Primjena algoritama strojnog učenja na graf bazi "Osnivači i startupi"

Glamočak, Nina

Undergraduate thesis / Završni rad

2023

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, School of Applied Mathematics and Informatics / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet primijenjene matematike i informatike**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:126:239828

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij Matematika i računarstvo

Primjena algoritama strojnog učenja na graf bazi *Osnivači i startupi*

Završni rad

Student:

Nina Glamočak

Osijek, 2023

Sveučilište Josipa Jurja Strossmayera u Osijeku

Fakultet primijenjene matematike i informatike

Sveučilišni prijediplomski studij Matematika i računarstvo

Primjena algoritama strojnog učenja na graf bazi *Osnivači i startupi*

Završni rad

Mentor:
izv. prof. dr. sc. Domagoj Matijević

Student:
Nina Glamočak

Komentor:
dr. sc. Mateja Đumić

Osijek, 2023

Sadržaj

Sažetak **1**

Summary **2**

1 Uvod **3**

2 Baze podataka i graf baze podataka **4**

3 Neo4j **6**
3.1 Povijest Neo4j 6
3.2 Neo4j ekosustav 6
3.3 Instalacija Neo4j Desktop 8
3.4 Uvoz baze podataka u Neo4j 9

4 Cypher **11**

5 Istraživanje i opis graf baze podataka s Neo4jem **13**
5.1 Opis grafa preko svojstava čvorova i bridova 13
5.1.1 Smjer veze 13
5.1.2 Težine veza 14
5.1.3 Vrste čvorova 14
5.1.4 Distribucija stupnjeva grafa 15
5.2 GDS biblioteka 15
5.2.1 Algoritmi 16
5.2.2 Modeli i cjevovodi strojnog učenja te Python klijent 18
5.2.3 Projiciranje grafa 18

6 Rad Neo4j Graph Data Science na bazi podataka *Osnivači i startupi* **20**
6.1 Baza podataka *Osnivači i startupi* 20
6.1.1 Stvaranje čvorova i veza te primjena algoritama 20

7 Najgora praksa i uobičajene greške **28**

Literatura **31**

Sažetak

U ovom završnom radu istražena je primjena graf baze podataka, preciznije primjena alata Neo4j Graph Data Science. Ukratko ćemo se upoznati s graf bazama podataka i proširenjima Neo4j baze podataka te ćemo vidjeti kako se alat Neo4j Graph Data Science može koristiti u svrhu boljeg razumijevanja i analize brojnih skupova podataka. Nadalje, kako bismo bolje razumjeli ovu temu bit će prikazana baza podataka *Osnivači i startupi* na kojoj su primijenjeni usvojeni postupci.

Ključne riječi

graf baza podataka, podatkovna znanost, analiza skupa podataka

The application of machine learning algorithms on graph database *Founders and startups*

Summary

In this paper, the application of graph database in data science will be explored, more precisely the application of the toolkit Neo4j Graph Data Science. We will briefly familiarize ourselves with graph databases and Neo4j database extensions and see how the toolkit Neo4j Graph Data Science can be used for better understanding and analysis of numerous data sets. Furthermore, in order to understand this topic even better, a database *Founders and startups* will be presented on which the adopted procedures were applied.

Keywords

graph database, data science, analysis of data sets

1 | Uvod

Graf baza podataka je vrsta NoSQL baza podatka koja omogućuje obradu skupova podatka i olakšano izvođenje samih upita. Koristeći odnose susjednih čvorova olakšava skupljanje i agregaciju informacija nad brojnim vezama i čvorovima. Neo4j učinkoviti je sustav otvorenog koda (engl. *open-source*) koji implementira model graf baze podataka te unutar kojeg se nalazi ekosustav alata Neo4j Graph Data Science. Prvi puta je predstavljen i uveden godine 2020.

U ovom radu istražit ćemo na koje se načine ovaj alat može primijeniti u podatkovnoj znanosti. U drugom poglavlju ukratko je opisano što su baze podatka i graf baze podataka s naglaskom na Neo4j graf baze podataka.

Zatim, u trećem poglavlju istražen je Neo4j, njegova povijest i ekosustav. Također je prikazana instalacija Neo4j Desktop aplikacije te način uvoza baze podataka u aplikaciju.

Četvrto poglavlje opisuje Cypher, upitni jezik u Neo4ju i kako pomoću njega uvesti CSV podatke u Neo4j.

U petom poglavlju opisana je graf baza podatka s Neo4jem preko svojstava čvorova i bridova. Uvedeni su pojmovi smjera i težine veza te vrste čvorova. Istražena je biblioteka alata Neo4j Graph Data Science, njezini algoritmi strojnog učenja i cjevovodi (engl. *pipelines*), Python klijent te projiciranje grafa i vraćanje rezultata.

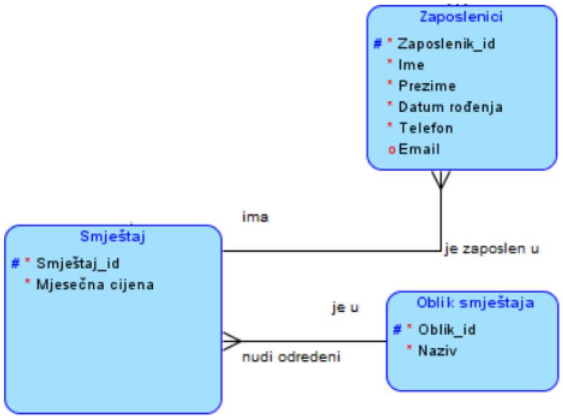
U šestom poglavlju, uvedena je baza podataka *Osnivači i startupi* u Neo4j, prikazano je uvođenje CSV podataka Cypherom, zatim izrada potrebnih i smislenih čvorova i veza te primjena algoritama u svrhu analize podataka.

Naposljetku, navedene su uobičajene greške koje se mogu dogoditi prilikom pokretanja algoritama te kako izbjeći probleme kod konfiguracije, kataloga i projekcije grafa.

2 | Baze podataka i graf baze podataka

Baze podataka način su organiziranja informacija tako da se u njima mogu brzo pronaći, izdvojiti, razvrstati, dodati ili obrisati željeni podaci. U smislu računarstva, to je sustav koji omogućuje pohranu podataka na računalu, telefonu ili bilo kojem drugom tehničkom uređaju. Jedna od njezinih prednosti je mogućnost upita nad podacima. Baze podataka se dijele na relacijske i NoSQL baze podataka.

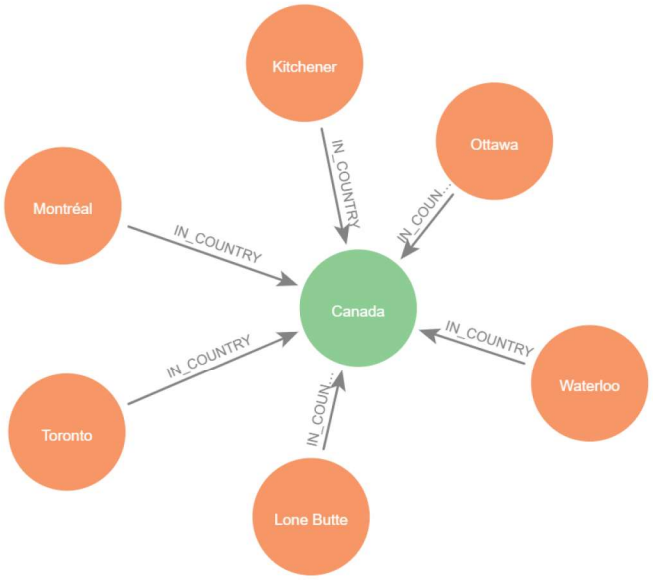
Relacijske baze podataka pohranjuju podatke u tablice u kojoj stupci predstavljaju atribute, a redovi instance entiteta. Imaju unaprijed zadanu shemu koja definira kako su podaci organizirani. Veze između entiteta ostvaruju se pomoću stranih ključeva.



