PROJEKTNA DOKUMENTACIJA

Stvaranje grafa, Clustering

Voditelj:

Zagreb, kolovoz 2020.

Nakon početnog grupnog istraživanja grafova znanja i termina BKT, DKT, IRT i sl., dio grupe se odvojio na dubinsko istraživanje BKT-a, a dio na proučavanje koncepata stvaranja grafa znanja. U kasnijoj fazi, nakon pronalaska i refaktoriranja obećavajućeg modela preporuka zadataka pod nazivom ExRec, definiran je i zadatak vizualnog grupiranja zadataka koji pripadaju različitim konceptima. Među zadacima stvaranja grafa i grupiranja primijećene su velike korelacije; njihovo nadovezivanje i upotpunjavanje.

1. Stvaranje grafa

1.1. Početna istraživanja

Većina prvih proučenih radova i modela je napuštena zbog nepoklapanja s glavnom idejom našeg projekta, prevelike matematičke kompliciranosti ili kompleknosti računarske izvedbe te zbog namjernog ili nenamjernog uskraćivanja informacija o pozadini funkcionalnosti u objavljenim radovima.

1.1.1. Obrada prirodnog jezika

Početna istraživanja mogućnosti stvaranja grafa odvela su nas u smjeru metoda s korištenjem NLP-a (engl. *natural language processing*). Iako je smjer ocijenjen kao prezahtjevan za potrebe projekta, proučene su osnovne funkcionalnosti Python biblioteka koje olakšavaju takve postupke - spaCy, networkX, seaborn (1). NetworkX je kasnije korišten za vizualizaciju grafova u BKT-u, kao u i Lentil pokušajima grupiranja zadataka po konceptima.

1.1.2. K12EduKG

Za edukacijske aplikacije pronađen je K12EduKG, sustav automatske konstrukcije grafa znanja gdje čvorovi i veze predstavljaju međusobno povezane koncepte (2), no

nejasan je način pretakanja te ideje u kod koji bi odgovarao našem problemu. Autori daju premalo iskoristive informacije. Odlučeno je nastaviti istraživanje u drugačijim smjerovima.

Novi smjerovi istraživanja od tog trenutka uključuju Deep Generative Models i GraphRNN. Proučeni su općeniti koncepti metoda nepovezani s edukacijskim aplikacijama kako bi poslužili kao inspiracija za nadogradnju na naš problem.

1.1.3. Deep Generative Models

Model je temeljen na klasi modela grafičkih neuronskih mreža (engl. *graph nets*). Uči reprezentaciju grafova; čvorova i veza prema propagaciji informacije. Sekvencijalno generira nove strukture (čvor ili vezu). Generiranje je slijed odluka o dodavanju gradivnih dijelova strukture predstavljen vjerojatnostima u zasebnim modulima:

- dodati čvor ili ne,
- dodati vezu ili ne,
- zabrati jedan čvor da se spoji s nekim novim.

Drugačiji poredak struktura označava različite odluke. Za graf se koristi tzv. vektor ugradnje čvorova (engl. *node embedding vector*). Računaju se "skrivena stanja" iz ulaza čvorova i propagiraju se grafom za dobivanje informacije lokalnog susjedstva. Vektor poruke računa se za svaku vezu (pomoću potpuno povezane neuronske mreže, GRU ili LSTM) pa svaki čvor dobiva tu informaciju i osvježava svoj prikaz (3). Spominje se i mogućnost uključivanja uvjetne informacije za proces generiranja. Kao i u prethodnom primjeru, autori ne objavljuju sve potrebne informacije. Oblik vektora ostaje nepoznat, kao i način izračuna "skrivenih stanja" za vektor ugradnje.

1.1.4. Graph RNN

Autoregresivni model. Autoregresivni modeli su sekvencijalni modeli, ali i s unaprijednom propagacijom; generativni, ali i pod nadzorom. Imaju velik potencijal kao alternativa povratnim neuronskim mrežama i GAN-ovima, generativnim suparničkim

mrežama (engl. *generative adversarial newtorks*) za obavljanje generiranja (4). Izlaz su im prediktivne uvjetne vjerojatnosti $P(x_{t+1}|x_1, \ldots, x_t)$. Uvjetovanje je moguće samo na podatcima (ne npr. na šumu kao kod GAN-ova). Konkretno, kod Graph RNN dijelovi matrice susjedstva generiraju se sekvencijalno (npr. jedan po jedan stupac) pomoću RNN. Daljnjim proučavanjem prepoznati su mnogi nedostaci ovog modela. Složenost je O(N^2), gdje je N broj čvorova. Nadalje, zbog sekvencionalnosti, dva bliska čvora grafa mogu biti jako udaljeni u procesu generiranja u RNN što otežava performanse. Prilično je bitno osigurati invarijantnost na permutacije u čvorovima zbog izračuna vjerodostojnosti. Jedan od nedostataka pristupa svakako je i korištenje BFS algoritma (engl. *breadth-first search*) za redanje čvorova u grafu; vrlo efikasnog računski, ali neoptimalnog.

