Upotreba algoritama strojnog učenja na graf bazi filmovi i tv emisije

Pejić, Ines

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij Matematika i računarstvo

Upotreba algoritama strojnog učenja na graf bazi *Filmovi i TV emisije*

Završni rad

Student:

Ines Pejić

Osijek, 2024

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij Matematika i računarstvo

Upotreba algoritama strojnog učenja na graf bazi *Filmovi i TV emisije*

Završni rad

Mentor:
doc. dr. sc. Mateja Đumić

Student:
Ines Pejić

Osijek, 2024

Sadržaj

1 | Uvod

Graf je objekt koji se sastoji od vrhova ili čvorova i bridova ili stranica, a baza podataka je organizirani skup podataka. Termin baza podataka nastao je u računarstvu, a podrazumijeva pohranu skupa podataka u računalo, telefon ili bilo koji elektronički uređaj. Graf baza podataka upotrebljava strukturu grafa za dohvaćanje, organizaciju i pohranjivanje podataka. Neo4j je sustav za upravljanje graf bazom podataka koji pruža prirodan način modeliranja entiteta i veza te uzima u obzir odnose susjednih čvorova, koji je često ključan za izvlačenje maksimuma iz podataka.

U ovom poglavlju dan je uvod u završni rad te pregled rada po poglavljima.

U drugom poglavlju ćemo u kratkim crtama objasniti vrste baza podataka te zašto su graf baze podataka korisne i popularne.

Treće poglavlje nas upoznaje s Neo4jem, sustavom za upravljanje graf bazom podataka.

Potom, u četvrtom poglavlju upoznajemo Cypher, upitni jezik stvoren od strane Neo4ja. Dodatno, u ovom poglavlju, bit će opisan postupak uvoza CSV podataka u Neo4j bazu pomoću Cyphera.

Zatim, u petom poglavlju upoznajemo graf baze podataka opisujući svojstva bridova i čvorova. Dodatno, upoznat ćemo se s Graph Data Science bibliotekom i alatima koje ova biblioteka sadrži. Ovi alati sadrže algoritme strojnog učenja, cjevovode, projiciranje grafa i Python klijent. U radu je objašnjeno na koji način radi dvanaest algoritama iz Graph Data Science biblioteke.

U šestom poglavlju dokumentiran je praktični dio ovog rada koji se sastoji od primjene navedenih algoritama na bazi *Filmovi i TV emisije*. Osim toga nabrojane su neke loše prakse i uobičajene greške prilikom pokretanja algoritama te kako spriječiti probleme konfiguracije, projekcije grafa i kataloga.

2 | Baze podataka

Baze podataka su organizirana zbirka povezanih i međusobno ovisnih podataka, koje su pohranjene u nekom od računalno čitljivih medija. Unutar baza podataka moguće je postavljanje upita nad spremljenim podacima. Baze podataka dijelimo na relacijske baze i NoSQL [1] baze podataka.

Relacijske baze podataka svoje podatke spremaju u relacije (tablice) na način da su stupci atributi[2], a redovi instance entiteta[3]. Podaci su organizirani prema unaprijed zadanoj shemi.

NoSQL baze podataka su baze podataka koje nisu primarno relacijske. Uobičajeno, NoSQL baze podataka se prema svojstvima dijele na četiri vrste baza podataka: dokumentna baza podataka (engl. *document database*), ključ-vrijednost (engl. *key-value database*), graf baze podataka (engl. *graph database*) i baza zasnovana na stupcima (engl. *column family database*).

Ključ-vrijednost baza podataka je jednostavna baza podataka u kojoj je ključ obično string, dok je vrijednost tipa blob[4]. Ovakav način pohrane je vrlo učinkovit za predmemoriju u kontekstu web aplikacija gdje vrijednost predstavlja HTML (engl. *HyperText Markup Language*) sadržaj stranice, a ključ URL (engl. *Uniform Resource Locator*) stranice.

Dokumentne baze podataka sliče ključ-vrijednost bazama podataka. Osnovni element ovog podatka je dokument: uređeni skup ključeva s pridruženim vrijednostima. Dokumentne baze podataka su prilagodljivije u odnosu na relacijske baze podataka, no u njima je teže modelirati odnose među entitetima.

Baza zasnovana na stupcima je baza koja sadrži stupce povezanih podataka. Stupci povezanih podataka su par ključ-vrijednost, gdje je ključ preslikan u vrijednost koja je skup stupaca, dok je stupac uređena trojka koja se sastoji od naziva stupaca, vremenske oznake i vrijednosti.

Graf baza podataka sprema podatke u čvorove i bridove umjesto u dokumente ili tablice. Podaci se pohranjuju bez ograničavanja na unaprijed definirani model, što omogućuje vrlo fleksibilan način razmišljanja i korištenja baze podataka.

---

[1] NoSQL - engl. *Not only structured query language*.

[2] Atributi - specifična svojstva entiteta.

[3] Entiteti - zajednički naziv za skup sličnih stvari.

[4] Blob - (engl. *binary large object*) koristi se za pohranu velikih datoteka, vrijednost je pohranjena kao niz bajtova.

Graf je matematički objekt određen skupom bridova i skupom čvorova, odnosno vrhova. Prednost grafova je ta što se njime može lako prolaziti, prelazeći s jednog čvora na drugi, prateći bridove. Pažljivom analizom mreže mogu se izvući brojne informacije, kao što su rangiranje čvorova i njezina struktura. Graf baza podataka omogućuje modeliranje veza i entiteta te uzima u obzir značenje čvora kako bi se iskoristio u cijelosti. Primarna uloga graf baze podataka je povezivanje podataka, što u suštini olakšava izradu upita.

3 | Neo4j

3.1 Početak i razvoj Neo4ja

Neo4j je sustav za upravljanje graf bazom podataka koji je razvila tvrtka Neo4j. Prvi dio koda je razradio Emil Eifrem, koji je izvršni direktor i osnivač tvrtke Neo4j. Johan Svensson, Peter Naubauer i Emil Eifrem su u 2000. godini primjetili velike troškove u radu na jednoj od njihovih aplikacija. Nakon provedbe različitih istraživanja vezano uz ovaj problem, započeli su s projektom razvoja Neo4ja. Dodatan razvoj i istraživanja dovela su Neo4j među vodeća rješenja pohrane podataka, pravdajući slogan na promotivnim materijalima „Vodeća svjetska graf baza podataka“ (engl. *„The World's Leading Graph Database“*).

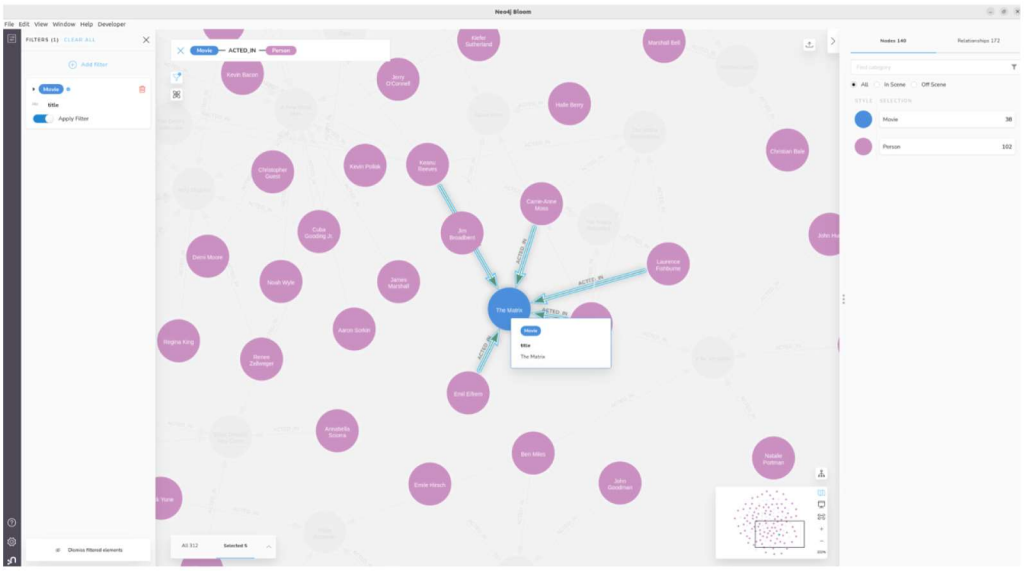
3.2 Neo4j ekosustav

Neo4j baza podataka sastoji se od velike količine proširenja, aplikacija i biblioteka. Neo4j desktop je aplikacija koja omogućuje upravljanje instaliranim dodacima i aplikacijama. Nakon instalacije aplikacije na Neo4j Desktop, aplikacije imaju pristup aktivnoj bazi. Osim instalacije aplikacija moguća je i izrada vlastite aplikacije što je jedan od razloga zašto postoji puno različitih aplikacija za Neo4j. Neo4j ekosustav sastoji se od aplikacija kao što su Neodash, Neo4j Browser i Neo4j Bloom.

