copy()
        self.scaled_tracks[self.continuous_columns] = self.scaler.fit_transform(self.features[self.continuous_columns])

        self.feature_matrix = self.scaled_tracks[self.continuous_columns + self.binary_columns].values
        self.track_ids = self.scaled_tracks['track_id'].values
```

Slika 5.9. Konstruktor klase *CBFRecommender*

Metoda *_mean_feature_recommendations* kreira lokalnu varijablu *mean_playlist_vector* koja se može smatrati profilom popisa za reprodukciju. Profil se uspoređuje sa svim pjesmama u skupu, a funkcija vraća listu najsličnijih pjesama sortiranu silazno po sličnosti.

```python
def _mean_feature_recommendations(self, playlist_tracks: set, num_of_predictions: int):
    playlist_features = self.scaled_tracks[self.scaled_tracks['track_id'].isin(playlist_tracks)]

    mean_continuous_vector = playlist_features[self.continuous_columns].mean(axis=0).values
    mode_binary_vector = playlist_features[self.binary_columns].mode().iloc[0].values
    mean_playlist_vector = pd.concat(
        [pd.Series(mean_continuous_vector), pd.Series(mode_binary_vector)]).values.reshape(1, -1)

    relevant_tracks = self.scaled_tracks[~self.scaled_tracks['track_id'].isin(playlist_tracks)].copy()
    rel_tracks_features = relevant_tracks[self.continuous_columns + self.binary_columns].values

    similarity = cosine_similarity(mean_playlist_vector, rel_tracks_features).flatten()

    relevant_tracks['similarity'] = similarity
    recommendations = relevant_tracks.sort_values(by='similarity', ascending=False).head(num_of_predictions)

    return recommendations['track_id'].tolist()
```

Slika 5.10. Metoda *_mean_feature_recommendations* klase *CBFRecommender*

Metoda *_track_specific_recommendations* uspoređuje svaku pjesmu na popisu za reprodukciju s ostalim pjesmama u skupu te pohranjuje vrijednost sličnosti u rječnik *track_similarity*. Ako je sličnost neke pjesme već pohranjena, ona se mijenja, ako je nova sličnost veća od te pohranjene. Povratna vrijednost metode je lista pjesama poredanih silazno po vrijednosti sličnosti.

```
def _track_specific_recommendations(self, playlist_tracks: set, num_of_predictions: int):
    track_similarity = defaultdict(float)
    relevant_tracks = self.scaled_tracks[~self.scaled_tracks['track_id'].isin(playlist_tracks)].copy()
    rel_tracks_features = relevant_tracks[self.continuous_columns + self.binary_columns].values

    for track in playlist_tracks:
        track_features = self.scaled_tracks[self.scaled_tracks['track_id'] == track][self.continuous_columns +
                                                                                     self.binary_columns].values
        similarity = cosine_similarity(track_features, rel_tracks_features).flatten()
        for i, track_id in enumerate(relevant_tracks['track_id'].values):
            if similarity[i] > track_similarity[track_id]:
                track_similarity[track_id] = similarity[i]

    sorted_tracks = sorted(track_similarity.items(), key=lambda x: x[1], reverse=True)
    sorted_recommendations = [track_id for track_id, _ in sorted_tracks[:num_of_predictions]]

    return sorted_recommendations
```

Slika 5.11. Metoda *_track_specific_recommendations* klase *CBFRecommender*

5.4. Implementacija hibridnog sustava za preporučivanje

Ideja je implementacije hibridnog sustava za preporučivanje odrediti koliko je svaki dosada implementirani sustav pogodan za davanje preporuke za predani popis za reprodukciju. Time određujemo koliku će težinu nositi preporuke pojedinog sustava, a izlaz hibridnog sustava bit će kombinacija preporuka svih osnovnih sustava. Unutar hibridnog sustava kombinirat će se sustav temeljen na suradničkom filtriranju te sustavi temeljeni na izvođačima i albumima.

Implementacija hibridnog sustava izvedena je klasom *HybridRecommender*. Konstruktoru klase se predaju podatkovni okviri *tracks* i *tracks_playlists* učitani iz datoteka *tracks.csv* i *tracks_playlists.csv* te argumenti potrebni za stvaranje osnovnih sustava. Težine pojedinih sustava unutar modela bit će određene linearnom regresijom, a u tu svrhu koristi se klase *LinearRegression* koja je dio Python biblioteke *sklearn.linear_model*.