1.2. Graph Recurrent Attention Networks

1.2.1. Općenito o GRAN-ovima

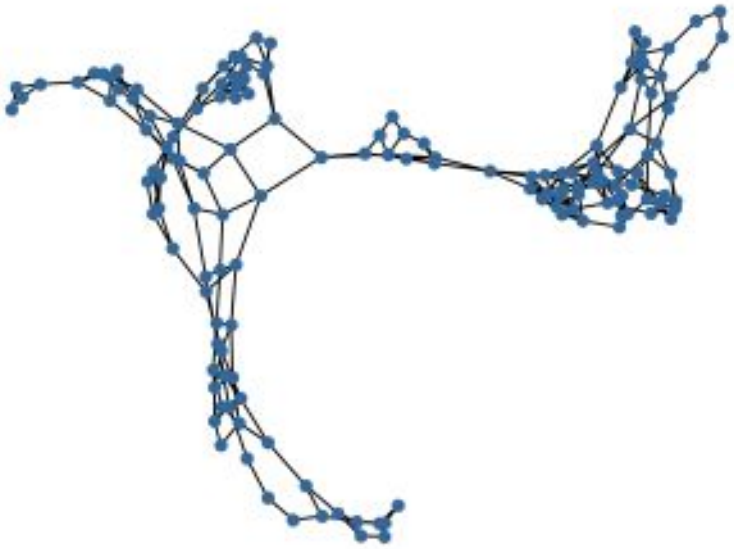
Istraživanjem rada (5) otkriveno je da gradi graf blok po blok. Mijenjanjem veličine blokova moguće je balansirati između kvalitete i efikasnosti. Blokovi predstavljaju određen broj redaka (stupaca) u matrici susjedstva (zadano je 2). Koriste se GNN koje bolje shvaćaju autoregresivnu povezanost (u odnosu na RNN) već generiranih dijelova grafa i onih koje još treba generirati.
Izlazna distribucija parametrizirana je korištenjem Bernoullijevih mješavina (korelacije generiranih veza unutar bloka).
Obećavajuća prednost ovog modela je što rješava neke probleme spomenutih GraphRNN. Složenost je manja, konkretno $O(N)$. Isto tako, GNN bolje razumije topologiju grafa; odluke o generiranju trenutnog bloka donosi izravno ovisno o strukturi grafa, ne koristi gore problematična "skrivena stanja" i efektnije modelira kompleksnost redanja čvorova.
Kao mogući problem prepoznata je općenita evaluacija generativnih modela.

1.2.2. Tijek rješavanja

Početni problem lokalne instalacije PyTorcha i potrebe za instalacijom kontradiktornih *requirementsa* riješen je prelaskom na Google Colab. Originalni proučeni program generira grafove proteina. Čvovori predstavljaju aminokiseline koje ih izgrađuju. Spajanjem čvorova vidljivo je kako se aminokiseline povezuju. Ideja je bila preinači originalni kod kako bi čvorovi predstavljali zadatke, a veze definirale povezanost zadataka unutar jednog koncepta.

1.2.3. Rezultati

Prilagođavanjem broja čvorova i epoha treniranja, popravljena je testna greška .pth datoteke koja sadrži informaciju o konfiguraciji. Vizualiziran je prikaz grafa proteina, što je vidljivo na slici 1.1:



Slika 1.1: Graf proteina

Nije odgovarala činjenica da je prikaz molekula proteina zapravo neusmjereni graf pa je matrica susjedstva čvorova simetrična, što je upola manje računskog posla od onog koji bi bio potreban za naš projekt jer se uzima samo donji trokut u izračunima. Cilj našeg projekta je dobiti usmjereni graf. U radu se spominje kako je moguće napraviti preinake matrice, izračunati i gornji trokut. Također, zbog specifične forme

dataseta, prilagodba na vlastiti dataset bila bi mukotrpna. Od pristupa se odustalo jer su pronađeni oni koji više obećavaju i kojima je potrebno manje prilagodbe.