Neo4j Browser je aplikacija koja omogućava vizualizaciju rezultata u obliku grafova i pisanje Cypher upita. Slika zaslona ove aplikacije prikazana je na Slici 3.1.

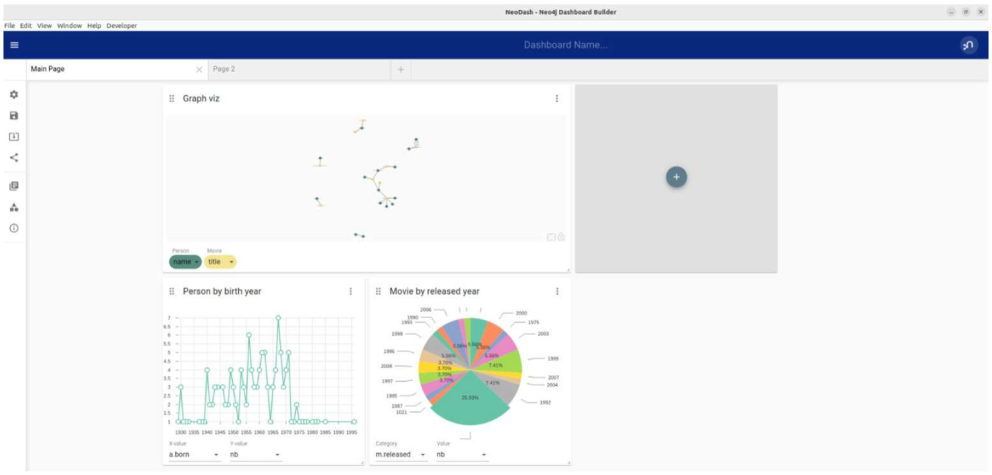
Slika 3.1: Neo4j Browser aplikacija

Neo4j Bloom je aplikacija za vizualni prikaz grafova koja ima mogućnost prilagodbe stilova čvorova, boja, veličine itd. Neo4j Bloom aplikacija ne zahtijeva nikakve programske vještine za pregled i analizu, a njezin prikaz zaslona dan je na Slici 3.2.



Slika 3.2: Neo4j Bloom aplikacija

Neodash (engl. *Dashboard*) je aplikacija za izvještaje pomoću koje možemo crtati dijagrame iz podataka pohranjenih u Neo4ju. Unutar aplikacije postoji mogućnost organizacije izvještaja koju je moguće podijeliti s drugim korisnicima. Slika zaslona ove aplikacije dana je na Slici 3.3.



Slika 3.3: Neodash aplikacija

3.3 Instalacija Neo4j Desktop aplikacije

Najjednostavniji način za korištenja Neo4ja je na lokalnom računalu uz korištenje Neo4j Desktop aplikacije, koja je dostupna za Linux, Windows i MaxOS operativne sustave.

Koraci za instalaciju Neo4j Desktop aplikacije na Windows operativni sustav su sljedeći:

1. Stisnuti link koji vodi na stranicu za preuzimanje: https://neo4j.com/download-center/
2. Pri dnu stranice stisnuti gumb Download Neo4j Desktop.
3. Popuniti obrazac koji zahtijeva podatke o korisniku (ime, prezime, e-mail i slično).
4. Stisnuti gumb Download Desktop.
5. Kopirati aktivacijski ključ. Primjer aktivacijskog ključa:

   eyJhbGciOiJQUzI1NiIsInR5cCI6IkpXVCJ9
   .eyJlbWFpbCI6Ii4rQC4rIiwibWl4cGFuZWxJZCI6IjE4NzFmNzFhZGZkMTIxN
   IlUQWpvNdRlQbZXCG3fE4zIZDo
   ...
   ...

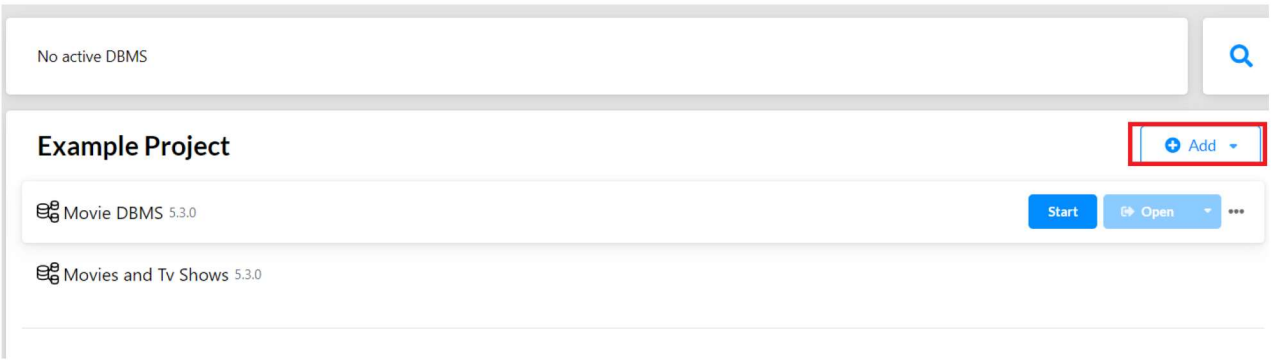
6. Naći instalacijski program, kliknuti na njega i pratiti korake.
7. Prilikom prvog pokretanja aplikacije, potrebno je unijeti aktivacijski ključ koji je kopiran prilikom preuzimanja.
8. Aplikacija će se pokrenuti.

3.4 Učitavanje baze podataka unutar Neo4ja

Kako bi uvezli bazu u Neo4j aplikaciju potrebno je prvo odabrati bazu. Baza može biti preuzeta s interneta ili njezini podaci mogu biti spremljeni u csv formatu. Za uvoz baze u Neo4j potrebno je:

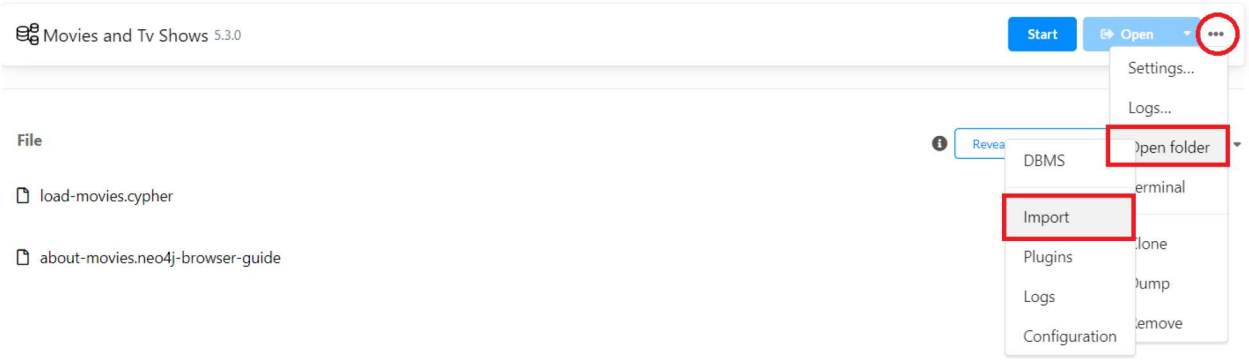
1. Pritisnuti gumb *Add* koji se nalazi pri vrhu aplikacije Neo4j Desktop.
2. U polje namijenjeno za ime baze upisati željeno ime npr. *"Movies and Tv Shows"* te u polje *password* unijeti željenu lozinku.
3. Pritisnuti gumb *Create*.
4. Nakon stvorene baze treba pritisnuti tri točkice u desnom kutu kraj imena baze.
5. Pritisnuti gumb *Import Folder* te zatim *Import*.
6. U nastalu mapu dodati csv datoteku u kojoj je spremljena baza podataka.
7. Pritisnuti gumb *Start*.