Slika 2.1: Primjer relacijske baze

NoSQL baze podataka su baze koje uglavnom nisu relacijske te se prema svojstvima mogu podijeliti u četiri tipa baza podataka: ključ - vrijednost, dokument baza podataka, baza temeljena na stupcima i graf baze podataka. Ključ - vrijednost jednostavna je baza podataka za pretraživanje u kojoj su vrijednosti pohranjene pomoću ključa koji je obično string. To je čini jednostavnijom od tradicionalne baze podataka te je posebice vrlo učinkovita za predmemoriju u kontekstu web aplikacija, gdje je ključ usklađeni lokator sadržaja, odnosno URL (engl. *Uniform Resource Locator*) stranice, a vrijednost HTML (engl. *HyperText Markup Language*) sadržaj stranice. Dokument baza podataka nalik je ključ - vrijednost

bazama podataka u kojima dokument predstavlja vrijednost te je puno fleksibilnija za razliku od relacijskih baza podataka. Ovdje svaki dokument može imati različita polja, međutim odnose je teže modelirati. Baza temeljena na stupcima (engl. *Column family*) baza je koja sadrži stupce povezanih podataka. Predstavlja par ključ - vrijednost, gdje je ključ preslikan u vrijednost koja je skup stupaca. Svaki je stupac član neke „obitelji stupaca" i uređena je trojka koja se sastoji od naziva stupca, vrijednosti i vremenske oznake. Graf baze podataka pohranjuje podatke u čvorove i veze. Podaci se pohranjuju bez ograničenja unaprijed definiranog modela, što omogućuje izuzetno fleksibilan pristup razmišljanju i uporabi.



Slika 2.2: Primjer graf baze podataka

Graf je matematički objekt definiran skupom vrhova, odnosno čvorova i skupom bridova. Prednost grafova je mogućnost laganog prelaska s jednog čvora na drugi, slijedeći bridove. Brojne informacije mogu se izdvojiti pažljivom analizom mreže, kao što su njezina struktura i rangiranje čvorova. U posljednjih nekoliko godina raste interes za graf bazama podataka. Graf baza podataka omogućuje modeliranje entiteta i veza te uzima u obzir stvarnu svrhu čvora kako bi se iskoristio u potpunosti. Njegova primarna funkcija je povezivanje podataka, što u konačnici olakšava izradu upita koji su gotovo nemogući u tradicionalnim bazama podataka temeljenim na SQL-u.

3 | Neo4j

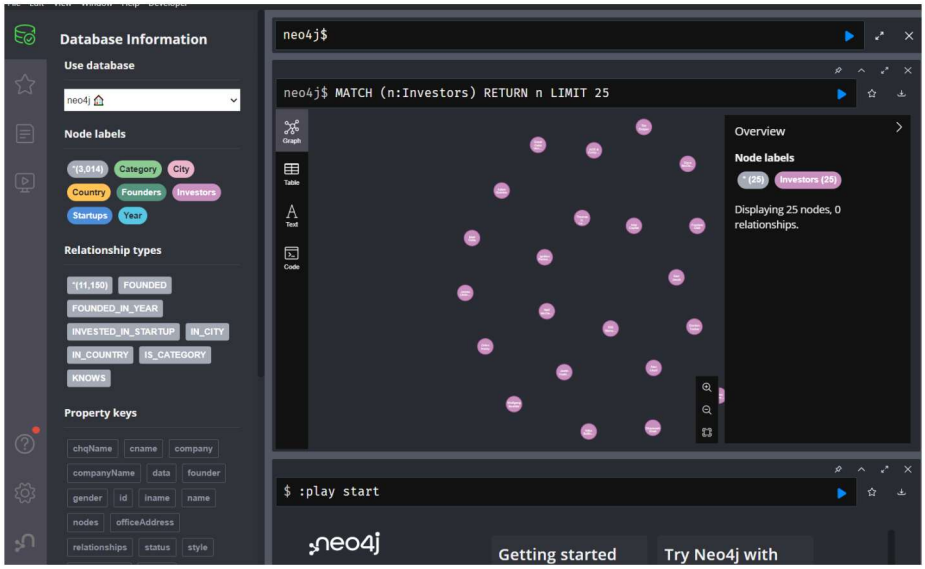
3.1 Povijest Neo4j

Neo4j je sustav za upravljanje graf bazom podataka koji je razvila tvrtka Neo4j, osnovana 2007. godine. Prvi dio koda skicirao je Emil Eifrem, osnivač i izvršni direktor tvrtke Neo4j, tijekom leta za Bombaj. Emil Eifrem, Johan Svensson i Peter Naubauer počeli su primjećivati visoke troškove u izvedbi i radu na jednoj od svojih aplikacija. Nakon provedenih različitih istraživanja, krenuli su u projekt s ciljem rješavanja izazova s kojima su se suočavali. Kroz dodatna istraživanja i razvoj, doveli su Neo4j među vodeća rješenja za graf baze podataka.

3.2 Neo4j ekosustav

Neo4j baza podataka je vrlo korisna već sama po sebi, no količina proširenja, aplikacija i biblioteka povezanih s njom čini je još potpunijim rješenjem. Neo4j Desktop je aplikacija koja omogućuje upravljanje instaliranim dodatcima i aplikacijama. Aplikacije instalirane na Neo4j Desktopu imaju pristup aktivnoj bazi podataka korisnika. Osim instalacije aplikacije postoji mogućnost izrade graf aplikacije koja će se pokrenuti unutar Neo4j Desktopa. Dvije baze ne mogu raditi istovremeno, prije no što se pokrene druga baza, prethodna se mora zaustaviti. Ekosustav se sastoji od aplikacija poput Neo4j Browser, Neo4j Bloom, Neodash, Graph Data Science Playground itd.

Neo4j Browser je jednostavna i moćna aplikacija koja omogućuje pisanje Cypher upita i vizualizaciju rezultata u obliku grafova.

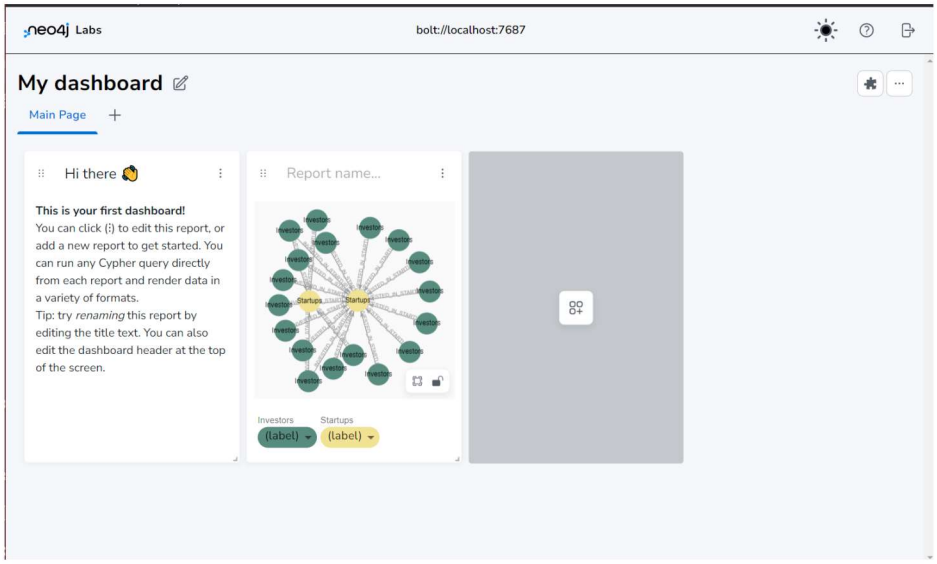


Slika 3.1: Neo4j Browser

Neo4j Bloom je aplikacija za vizualizaciju grafova u kojoj se mogu prilagoditi stilovi čvorova, poput veličine, boja, prikazanog imena itd. Za pregled i analizu podataka ne zahtijeva nikakve kodne ili programske vještine.