1.3. Latent Skill Embedding (Lentil)

1.3.1. Općenito o Lentil-u

Početno je istražen i opisan u Izvještaju prvog tjedna, staviti poveznicu na gornji dio teksta (**?**). Daljnjim istraživanjem pokazuje velik potencijal. Metoda je upravljana podacima, uči reprezentaciju sadržaja koja ne zahtijeva *a priori* znanje preslikavanja *content-to-concept*. Proširuje ideje *sparse factor analysis* (SPARFA) i multidimenzionalne Item Response Theory.
Model uči prikaz korisničkog znanja, kao i edukacijski sadržaj kako bi se prikazale preporuke, personalizirane upute o učenju. Probabilistički je model. Problem je formuliran kao regularizirani *maximum-likelihood embedding* korisnika, nastavnih cjelina (lekcija) i procjena znanja (ispita) u zajednički semantički prostor iz skupa podataka o prošlosti interakcije korisnik-sadržaj (*access traces*). Nastavne cjeline i procjene znanja su moduli sadržaja. Fiksni su, a korisnici imaju putanje latentnim prostorom vještina. Stvara se multidimenzionalno okružje - studentsko znanje "leži" u kontinuiranom prostoru stanja, a preduvjeti lekcija moduliraju dobitke znanja lekcijskih modula. Ugradnja u taj prostor ne koristi simetrične udaljenosti komponenata, već se bilježi prirodni napredak težine procjene znanja i rasta korisničkog znanja.
Korisnik je prikazan kao skup latentnih vještina, nastavna cjelina kao vektor dobitaka vještina i skupa preduvjeta, a procjena znanja kao skup potrebnih vještina. Korisniku se procjenjuje znanje nekog modula (ne zna ili zna, 0 ili 1), a vjerojatnost da će proći je veća ako korisnik ima visoku razinu vještine koja nadmašuje potrebne vještine za procjenu. Korisnik može poboljšati vještinu vremenom (koje je diskretizirano). Naglašava se da bi za poboljšanje vještine trebalo gledati korisničko predznanje (ovisno o predznanju podešavati težine u jednadžbi modela).

1.3.2. Ideje

U radu je istražena i implementirana ideja određivanja slijeda lekcija pri praćenju znanja što se ispostavilo kao veoma primjenjivo na naš problem. Temelj eksperimenata s ovim modelom bila je vizualizacija slijeda zadataka. U kasnijoj etapi projekta, povratkom na ovaj obećavajući model, u originalni se kod implementiralo grupiranje zada-

taka kada su poznate pripadnosti koncepata i zadataka. Velika početna prednost modela je što koristi pronađeni Assistments dataset koji je prvi pronađen u skupu projekta i koji služi kao glavni temelj za usporedbu. Razmišljalo se na koji način je prigodnom veličinom dataseta moguće osigurati dovoljnu varijabilnost puteva učenja.
U originalnim .ipynb bilježnicama primijećene su mnoge instance korisničkih putanja koje dijele iste lekcijske module na početku i module procjene na kraju, ali se sastoje od različitih lekcija tijekom učenja → stvaraju se mjehurići, *bubbles*, koji su zapravo eksperimentalni dokaz dobrih svojstava različitih pristupa učenju. Mjehurići se koriste za evaluaciju sposobnosti ugradnje preporuke slijeda lekcija koja vodi do uspješnog ostvarenja ciljeva učenja. Model logističke regresije s L2 regularizacijom korišten za procjenu vjerojatnosti da će korisnik slijediti preporučenu "granu" mjehurića. Unutar svakog mjehurića, korisnici koji su krenuli preporučenim putem spojeni su s najbližim susjedom iz grupe onih koji nisu slijedili preporučeni put prema razlici u *propensity scoreu*.

1.3.3. Problemi i zadaci

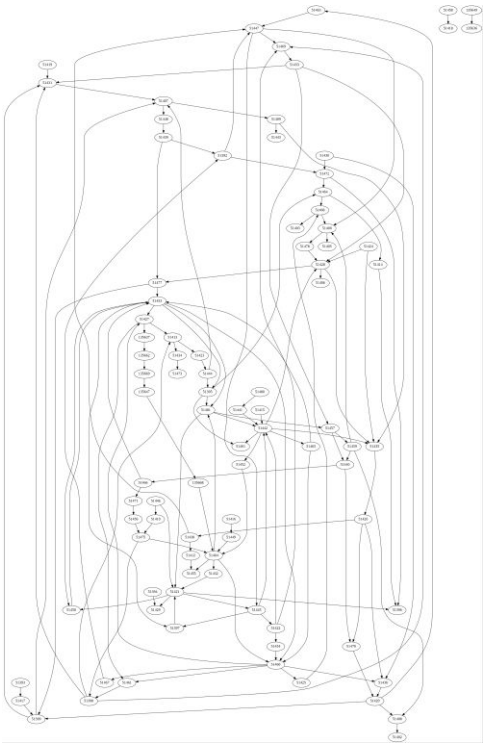
Kao početni potencijalni problem izvedbe modela pokazala se činjenica da su ocjene znanja modelirane na različitim osima grafičkih prikaza što za sobom povlači neovisnost lekcija i vještina, što ne mora u stvarnosti biti slučaj. Također, u prikazu putanja, u originalnim ispitnim podacima u radu vidljiva je pristranost korisničkih putanja jednoj vrsti temeljne, česte putanje zbog sličnosti pozadinskog znanja ispitanika.
Isto tako, pronađena je greška u originalnom kodu; činjenica da se prijeđene putanje korisnika ne broje na odgovarajući način (iteriranje liste tupleova bezuspješno, uvijek 0).
Pri početnom ostvarivanju vizualizacije slijeda zadataka kao problem je okarakterizirana činjenica da crta vektore, a ne pravi graf koji "povezuje" korisnike preko procjena ovisno o vještinama dobivenih nakon neke lekcije (više ukazuje na smjer potencijalnog puta, što isto donekle odgovara cilju projekta).
Izrađena je prilagodba na Google Colab, koja se pokazala izazovnijom od očekivanog zbog zastarjelosti originalnog koda i činjenice da je Google Colab službeno ukinuo podršku za Python 2.7 (iako je ipak i dalje moguće pokretati sadržaj).
Nakon uspješne vizualizacije slijeda rješavanja zadataka Assistments dataseta, ideja je bila pokušati prilagoditi model našem "Biologija" datasetu. Potrebno je bilo riješiti problem prisutnosti varijabli trajanja i vremenskih koraka u originalnim funkcijama jer mi to ne gledamo i ne trebamo te spajati čvorove zadataka ovisno o pripadnosti

pojedinom konceptu ili još bolje, spajati koncepte.