Prikazi zaslona odgovarajućih koraka dani su na Slikama 3.4. - 3.7. Nakon uspješnog uvoza baze moguće je njezino daljnje korištenje.

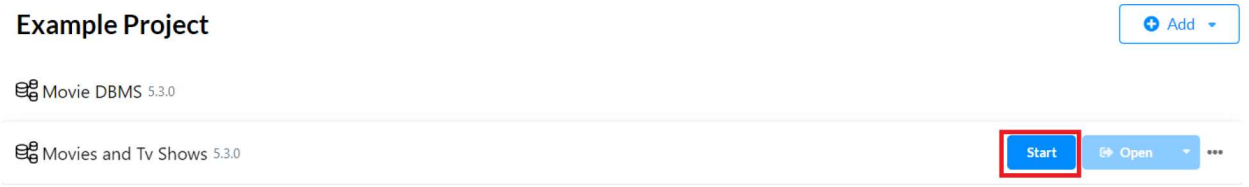


Slika 3.4: Učitavanje baze podataka (Korak 1)

Slika 3.5: Unošenje proizvoljnog imena i lozinke za učitanu bazu (Koraci 2 i 3)



Slika 3.6: Dodavanje csv datoteke (Koraci 4 i 5)



Slika 3.7: Pokretanje baze (Korak 7)

4 | Cypher

Cypher je upitni jezik koji omogućuje pretraživanje, stvaranje i manipulaciju podataka unutar Neo4j baze podataka. Često se koristi iz razloga što je lak za razumjeti i naučiti posebice za one koji razumiju SQL. Osim toga Cypher je jezik otvorenog tipa što bi značilo da je njegov izvorni kod dostupan svim korisnicima koji ga žele poboljšati ili modificirati.

U Cypheru oznaka za svojstvo je {}, za čvor (), dok su veze prikazane s []. Kombinacijom osnovnih upita moguća je izrada složenijih upita. Tri osnovne Chyper naredbe uz pomoć kojih se upravlja Neo4j bazom podataka su CREATE, MATCH i SET.

CREATE je naredba pomoću koje se stvaraju čvorovi i relacije sa svojstvima i bez njih te je identična istoimenoj naredbi u SQL-u.

Primjer 1. *Koristeći CREATE naredbu stvorite čvor TV Show koji sadrži svojstva: id, naslov i godina izdanja.*

Rješenje:

CREATE (ts: TV Show { ID: 's7', naslov: 'Hawkeye', godina_izdanja: '2021', trajanje: 'jedna sezona' })

Na sljedeći način moguće je kreirati više čvorova istovremeno:

CREATE (ts: TV Show { ID: 's7', naslov: 'Hawkeye', godinaizdanja: '2021.', trajanje: 'jedna sezona'}), (di: Director { ime:'John', prezime: 'Cherry'})

Kreiranje veze je moguće pomoću naredbe CREATE na sljedeći način:

CREATE (di: Director)-[VODI]->(tv: Tv Show)

MATCH je naredba koja omogućava određivanje obrazaca koje će Neo4j pretražiti u bazi podataka.

Primjer 2. *Stvorite vezu između određenih TV emisija.*

Rješenje:

MATCH(ts1: TV Show{naslov: 'Hawkeye'}), (ts2:TV Show{naslov: 'Olaf Presents'})
CREATE(ts1)-[IMAJU ISTOG DIREKTORA]->(ts2) RETURN a, b

U naredbama korištenim u Primjeru 2. pomoću MATCH naredbe pronađeni su čvorovi koji zadovoljavaju određeni uzorak (imaju naslove 'Hawkeye' i 'Olaf Presents'), a zatim je pomoću naredbe CREATE kreirana veza s nazivom 'imaju istog direktora' između ta dva čvora.

Naredba SET služi za ažuriranje svojstava čvorova i veza.

Primjer 3. *Ažuriraj trajanje TV emisije.*

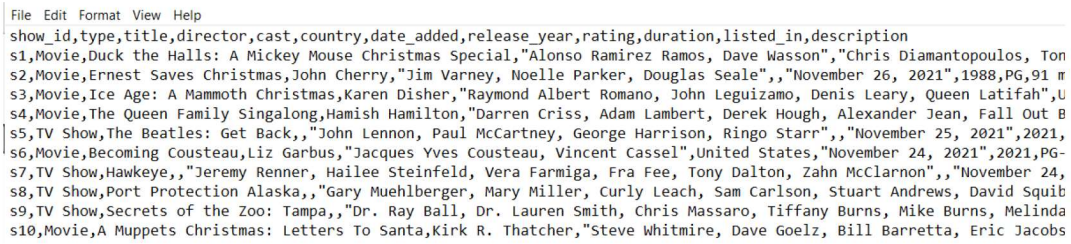
Rješenje:

MATCH (ts: TV Show{naslov: 'Hawkeye'}) SET ts.trajanje = 'dvije sezone' RETURN ts.naslov, ts.trajanje

U Primjeru 3. pomoću MATCH naredbe pronađen je čvor koji zadovoljava uzorak (ima naslov 'Hawkeye') te je uz pomoć SET naredbe ažurirano svojstvo trajanje na 'dvije sezone', zatim je naredbom RETURN ispisan naslov i trajanje zadanog čvora.

4.1 Učitavanje CSV podataka u Neo4j sa Cypherom

Format datoteke čije su vrijednosti odvojene zarezom nazivamo CSV[1] datoteka. Primjer jedne csv datoteke dan je na Slici 4.1. Takav format datoteke čini više od 57% skupova svih podataka, dok JSON datoteke čine manje od 10%.



Slika 4.1: Primjer CSV datoteka

LOAD CSV je naredba, pomoću koje uvozimo podatke iz CSV datoteke u Neo4j, koja sadrži FROM dio u kojem je specificirano gdje se nalazi datoteka koju treba učitati te nakon nje stoji dio u kojem se precizira što je potrebno učiniti s podacima koji se dohvaćaju. Na Slici 4.2. u nastavku možemo vidjeti primjer uvoza podatka o državi (engl. *country*) u bazu.

```
1 LOAD CSV WITH HEADERS
2 FROM 'file:///disney_plus_titles.csv' AS row
3 WITH split(row.country, ",") as country_list
4 UNWIND country_list AS country_name
5 MERGE (:Country{countryName:country_name})
6
```

Slika 4.2: Primjer uvoza podataka o državi iz csv datoteke

[1] engl. *comma-separated values* - CSV

5 | Ispitivanje i karakterizacija graf baze podataka pomoću Neo4ja

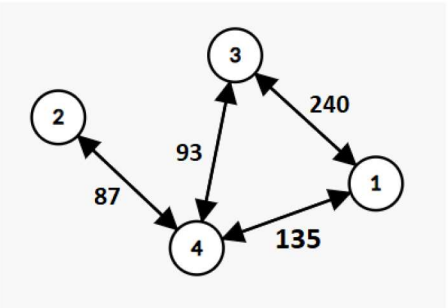
5.1 Karakterizacija grafa

Graf je objekt koji se sastoji od vrhova ili čvorova i bridova ili stranica. Grafovi mogu biti neusmjereni i usmjereni te težinski i bestežinski. Ako su u grafu svi bridovi usmjereni, tada je taj graf usmjeren, u suprotnom je neusmjeren. Kod težinskih grafova svaki brid ima pridruženu težinu, dok bestežinski grafovi nemaju težinu na bridovima. Grafovi se koriste za opisivanje veza između dva čvora iz određene kategorije.