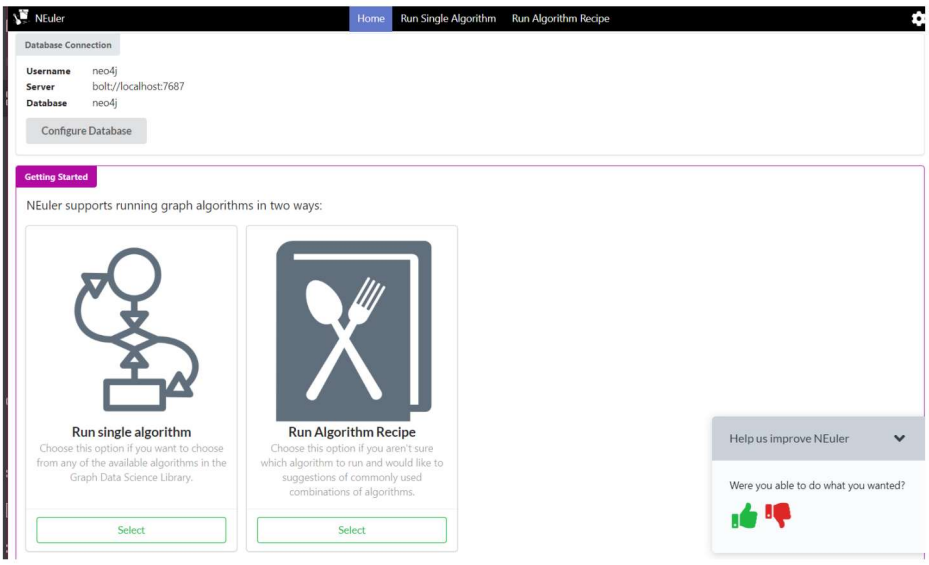
Slika 3.2: Neo4j Bloom

Neodash je aplikacija za izvještaje (engl. *dashboard*) koja omogućuje crtanje dijagrama iz podataka pohranjenih u Neo4j. Dijagrami se mogu organizirati u lijepe izvještaje koji se mogu dijeliti s drugim korisnicima.



Slika 3.3: Neodash

Graph Data Science Playground, odnosno Neuler, je korisničko sučelje bez koda koje podržava izvođenje svakog algoritma grafova iz GDS biblioteke. Također pruža mogućnost pregleda rezultata te Cypher upita za reprodukciju rezultata.



Slika 3.4: Graph Data Science Playground

3.3 Instalacija Neo4j Desktop

Najlakši način za korištenje Neo4ja na lokalnom računalu je korištenje aplikacije Neo4j Desktop, koja je dostupna za Windows, MacOS i Linux operativne sustave. Za instalaciju na sustav Windows potrebno je:

1. Posjetiti Neo4j centar za preuzimanje na https://neo4j.com/download-center/

2. Pritisnuti gumb Download Neo4j Desktop pri dnu stranice.
3. Ispuniti obrazac koji traži podatke o trenutnom korisniku (ime, e-mail, tvrtka i slično).
4. Pritisnuti gumb Download Desktop.
5. Spremiti aktivacijski ključ koji je prikazan na sljedećoj stranici. Aktivacijski ključ izgleda ovako:

   ezJhcGciOiJQUzI1NiIsInR5cCI6IkpXVCJ8
   ezJlcWFpbCI6InN0ZWxsYTBvdWhAZ21haWwuY29tIiwibWl4cGFuZWxJZ
   CI6Imdvb2dsZS1vYXV0a
   ...
   ...

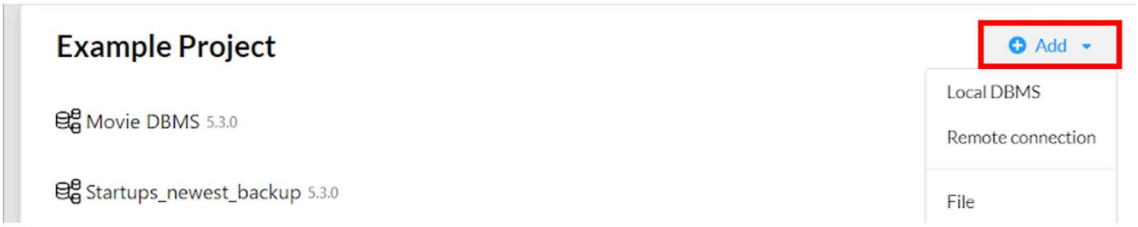
6. Pronaći instalacijski program, dvaput kliknuti na njega i slijediti navedene korake.
7. Kada se prvi put pokrene aplikacija, potrebno je unijeti aktivacijski ključ koji je bio spremljen prilikom preuzimanja izvršne datoteke. Nakon toga, aplikacija će se pokrenuti.

3.4 Uvoz baze podataka u Neo4j

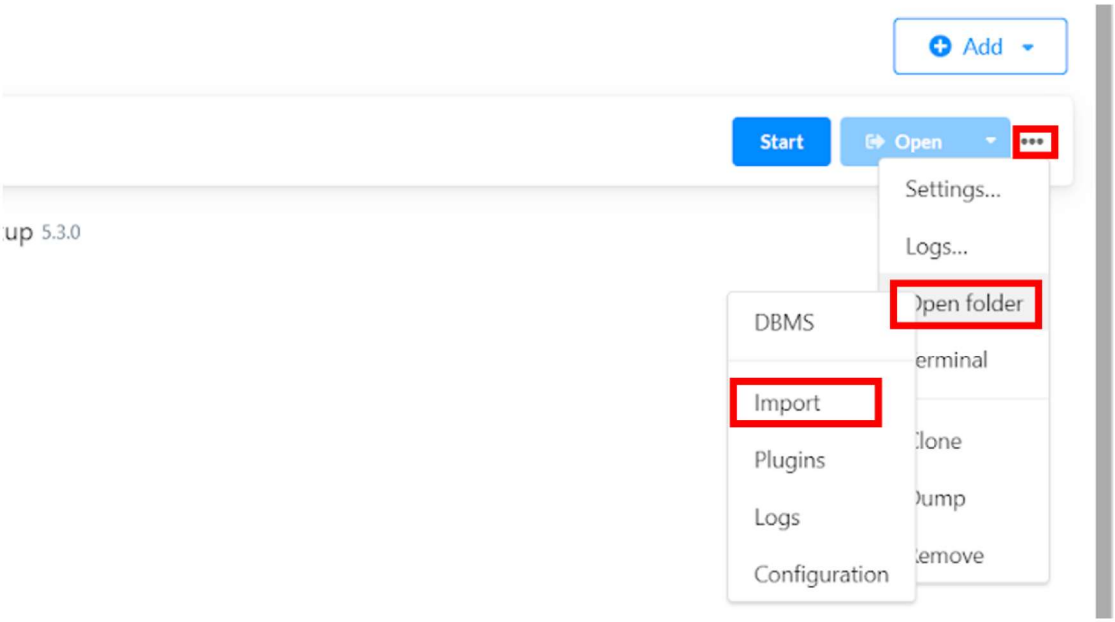
Kako bi se baza podataka mogla uvesti u Neo4j potrebno je prvo odabrati neku koja će odgovarati potrebama korisnika. Baza može biti preuzeta s interneta ili proizvoljno napravljena kao csv datoteka. Za uvođenje baze u Neo4j potrebno je:

1. U Neo4j Desktop potrebno je pritisnuti gumb Add prvi vrhu aplikacije.
2. U Name unijeti proizvoljno ime baze npr. “Movies DBMS” te u password upisati željenu lozinku.
3. Kliknuti gumb Create.
4. Nakon što je baza stvorena potrebno je kliknuti tri točkice u desnom kutu kraj imena baze.
5. Kliknuti Import Folder i zatim Import.
6. U stvorenu mapu staviti csv datoteku, odnosno bazu.
7. Kliknuti gumb Open.
8. Kliknuti gumb Neo4j Browser.

Nakon što je baza uspješno uvezena u Neo4j dostupna je za daljnje korištenje.



Slika 3.5: Korak 1.



Slika 3.6: Korak 4. i 5.

4 | Cypher

Cypher je upitni jezik koji je razvio Neo4j za stvaranje, pretraživanje i manipulaciju podataka. Postoji nekoliko jezika za postavljanje upita, no Cypher se najčešće koristi. Prednost Cyphera je što je lagan za naučiti i razumjeti posebno za one koji poznaju SQL te je zbog toga odlična osnova za učenje o grafovima. Nadalje, jezik je otvorenog koda što bi značilo da je softver dostupan s izvornim kodom kojeg svatko može pregledati, poboljšati i modificirati.

U Cypheru oznaka za čvor je (), oznaka za svojstvo je {}, dok su veze prikazane s []. Osnovne upite moguće je kombinirati kako bi se dobili složeniji upiti. Naredbe CREATE, MATCH i SET osnovne su i koriste se za upravljanje bazom podataka u Neo4j.

CREATE je identična naredbi INSERT u SQL-u te ima ulogu kreiranja čvorova i relacija sa svojstvima i bez njih.