```
class HybridRecommender:
    def __init__(self, tracks: pd.DataFrame, tracks_playlists: pd.DataFrame, metric: str, algorithm: str, k: int):
        self.tracks = tracks
        self.tracks_playlists = tracks_playlists
        self.collaborative_model = KNNRecommender(tracks_playlists, metric, algorithm, k)
        self.content_artists_model = KNNRecommender(tracks_playlists, metric, algorithm, k)
        self.content_albums_model = KNNRecommender(tracks_playlists, metric, algorithm, k)
        self.hybrid_model = LinearRegression()
```

Slika 5.12. Konstruktor klase *HybridRecommender*

Za treniranje sustava potrebno je podijeliti argument *training_playlist* metode *build_model* na skup podataka za izgradnju osnovnih modela i na skup podataka za treniranje modela linearne regresije. Ta funkcionalnost sadržana je u funkciji *split_data*

koja se nalazi u modulu *data_utils*. Podaci za treniranje modela linearne regresije moraju biti u obliku para (*ulazna vrijednost*, *ciljna vrijednost*), a za to je korištena metoda *_prepare_hybrid_model_data*.

Ulazne vrijednosti modela linearne regresije biti će podaci za koje smatramo da imaju utjecaj na uspješnost preporučivanja pojedinog sustava. Kao takve smatramo duljinu predanog popisa za reprodukciju, *playlist_length*, heterogenost popisa što se tiče broja različitih izvođača, *artist_heterogenity*, te heterogenost popisa što se tiče broja različitih albuma koje sadrži, *album_heterogenity*. Evaluacija pojedinih sustava bit će pobliže opisana u 6. poglavlju, ali ona je već ovdje korištena za određivanje izlaznih vrijednosti linearnog model. Izlazne vrijednosti su r-preciznosti preporuka dobivenih osnovnim sustavima. Opisana funkcionalnost prikazana je na slici 5.13.

```
x = []
y = []

for i, playlist in enumerate(tqdm(collaborative_test_matrix, desc: "Preparing linear regression model")):
    row_to_pid = collaborative_dict[0]
    held_out = set(x[1] for x in holdout_set if x[0] == row_to_pid[i])
    num_of_predictions = len(held_out)

    playlist_tracks = playlist.indices
    collaborative_recs = self.collaborative_model.predict(playlist, 15 * num_of_predictions, playlist_tracks)
    artists_recs = self.content_artists_model.predict(
        artists_test_matrix.getrow(artists_dict[1][row_to_pid[i]]), 15 * num_of_predictions, playlist_tracks)
    albums_recs = self.content_albums_model.predict(albums_test_matrix.getrow(albums_dict[1][row_to_pid[i]]),
                                                    15 * num_of_predictions, playlist_tracks)

    collaborative_r_precs = r_precision(collaborative_recs, held_out)
    artists_r_precs = r_precision(artists_recs, held_out)
    albums_r_precs = r_precision(albums_recs, held_out)

    playlist_length = len(playlist_tracks)
    artist_heterogeneity = self._calculate_heterogeneity(playlist_tracks, feature: 'artist_id')
    album_heterogeneity = self._calculate_heterogeneity(playlist_tracks, feature: 'album_id')

    x.append([playlist_length, artist_heterogeneity, album_heterogeneity])
    y.append([collaborative_r_precs, artists_r_precs, albums_r_precs])

return np.array(x), np.array(y)
```

Slika 5.13. Isječak koda metoda *_prepare_hybrid_model_data*

Metoda *predict* poziva se za dobivanje preporuka hibridnog sustava. Njena implementacija prikazana je na slici 5.12. Potrebno je izračunati *playlist_length*, *artist_heterogenity*, *album_heterogenity* popisa za reprodukciju nad kojim se odvija preporučivanje i predati ih kao ulazne vrijednosti linearnom modelu. On na temelju njih predviđa očekivane r-preciznosti osnovnih modela. R-preciznosti se koriste kao težine pojedinih modela pri kombiniranju njihovih preporuka. Povratna vrijednost metode lista je preporučenih pjesama poredana silazno po očekivanoj vrijednosti relevantnosti

preporuke.