5.1.1 Težina veze

Težina veze (engl. *link weight*) se određuje tako da provjerimo nose li bridovi obilježje koje kvantificira snagu veze među čvorovima.

Primjer 4. *Cestovnu mrežu možemo prikazati pomoću težinskog grafa prikazanog na Slici 5.1. gdje vrhovi predstavljaju gradove, a težinu bridova predstavlja udaljenost između gradova u kilometrima.*



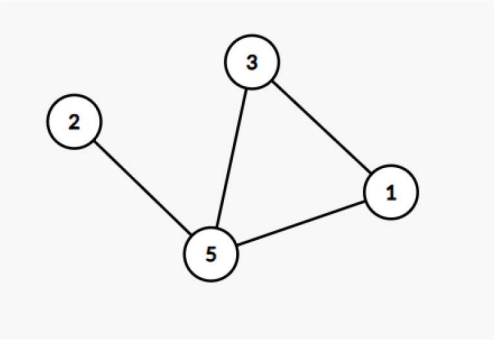
Slika 5.1: Primjer težinskog grafa

5.1.2 Smjer veze

Razlikujemo usmjerene (lukovi) i neusmjerene (bridovi) veze između dva čvora. U teoriji postoji jasna razlika između ove dvije vrste veza, no u graf bazama podataka sve se veze nazivaju bridovi ili jednostavnije samo veze. Smjer veze (engl.

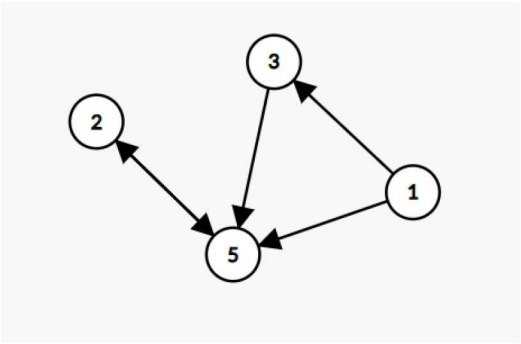
link direction) se može koristiti u Neo4j bazi kod upita, a obavezan je kod kreiranja veza između čvorova.

Primjer 5. *Dodavanje poznanika na društvenoj mreži Facebook grafički možemo prikazati kao što je na Slici 5.2. pri čemu vrhovi označeni s brojevima označavaju korisnike, odnosno njihove profile na društvenoj mreži. Oba korisnika kod stvaranja prijateljstva pristaju na međusobno praćenje njihovih aktivnosti na društvenoj mreži Facebook što bi predstavljalo neusmjeren graf.*



Slika 5.2: Primjer neusmjerenog grafa

Primjer 6. *Usmjeren graf možemo predstaviti pomoću društvene mreže Twitter kao što je prikazano na Slici 5.3. gdje slično kao u prethodnom primjeru čvor označava korisnika odnsosno njegov profil. Na ovoj društvenoj mreži korisnik ima mogućnost pratiti nekoga tko njega ne prati i obrnuto. Na primjer, na Slici 5.3. vidimo da korisnik 1 prati korisnike 3 i 5, dok korisnici 3 i 5 ne prate korisnika 1. S druge strane, korisnici 2 i 5 se međusobno prate.*



Slika 5.3: Primjer usmjerenog grafa

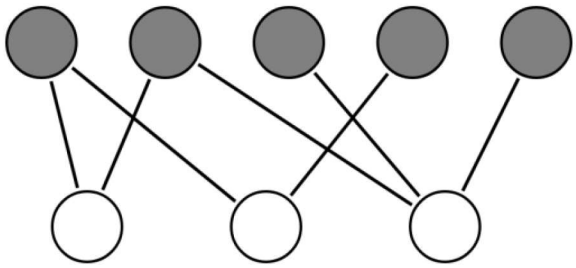
5.1.3 Distribucija stupnjeva grafa

Distribucija stupnjeva grafa (engl. *graph degree distribution*) je broj bridova povezanih s tim čvorom. Kod usmjerenih grafova postoje izlazni, dolazni i ukupni stupanj čvora dok kod neusmjerenih grafova postoji samo jedan stupanj. Dolazni stupanj je broj bridova koji ulaze u čvor, izlazni stupanj predstavlja broj bridova koji izlaze iz čvora, a ukupni stupanj čvora prebrojava sve bridove spojene sa čvorom.

5.1.4 Vrste čvorova

Vrsta čvora (engl. *node type*) je skup čvorova koji posjeduju istu svrhu. Grafovi koji sadrže samo čvorove iste vrste u bazi podataka nazivamo homogeni grafovi, a grafovi koji sadrže različite vrste čvorova nazivaju se heterogeni grafovi. Homogeni grafovi u Neo4ju imaju jednu oznaku čvorova, a heterogeni grafovi imaju više oznaka čvorova.

Specifična vrsta heterogenog grafa naziva se bipartitni graf te je primjer jednog takvog grafa dan na Slici 5.4. Bipartitni graf sadrži dvije vrste čvorova. Vezu je moguće uspostaviti jedino između dva čvora različite vrste.



Slika 5.4: Grafički prikaz jednog bipartitnog grafa

Primjer 7. *E-trgovina koja sadrži čvorove proizvod i korisnik koji su povezani vezom kupovina predstavlja bipartitni graf jer ne postoje veze između dva proizvoda ili dva korisnika, samo između korisnika i proizvoda.*

5.2 GDS biblioteka

Graph Data Science (GDS) je biblioteka predstavljena 2020. godine. GDS biblioteka sadrži alate koji se koriste u podatkovnoj znanosti kao što su algoritmi vezani za modele, graf algoritme, algoritmi vezani za put, Python klijent i cjevovode strojnog učenja. U smislu standardizacije i performansi GDS biblioteka je poboljšana kroz vrijeme, a dodano je i puno novih značajki u pogledu parametrizacije algoritama i novih vrsta algoritama.

5.2.1 Algoritmi

Algoritme centraliteta upotrebljavamo kako bi razumjeli uloge pojedinih čvorova u grafu i njihov značaj za taj graf. Ovi algoritmi se koriste za identifikaciju najvažnijih čvorova i omogućuju shvaćanje grupne dinamike, kao što je vjerodostojnost, pristupačnost, mostovi između grupa i brzina kojom se stvari šire.

PageRank algoritam

PageRank se u početku upotrebljavao za rangiranje web stranica na osnovi rezultata pretraživanja korisnika. Ovaj algoritam unutar grafa promatra neophodnost svakog pojedinog čvora. Na temelju broja dolaznih veza i važnosti izvornih čvorova algoritam PageRank mjeri važnost svakog čvora u grafu. Osnovna tvrdnja jest da je stranica bitna onoliko koliko su bitne stranice koje ukazuju na nju.