Primjer 1. *Kako bi se kreirao čvor Osnivač koji sadrži svojstva poput ime, prezime ,tvrtka i slično, potrebno je napisati sljedeću naredbu:*

CREATE (os: Osnivač {ime: 'Adam', prezime: 'Smith', tvrtka: 'TechnoS', grad: 'Zagreb'})

Moguće je također kreirati i više čvorova istovremeno na sljedeći način:

CREATE (os: Osnivač {ime: 'Adam', prezime: 'Smith', tvrtka: 'TechnoS', grad: 'Zagreb'}), (tv: Tvrtka {tvrtka: 'TechnoS', godina: '2005'})

Osim toga, naredbom se mogu kreirati i veze:

CREATE (os: Osnivač)-[OSNOVAO]->(tv: Tvrtka)

MATCH je identična naredbi JOIN u SQL-u i omogućuje određivanje obrazaca koje će Neo4j pretražiti u bazi podataka.

Primjer 2. *Kako bi se kreirala veza između određenih osnivača potrebno je napisati sljedeće:*

MATCH (os1: Osnivač {ime: 'Adam', prezime: 'Smith'}),
(os2: Osnivač {ime: 'Lucia', prezime: 'Bond'})
CREATE (os1)-[POZNAJE]->(os2) RETURN a, b

SET služi ažuriranju veza i svojstava čvorova.

Primjer 3. *Kako bi se ažuriralo ime tvrke koju je osnovao osnivač s imenom Adam i prezimenom Smith potrebno je napisati sljedeću naredbu:*

MATCH (os: Osnivač {ime: 'Adam', prezime: 'Smith'}) SET os.tvrtka = 'Apple' RETURN os.ime, os.tvrtka

Datoteka razdvojena zarezom ili CSV datoteka (engl. *CSV-comma-separated values*) obična je tekstualna datoteka koja dopušta spremanje podataka u format tablica. Koristi se za razmjenu podataka i predstavlja više od 57% svih skupova podataka, dok JSON datoteke čine manje od 10%. Dovoljno je samo pročitati nazive stupaca kako bi se razumjelo što koji stupac predstavlja te u usporedbi s JSON datotekama nema skrivenih polja.

	A	B	C	D	E	F
1	founder,company,gender					
2	Aakash Patel,Flytenow,Male					
3	Aarjav Trivedi,InstantCab,Male					
4	Aaron Cheung,HomeJoy,Male					
5	Aaron Epstein,ChromaOm,Male					
6	Aaron Feuer,Panorama Education,Male					
7	Aaron Grant,Thalmic Labs (Myo),Male					
8	Aaron Harris,Tutorspree,Male					
9	Aaron Iba,Apniet,Male					

Slika 4.1: CSV datoteka

Za uvoz podataka iz CSV datoteke u Neo4j upitu rabi se naredba LOAD CSV. Zatim se upisuje direktorij baze te se koristi naredba CREATE za stvaranje čvora zajedno s njegovim svojstvima.

```
1 LOAD CSV WITH HEADERS
2 FROM 'file:///Startups.csv' AS row
3 WITH split(row.cityHq, ",") as city_list
4 UNWIND city_list AS city_name
5 MERGE (:City{name: city_name})
```

Slika 4.2: Primjer kako bi se podatak Grad (engl. *City*) uveo u bazu

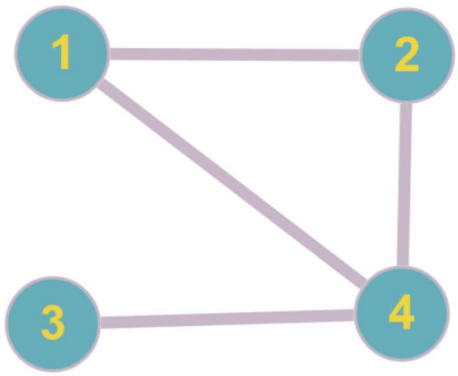
5 | Istraživanje i opis graf baze podataka s Neo4jem

5.1 Opis grafa preko svojstava čvorova i bridova

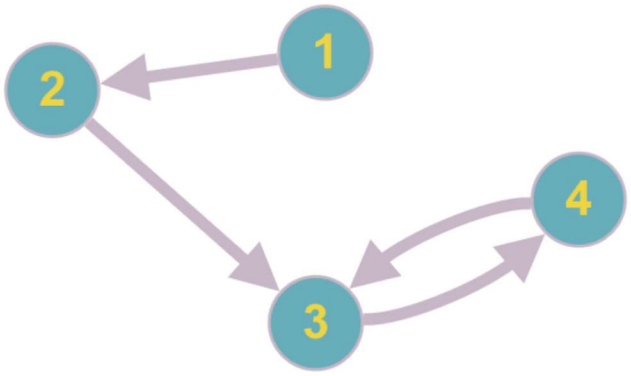
5.1.1 Smjer veze

Veze među čvorova dijele se na usmjerene (lukovi), gdje bridovi asimetrično povezuju dva vrha, i neusmjerene (bridovi), gdje bridovi simetrično povezuju dva vrha. U graf bazama podataka sve se veze zovu bridovi ili samo veze.

Na primjer, dodavanjem prijatelja na socijalnoj mreži Facebook, svaki korisnik sada dobiva puni pristup javnom sadržaju drugog korisnika te to predstavlja usmjeren graf. Primjer neusmjerenog grafa u stvarnom životu su pratitelji na socijalnoj mreži Instagram. Kada jedan korisnik zaprati drugog korisnika, taj novi korisnik ne mora nužno uzvratiti praćenje prethodnom korisniku.



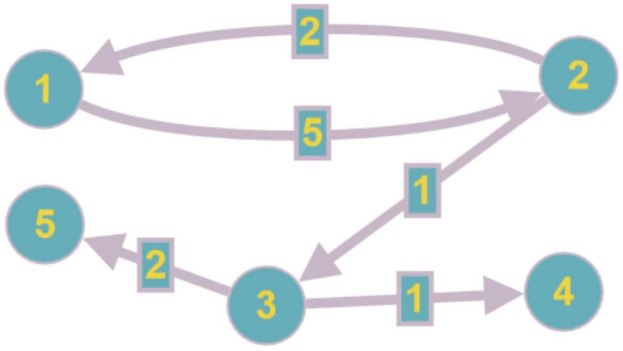
Slika 5.1: Neusmjereni graf



Slika 5.2: Usmjereni graf

5.1.2 Težine veza

Graf je težinski ili netežinski, ovisno o tome nose li bridovi svojstvo koje kvantificira snagu veza između čvorova. Na primjer, u grafu prijateljstava, veza između dvije osobe može sadržavati broj godina koliko se te osobe poznaju.



Slika 5.3: Težinski graf

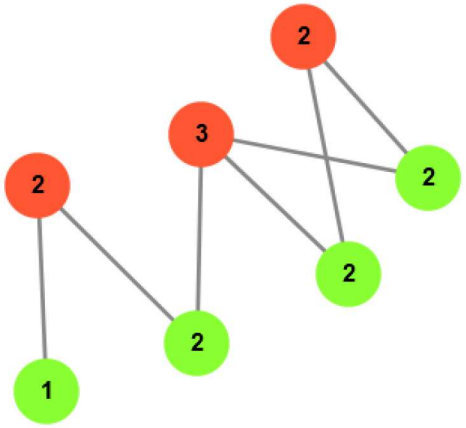
Usmjereni graf se može tretirati kao neusmjeren, a težinski graf se uvijek može promatrati kao netežinski ako je potrebno, ali obrnuto nije moguće.

5.1.3 Vrste čvorova

Homogeni grafovi u bazi podataka su grafovi koji sadrže čvorove iste vrste, dok se grafovi koji sadrže različite entitete zovu heterogeni. U Neo4ju heterogeni grafovi imaju više oznaka čvorova, dok homogeni grafovi samo jednu oznaku čvora.

Posebna vrsta heterogenog grafa je bipartitni graf. Bipartitni graf sadrži dvije vrste čvorova, a veze između čvorova se ostvaruju samo između dva čvora različitih vrsta. Često se koristi za predviđanje preferenci poput filmova ili hrane.

Primjerice, u grafu filmova i osoba, osobe su povezane s filmovima koje su pogledali, ali međusobno osobe i filmovi nisu povezani.



Slika 5.4: Bipartitni graf

5.1.4 Distribucija stupnjeva grafa

Stupanj čvora daje informaciju jesu li bridovi jednako podijeljeni po čvorovima ili su neki monopolizirali gotovo sve veze, ostavljajući ostale nepovezanim. Odnosno, to je broj veza povezanih s tim čvorom. U Neo4ju stupanj se može izračunati Cypherom te se distribucije mogu nacrtati pomoću NeoDash aplikacije za grafove.