1.3.4. Rezultati

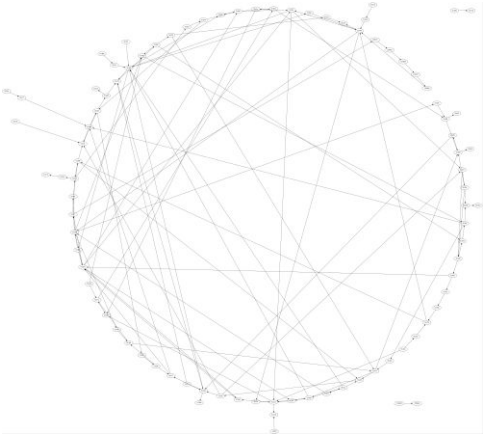
Nakon početnog ispravljanja grešaka u kodu, dobiven je string koji označuje povezanosti čvorova dataseta na način da su strelicom povezani zadaci koje jedan korisnik rješava jedan za drugim prema pojavi u datasetu. Vizualizacija toga na cijelom datasetu generirala se preko 8 sati jer PyGraphviz, Pythonovo sučelje Graphviza (alata za crtanje grafova), ne iskorištava mogućnosti ubrazanja u Google Colabu. Odlučeno je prilagoditi i smanjiti dataset (uzeti prvih 200 redaka). Nakon toga uspješno je izvršena vizualizacija dvije vrste grafova koji prikazuju povezanost zadataka:

- graf tijeka (engl. *flow grah*) - zadaci Assistment dataseta predstavljaju čvorove grafa koji se povezuju ovisno o korisničkom pristupu konkretnom problemu. Prikaz je vidljiv na slici 1.2.
- graf veza (engl. *connection grah*) - čvorovi su i korisnici i zadaci kojima oni pristupaju. Prikaz je vidljiv na slikama 1.5 i 1.6.

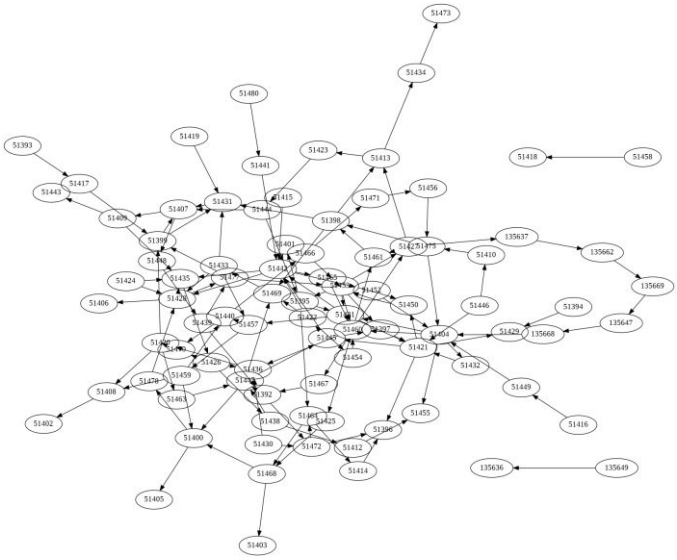


Slika 1.2: Originalni "dot" graf tijeka s podacima smanjenog Assistment dataseta

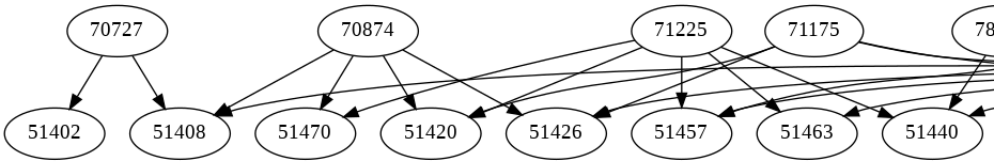
Eksperimentiranjem s PyGraphvizom moguće je dobiti različite vizualne prikaze istog grafa (odnosno rasporede - dot, neato, twopi, circo, fdp, sfdp), pa je ovisno o namjeni moguće izabrati najprikladniji. Neki od prikaza vidljivi su na slikama 1.3 i 1.4.