Algoritam središnje centralnosti

Algoritam središnje centralnosti (engl. *Betweenness centrality*) se upotrebljava za otkrivanje koliki utjecaj ima čvor na protok informacija unutar grafa. Koristi se često prilikom pronalaska čvorova koji se upotrebljavaju kao mostovi za prijelaz s jednog dijela na drugi dio grafa. Osim za pronalazak mostova koristi se i za izračunavanje najkraćeg puta između parova čvorova unutar grafa. Implementiran je za grafove čiji bridovi nemaju težine ili su težine pozitivne. Implementacija GDS-a zasnovana je na Dijkstra algoritmu za težinske grafove, dok se za ne težinske grafove upotrebljava Brandesov približni algoritam.

Algoritam stupnja centralnosti

Algoritam stupnja centralnosti (engl. *Degree centrality*) se može koristiti na netežinskim, ali i na težinskim grafovima u Neo4ju. Ovaj algoritam prebrojava broj odlaznih ili dolaznih bridova iz čvora.

Svojstvo mnogih grafova su zajednice[1], neki grafovi mogu imati i više zajednica te čvorovi u zajednici mogu biti gusto povezani. Tehnike za otkrivanje zajednica koriste se za algoritme društvenih mreža s ciljem otkrivanja ljudi čiji su interesi zajednički. Osim toga, otkrivanje zajednica korisno je u strojnom učenju za izdvajanje grupa i za pronalazak grupa sa sličnim svojstvima.

Louvain algoritam

Louvain algoritam je algoritam za pronalazak zajednica unutar velikih mreža koji za svaku zajednicu maksimizira ocjenu modularnosti. *Louvain* algoritam je hijerarhijski algoritam za grupiranje koji rekurzivno spaja zajednice u jedan čvor i obavlja modularno klasteriranje na tako sažetom grafu. Modularnost kvantificira kvalitetu dodjele čvora zajednici, odnosno to znači da procjenjuje koliko su gusto

[1] Zajednice - manji dijelovi grafa koji su međusobno bolje povezani nego što su povezani s ostatkom grafa.

vezani čvorovi u zajednici, u usporedbi s time kako bi bili povezani unutar slučajne mreže.

Broj trokuta

Trokut je definiran s tri povezana čvora. Broj trokuta (engl. *Triangle count*) za čvor n pronalazimo tako da provjeravamo njegove susjede i jesu li oni također povezani s drugim susjedima od n. Unutar GDS biblioteke algoritam pronalazi samo trokute unutar neusmjerenih grafova. Broj trokuta se često koristi prilikom analize društvenih mreža gdje se upotrebljava za otkrivanje zajednice.

Slabo povezane komponente

Algoritam slabo povezanih komponenti (engl. *Weakly Connected Components* (*WCC*)) se koristi za pronalaženje skupova povezanih čvorova u neusmjerenim i usmjerenim grafovima. Za dva čvora između kojih postoji put kažemo da su povezani. Komponentu čini skup svih čvorova koji su povezani. Algoritam slabo povezanih komponenti između dva čvora ne promatra smjer odnosa za razliku od algoritma za pronalazak snažno povezanih komponenti. Algoritam za pronalazak slabo povezanih komponenti često je upotrebljavan na samom početku analize u svrhu lakšeg tumačenja strukture grafa. Naime u GDS-u, unutar grafa bez nepovezanih čvorova i dalje je moguće stvaranje grupa ili klastera.

Sličnost čvorova

Algoritam sličnosti čvorova (engl. *Node similarity*) uspoređuje skup čvorova na osnovi čvorova s kojima su čvorovi u tom skupu povezani. Ako dva čvora dijele puno istih susjeda tada se oni smatraju sličnim. Na temelju Jaccardove metrike[2] računamo sličnost čvorova u paru.

Filtrirana sličnost čvorova

Algoritam filtrirane sličnosti čvorova (engl. *Filtered Node similarity*) predstavlja proširenje algoritma sličnosti čvorova. Algoritam pridodaje filter na ciljnim čvorovima[3], izvornim čvorovima ili oba čvora.

K najbližih susjeda

K najbližih susjeda (engl. *K-Nearest Neighbors*) je algoritam pomoću kojega računamo udaljenost za sve parove čvorova i stvaramo nove odnose među čvorovima i njegovih k susjeda. Na temelju obilježja čvorova računamo udaljenost.

Unutar raznih primjena (npr. logistika, videoigre, robotika) ključna komponenta su algoritmi za pronalazak puta. Pronalazak najučinkovitijeg ili najkraćeg puta između dvije točke pomaže ljudima za efikasnu navigaciju.

[2]Jaccardova metrika - mjeri različitost između skupova, a dobiva se dijeljenjem razlike veličina unije i presjeka dva skupa s veličinom unije.

[3]Ciljni čvor - čvor u grafu koji zadovoljava definirane kriterije

A* najkraći put

Pomoću algoritma A* najkraći put (engl. *A* Shortest Path*) pronalazimo najkraći put između dva čvora. Ovaj algoritam upotrebljava heurističku funkciju za obilazak po grafu, osim toga podržava težinske grafove s težinama koje su pozitivne.

Minimalno razapinjuće stablo

Algoritam minimalno razapinjujućeg stabla (engl. *The minimum weight spanning tree*) je razapinjuće stablo[4] čiji je zbroj težina bridova što manji. Najpoznatiji i najjednostavniji ovakav algoritam je Primov algoritam, osim njega slično djeluje i Dijkstrin algoritam.

Najkraći put svih parova

Algoritam najkraćeg puta svih parova (engl. *All pairs shortest path*) računa najkraći put između svih parova čvorova. Algoritam najkraćeg puta svih parova je brži za razliku od algoritma najkraćeg puta iz jednog početnog čvora.

5.2.2 Cjevovodi i modeli strojnog učenja

Proces prelaska visokodimenzionalnog objekta (npr. graf, tekst ili slika) u objekt nižih dimenzija (npr. vektor) uz očuvanje ključnih značajki naziva se ugrađivanje (engl. *embedding*).

Osim implementacije vrlo korisnih algoritama za ugradnju, GDS biblioteka može spremiti utrenirane modele u katalog modela kako bi se model kasnije mogao koristiti kod predviđanja. Osim toga pomoću GDS biblioteke možemo stvoriti cjevovode[5] i spremati ih u katalog za kategorizaciju čvorova.

GDS biblioteka se može koristiti u Cypheru korištenjem CALL procedura. Python klijent omogućava da se bez pisanja Cypher upita poziva GDS procedura.

5.2.3 Projiciranje grafa

Graf koji se upotrebljava prilikom pokretanja svih algoritama iz GDS-a se naziva projicirani graf. Sve veze, svojstva i čvorovi projiciranog grafa mogu se konfigurirati. Dva načina za kreiranje projiciranih grafova su izvorna projekcija i Cypher projekcija.

Izvorna projekcija sadrži čvorove, veze i svojstva koja se odabiru iz Neo4j baze podataka sa standardiziranom konfiguracijom.

[4]Razapinjuće stablo - je podgraf grafa koji se sastoji od vrhova i bridova tako da postoji put između svaka dva vrha.

[5]Cjevovodi - (engl. *pipelines*) skup serijski povezanih elemenata za obradu podataka, pri čemu je izlaz jednog elementa ulaz sljedećeg elementa.

Cypher projekcija sadrži entitete koji se izdvajaju iz Neo4ja ili se stvaraju usputno, pomoću Cypher upita.

Druga ključna značajka je mogućnost vraćanje rezultata algoritama na nekoliko načina. Jednostavni način provođenja, gdje se učinci algoritama arhiviraju u memoriju i odlaze ka korisniku naziva se tok. Rezultat možemo vratiti tako da se on ne arhivira u memoriji već da se zapisuje natrag u Neo4j kao nova veza ili svojstvo čvora. Mutacije se koriste kada Neo4j graf nije ažuriran, ali projicirani graf je, što je korisno za ulančavanje algoritama. Statistički način rada (engl. *stats mode*) ne vraća zasebni rezultat za svaki pojedini čvor, nego vraća sveukupnu statistiku za algoritam kao što je distribucija izračunate metrike i vrijeme izvođenja.