```
def predict(self, playlist: sp.csr_matrix, playlist_artists: sp.csr_matrix, playlist_albums: sp.csr_matrix,
            num_of_predictions: int):
    playlist_length = playlist.shape[0]
    artist_heterogeneity = self._calculate_heterogeneity(playlist.indices, feature: 'artist_id')
    album_heterogeneity = self._calculate_heterogeneity(playlist.indices, feature: 'album_id')

    features = np.array([[playlist_length, artist_heterogeneity, album_heterogeneity]])

    weights = self.linear_model.predict(features).flatten()

    collaborative_recs = self.collaborative_model.predict(playlist, num_of_predictions,
                                                          playlist.indices)
    artists_recs = self.content_artists_model.predict(playlist_artists, num_of_predictions,
                                                      playlist.indices)
    albums_recs = self.content_albums_model.predict(playlist_albums, num_of_predictions,
                                                    playlist.indices)

    recs = defaultdict(float)
    for i in range(len(collaborative_recs)):
        recs[collaborative_recs[i]] += (1 / (i + 1)) * weights[0]
    for i in range(len(artists_recs)):
        recs[artists_recs[i]] += (1 / (i + 1)) * weights[1]
    for i in range(len(albums_recs)):
        recs[albums_recs[i]] += (1 / (i + 1)) * weights[2]

    predictions = heapq.nlargest(num_of_predictions, recs, key=recs.get)

    return predictions
```

Slika 5.14. Metoda *predict* klase *HybridRecommender*

6. Evaluacija implementiranih sustava

Evaluacija implementiranih sustava napravljena je korištenjem mjera r-preciznost (eng. r-precision) i NDCG (eng. Normalized Discounted Cumulative Gain), a korištene funkcije nalaze se u modulu *model_performance*.

Neka je R skup koji sadrži preporuke sustava te neka je G skup koji sadrži očekivane, relevantne pjesme za pojedini popis za reprodukciju. R-preciznost je omjer broja relevantnih preporuka (presjek skupova R i G) i ukupnog broja relevantnih pjesama.

$$R\text{-}precision = \frac{|R_{1:|G|} \cap G|}{|G|}$$

NDCG je omjer mjera DCG (eng. Discounted Cumulative Gain) i IDCG (eng. Ideal Discounted Cumulative Gain). DCG mjeri kvalitetu dobivene preporuke, koja je ovisna o tome na kojim pozicijama u listi se nalaze relevantne pjesme. DCG je veći što se više pjesama nalazi pri početku liste. IDCG je vrijednost DCG-a kada su sve relevantne pjesme u preporuci poredane na početku liste.

$$DCG = rel_0 + \sum_{i=1}^{|R|} \frac{rel_i}{\log_2 i}$$

$$IDCG = 1 + \sum_{i=1}^{|R \cap G|} \frac{1}{\log_2 i}$$

$$NDCG = \frac{DCG}{IDCG}$$

U svrhu evaluacije, skup podataka je podijeljen na dva dijela: *train_set* i *test_set*. Za to se koristila funkcija *data_split* modula *data_utils*. U tijelu te funkcije korištena je funkcija *train_test_split* iz Python biblioteke *sklearn.model_selection*. Iz svakog popisa za reprodukciju sadržanog u skupu *test_set* izuzet je određen postotak pjesama. Uspješnost sustava mjeri se kao njegova mogućnost da preporuči upravo te pjesme. Evaluacija svih sustava provedena je u Jupyter bilježnici *main.ipynb*.

Na slici 6.1. prikazani su rezultati evaluacije sustava za preporučivanje temeljenog na suradničkom filtriranju opisanom u potpoglavlju 5.2.