Slika 1.3: Originalni "circo" graf tijeka s podacima smanjenog Assistment dataseta



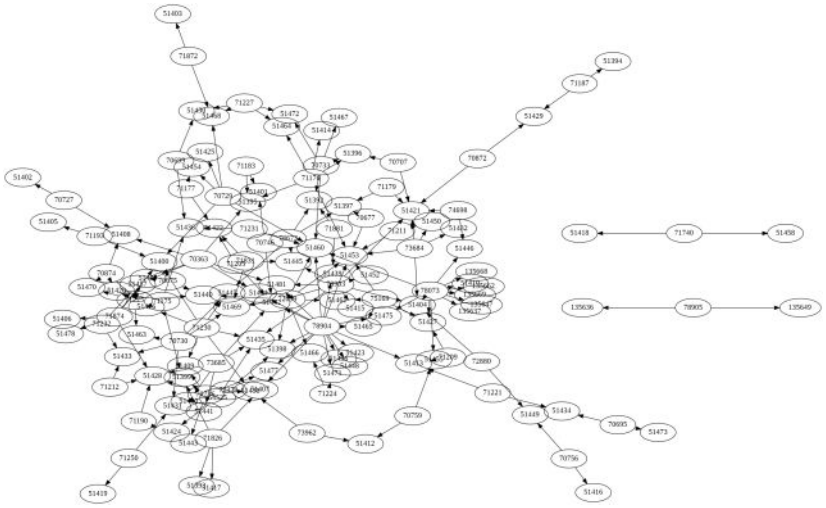
Slika 1.4: Originalni "sfdp" graf tijeka s podacima smanjenog Assistment dataseta



Slika 1.5: Originalni "dot" graf veza s podacima smanjenog Assistment dataseta

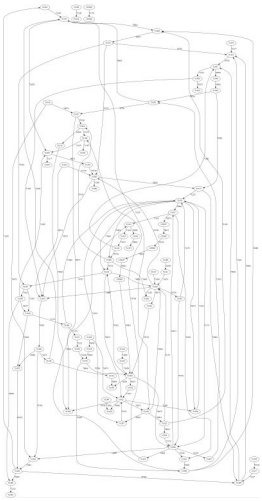
Nakon ovakvog prikaza moguće je bilo zaključiti prednosti i nedostatke pojedine vrste prikaza.

Graf veza mnogo je manje intuirivan i manje pregledan od grafa tijeka. Za daljnje eksperimentiranje, kao prikladniji graf uzet je graf tijeka zbog veće preglednosti i veće



Slika 1.6: Originalni "sfdp" graf veza s podacima smanjenog Assistment dataseta

prilagođenosti našem problemu. No, iako je preglednost znatno bolja nego kod grafa veza, svejedno nije u potpunosti zadovoljen kriterij preglednosti. Zadaci različitih korisnika nisu obojeni različitim bojama, a ni pristup samim zadacima nije kronološki. Za intuitivniju vizualizaciju putanja dodane su oznake pojedinih korisnika između čvorova, što je vidljivo na slici 1.7. Ideja je poboljšati dodavanjem različitih boja svakom korisniku.



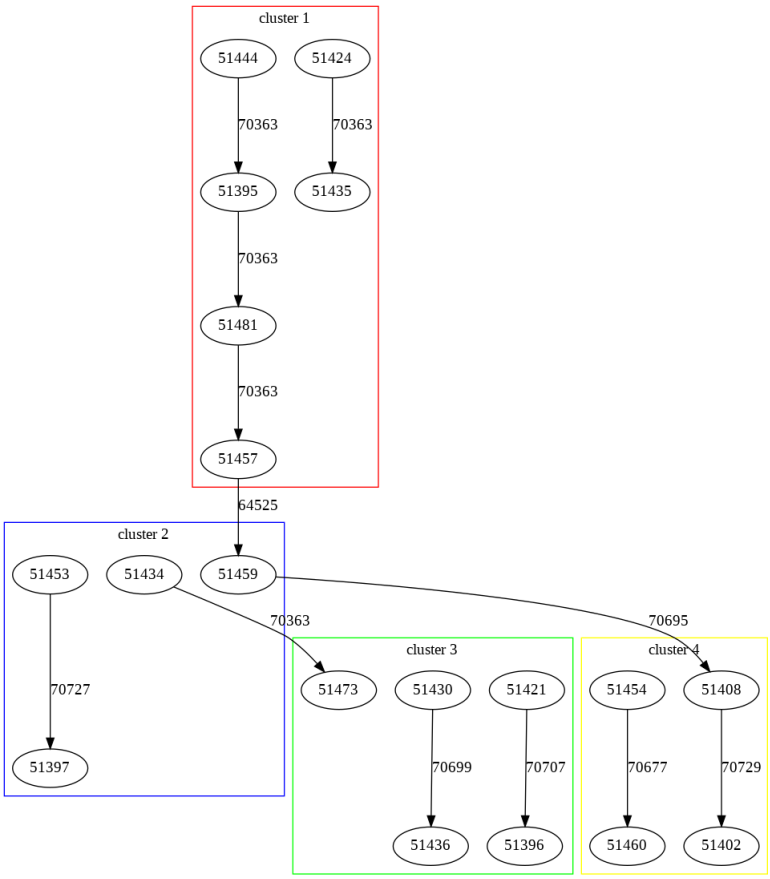
Slika 1.7: "dot" graf tijeka s podacima smanjenog Assistment dataseta uz dodane oznake korisnika

2. Clustering

Početno se razmatralo na osnovu čega bismo u modelu preporuka mogli napraviti ostvarenje vizualizacije grupiranja (engl. *clustering*) zadataka u koncepte ako nemamo označenu eksplicitnu pripadnost. Kao moguća ideje spominjao se redoslijed rješavanja, no zbog nerealnosti takve situacije najbrže je odbačena. Dodatno, moglo bi se gledati točnost pojedinih odgovora velikog broja ispitanika.

