Sveučilište u Zagrebu
Fakultet Elektrotehnike i računarstva

Dora Pavelić

Metoda samonadziranog učenja za hladni start sustava preporuke znanstvenih članaka temeljena na slučajnim šetnjama

Zagreb, lipanj 2022.

Ovaj rad izrađen je u sklopu studentskog istraživačkog rada pod vodstvom doc. dr. sc. Marija Brčića, predan je na natječaj za dodjelu Rektorove nagrade u akademskoj godini 2021./2022.

Sadržaj

1 Uvod

Razvojem tehnologije i digitalnog svijeta sve je veći broj informacija nadohvat ruke. Kako bi iskustvo pregledavanja i pretraživanja mnoštva informacija bilo što ugodnije i jednostavnije, potrebno je na neki način isfiltrirati sav sadržaj i prikazati samo određene relevantne stavke. U tome nam uvelike pomažu sustavi za preporuku sadržaja. Sustavi za preporučivanje za cilj imaju korisnicima preporučiti sadržaj koji bi za njih mogao biti relevantan i u nekim slučajevima otkriti njima zanimljiv sadržaj koje sami ne bi pronašli. Sustavi za preporučivanje postali su dio našeg svakodnevnog života te ih nesvjesno susrećemo pri izboru filma, glazbe, kupovine na internetu, na društvenim mrežama.

Postoje četiri vrste metoda sustava za preporuku sadržaja. Metoda kolaborativnog filtriranja (engl. *collaborative filtering*), metoda filtriranja temeljenog na sadržaju (engl. content based filtering), hibridna metoda te metoda temeljena na grafu (engl. *graph-based method*). Sve te metode razlikuju se po značajkama koje uzimaju u obzir prilikom pretraživanja kandidata za preporuku. Kolaborativno filtriranje se bazira na interakciji između korisnika i sadržaja tako da pamti i bilježi korisničku povijest posjećivanja sadržaja te na temelju tih podataka nastoji prepoznati slične korisnike i preporučiti im sadržaj posjećen od strane sličnih korisnika. Metoda preporuke temeljene na sadržaju prvenstveno prati interakciju korisnik-sadržaj, no uključuje i dodatne informacije o korisniku koje kasnije koriste u rangiranju sadržaja tokom preporuke. Ova metoda generira preporuke koje su često presličine već pregledanom sadržaju. Hibridna metoda kombinira kolaborativno filtriranje i filtriranje temeljeno na sadržaju dok je metoda temeljena na grafu fokusirana na konstruiranje grafa u kojemu bridovi predstavljaju recenzije a za procjenu relevantnosti koriste se šetnje po grafu.

Učinkovitost svih navedenih metoda ovisi o količini podataka koje povezuju korisnike i ponuđeni sadržaj. Za učinkoviti rad potrebna je velika količina podataka koju je često teško dobaviti. Iz tog razloga jedan od glavnih problema je tzv. problem hladnoga početka (engl. *cold start*) koji nastaje kada u sustavu nemamo dostupne informacije o preferencijama korisnika i njihovim interakcijama s ponuđenim sadržajem na temelju kojih bismo mogli generirati preporuke. Nadalje, problemi s kojima se susreću sustavi preporuke su skalabilnost i pozitivna slučajnost. Skalabilnost je mjera koja ocjenjuje može li sustav raditi efikasno s dinamičnom i velikom količinom podataka. Pozitivna slučajnost odnosi se na preporuke koje odstupaju od tipičnih korisničkih preferencija no koje su od velike značajnosti korisniku.

U sklopu ovoga istraživačkog rada razvijena je metoda za memorijski i vremenski efikasno generiranje preporuka znanstvenih članaka koja nudi rješenje za djelomično

rješavanje hladnog starta. Uspjeh ove metode bazira se na konceptu samonadziranog učenja u sklopu kojega je razvijen algoritam ponderirane slučajne šetnje sa iznenađenjem WRS (engl. *Weighted Random Surprise*) i prikladnu arhitekturu duboke neuronske mreže.

Poglavlje 2 predočuje problematiku koja je u sklopu ovoga rada razrađena kao i pretpostavke nužne za daljnje definiranje rješenja. Kroz poglavlje 3 razrađene su faze i metode koje su korištene i razvijene u sklopu ovoga istraživanja. Unutar 4. poglavlja izlažu se dobiveni rezultati WRS metode dok se kroz 5. i 6. poglavlje daje uvid u moguće nadogradnje kao i moguće primjene razvijene metode u komercijalnom rješenju.