```
COLLABORATIVE FILTERING MODEL
Avg. r-precision: 0.12996916136007136
Avg. ndcg: 0.4139471061700259 5
```

Slika 6.1. Rezultati evaluacije sustava za preporučivanje temeljenog na suradničkom filtriranju

Na slikama 6.2. i 6.3. prikazani su rezultati evaluacija sustava za preporučivanje temeljenih na sadržajnim značajkama, pri čemu su odabrane značajke bile izvođači sadržani u popisu za reprodukciju, odnosno albumi. Sustav je opisan u potpoglavlju 5.2.

```
CONTENT BASED(ARTIST) MODEL
Avg. r-precision: 0.10695226195467444
Avg. ndcg: 0.3626843598199184
```

Slika 6.2. Rezultati evaluacije sustava za preporučivanje temeljenog na izvođačima

```
CONTENT BASED(ALBUM) MODEL
Avg. r-precision: 0.12383172318632342
Avg. ndcg: 0.40820918642684834
```

Slika 6.3. Rezultati evaluacije sustava za preporučivanje temeljenog na albumima

Na slikama 6.4. i 6.5. prikazani su rezultati evaluacija sustava za preporučivanje temeljenih na sadržajnim značajkama, pri čemu su odabrane značajke bile akustične zna-

čajke pjesama na popisu za reprodukciju. Na slici 6.4. su prikazani rezultati kada je za predviđanje korištena prosječna vrijednost značajki u popisu, a na slici 6.5. kada su za predviđanje korištene vrijednosti značajki svake pojedine pjesme na popisu. Sustav je opisan u potpoglavlju 5.3.2.

```
CONTENT BASED(FEATURES) MODEL
Avg. r-precision: 0.01409687236067991 8
Avg. ndcg: 0.03461269478217441 5
```

Slika 6.4. Rezultati evaluacije sustava za preporučivanje temeljenog na akustičnim značajkama (1)

```
CONTENT BASED(FEATURES) MODEL
Avg. r-precision: 0.013443709800082053
Avg. ndcg: 0.03485635452138529 5
```

Slika 6.5. Rezultati evaluacije sustava za preporučivanje temeljenog na akustičnim značajkama (2)

Na slici 6.6. prikazani su rezultati evaluacije hibridnog sustava za preporučivanje opisanog u potpoglavlju 5.4.