2.1. Latent Skill Embedding (Lentil)

Nakon kratkotrajnog napuštanja Lentil modela zbog bavljenja modelima koji više obećavaju, napravljen je povratak u sklopu istraživanja za zadatak grupiranja zadataka ako nemamo eksplicitno definiranu pripadnost konceptima.
Prvo je istražena mogućnost vizualnog grupiranja zadataka u okviru Lentil koda ako imamo pripadnost konceptima. Takva vizualizacija služila bi za usporedbu točnosti s modelom koji bismo osmislili. To je uspješno izvršeno. Dodan je stupac informacije o konceptima dataseta Assisstments u kod. Smanjenom datasetu za probu (20 pitanja) dodijeljene su različite oznake koncepata te je izvršeno grupiranje zadataka po konceptima izmjenom funkcije koja stvara graf tijeka manipulacijama s Pandasom (čvorovi su zadaci, na vezama oznake korisnika koje su ih riješile). Vizualizacija se nalazi na slici 2.1.
Isti rezultat dobiven eksperimentom s drugačijim pristupom (izračunom matrice prijelaza između stanja (čvorova) Markovljevog modela). Grupiranje zadataka je identično kao na slici 2.1, ali je prikaz manje interpretabilan jer u ovoj fazi ne prikazuje oznake korisnika. Vidljiv je na slici 2.2.
Konkretno, koristi se provjera različitosti pojedinih pitanja. Svaki zadatak teorijski je modeliran kao stanje Markovljevog procesa (diskretni stohastički proces je Markovljev proces ako vjerojatnost prijelaza u određenom diskretnom trenutku ovisi o stanjima u svim trenutcima prije). Stvara se matrica prijelaza između stanja (čvorova koji predstavljaju pitanja). Na osnovi takve matrice crta se graf. Kasnije se računa matrica vje-



Slika 2.1: Grupiranje zadataka smanjenog Assistments dataseta prema pripadnosti konceptima

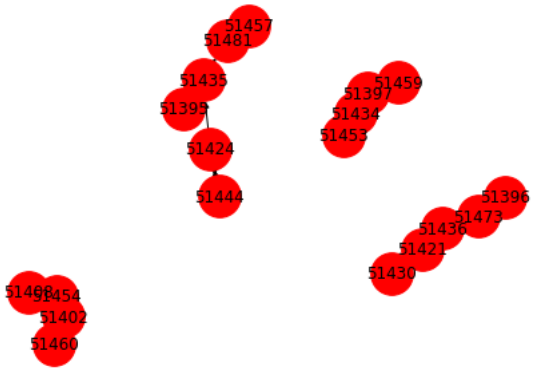
rojatnosti prijelaza stanja pa stacionarna distribucija (vjerojatnosna distribucija koja se ne mijenja u Markovljevom lancu kako vrijeme prolazi).

Ponovno čitanje rada (**?**) inspiriralo je za ideju korištenja izračuna entropije za određivanje koncepata. Entropija se računa kao negativni skalarni produkt stacionarne distribucije i $Plog(P)$, gdje P predstavlja matricu vjerojatnosti prijelaza među stanjima. Sama korist izračuna entropije još nije prikazana. Entropija inače predstavlja "porast informacije u podatcima". Potencijalno, ideja je da, s obzirom da se računa za svaku matricu vjerojatnosti prijelaza, za svaku putanju različitih korisnika računati entropiju i utvrditi postoje li pravilnosti koje pomažu odrediti koncepte.

2.2. K-Medoids

Kao odvojeni pristup, napravljena je skripta koja radi clustering pomoću algoritma K-Medoids.

Kao vektor, odnosno element dataseta koji treba grupirati, uzeta je lista odgovora svih ispitanika na određeno pitanje (jedan element predstavlja jedno pitanje). Za metriku



Slika 2.2: Grupiranje zadataka smanjenog Assistments dataseta prema pripadnosti konceptima izračunom matrice prijelaza

udaljenosti napravljena je funkcija koja gleda sumu razlika dva vektora (gleda se razlika između $x[i]$ i $y[i]$ za $i \in [0, brojispitanika - 1]$). Može se uzeti i kvadratni korijen te sume.
Definirana je i funkcija koja računa grešku kao sumu elemenata koji nisu dobro grupirani. Uzeto je da je "prava" klasa ona kojoj pripada najviše elemenata iz tog koncepta, a oni koji nisu u njoj pridodani su ukupnoj sumi greške. Grupiranje se radi na standardiziranim i nestandardiziranim podacima i za obje verzije računaju se *rand index*, *adjusted rand index*, NMI vrijednost (*normalized mutual information*) te ranije spomenuta funkcija koja računa broj krivo grupiranih primjera.
Napravljena je i funkcija koja metodom lakta određuje idealan broj klasa, u slučaju da on nije otprije poznat.