Za neusmjerene grafove postoji samo jedan stupanj, dok se kod usmjerenih grafova razlikuju dolazni, izlazni i ukupni stupanj. Dolazni stupanj broji samo bridove koji pokazuju prema čvoru. Izlazni broji samo bridove koji su usmjereni prema van od čvora. Ukupni stupanj broji sve čvorove, neovisno o njihovom smjeru.

5.2 GDS biblioteka

Graph Data Science (GDS) biblioteka prvi put je objavljena 2020. godine, a od tada je doživjela mnoga poboljšanja u pogledu performansi i standardizacije. Također su dodane brojne nove značajke, uključujući parametrizaciju algoritama i nove vrste algoritama.

Koristeći podatke pohranjene u Neo4ju, GDS biblioteka sadrži alate koji se koriste u projektima podatkovne znanosti. To uključuje algoritme vezane za put, graf algoritme, modele i cjevovode strojnog učenja te Python klijent.

5.2.1 Algoritmi

Algoritmi centraliteta koriste se za razumijevanje uloga određenih čvorova u grafu i njihov utjecaj na tu mrežu. Korisni su jer identificiraju najvažnije čvorove i omogućavaju razumijevanje grupne dinamike, kao što su vjerodostojnost, pristupačnost, brzina kojom se stvari šire i mostovi između grupa.

Algoritam ranga stranice (engl. *PageRank*)

Nazvan po suosnivaču Googlea Larryju Pageu, algoritam se prvotno koristio za rangiranje web stranica u Googleu prema rezultatima pretraživanja korisnika. Mjeri važnost svakog čvora unutar grafa, na temelju broja dolaznih veza i važnosti odgovarajućih izvornih čvorova. Temeljna pretpostavka jest da je stranica važna onoliko koliko su važne stranice koje pokazuju na nju.

Algoritam središnje centralnosti (engl. *Betweenness centrality*)

Algoritam služi otkrivanju količine utjecaja koji čvor ima na protok informacija u grafu. Često se koristi za pronalaženje čvorova koji služe kao most od jednog dijela grafa do drugog. Izračunava najkraće staze između svih parova čvorova u grafu. Svaki čvor dobiva rezultat na temelju broja najkraćih puteva koji prolaze kroz čvor. Implementiran je za grafove bez težina ili s pozitivnim težinama. Implementacija GDS-a temelji se na Brandesovom približnom algoritmu za ne težinske grafove, dok se za težinske grafove koristi Dijkstra algoritam.

Algoritam stupnja centralnosti (engl. *Degree centrality*)

Algoritam se može koristiti za pronalaženje popularnih čvorova unutar grafa. Mjeri broj dolaznih ili odlaznih (ili oba) odnosa iz čvora, ovisno o orijentaciji projekcije odnosa. U Neo4ju se može primijeniti na težinske ili netežinske grafove.

Zajednice su svojstvo mnogih mreža u kojima određena mreža može imati više zajednica tako da su čvorovi unutar zajednice gusto povezani. Prilikom analize različitih mreža, može biti važno otkriti zajednice unutar njih. Tehnike otkrivanja zajednica korisne su za algoritme društvenih medija kako bi otkrili ljude sa zajedničkim interesima i držali ih čvrsto povezanima. Otkrivanje zajednice može se koristiti u strojnom učenju za otkrivanje grupa sa sličnim svojstvima i izdvajanje grupa iz raznih razloga.

Louvain algoritam

Algoritam je za otkrivanje zajednica u velikim mrežama te maksimizira ocjenu modularnosti za svaku zajednicu, pri čemu modularnost kvantificira kvalitetu dodjele čvorova zajednicama. To znači procijeniti koliko su gušće povezani čvorovi unutar zajednice, u usporedbi s time koliko bi bili povezani u slučajnoj mreži.

Modularnost je mjera koliko su dobro grupe podijeljene u klastere. Uspoređuju se odnosi unutar klastera s onim što bi se očekivalo za nasumičan broj veza. Hi-

jerarhijski je algoritam grupiranja koji rekurzivno spaja zajednice u jedan čvor i izvršava modularno klasteriranje na sažetim grafovima.

Broj trokuta (engl. *Triangle count*)

Naziv je sam po sebi razumljiv, ali trokut definiraju tri povezana čvora. U usmjerenom grafu, potrebno je uzeti u obzir orijentaciju ruba. Za dani čvor, n, njegov broj trokuta nalazi se provjerom jesu li njegovi susjedi također povezani s drugim susjedom od n. Algoritam u GDS biblioteci pronalazi trokute samo u neusmjerenim grafovima. Popularan je u analizi društvenih mreža, gdje se koristi za otkrivanje zajednica i mjerenje kohezivnosti tih zajednica.

Slabo povezane komponente (engl. *Weakly connected components* (*WCC*))

Pronalazi skupove povezanih čvorova u usmjerenim i neusmjerenim grafovima. Dva su čvora povezana ako između njih postoji put. Skup svih čvorova koji su međusobno povezani čine komponentu. Za razliku od snažno povezanih komponenti (SCC), ne uzima se u obzir smjer odnosa na putu između dva čvora. Često se koristi na početku analize za razumijevanje strukture grafa. Ovaj algoritam, iako koristan, pomalo je ekstreman u smislu da će pronaći samo čvorove koji imaju nula veza. Međutim u GDS-u, u grafu bez nepovezanih čvorova još je uvijek moguće stvoriti klastere ili grupe.

Algoritmi sličnosti dodjeljuju ocjenu paru vrhova na temelju toga koliko su slični. To obično uključuje provjeru imaju li iste ili slične četvrti.

Sličnost čvorova (engl. *Node similarity*)

Uspoređuje skup čvorova na temelju čvorova s kojima su oni povezani. Dva se čvora smatraju sličnim ako dijele mnogo istih susjeda. Sličnost čvorova izračunava sličnosti u paru na temelju Jaccardove metrike, također poznate kao Jaccardova ocjena sličnosti ili koeficijent preklapanja, također poznatog kao Szymkiewicz–Simpsonov koeficijent.

Filtrirana sličnost čvorova (engl. *Filtered Node similarity*)

Proširenje je algoritma sličnosti čvorova. Dodaje podršku za filtriranje na izvornim čvorovima, ciljnim čvorovima ili oboje. Filter čvora smanjuje prostor čvora za koji će algoritam dati rezultate.

K najbližih susjeda (engl. *K-Nearest Neighbors*)

Algoritam izračunava vrijednost udaljenosti za sve parove čvorova u grafu i stvara nove odnose između svakog čvora i njegovih k najbližih susjeda. Udaljenost se izračunava na temelju svojstava čvora.

Algoritmi za pronalazak puta ključna su komponenta u raznim primjenama, uključujući videoigre, robotiku i logistiku. Omogućuju strojevima učinkovitu naviga-

ciju kroz složena okruženja pronalaženjem najkraćeg ili najučinkovitijeg puta između dvije točke.

A* najkraći put (engl. *A* Shortest Path*)

Algoritam izračunava najkraći put između dva čvora. A* je informirani algoritam pretraživanja jer koristi heurističku funkciju za vođenje obilaska grafa. Algoritam podržava težinske grafove s pozitivnim težinama veza.

Minimalno razapinjuće stablo (engl. *The Minimum Weight Spanning Tree*)

Algoritam počinje od zadanog čvora, pronalazi sve njegove dostupne čvorove i vraća skup odnosa koji povezuju te čvorove s minimalnom mogućom težinom. Primov algoritam je jedan od najjednostavnijih i najpoznatijih algoritama minimalnog razapinjućeg stabla. Djeluje slično Dijkstrinom algoritmu najkraćeg puta, ali umjesto da minimizira ukupnu duljinu puta koji završava na svakom odnosu, on minimizira duljinu svakog odnosa pojedinačno.

Najkraći put svih parova (engl. *All Pairs Shortest Path*)

Izračunava najkraći (težinski) put između svih parova čvorova. Ovaj algoritam je optimiziran, što ga čini bržim od algoritma najkraćeg puta iz jednog početnog čvora (engl. *Single Source Shortest Path*) za svaki par čvorova u grafu.

Ovo su samo od neki od algoritama koje je moguće koristiti u alatu Neo4j Graph Data Science.