```
HYBRID MODEL
Avg. r-precision: 0.1546100887305634 5
Avg. ndcg: 0.4081290940136488
```

Slika 6.6. Rezultati evaluacije hibridnog sustava za preporučivanje

7. Zaključak

U ovom završnom radu predstavljen je problem preporučivanje te je dana formalna definicija problema (2. poglavlje). Napravljena je osnovna podjela sustava za preporučivanje te su detaljno opisani sustavi temeljeni na suradničkom filtriranju (3. poglavlje) i sustavi temeljeni na sadržajnim značajkama (4. poglavlje). U 5. poglavlju detaljno je opisana implementacija više vrsta sustava za preporučivanje, uključujući i jedan hibridni sustav. Implementacija je napravljena u programskom jeziku Python, a za skup podataka korišten je Spotifyev *The Million Playlist Dataset*.

Evaluacija napravljena u 6. poglavlju i njezini rezultati govore o prikladnosti korištenja pojedinih sustava za problem preporučivanje glazbe. Uspješnost sustava za preporučivanje temeljenog na suradničkom filtriranju ukazuje na socijalni aspekt glazbe. Korisnici sličnih ukusa utječu jedni na druge po pitanju pjesama koje slušaju te se tako iste pjesme često nađu na popisima za reprodukciju više korisnika. Trendovi također imaju svoju važnost jer je vjerojatnije da će se popularne pjesme pojaviti na nekom popisu za reprodukciju. Vrlo loši rezultati dobiveni pri korištenju sustava temeljenog na akustičnim značajkama indiciraju da akustične značajke ne igraju važnu, ili ikakvu, u korisnikovu stvaranju popisa za reprodukciju. Veća uspješnost sustava temeljenih na izvođačima i albumima pokazuje da korisnici najčešće stvaraju popise za reprodukciju tako da dodaju sve pjesme s istog albuma jednu za drugom ili tako da dodaju sve pjesme istog izvođača. Hibridni sustav ima najveću uspješnost i to nam govori da je najbolji pristup rješavanju problema preporučivanja glazbe upravo kombinacija više sustava.

U budućnosti bi se kao dorada na ovaj rad mogli implementirati i sustavi temeljeni na modelu. Očekivanje je da bi ti sustavi potencijalno još bolje mogli riješiti problem zbog svoje kompleksnosti i mogućnosti otkrivanja latentnih faktora koji imaju utjecaj na korisnika.

Literatura

[1] Artificial Intelligence - All in One, “Mining massive datasets - stanford university [full course]”, https://www.youtube.com/playlist?list=PLLssT5z_DsK9JDLcT8T62VtzwyW9LNepV, 2016., [mrežno; stranica posjećena: lipanj 2024.; relevantni videozapisi: 41-45].

[2] C. W. Chen, P. Lamere, M. Schedl, i H. Zamani, “Recsys challenge 2018: Automatic music playlist continuation”, u *Proceedings of the 12th ACM Conference on Recommender Systems (RecSys ’18)*, 2018.

[3] Z. Damijanić, “Sustavi za davanje preporuka”, Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, Zagreb, 2017., [pristupljeno 07.06.2024.] Dostupno na: https://urn.nsk.hr/urn:nbn:hr:217:058735.

[4] Github, https://github.com/tmscarla/spotify-recsys-challenge, [mrežno; stranica posjećena: lipanj 2024.].

[5] FER - Analiza velikih skupova podataka, “Recommender systems”, https://www.fer.unizg.hr/_download/repository/AVSP_11_RecSys[1].pdf, [mrežno; stranica posjećena: lipanj 2024.].

[6] A. Rajaraman i J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2011.

[7] J. Šumečki, “Sustavi za preporučivanje”, https://web.math.pmf.unizg.hr/nastava/su/index.php/download_file/-/view/34/, 2010., [mrežno; stranica posjećena: lipanj 2024.].

Sažetak

Hibridni sustav za preporučivanje glazbe zasnovan na suradničkom filtriranju i sadržajnim značajkama

Ana Vuksanović

U ovom radu napravljen je osnovni pregled sustava za preporučivanje te su detaljno opisani sustavi zasnovani na suradničkom filtriranju i oni zasnovani na sadržajnim značajkama. Korištenjem skupa podataka *The Million Playlist Dataset*, koji je objavljen u sklopu istraživačkog izazova *Spotify Million Playlist Dataset Challenge*, napravljene su implementacije opisanih sustava te je također napravljena implementacija hibridnog sustava za preporučivanje koji se koristi značajkama suradničkog filtriranja i sadržajnim značajkama. Sustavi su evaluirani koristeći mjere r-preciznost i NDCG.

Ključne riječi: sustav za preporučivanje; hibridni sustav za preporučivanje glazbe; suradničko filtriranje; sustav za preporučivanje zasnovan na sadržajnim značajkama

Abstract

Hybrid Recommender System for Music Based on Collaborative Filtering and Content Features

Ana Vuksanović

This paper provides basic overview of recommender systems and describes recommender systems based on collaborative filtering and recommender systems based on content features in greater detail. The paper outlines the implementation of the described systems made using *The Million Playlist Dataset*, which was published as part of the *Spotify Million Playlist Dataset Challenge*. The paper also provides an implementation of a hybrid recommender system for music based on collaborative filtering and content features. The systems were evaluated using r-precision and NDCG.

Keywords: recommender system; hybrid recommender system for music; collaborative filtering; content-based model