6 | Upotreba Neo4j GDS-a na bazi podataka *Filmovi i TV emisije*

6.1 Opis baze podataka *Filmovi i TV emisije*

Disney+ jedna je od najpopularnijih platformi za prijenos medija i videa. Trenutno imaju blizu 1300 filmova ili TV emisija dostupnih na svojoj platformi te od 2021. godine imaju preko 116 milijuna pretplatnika diljem svijeta. Ova baza podataka sastoji se od popisa svih filmova i TV emisija s Disney+ platforme.

6.1.1 Kreiranje veza i čvorova te uporaba algoritama

Kako bismo mogli manipulirati s bazom podataka i raditi s algoritmima potrebno je stvoriti čvorove.

Čvorovi koji se nalaze u bazi podataka *Filmovi i TV emisije* su:

1. Filmovi i TV emisije (Movies i TV shows) koji će imati sljedeća svojstva: show_id, godina izlaska (release_year), naziv (title),
2. Redatelji i glumci (People) koji će sadržavati svojstvo: ime (name),
3. Trajanje (Duration) koji će sadržavati svojstvo: trajanje (duration_time),
4. Ocjene (Rating) koji će sadržavati svojstvo: ocjena (rating),
5. Država (Country) koji će sadržavati svojstvo: država (country),
6. Žanr (Genre) koji će sadržavati svojstvo: žanr (genre).

Na Slici 6.1. vidimo primjer stvaranja čvora TV Show.

```
1 LOAD CSV WITH HEADERS FROM 'file:///disney_plus_titles.csv' AS row
2 WITH row WHERE row.type = 'TV Show'
3 MERGE (s:TvShow{show_id:
  row.show_id,title:row.title,release_year:row.release_year})
4 RETURN count(s);
```

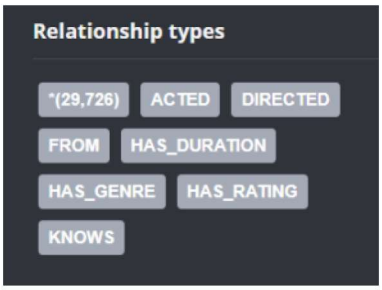
Slika 6.1: Kreiranje čvora TV Show u bazi Filmovi i TV emisije

Da bi graf baze podataka imao smisla neophodno je stvaranje veza među čvorovima. Primjer naredbi za kreiranje veza u ovoj bazi prikazani su na na Slici 6.2.

```
1  LOAD CSV WITH HEADERS
2  FROM 'file:///disney_plus_titles.csv' AS row
3  MATCH (m:Movies{show_id:row.show_id})
4  WITH m, split(row.director, ",") as directorm_list
5  UNWIND directorm_list AS directorm_name
6  MATCH (d:People {name: directorm_name})
7  CREATE (m)<-[:DIRECTED]-(d)
8
9  LOAD CSV WITH HEADERS
10 FROM 'file:///disney_plus_titles.csv' AS row
11 MATCH (m:Movies{show_id:row.show_id})
12 WITH m, split(row.cast, ",") as actor_list
13 UNWIND actor_list AS actor_name
14 MATCH (d:People {name: actor_name})
15 CREATE (m)<-[:ACTED]-(d)
```

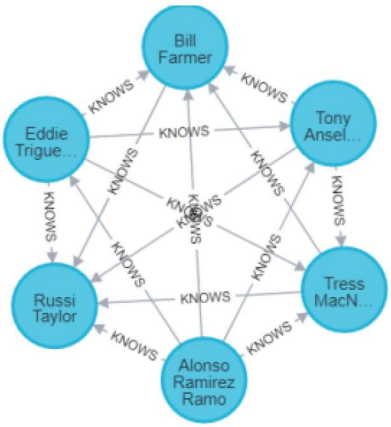
Slika 6.2: Upiti za stvaranje veza REŽIRA (engl. *DIRECTED*) i GLUMI (engl. *ACTED*) u bazi Filmovi i TV emisije

Pritiskom na oznaku veze moguće je prikazati uspješno stvorene veze. Prikaz uspješno stvorenih veza za ovu bazu dan je na Slici 6.3.



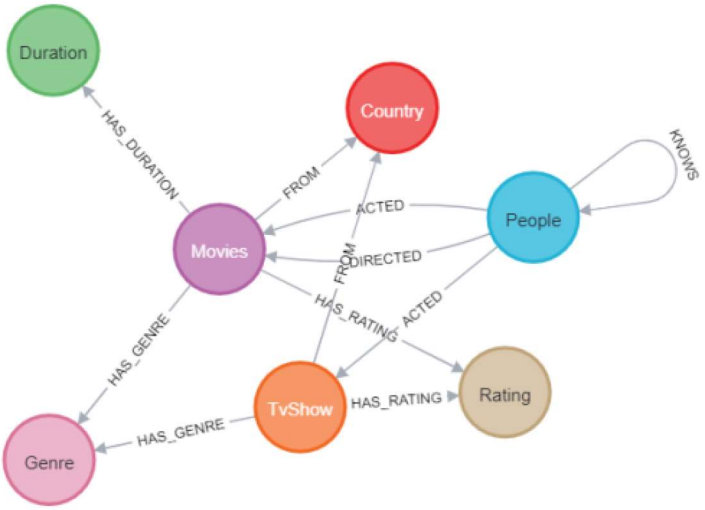
Slika 6.3: Popis uspješno stvorenih veza u bazi Filmovi i TV emisije

Grafički prikaz uspješno stvorene veze KNOWS vidimo na Slici 6.4.



Slika 6.4: Grafički prikaz veze POZNAJE (engl. *KNOWS*) u bazi Filmovi i TV emisije

Ako želimo grafički prikazati cijeli graf kao na Slici 6.5. to možemo učiniti naredbom CALL db.schema.visualization().



Slika 6.5: Grafički pregled grafa baze podataka Filmovi i TV emisije

U trenutku kada smo stvorili graf moguće je započeti s primjenom algoritama kako bismo saznali bitne informacije za bazu podataka.

Algoritam središnje centralnosti

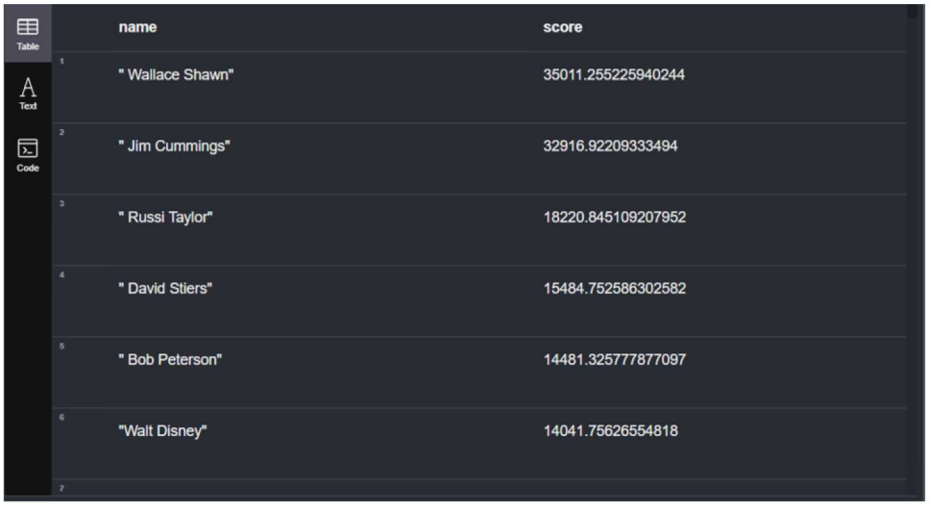
Primjenom ovog algoritma na kreiranu bazu možemo pronaći osobe koje imaju najviše veza s drugim osobama u grafu, odnosno osobe koje smatramo najpoznatijima. U slučaju algoritma središnje centralnosti što je broj veza veći to je čvor (osoba) važnija.