Za različite datasetove skripta pokazuje funkcionalnost na različite načine.
Biologija dataset:
Za sad je algoritam isproban samo na datasetu Biologija (84 odgovora) jer je Assistments malo prevelik, ali sljedeći korak je testiranje i na Assistments.
Dataset ima 30 pitanja, znači imamo 30 ulaznih vektora. Algoritam je isproban na standandiziranim i nestandardiziranim podacima, te daje nešto malo bolje rezultate na nestandandiziranim, ali to može biti i do dataseta. Rand index na nestandardiziranim podacima je 0.83, a adjusted rand index 0.41, za što kažu da spada u "srednje dobar rezultat". NMI vrijednost je oko 0.58, a minimum koji se dobiva funkcijom za sumu greške jest 9 (od ukupno 30 pitanja). Lošiji rezultati mogu biti i do dataseta Biologija koji je svakako daleko od idealnog, jer ni koncepti nisu dobro definirani (npr. ispitanik možda ne zna živčanu stanicu, ali je čuo odgovor na par pitanja iz psihologije). Ko-

rištenjem metode lakta za određivanje broja klasa dobije se 6 klasa, dok je pravi broj klasa 5.
Assistments dataset:
Testiranje na Assistments u početku je okarakterizirano kao nemoguće jer K-Medoids algoritam zahtijeva da su svi vektori iste dimenzionalnosti odnosno da su svi ispitanici odgovorili na sva pitanja. To u Assistments nije zadovoljeno. Eventualno se može probati s nekakvom redukcijom dimentionalnosti vektora (npr. *principal component analysis*, međutim budući da ima pitanja na koja je samo jedan student odgovorio, na kraju bismo trebali završiti s dimenzionalnošću 1 (što je puno premalo da bi obuhvaćalo potrebne informacije).
U nastavku rada na ovom pristupu, napravljena je prilagodba za učitavanje smanjenog Assistments dataseta (20 pitanja). No, početna pretpostavka da testiranje s Assistemntsom neće biti moguće pokazala se ispravnom. Isprobana je metoda grupiranja slična K-Medoids, K-Means, ali nije pokazala željenu funkcionalnost.

2.3. Ostatak istraživanja

Mnogi radovi pokazali su se u cjelovitosti ne toliko relevantnima da bismo ih mogli u potpunosti primijeniti na naš problem, ali su predstavili mnoge iskoristive koncepte i algoritme vrijedne spomena.

U radu (10) spominje se da je razumijevanje krajnjeg cilja svakog studenta praćeno pomoću posebne varijable mastery_score koja se osvježava svaki puta kada se vrši procjena studentovog znanja.
Spominje se da identificiraju i vizualiziraju grupe povezanih koncepata. Točnije, *spreading algorithm* pronalazi uzorak u povezanosti koncepata i dokumenata te dokumenata i koncepata. Analizirana je struktura *spreading algorithma* (11).
Problem je što je potrebna neka vrsta reference, ekspertno određenog početnog grafa koji će se evaluirati tim algoritmom.

U radu (**?**) predstavljene su hijerarhijske veze pitanja i koncepata modelirane pomoću hinge loss skalarnog produkta ugrađenih koncepata i pitanja. Problem: povezanost koncepata i pitanja određena ekspertima i zapisana u matrici ugradnje, a eksperte želimo izbjeći. Korišten je t-distributed stochastic neighbor embedding (tSNE) algoritam. tSNE je algoritam strojnog učenja za vizualizaciju; nelinearna tehnika za redukciju dimenzionalnosti. Koristi se za ugradnju visokodimenzionalnih podataka

za vizualizaciju u niskodimenzionalni prostor (2 ili 3 dimenzije). Slični objekti su s visokom vjerojatnosti modelirani točkama koje su blizu u prostoru, a različiti objekti udaljenim točkama (**?**). Problem: velika osjetljivost algoritma na promjenu parametara.

Rad (**?**) primarno radi poveznicu između samih koncepata, ali i između koncepata i objekata učenja (zadataka, primjera, itd.) uz pretpostavke da je broj objekata učenja poznat. Glavni problem i razlog napuštanja ovakvog modela je što se *preprocessing* objekata učenja radi se pomoću *text-mininga*. Isto tako, rad pretpostavlja da je netko nekad označio povezanost koncepata i objekata učenja pa da može koristiti tu referencu. Nad tom referencom oblikuje se tzv. *contextual network* koja sadrži više vrsta čvorova. Za izračunavanje sličnosti koncepata koriste se dva načina: već spomenuti *spreading activation algorithm* i *PageRank with Priors*. *PageRank with Priors* koristi se kao kvantitativna mjera sličnosti čvorova prema nekom drugom čvoru. Čestu primjenu ima u sustavima kategorizacije. U konkretnom primjeru, propagiraju se stvarne karakteristike modela domene znanja u eksplicitne veze među konceptima.