2 Hipoteza

Sustavi preporuke zahtijevaju opširnu povijest interakcije korisnika sa sadržajem. Ova interakcija može biti eksplicitna u obliku recenzija ili implicitna u obliku posjećivanja sadržaja. Eksplicitne interakcije daju preciznije, a time i kvalitetnije informacije, no takav oblik interakcije je u praksi teško dobavljiv. Zbog toga se u praksi većinski radi sa implicitnom interakcijom. U slučaju kada sustav nema povijest interakcije ili je ima nedovoljno kao što je to primjer za nove proizvode, nove korisnike ili općenito nove sustave, znatno je teže ponuditi korisniku njemu relevantni sadržaj. Prethodno je aktualan izazov s kojima se susreće svakodnevno mnogo platformi. Kako bi se dao prijedlog mogućeg rješenja koji bi odgovorio na neke navedene izazove, u sklopu ovoga istraživanja razrađen je sustav preporuke znanstvenih članaka.

Sustav preporuke znanstvenih članaka idealan je za testiranje mogućeg rješenja navedenih izazova iz više razloga. U trenutku pisanja ovoga rada, ne postoji javni skup podataka koji veže korisnike sa znanstvenim radovima od interesa, kao što je to slučaj za filmove ili knjige za koje postoje skupovi podataka koji vežu korisnika i film/knjigu putem recenzije. Nadalje, sama količina članaka koja se godišnje izdaje se procjenjuje na preko 7 milijuna primjeraka godišnje [1]. Dakle, radi se o velikoj i raznovrsnoj količini podataka za koju je potrebno konstruirati sustav koji je brz, memorijski ne-zahtjevan i koji ne gleda čistu sličnost preporučenih članaka sa onim već posjećenim već uzima u obzir i neke druge karakteristike.

Predloženo rješenje mora zaobići prepreku nedostatka podataka za treniranje, iskorištavajući pritom prirodne veze i kontekst sadržaja na smislen način. Sljedeći korak, nakon ekstrakcije znanja, mora uključivati sažimanje tog znanja u model koji će se dalje iskorištavati za generiranje preporuka.

Glavna ideja ovakvog sustava za preporuku je preslikavanje sadržaja i korisnika u visokodimenzionalni prostor na način da korisnici i njima relevantan sadržaj budu što bliže u prostoru. U literaturi za ovaj proces smještaja korisnika i sadržaja u visokodimenzionalni prostor možemo naći različite metode koje također uvelike ovise o tipu sadržaja (koji može biti multimedijalan) kao i količini dostupnih podataka.

Rješenje prepreke nedostatka podataka kao i zadaće preslikavanje sadržaja i korisnika u visokodimenzionalni prostor može ponuditi samonadzirano učenje. U sklopu samonadziranog učenja podaci potrebni za trening modela neuronske mreže dobivaju se automatiziranim putem. Nadalje, ukoliko je logika automatiziranog generiranja podataka ispravna, učenjem neuronske mreže nad tim podacima biti će moguće ugrubo smjestiti sadržaj i korisnike u vektorski prostor.

Kako ne postoji gotov skup podataka koji veže stupanj preferencije članka sa pojedinim korisnikom, prvi korak je osmisliti način s kojim se može korisnike povezivati sa skupom članaka. Ideja je preslikati javno dostupan skup citata u graf, gdje vrhove predstavljaju članci a bridove predstavljaju reference. Pretpostavka je da ukoliko je neki članak *c* od interesa nekom korisniku, članci koji su referencirani unutar tog članka ili članci koji referenciraju članak *c* imaju velike šanse također biti od interesa iako možda nisu iste grane znanosti. Što se dalje udaljavamo od početnog članka *c* za koji smo sigurni da je od interesa korisniku, to članci postaju manje relevantni.

Ako je ova prethodna pretpostavka točna možemo generirati skup podataka u kojemu će pojedinačni vrh i njegovo bliže susjedstvo predstavljati interes pojedinog korisnika.

Nadalje, biti će potrebno konstruirati duboki model koji će biti sposoban učiti nad visoko nelinearnim generiranim podacima. Preferencije korisnika biti će određene člancima, točnije tekstom koji čine članci. Za potrebe treniranja neuronske mreže biti će potrebno prikazati te članke vektorima što će biti učinjeno aproksimativnim smještajem u vektorski koristeći već trenirane modele nad tekstom.