Projiciramo najprije graf te zatim pozovemo željeni algoritam kako je prikazano na Slici 6.6.

```
1 CALL gds.graph.project('PeopleGraph', 'People', {KNOWS: {properties: 'weight'}})
2
3 CALL gds.betweenness.stream('PeopleGraph', {relationshipWeightProperty: 'weight'})
4 YIELD nodeId, score
5 RETURN gds.util.asNode(nodeId).name AS name, score
6 ORDER BY score DESC
7
```

Slika 6.6: Upotreba algoritma središnje centralnosti za pronalazak "najvažnijih" osoba u bazi Filmovi i TV emisije

Pogledavši rezultate sa Slike 6.7., vidljivo je, da u našem grafu u prvih pet "najvažnijih" osoba spadaju: Wallace Shawn, Jim Cummings, Russi Taylor, David Stiers i Bob Peterson.



	name	score
1	" Wallace Shawn"	35011.255225940244
2	" Jim Cummings"	32916.92209333494
3	" Russi Taylor"	18220.845109207952
4	" David Stiers"	15484.752586302582
5	" Bob Peterson"	14481.325777877097
6	"Walt Disney"	14041.75626554818
7		

Slika 6.7: Rezultati upotrebe algoritma središnje centralnosti na bazi Filmovi i TV emisije

Broj trokuta

Zamislimo da čvorovi predstavljaju korisnike društvene mreže, a bridovi predstavljaju prijateljstvo. Ako je korisnik A prijatelj s korisnicima B i C te tada postoji određena vjerojatnost da se korisnici B i C poznaju.

U našem primjeru grafa želimo iskoristiti algoritam broj trokuta, prikazano na Slici 6.8., kako bi proučili moguća nova prijateljstva. Na primjer, ako imamo tri čvora Marko, Maja i Iva koji predstavljaju ljude i Marko poznaje Maju i Ivu, onda postoji mogućnost da se Maja i Iva također poznaju.

Dakle, ponovno napravimo projekciju, no ovog puta stavimo da su veze neusmjerene.

```
1 CALL gds.graph.project(  'PeopleGraph','People',{ KNOWS: {orientation: 'UNDIRECTED'} })
2
3 CALL gds.triangleCount.stream('PeopleGraph')
4 YIELD nodeId, triangleCount
5 RETURN gds.util.asNode(nodeId).name AS name, triangleCount
6 ORDER BY triangleCount DESC
7
```

Slika 6.8: Upotreba algoritma broj trokuta na bazi Filmovi i TV emisije

U rezultatima prikazanim na Slici 6.9. vidljivo je da Jim Cummings ima najveći broj trokuta, preciznije 261, dok primjerice John Kahrs nema niti jedan trokut.

Slika 6.9: Rezultati upotrebe algoritma broj trokuta na bazi Filmovi i TV emisije

Slabo povezane komponente

Ako želimo pronaći izolirane grupe ljudi, odnosno grupe čvorova koji nisu povezani s ostatkom grafa koristimo algoritam slabo povezanih komponenti. Kod primjene algoritma može se postaviti prag za vrijednost težine, kako bi algoritam uzeo u obzir samo one veze čija je težina veća od vrijednosti praga. To činimo navođenjem vrijednosti praga s konfiguracijskim parametrom praga. Ako odnos nema navedeno svojstvo težine, algoritam se vraća na upotrebu zadane vrijednosti nula, odnosno radi na netežinskom grafu. Primjer upotrebe algoritma za pronalazak slabo povezanih komponenti prikazan je na Slici 6.10.

```
1 CALL gds.graph.project( 'PeopleGraph', 'People', 'KNOWS',  {relationshipProperties: 'weight'})
2
3 CALL gds.wcc.stream('PeopleGraph', {
4   relationshipWeightProperty: 'weight',
5   threshold: 1.0
6 }) YIELD nodeId, componentId
7 RETURN gds.util.asNode(nodeId).name AS Name, componentId AS ComponentId
8 ORDER BY ComponentId,Name
9
```

Slika 6.10: Upotreba algoritma slabo povezanih komponenti na bazi Filmovi i TV emisije

Algoritam za svaki čvor vraća broj, vrijednost *componentId*, koja predstavlja zajednicu kojoj čvor pripada. Pogledajmo rezultate sa Slike 6.11. gdje uočavamo da osobe Jennifer Hale, Jim Cummings i Wayne Allwine predstavljaju potpuno izolirane čvorove.

	Name	ComponentId
32	"Jennifer Hale"	0
33	"Jim Cummings"	0
34	"Wayne Allwine"	0
35	"John Cherry"	2
36	"Karen Disher"	3
37	"Hamish Hamilton"	4

Slika 6.11: Rezultati upotrebe algoritma slabo povezanih komponenti na bazi Filmovi i TV emisije

Filtrirana sličnost čvorova

Pogledajmo dva rezultata sličnosti:

$$A = (\text{Alice})\text{-}[:\text{SIMILAR\_TO}]\rightarrow(\text{Bob}) \text{ i } B\ (\text{Bob})\text{-}[:\text{SIMILAR\_TO}]\rightarrow(\text{Alice}).$$

Rezultat A će se dobiti ako čvor (Alice) odgovara filteru izvornog čvora, a čvor B (Bob) odgovara filteru ciljanog čvora. Ako se čvor A (Alice) ne podudara s filtrom ciljanog čvora B ili čvor B (Bob) ne odgovara filtru izvornog čvora, rezultat B neće biti vraćen.

U našem primjeru sa Slike 6.12. želimo vidjeti osobe koje su najmanje glumile u istim serijama pa projiciramo graf i pozovemo algoritam.

```
1 CALL gds.graph.project( 'PeopleTV', ['People','TvShow'], 'ACTED')
2
3 CALL gds.alpha.nodeSimilarity.filtered.stream('PeopleTV' )
4 YIELD node1, node2, similarity
5 RETURN gds.util.asNode(node1).name AS Person1, gds.util.asNode(node2).name AS Person2, similarity
6 ORDER BY similarity ASC, Person1, Person2
7
```

Slika 6.12: Upotreba algoritma za filtriranje sličnosti čvorova na bazi Filmovi i TV emisije

Iz rezultata danih na Slici 6.13. vidljivo je da su Brenda Song i Bill Farmer te Russi Taylor i Corey Burton glumili u najmanjem broju istih serija.

	Person1	Person2	similarity
1	" Brenda Song"	" Bill Farmer"	0.09090909090909091
2	" Russi Taylor"	" Corey Burton"	0.09090909090909091
3	" Christine Cavanaugh"	" Jim Cummings"	0.1
4	" Chuck McCann"	" Jim Cummings"	0.1
5	" Daniel Ross"	" Corey Burton"	0.1
6	" James Avery"	" Jim Cummings"	0.1
7			

Slika 6.13: Rezultati upotrebe algoritma za filtriranje sličnosti čvorova na bazi Filmovi i TV emisije

6.2 Česte greške i loša praksa

Prilikom pokretanja algoritma na grafu, algoritam promatra svaki čvor i njegove veze te koristeći te veze prelazi na susjedne čvorove. Čvor koji nema susjeda, odnosno nema nikakvih bridova se zove izolirani čvor. Takva vrsta čvorova nam najčešće nije od koristi. Prilikom pronalaska izoliranog čvora takav čvor se preskoči i algoritam nastavi s radom nad preostalim čvorovima. Ako su prilikom pokretanja algoritma svi čvorovi ostali na svojim početnim pozicijama u grafu, tj. ako su inicijalizirani u nule, tada veze unutar grafa ne postoje.