5.2.2 Modeli i cjevovodi strojnog učenja te Python klijent

Ugrađivanje (engl. *Embedding*) je proces transformacije visokodimenzionalnog objekta, kao što je tekst, slika ili graf, u vektor niske dimenzije uz očuvanje nekih ključnih karakteristika izvornog objekta, kao što je značenje za tekst ili topologija za graf. Biblioteka GDS također može pohraniti uvježbane modele u katalog modela kako bi se modeli kasnije mogli učitati za predviđanje. Također može stvoriti cjevovode (engl. *pipelines*) i pohraniti ih u katalog za klasifikaciju čvorova ili zadatke predviđanja veze. Godine 2021. pušten je Python klijent, koji omogućuje pozivanje GDS procedure bez pisanja Cyphera, već samo Pythona.

5.2.3 Projiciranje grafa

Projiciran graf je graf koji se koristi za pokretanje svih algoritama iz GDS-a. Može biti identičan Neo4j grafu, ali ne mora. Svi čvorovi, veze i svojstva projiciranog grafa mogu se konfigurirati, a u GDS biblioteci dva su načina stvaranja projiciranih grafova. U izvornu projekciju spadaju čvorovi, veze i svojstva, dok se u Cypher projekciji entiteti filtriraju iz Neo4ja ili se kreiraju u hodu pomoću Cypher upita.

Ovisno o potrebama postoji više načina vraćanja rezultata. Tok je jednostavniji način izvođenja, gdje se rezultati algoritma pohranjuju u memoriji i odlaze prema

korisniku. Kod pisanja rezultati algoritma se ne pohranjuju u memoriji već se zapisuju natrag u Neo4j graf kao svojstvo čvora ili nova veza. Mutacija rezultate algoritma vraća u projektirani graf. Statistika ne vraća pojedinačni rezultat za svaki čvor, ali pruža ukupnu statistiku za algoritam kao što su njegovo vrijeme izvođenja i distribucija izračunate metrike.

6 | Rad Neo4j Graph Data Science na bazi podataka *Osnivači i startupi*

6.1 Baza podataka *Osnivači i startupi*

Baza sadrži informacije o gotovo 700 startup tvrtki koje podržava Y Combinator između 2005. i 2014. godine. Podaci se prikupljaju i agregiraju iz SeedDBa, CrunchBasea i AngelLista.

6.1.1 Stvaranje čvorova i veza te primjena algoritama

Prethodno kreiranje čvorova je potrebno kako bi se mogla izvršiti manipulacija bazom podataka i primijeniti algoritmi.

Nabrojat će se čvorovi koji se nalaze u bazi *Osnivači i startupi* s pripadnim sadržajima za svaki čvor, pri čemu je naveden prvo izvorni, engleski naziv koji se nalazi u bazi podataka, a u zagradi se navodi njegovo značenje na hrvatskom jeziku.

Dakle, čvorovi sadržani u bazi *Osnivači i startupi* su:

1. Founders (Osnivači) sadrži: gender (spol), founder (osnivač), company (tvrtka)
2. Investors (Investitori) sadrži: investor
3. Startups (Startupi) sadrži: companyName (ime tvrtke), officeAddress (adresa ureda), status
4. Country (Država) sadrži: country (država)
5. City (Grad) sadrži: city (grad)
6. Category (Kategorija) sadrži: categories (kategorije)
7. Year (Godina) sadrži: year (godina)

Također kako bi se čvorovi kreirali korištene su naredbe LOAD CSV i CREATE. Sljedeće slike prikazuju neke od primjera upita.

```
1  //create startup node
2  LOAD CSV WITH HEADERS FROM 'file:///Startups.csv' AS row
3  WITH row WHERE row.companyName IS NOT NULL
4  MERGE (c:Startups {companyName: row.companyName,officeAddress:
   coalesce(row.officeAddress,"Unknown"),status:row.status})
5  SET c.officeAddress = CASE trim(row.officeAddress) WHEN "" THEN null ELSE row.officeAddress END
6  RETURN count(c);
7
8  //create founders node
9  LOAD CSV WITH HEADERS FROM 'file:///Founders.csv' AS row
10 MERGE (f:Founders {founder: row.founder, gender: row.gender,company:row.company})
11 RETURN count(f);
```

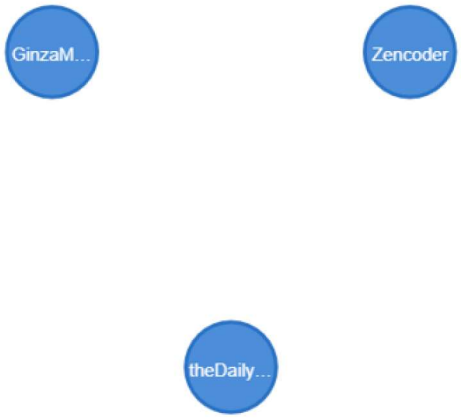
Slika 6.1: Izrada čvorova Startupi (engl. *Startups*) i Osnivači (engl. *Founders*)

```
File  Edit  View  Window  Help  Developer
1  LOAD CSV WITH HEADERS
2  FROM 'file:///Startups.csv' AS row
3  WITH split(row.categories, ",") as categories_list
4  UNWIND categories_list AS categories_name
5  MERGE (:Category{cname: categories_name})
6
7  LOAD CSV WITH HEADERS FROM 'file:///Startups.csv' AS row
8  MERGE (e:Country {chqName: coalesce(row.countryHq,"Unknown")})
9  WITH e, row
10 UNWIND split(row.cityHq, ',') AS city
11 MERGE (s:City{name: city})
12 MERGE (e)<-[r:IN_COUNTRY]-(s)
```

Slika 6.2: Izrada čvorova Kategorija (engl. *Category*) i Grad (engl. *City*)

Uspješno stvoreni čvorovi mogu se prikazati pojedinačno klikom na njihovu oznaku (engl. *label*) .

Slika 6.3: Lista uspješno stvorenih čvorova



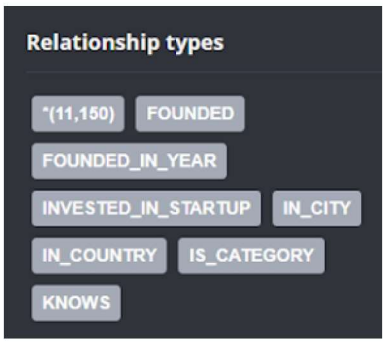
Slika 6.4: Grafički prikaz čvora Startupi (engl. *Startups*)

Kako bi graf baza podataka bila smislena potrebno je kreirati veze između čvorova. Sljedeće slike prikazuju neke od primjera upita za kreiranje veza.

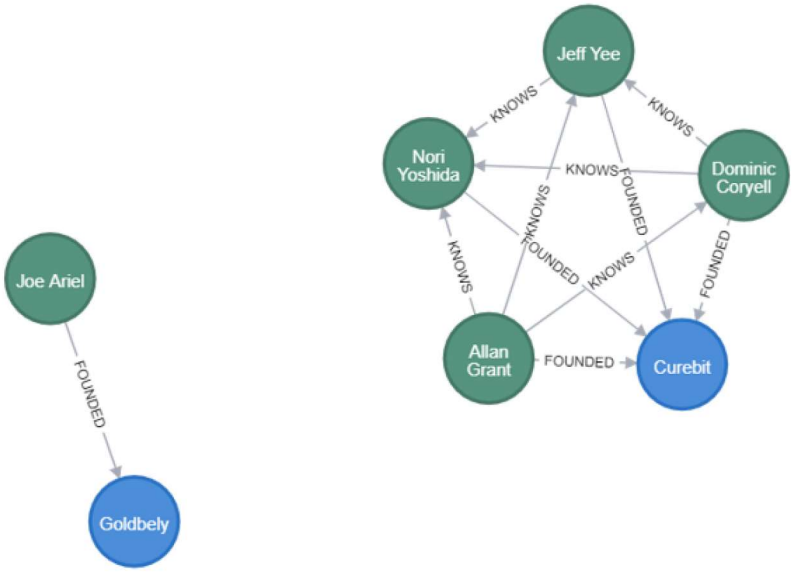
```
1  LOAD CSV WITH HEADERS
2  FROM 'file:///Startups.csv' AS row
3  MATCH (s:Startups {companyName:row.companyName})
4  WITH s, split(row.cityHq, ",") as city_list
5  UNWIND city_list AS city_name
6  MATCH (c:City {name: city_name})
7  CREATE (c)<-[:IN_CITY]-(s)
8
9  LOAD CSV WITH HEADERS FROM 'file:///Founders.csv' AS row
10 MATCH (f1:Founders)-[:FOUNDED]->(common:Startups)<-[:FOUNDED]-(f2:Founders)
11 WHERE id(f1) < id(f2)
12 MERGE (f1)-[:KNOWS]->(f2)
```

Slika 6.5: Upiti za kreiranje veza U_GRADU (engl. *IN_CITY*) i POZNAJE (engl. *KNOWS*)

Uspješno kreirane veze također se mogu prikaza klikom na njihovu oznaku.