U radu (**?**) veoma je izraženo shvaćenje da mnogi radovi poistovjećuju koncepte i zadatke i to pojednostavljenje uzrokuje probleme.
Prepoznaju da grafovi znanja ciljaju da predstave znanje u obliku grafova tripleta; triju činjenica - (početni entitet, veza, završni entitet).
Razlikuju povezanost koncepta i zadataka, koncepata međusobno te zadataka međusobno. Ovisno o vrsti povezanosti, koriste različite funkcije gubitka.
Koncepte enkodiraju kao sfere, a zadatke kao vektore u istom semantičkom prostoru.
Metoda je evaluirana pomoću predikcije veza i klasifikacije tripleta. Potonja ocjenjuje točnost tripleta. Predikcija veza predviđa početni ili završni entitet iz triplea, ovisno koji nedostaje. Algoritmu je potrebno dati rangirane preporuke zadataka, ne samo najbolji rezultat. Sve mogućnosti se rangiraju kao moguće upopunjavanje traženog praznog mjesta prema udaljenosti u određenoj funkciji gubitka. Za evaluaciju koriste se dvije mjere:

1. srednja recipročna mjera svih točnih instanci (MRR) i
2. razmjer točnih zadataka koji rangiraju ne više od N (Hits@N).

Glavni problem, kao i u većini istraženih metoda: znaju koliko je koncepata.

Rad (**?**) slične vježbe grupira u koncepte naziva “problem schemas”. Koristi se

hierarchical graph neural network. Konkretno, ovdje je mreža dvoslojna: donji je sloj zadataka (svaki čvor je jedan zadatak), gornji je sloj problemskih shema. Povezanost zadataka i problemskih shema modelira se *assignment matrixom* koji se dobije pomoću *hierarchical clustering analysis (HCA)*. HCA je metoda analize grupiranja bez nadzora korištenjem aglomerativnih ili divizivnih strategija za izgradnju hijerarhije grupa. Za konstrukciju hijerarhijskog grafa zadataka spominje se iskorištavanje semantičke informacije zadataka i shema, ali kasnije se spominje treniranje mreže samo pomoću HCA pa je i to jedan od pristupa koji valja detaljnije istražiti.

Istražen je i alat Ampligraph (**? ?**). Površna analiza definirala ga je kao dosta moćan alat. Funkcionalnosti obuhvaćaju stvaranje grafa znanja, treniranje ugradbenog modela na tripletima, evaluaciju modela, vizualno grupiranje pojmova. Problem: s obzirom da koristi triplete (subjekt, predikat, objekt), potrebno je poznavati povezanost pitanja.

3. Literatura

[1] https://www.analyticsvidhya.com/blog/2019/10/how-to-build-knowledge-graph-text-using-spacy/

[2] K12EduKG, https://aic-fe.bnu.edu.cn/docs/20181205101832069569.pdf

[3] Deep Generative Models, https://arxiv.org/abs/1803.03324

[4] https://eigenfoo.xyz/deep-autoregressive-models/

[5] Graph Recurrent Attention Networks, https://arxiv.org/abs/1910.00760

[6] Latent Skill Embedding (Lentil), https://arxiv.org/pdf/1602.07029.pdf

[7] http://math.ubbcluj.ro/~tradu/TI/coverch4_article.pdf

[8] https://www.slideshare.net/bamparopoulos/entropy-based-measures-for-graphs

[9] https://www.educationaldatamining.org/EDM2015/proceedings/short360-363.pdf

[10] https://www.researchgate.net/publication/333828640_INTERACTIVE_LEARNING_IN_A_CONVERSATIONAL_INTELLIGENT_TUTORING_SYSTEM_USING_STUDENT_FEEDBACK_CONCEPT_GROUPING_AND_TEXT_LINKING

[11] https://en.wikipedia.org/wiki/Spreading_activation

[12] Deep Hierarchical Knowledge Tracing, http://www.personal.psu.edu/ffm5105/files/2019/edm19.pdf

[13] https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding

[14] Automatic Concept Relationships Discovery for an Adaptive E-course, https://www.educationaldatamining.org/EDM2009/uploads/proceedings/simko.pdf

[15] Differentiating Concepts and Instances for Knowledge Graph Embedding, https://www.aclweb.org/anthology/D18-1222/

[16] HGKT : Introducing Problem Schema with Hierarchical Exercise Graph for Knowledge Tracing, https://arxiv.org/pdf/2006.16915v2.pdf

[17] Ampligraph, https://docs.ampligraph.org/en/1.3.1/

[18] https://github.com/Accenture/AmpliGraph/blob/master/docs/tutorials/ClusteringAndClassificationWithEmbeddings.ipynb