Treniranje smještaja interesa korisnika putem neuronske mreže se može svesti na kontrastno samonadzirano učenje u kojemu bi postojale dvije klase, klasa dobrih preporuka i klasa loših preporuka. U literaturi [2], [3] formulacija ovoga zadatka zahtjeva skup podataka koji ima naznačene pozitivne i negativne primjerke. Treniranje neuronske mreže se svodi na učenje približavanja pozitivnih primjera referentnom i udaljavanja negativnih. Ovi skupovi pozitivnih i negativnih primjera biti će dobiveni šetnjama po grafu citata.

Rezultat navedenog pristupa trebao bi biti model koji će za dani referentni tekst uspješno raspoznavati znanstvene članke koji su dobar odabir on onih koji to nisu, uzimajući u obzir i pozitivnu slučajnost. Pristup bi tako ponudio rješenje za problem hladnog starta koji bi uz svoju brzinu i preciznost kao i memorijski utrošak mogao poslužiti i u produkcijskom okruženju.

3 Plan rada i metode

Nakon odluke o korištenom skupu podataka potrebno je definirati način na koji će se generirati skupa podataka sa oznakama potrebnim za treniranje neuronske mreže. S obzirom na postavljene izazove, taj skup biti će izgrađen u sklopu samonadziranog učenja koji će biti opisan u nastavku. Kako bi se iskoristile prave veze i uzorci u generiranim podacima, od velike je važnosti ispravno konstruirati arhitekturu neuronske mreže. Iako se u literaturi mogu naći određeni naputci, svaki problem zahtijeva specifičan pristup stoga je potrebno eksperimentirati sa različitim arhitekturama.

3.1 Metoda samonadziranog učenja

Strojno učenje dijeli se na tri glavne grane; nadzirano, nenadzirano i podržano. Nadziran pristup zahtjeva skup označenih ulaznih podataka na principu {ulazni podatak, željeni izlaz} na temelju kojih algoritam uči klasificirati podatke ili aproksimirati funkciju koju opisuje dani skup podataka. U nenadziranom pristupu ulazni podatci nisu označeni stoga vršimo grupacije (engl. *clustering*) podataka na temelju značajki koje se izvode iz samog skupa podataka. U slučaju podržanog učenja postoje tzv. agenti (algoritmi) koji uče izvršavajući akcije na temelju povratnih informacija koje su definirane pravilima.

U zadnje vrijeme, veliku pozornost privlači metoda samonadziranog učenja (engl. *Self-supervised learning*). Radi se o pristupu strojnog učenja koji je najsličniji nenadziranom učenju, no za razliku od nenadziranog učenja postoji izlazna oznaka. U sklopu samonadziranog učenja, te oznake dobivaju se automatiziranim (umjetnim) putem proizvoljno definiranog pomoćnog zadatka (engl. *auxiliary task*).

Suvremene metode samonadziranog učenja se općenito mogu podijeliti u tri klase metoda: generativne, kontrastne i generativno-kontrastne [4].

- Generativne: učenje neuronske mreže da kodira ulazni podatak *x* u vektor *z, (*drugim riječima smjesti ulaz x u vektorski prostor) zatim učenje rekonstrukcije vektora *x* iz *z.*
- Kontrastne: učenje smještaja ulaznih podataka s ciljem lakšeg raspoznavanja podataka različitih klasa. Kontrastno učenje omogućava učenje generalnih značajki putem usporedbe parova podataka koji su slični (pozitivni primjerci) ili različiti (negativni primjerci).
- Generativno-kontrastne: učenje neuronske mreže imitiranju ulaznih podataka i zatim raspoznavanju umjetno generiranih od pravih.

S ciljem učenja dobrih preporuka (pozitivnih primjera) i loših preporuka (negativnih primjera), zadatak se može svesti na kontrastno učenje koje će biti opisano u nastavku.

Princip je sljedeći, dostupni podaci se proizvoljno modificiraju (pomoćni zadatak) generirajući oznake u samom procesu. Iz razloga što ne moramo ručno označavati podatke, jer je ono automatizirano, imamo mogućnost trenirati neuronsku nad mnogo više podataka. Jednom dobivene oznake dalje koristimo za učenje neuronske mreže reprezentacijama tog skupa podataka. Reprezentacije u kontekstu samonadziranog učenja predstavlja niz značajki koji ugrubo predstavljaju ciljani skup podataka. U ovom slučaju, ciljani skup bili bi podaci o preferencijama pravih korisnika. Uz pomoć naučenih reprezentacija moguće je naknadno učenje modela (dodatno treniranje) nad realnim skupom podataka dobivenim u produkcijskom okruženju uz koje se postiže veća preciznost.