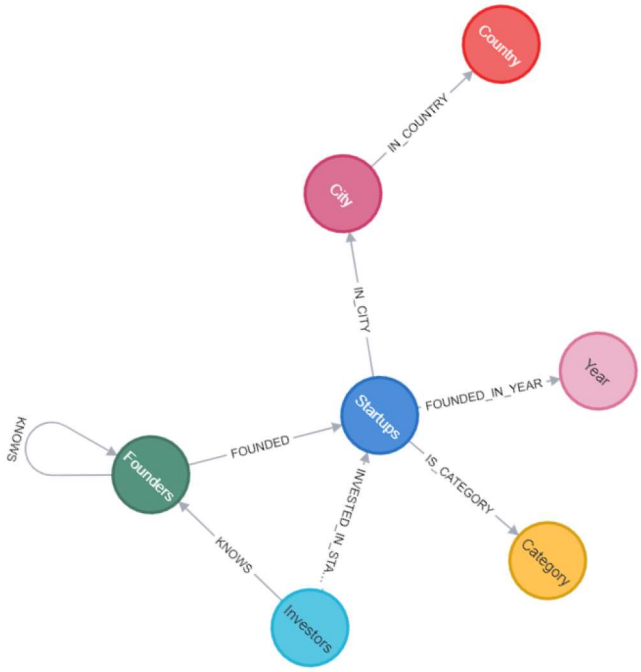


Slika 6.6: Lista uspješno stvorenih veza



Slika 6.7: Grafički prikaz veze POZNAJE (engl. *KNOWS*)

Cijeli graf, uključujući sve njegove veze i čvorove, može biti prikazan naredbom CALL db.schema.visualization().



Slika 6.8: Grafički prikaz cijelog grafa baze podataka

Nakon što je smisleni graf kreiran mogu se početi primjenjivati algoritmi kako bi se dobile i saznale neke bitne informacije vezane za graf, odnosno bazu podataka.

Algoritam stupnja centralnosti

Kako bi se pronašli investitori koji su uložili u najviše tvrtki primijenjen je algoritam stupnja centralnosti.

```
1 CALL gds.graph.project('InvestorStartup',['Investors', 'Startups'], ['INVESTED_IN_STARTUP']);
2
3 CALL gds.degree.stream('InvestorStartup')
4 YIELD nodeId, score
5 RETURN gds.util.asNode(nodeId).iname AS name, score
6 ORDER BY score DESC
7
```

Slika 6.9: Projekcija grafa i primjena algoritma stupnja centralnosti

U rezultatima je vidljivo da je među investitorima u startupe najviše uložila firma Y Combinator, SV Angel, Andreessen Horowitz i Alexis Ohanian.

2	" Y Combinator"	264.0
3	" SV Angel"	81.0
4	" Andreessen Horowitz"	44.0
5	" Alexis Ohanian"	41.0
6	" Paul Buchheit"	38.0

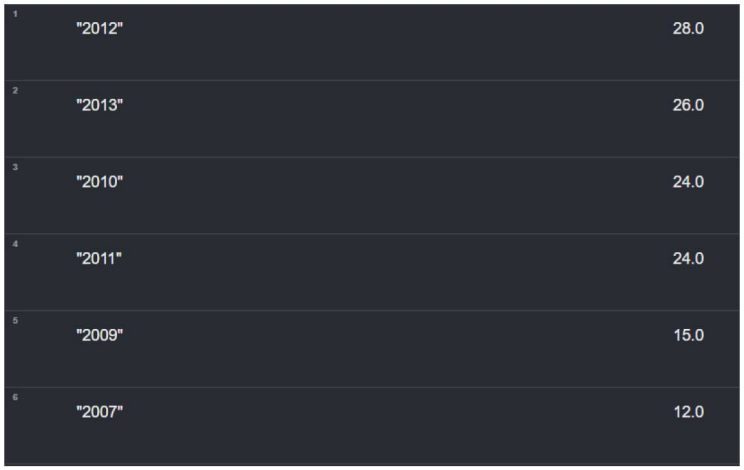
Slika 6.10: Rezultati primjena algoritma za investitore s najviše ulaganja

Kako bi se pronašla najunosnija godina, odnosno godina u kojoj je osnovano najviše tvrtki također je primijenjen algoritam stupnja centralnosti.

```
1 CALL gds.graph.project('YearStartups',['Startups', 'Year'], {FOUNDED_IN_YEAR:{   orientation:
  'REVERSE'}});
2
3 CALL gds.degree.stream('YearStartups')
4 YIELD nodeId, score
5 RETURN gds.util.asNode(nodeId).yname AS name, score
6 ORDER BY score DESC
7
```

Slika 6.11: Projekcija grafa i primjenu algoritma stupnja centralnosti.

U rezultatima je vidljivo da je 2012. godine osnovano najviše startupa, dok je 2000. i 1996. godine osnovan samo jedan.



1	"2012"	28.0
2	"2013"	26.0
3	"2010"	24.0
4	"2011"	24.0
5	"2009"	15.0
6	"2007"	12.0

Slika 6.12: Rezultati primjene algoritma na godinama

Sličnost čvora

Kako bi se pronašla sličnost između investitora, odnosno koliko je investitora uložilo u iste startupe potrebno je primijeniti algoritam sličnosti čvora.

Ako su korištene zadane vrijednosti za konfiguracijski parametar procedure, TopK je postavljen na 10, dok je topN postavljen na 0. Zbog toga skup rezultata sadrži 10 najboljih rezultata sličnosti za svaki čvor.

No, u ovom slučaju potrebno je ograničiti rezultate kako bi se vidio samo najbolji rezultat za svaki čvor, to jest za svakog investitora. Postoje četiri ograničenja koja se mogu primijeniti na rezultate sličnosti. TOP ograničava rezultat na najviše rezultate sličnosti, odnosno najveći broj istih značajki. Primjerice, ako su tvrtka A i tvrtka B uložile u tri iste kompanije, a tvrtka A i tvrtka C u sedam istih kompanija, tada tvrtke A i C imaju najviši rezultat sličnosti. BOTTOM ograničava rezultat na najniže rezultate sličnosti to jest algoritam ispisuje samo parove tvrtki koje su uložile u najmanje istih startupa. I gornja i donja granica mogu se primijeniti na rezultat kao cjelinu ("N") ili na rezultat po čvoru ("K").

```
1 CALL gds.nodeSimilarity.stream('InvestorStartup', { topK:1 })
2 YIELD node1, node2, similarity
3 RETURN gds.util.asNode(node1).iname AS Investor1, gds.util.asNode(node2).iname AS Investor2,
  similarity
4 ORDER BY similarity DESC
```

Slika 6.13: Projekcija grafa i primjena algoritma sličnosti čvora

Rezultati pokazuju da je investor Dharmesh Shah najsličniji Alexu Lloydu, odnosno uložili su u najviše istih tvrtki, dok je Thomas D. Lehrman najsličniji Adamu Quintonu.

" Alex Lloyd"	" Dharmesh Shah"	1.0
" Gordon Tucker"	" Dharmesh Shah"	1.0
" Stage One Capital"	" Dharmesh Shah"	1.0
" Thomas D. Lehrman"	" Adam Quinton"	1.0
" Adam Quinton"	" Thomas D. Lehrman"	1.0

Slika 6.14: Rezultati primjene algoritma na investitorima

Rang stranice (engl. *PageRank*)

Kako bi se pronašla najpopularnija kategorija u koju spadaju startupi, odnosno kojeg su tipa najveći broj osnovanih kompanija potrebno je primijeniti algoritam ranga stranice.

```
CALL gds.graph.project('CategoryStartup',['Category', 'Startups'], ['IS_CATEGORY']);

CALL gds.pageRank.stream('CategoryStartup')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).cname AS name, score
ORDER BY score DESC, name ASC
```

Slika 6.15: Projekcija grafa i primjena algoritma ranga stranice

Rezultati pokazuju da najveći broj osnovanih startupa spada u kategoriju “Curated Web”, zatim “Software” i “Mobile”.