Kada se algoritam pokrene svi čvorovi su smješteni u jednu veliku zajednicu. Kažemo da je graf gusto povezan ako postoji velik broj veza te je tada algoritmu gotovo nemoguće odabrati mjesto razbijanja te veze. Kako bi se riješio takav problem mogu se koristiti težine ili pragovi kako bi se veze koje nisu zanimljive samo preskočile. Umjesto korištenja samo veza, koristimo i heuristiku kako bismo zanemarili neke veze u grafu te tako smanjili gustoću veza i omogućili algoritmu da razbija veze.

Algoritam prilikom pokretanja ne može znati da čvorovi koji se promatraju ustvari projiciraju stvari iz života. Pokretanjem algoritma za pronalaženje zajednica, to želimo učiniti na stvarima koje projiciraju istu stvar. Ako pokrenemo algoritam na hrpi čvorova različitog tipa i pustimo da ih algoritam gleda kao istu stvar, algoritam nas neće upozoriti da su to različiti tipovi.

Ako algoritam pokrenemo dva puta nad istim podacima i on nam vrati različita rješenja postoji mogućnost da je došlo do greške ili da jednostavno nije smislena

upotreba tog algoritma. Priličan broj algoritama je stohastičan, tj. to su algoritmi koji koriste heuristiku koja nije deterministička. Kada algoritam dođe u situaciju da mora odlučiti hoće li poći desno ili lijevo, prilikom prvog pokretanja može poći lijevo, a prilikom drugog pokretanja desno. Zbog toga kod stohastičnog algoritma, nemamo jamstvo da ćemo njegovim ponovnim pokretanjem dobiti isti rezultat.

Postoje algoritmi koji su preskupi, odnosno imaju dugo vrijeme izvršavanja. Primjeri ovih algoritama su: algoritam sličnosti čvora, središnje centralnosti ili bilo koji algoritam sličnosti. Postoje konfiguracijske opcije u slučaju da je algoritam preskup, kako bi mogli nadzirati koliko vremena će trebati da algoritam završi. Problem uvijek možemo rastaviti, na primjer ako imamo veliki graf, ali ustvari nama treba samo jedan zanimljiviji dio tada algoritam primjenimo na tom dijelu, a ne na cijelom grafu. Postoje brojni algoritmi koji nam mogu pomoći otkriti koji je to dio, kao što je naprimjer algoritam slabo povezanih komponenti.

Čuvar memorije (engl. *memory guard*) je definiran unutar sustava kako bi spriječio da dođe do urušavanja baze podataka. Ako se algoritam nije pokrenuo zbog čuvara memorije to znači da će se prilikom pokretanja algoritma prekoračiti memorijska granica te će se samim time sustav urušiti. Moguće je isključiti to svojstvo, no nije preporučljivo.

Neka od mogućih posljedica isključivanja čuvara memorije je da JVM ostane bez memorije, no to je moguće da se dogodi i kada je on uključen. Preporučljivo je brisati nekorištene veze, svojstva i grafove kako bismo oslobodili prostor u memoriji za algoritme kojima je jednostavno neophodno puno memorije.

Kako bismo spriječili moguće konflikte kod konfiguracije koristimo procjenu memorije. Potrebno je postaviti da Neo4j koristi hrpu (engl. *heap*) što je više moguće te pokretati algoritame na jednoj instanci, a ne na klasteru. Kod projekcije grafa treba učitati samo veze i čvorove koje ćemo koristiti. Katalog[1] koristimo ako na istom grafu imamo pokrenuto više algoritama te se tada brišu grafovi koji nam više nisu potrebi.

[1]Katalog - repozitorij koji prati konfiguracijske podatke i status vremena izvođenja određene konfiguracije.

Popis slika

Literatura

[1] T. Cormen, dr. C. Leiserson, R. Rivest, C. Stein, *Introduction to Algorithms Fourth Edition*, The MIT Press, Cambridge, Massachusetts London, England, 2022.

[2] J. Depeau, dr. A. Frame, L. Gannon, *Neo4j Graph Data Science Configuration Guide*, Neo4j Inc., 2022.

[3] dr A. Frame, Z. Blumenfeld, *Graph Data Science For Dummies, Second Edition*, John Wiley & Sons, 2021.

[4] C. Kemperl, *Beggining Neo4j*, Apress, 2015.

[5] M. Needham, A.E. Holder, *Graph Algorithms*, Neo4j Inc., 2023.

[6] Neo4j Team, *The Neo4j Graph Algorithms User Guide v3.4*, Neo4j Inc., 2019.

[7] Neo4j Team, *Graph Data Science Use Case Selection Guide*, O'Reilly Media, 2019.

[8] E. Scifo, *Graph Data Science with Neo4j*, Packt Publishing, 2023.

[9] dr. J. Webber, R.B. Brugen, *Graph Databases For Dummies*, John Wiley & Sons, 2020.

[10] Web izvor dostupan na https://www.youtube.com/watch?v=TmVlq33u6l4&list=PL9Hl4pk2FsvWNjrgm5xR47x70nNxaCmGI&index=4&t=749s&ab_channel=Neo4j (*Graph Data Science worst practices*)

[11] Web izvor dostupan na https://www.youtube.com/watch?v=spmLJ3AHMPY&ab_channel=Neo4j (*Neo4j Live: Graph Data Science 2.0*)

[12] Web izvor dostupan na https://www.youtube.com/watch?v=APOGvq7P2gY&list=PL9Hl4pk2FsvWNjrgm5xR47x70nNxaCmGI&ab_channel=Neo4j (*Graph Embeddings with the Graph Data Science Library*)

[13] Web izvor dostupan na https://neo4j.com/docs/graph-data-science/current/ (*Neo4j Graph data science dokumentacija*)

Sažetak

U ovom završnom radu istražit ćemo kako primijeniti graf bazu podataka u podatkovnoj znanosti, s posebnim naglaskom na Neo4j bazu podataka i Neo4j Graph Data Science biblioteku. U prvom dijelu upoznat ćemo se s graf bazama podataka, nakon čega ćemo se više upoznati s Neo4j sustavom kroz njegovu povijest, ekosustav i postupak instalacije na osobno računalo. Zatim, upoznat ćemo se s upitnim jezikom Cypher pomoću kojeg je moguće pretraživati, kreirati i manipulirati podacima u graf bazama podataka. Koristeći Cypher kreirat ćemo bazu podataka *"Filmovi i TV emisije"* te na nju primijeniti neke od algoritama strojnog učenja. Na samom kraju, osvrnut ćemo se na česte greške koje se javljaju tijekom primjene algoritama strojnog učenja na graf bazama podataka.

Ključne riječi

graf baza podataka, Neo4j, podatkovna znanost, Cypher, algoritmi strojnog učenja

The use of machine learning algorithms on graph database *Movies and TV Shows*

Summary

In this final paper, we will explore how to apply a graph database in data science, focusing on Neo4j Graph Data Science and Neo4j Graph Data Science library. In the first part, we will get acquainted with graph databases, followed by an introduction to the Neo4j system through its history, ecosystem, and ins- tallation procedure on a personal computer. Then, we will learn about the query language Cypher, which allows us to search, create, and manipulate data in graph databases. Using Cypher, we will create a database called "Movies and TV Shows" and apply some machine learning algorithms to it. Finally, we will discuss common errors that occur while applying machine learning algorithms to graph databases.

Keywords

graph databases, Neo4j, data science, Cypher, machine learning algorithms

Životopis

Rođena sam u Slavonskom Brodu 09. 08. 2001. godine. Svoje obrazovanje započinjem 2008. godine u Osnovnoj školi "Vladimir Nazor" u Slavonskom Brodu. Nakon osnovnoškolskog obrazovanja srednjoškolsko obrazovanje započinjem 2017. godine u Gimnaziji "Matija Mesić" u Slavonskom Brodu te 2020. godine upisujem Preddiplomski studij Matematike i računarstva na Odjelu za matematiku, sadašnjem Fakultetu primijenjene matematike i informatike.