Umjetni zadatak u ovome radu predstavlja generiranje trojke (referentni, pozitivni primjerak, negativni primjerak) koji se dobio šetnjom po utežanom grafu citata. Naknadnim učenjem reprezentacija korištene su za smještaj korisnika i znanstvenih radova u vektorski prostor.

3.2 Kreiranje i adaptacija grafa citata

Skup podataka nad kojim je provedeno istraživanje je javno dostupan skup citata kreiran za svrhe znanstvenog istraživanja [5]. Skup sadrži 5,354,309 članaka i 48,227,950 citata. Od sveukupno 27 atributa, u obzir su uzeti identifikacijski broj članka, pripadne kategorije članka, reference, naslov, sažetak i ključne riječi. Od cijelog skupa 28% članaka nije sadržavalo sažetak koji je ključan za naredne korake zbog čega su uklonjeni. Graf je kreiran uz pomoć specijalizirane biblioteke iGraph python [6].

Pod pretpostavkom da članci koji dijele veći broj kategorija imaju veće šanse biti međusobno relevantniji, definirane su težine nad bridovima koje su se iskoristile tokom generiranja korisnika šetnjom po grafu. Težine između dva članka (i, j) sa kategorijama C_1 i C_2 su dodijeljene po $W_{ij} = max\left(1, \frac{C_1 \cap C_2}{C_1 \cup C_2} * 100\right)$. Nakon ovog postupka, prosječna težina brida iznosila je 14, od toga je 8% bridova imalo težinu 1 što znači da između dana dva znanstvena rada nije bilo niti jedne dijeljene kategorije ili druga mogućnost, da jedan ili oba znanstvena rada nisu imali zapis o kategoriji.

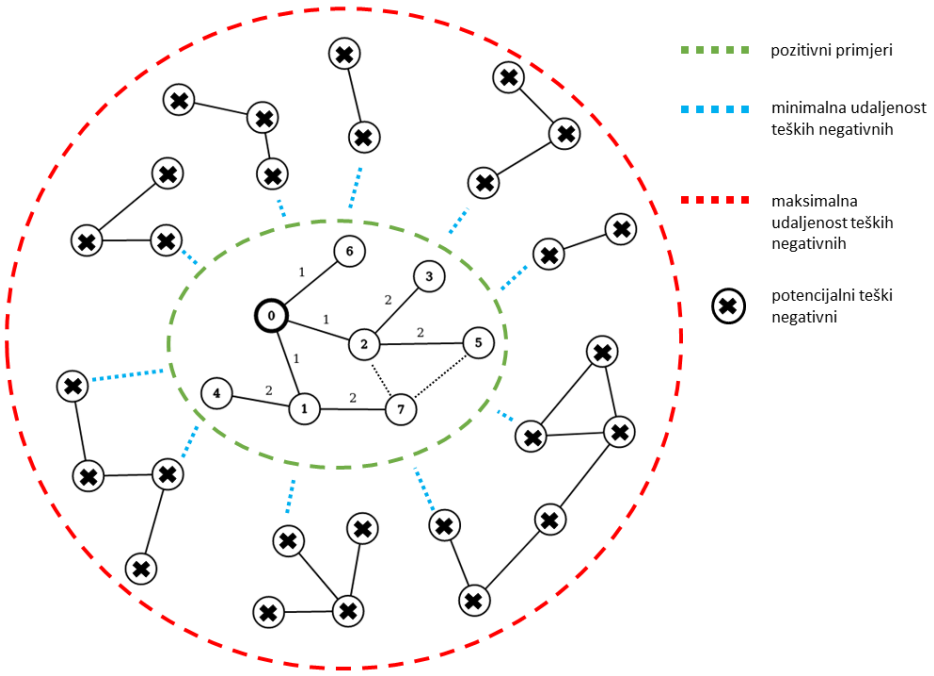
Naknadnim testiranjem utvrđeno je poboljšanje kvalitete generiranih podataka uklanjanjem vrhova određenog stupnja. Tako su dodatno uklonjeni vrhovi stupnja većeg od 100 kao i vrhovi koji predstavljaju pregledne radove s pretpostavkom da povezuju susjedstva niže relevantnosti. Pregledni članci su filtrirani putem naslova radova uz niz ključnih riječi koje upućuju na takav članak poput „review“, „survey“ ili „overview“.