"Curated Web"	6.588750000000006
"Software"	4.676250000000002
"Mobile"	2.901874999999999

Slika 6.16: Rezultati primjene algoritma na kategorije i startupa

Louvain algoritam

Kako bi se otkrilo koliko koja osoba ima utemeljenih zajednica, u ovom slučaju

na POZNAJE (engl. *KNOWS*) relaciji, odnosno broj zajednica u velikoj mreži potrebno je primijeniti Louvain algoritam.

```
1 CALL gds.graph.project('InvestorFounder',['Investors', 'Founders'], ['KNOWS']);
2
3 CALL gds.louvain.stream('InvestorFounder')
4 YIELD nodeId, communityId, intermediateCommunityIds
5 RETURN gds.util.asNode(nodeId).founder AS name, communityId
6 ORDER BY name ASC
```

Slika 6.17: Projekcija grafa i primjena Louvain algoritma

U rezultatima je vidljivo da AJ Asver ima 502 zajednice, AJ Forsythe 93, dok Aarjav Trivedi ima samo jednu što bi značilo da ima najmanje poznanstava (povezanosti) s ostalim osnivačima i investitorima.

1	"AJ Asver"	772
2	"AJ Forsythe"	91
3	"Aakash Patel"	601
4	"Aarjav Trivedi"	1
5	"Aaron Cheung"	16
6	"Aaron Epstein"	3

Slika 6.18: Rezultati primjene algoritma na osnivačima

7 | Najgora praksa i uobičajene greške

Za svaki čvor algoritam obično razmatra veze svih čvorova te tim vezama prelazi na susjede. Ako čvor nema nikakvih veza, odnosno nema susjeda nazivamo ga izolirani čvor. Izolirani čvorovi većinom nisu korisni. Jednom kada je takav čvor pronađen samo je preskočen i algoritam se nastavlja nad ostalim čvorovima u grafu. No, to nije najbolja praksa jer se primjerice prilikom pokretanja algoritama zajednice gube bitne informacije.

Kada je algoritam pokrenut svi čvorovi su unutar iste grupe, odnosno svi su čvorovi unutar grafa smješteni u jednu veliku zajednicu. U ovom slučaju graf je gusto povezan i ima previše veza. Tada je algoritmu teško da odredi gdje da ukloni veze. Nadalje, ako je to slučaj u samo jednoj zajednici, tada se mogu probati koristiti drugi algoritmi. Također se mogu koristiti i težine ili pragovi koji omogućuju da se neke veze preskoče ako nisu dovoljno interesantne. Koristi se heuristika da bi se isključile, tj. zanemarile neke veze iz grafa i tako smanjila gustoća veza pa je algoritmu lakše da razbije veze i stvori manje zajednice.

Algoritam ne zna da čvorovi koje vidi reprezentiraju stvari iz života koje su međusobno različite. Ako postoji graf s osobama i stvarima, trgovinama i zemljama to su drugačije stvari i međusobno drugačije vrste veza, no sve se mogu svrstati u jedan isti graf. Kada se pokrene algoritam za pronalazak zajednica, to je najbolje napraviti na stvarima koje reprezentiraju istu stvar. Moguće je projicirati hrpu čvorova različitog tipa u istu stvar i pustiti algoritam da ih gleda tako. Algoritam neće moći reći da to nisu isti tipovi.

Algoritmi mogu za iste podatke dati različite rezultate. Ako je algoritam dva puta pokrenut i dao različite rezultate mogućnost je da postoji greška ili da nema smisla koristiti određeni algoritam. No, velik broj algoritama je stohastičan, tj. algoritam koristi heuristiku koja nije deterministička. Primjerice, algoritam je došao do točke da mora donijeti odluku, kako bi ju donio koristi nasumičnost, u ovom slučaju hoće li ići lijevo ili desno, jedanput može otići lijevo, a ponovnim pokretanjem algoritma ode desno.

Pojedini algoritmi su preskupi, odnosno po prirodi su prespori specifično algoritam središnje centralnosti, sličnost čvora ili bilo koji drugi algoritam sličnosti. No, kad god je algoritam preskup postoje konfiguracijske opcije pomoću kojih se

može kontrolirati koliko će vremena trebati da se algoritam izvrši. Postoje brojni načini kako smanjiti veličinu problema za određeni algoritam, primjerice za sličnost su to: topK, topN, degreeCutOff. Postoji mogućnost razdvajanja problema, odnosno ako postoji neki veliki graf, ali u stvarnosti je korisniku potreban samo jedan njegov zanimljiv dio, tada postoje algoritmi koji prepoznaju koji su čvorovi važni unutar grafa.

Algoritam se nije pokrenuo zbog čuvara memorije. Čuvar memorije je svojstvo unutar sustava koji sprječava rušenje baze podataka. Ako postoji velika mogućnost da će pokretanje algoritma na podacima prijeći memorijske granice te da će se sustav srušiti, onda ga čuvar zabrani. Svojstvo se može isključiti, no to se ne preporučuje.

JVM je ostao bez memorije. Jedna je od mogućih posljedica isključivanja čuvara memorije, no u nekim slučajevima može se dogoditi i kada je on uključen. Neki algoritmi zahtijevaju puno memorije, kako bi oslobodili mjesta u memoriji preporučeno je ukloniti nekorištene grafove te izbrisati nekorištena svojstva i veze.

Kako bi se problemi izbjegli kod konfiguracije se preporučuje korištenje procjene memorije kako bi se saznali zahtjevi memorije. Također je preporučljivo konfiguriranje Neo4ja da koristi hrpu (engl. *heap*) što više, da se algoritmi pokreću na jednoj instanci te također da se ne pokreću na klasterima. Kod projekcije grafova potrebno je učitavati samo one čvorove i veze koje će se koristiti, učitavaju se samo potrebna svojstva i izbjegava projekciju suvišnih, to jest sličnih veza. Katalog se koristi ako je više algoritama pokrenuto na istom grafu, tada se ispuštaju, to jest brišu grafovi koji više nisu potrebni, jedan veći graf unutar memorije je bolji od većeg broja manjih.

Popis slika

2.1 Primjer relacijske baze 4
2.2 Primjer graf baze podataka 5

3.1 Neo4j Browser 7
3.2 Neo4j Bloom 7
3.3 Neodash 8
3.4 Graph Data Science Playground 8
3.5 Korak 1. 10
3.6 Korak 4. i 5. 10

4.1 CSV datoteka 12
4.2 Primjer kako bi se podatak Grad (engl. *City*) uveo u bazu 12

5.1 Neusmjereni graf 13
5.2 Usmjereni graf 14
5.3 Težinski graf 14
5.4 Bipartitni graf 15

6.1 Izrada čvorova Startupi (engl. *Startups*) i Osnivači (engl. *Founders*) 21
6.2 Izrada čvorova Kategorija (engl. *Category*) i Grad (engl. *City*) . . . 21
6.3 Lista uspješno stvorenih čvorova 21
6.4 Grafički prikaz čvora Startupi (engl. *Startups*) 22
6.5 Upiti za kreiranje veza U_GRADU (engl. *IN_CITY*) i POZNAJE (engl. *KNOWS*) 22
6.6 Lista uspješno stvorenih veza 22
6.7 Grafički prikaz veze POZNAJE (engl. *KNOWS*) 23
6.8 Grafički prikaz cijelog grafa baze podataka 23
6.9 Projekcija grafa i primjena algoritma stupnja centralnosti 24
6.10 Rezultati primjena algoritma za investitore s najviše ulaganja 24
6.11 Projekcija grafa i primjenu algoritma stupnja centralnosti. 24
6.12 Rezultati primjene algoritma na godinama 25
6.13 Projekcija grafa i primjena algoritma sličnosti čvora 25
6.14 Rezultati primjene algoritma na investitorima 26
6.15 Projekcija grafa i primjena algoritma ranga stranice 26
6.16 Rezultati primjene algoritma na kategorije i startupa 26
6.17 Projekcija grafa i primjena Louvain algoritma 27
6.18 Rezultati primjene algoritma na osnivačima 27

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*)