Od nastalog grafa uzeti je maksimalni podgraf koji je činilo 87% sveukupnih vrhova i 99% bridova. Graf je tako činilo oko 3.2 milijuna vrhova i 20 milijuna bridova, pritom je prosječni stupanj vrha iznosio 10. Iako se u literaturi češće nalazi usmjereni graf citata od neusmjerenog kako bi se iskoristila dodatna informacija o vezi među radovima, odlučeno je graf ostaviti neusmjerenim. Razlog ovome je prosječni stupanj vrhova grafa koji je bio dovoljno mali kao i svrhe grupiranja tih vrhova, gdje usmjereni bridovi nisu od značaja.

3.3 Generiranje umjetnih korisnika

Generiranje trojke *(referentni, pozitivni primjerak, negativni primjerak)* za svrhe učenja reprezentacija mogli bi se iskoristiti algoritmi poput detekcije zajednica no oni se isplate provoditi nad znatno manjim grafovima od onog konstruiranog. Imajući na umu veličinu grafa pa time i vremensku i prostornu složenost, za svrhe efikasnog grupiranja članaka bilo je potrebno osmisliti efikasan algoritam.

Predloženi algoritam temelji se na modificiranoj slučajnoj šetnji. Iako je unutar poglavlja dan pseudokod algoritma, u nastavku će se ukratko osvrnuti na osnovni princip šetnje i ulogu hiperparametara šetnje.



Slika 3.1 Položaj teških negativnih naspram pozitivnih. Šetnja pozitivnih kreće u vrhu nasumičnom korijenskom vrhu 0, šetnjom pamtimo udaljenosti od tog vrha.

Prilikom svakog generiranja korisnika kreće se od vrha koji je nasumično izabran iz skupa svih vrhova grafa. Ovaj vrh možemo nazvati korijenskim. Od njega dalje započinje slučajna šetnja koja obilazi zadani broj vrhova dobiven slučajnim odabirom iz intervala *min članaka* i *max članaka*. Interval označava koliko će se pozitivnih primjera prikupiti šetnjom.
Radi same izvedbe šetnje, povećanjem broja članaka koji predstavljaju pozitivne primjere ponekad produljimo šetnje u preveliku dubinu čime stupanj relevantnosti pada. Generiranjem većeg broja korisnika putem kraćih šetnji pokriva se dovoljno velika površina grafa ali zbog manjeg broja grupiranih pozitivnih primjeraka, šetnje grupiraju više međusobno relevantne znanstvene radove. Također, mora biti dovoljno pozitivnih članaka kako bi se dio izdvojio za referentne primjere. Ti referentni primjeri označavaju preferencije korisnika. U sklopu kontrastnog učenja to će biti oni primjeri kojima želimo približiti pozitivne primjere u vektorskom prostoru.
Nakon što se jednom odabere korijenski vrh, svi njegovi susjedni vrhovi spremaju se u set iz kojega će se u narednim koracima nasumično birati kao i dodavati novi vrhovi.

Pozitivni primjeri se biraju tako da se slučajnim odabirom izabere čvor iz prethodno spomenutog seta vrhova. Za potencijalni pozitivni primjerak iteracije razmatra se susjedstvo prvoga reda izabranog vrha. Odabir pozitivnog primjerka djelomično ovisi o težinama bridova koje izabrani vrh dijeli sa njima. Što je veća težina brida, to su veće šanse da će dotični vrh biti izabran. Kao primjer možemo uzeti nasumično izabran vrh A i njegove susjede B i C. Neka je težina između A i B je 99, a između A i C je 1. Vjerojatnost da idući čvor obiđen u slučajnoj šetnji bude B je 99/100, a vjerojatnost da bude C je 1/100. Inspiracija za algoritam odabira sljedećeg čvora je inspirirana algoritmom kotača ruleta (engl. *roulette wheel algorithm*) koji se u velikoj mjeri koristi u genetskim algoritmima.
Nakon što se izračuna doprinos svakog brida sa susjednim vrhom, skalira se na raspon od 0 do 1. Dobiveni raspon se dalje dijeli na onoliko segmenata koliko ima susjeda. Podjelom svaki segment zauzima razmjerno onoliki raspon kolika mu je vjerojatnost. Krajnji odabir vrha predstavlja vrtnju ruleta u kojemu se nasumično bira decimalni broj od 0 do 1. Ovisno o iznosu dobivenog segmenta, biramo susjedni vrh kojemu taj segment odgovara. Izabrani vrh se dodaje u set vrhova koji će se izvlačiti u narednim iteracijama.
Za svaki dodani vrh u setu pamti se i udaljenost od korijenskog čvora od kojega je šetnja krenula. Ove vrijednosti su aproksimativne i uvedene radi manje vremenske složenosti.

Nakon generiranja pozitivnih primjera, određuju se teški negativni primjeri. Za teške negativne mora se odrediti iz koje okolice smiju biti birani. Proces je sličan generiranju pozitivnih uz nekoliko izmjena. U svakoj iteraciji prate se posjećeni vrhovi i „fronta“ u kojoj se nalaze vrhovi koji još nisu posjećeni. U set vrhova fronte dodaju se svi vrhovi susjedstva

izabranog vrha iteracije. Ovo se na prvu ne čini kao najefikasnije rješenje, no ono je zadovoljavajuće imajući na umu da se radi o relativno kratkim šetnjama, nastojanju što manjeg dodatnog uskraćivanja mogućih puteva kao i prosječni stupanj vrha grafa. Iako se u set fronte pohranjuju svi susjedi, samo jedan od susjeda ulazi u set teških negativnih. Ukoliko susjedstvo ima aproksimativnu udaljenost od korijenskog vrha unutar dopuštenog intervala, izbor ulazi li vrh u negativne ili ne bira se već spomenutim algoritmom selekcije uz razliku da veće šanse ima onaj susjedni vrh koji dijeli manju težinu brida sa trenutnim. Ukoliko udaljenost nije unutar intervala, iteracija samo pohranjuje susjede. Dopušteni interval udaljenosti je od maksimalne udaljenosti unutar seta vrhova pozitivnih primjera *tm* uvećana za minimalnu (dodatnu) udaljenost do *tm* vrijednosti uvećane za maksimalnu (dodatnu) udaljenost teških negativnih. Ovo je jedna od mjera predostrožnosti s kojim je cilj odrediti dovoljno veliku udaljenost kako bi se izbjegli mogući konflikti (preslični primjeri označeni kao negativni) koji bi utjecali na stabilnost učenja neuronske mreže.

Biranje lakih negativnih primjera se svodi na trivijalni nasumičan odabir vrhova iz seta vrhova cijeloga grafa.

Hiperparametri koje je bilo nužno utvrditi su minimalni i maksimalni broj pozitivnih članaka, broj teških i lakih negativnih, dodana minimalna i maksimalna udaljenost teških negativnih od korijenskog vrha i broj generiranih korisnika.

Algoritam 1: Generiranje trojke (referentni, pozitivni primjerak, negativni primjerak)

Ulaz: *broj korisnika koje je potrebno generirati //korisnik reprezentiran setom članaka*

Izlaz: *lista setova članaka*

// A := set svih setova članaka korisnika
// C := set članaka (vrhova grafa)
// VH := rječnik udaljenosti vrhova od početnog vrha
// G := graf citata
// HN := set teških negativnih primjera
// EN := set laganih negativnih primjera

```
A ← ∅
dok |A| < broj korisnika radi
    C ← ∅
    VH ← ∅
    v ← nasumičan vrh grafa G                    // korijenski vrh
    VS ← susjedstvo vrha v
    C ← v
    VH[v] ← 0
    np ← random(min članaka, max članaka)  // broj pozitivnih članaka iz zadanog intervala
    C, VH ← generirajPozitivne(C, np, VH )  // algoritam 2
    HN ← generirajTeškeNegativne(C, VH)   // algoritam 3
    EN ← random(svi vrhovi G, broj lakih negativnih)
    A.append( (C, HN, EN) )
    ako |A| ≥ broj korisnika
        vrati A
```

Algoritam 2: Generiranje pozitivnih primjera ponderiranom slučajnom šetnjom

```
    Ulaz: set inicijalnih članaka, // C iz algoritma 1
          broj članaka koje je potrebno izgenerirati,
          rječnik udaljenosti vrhova od početnog vrha      // VH iz algoritma 1
    Izlaz: set pozitivnih članaka(vrhova),
          rječnik udaljenosti vrhova od početnog vrha

    // lh := udaljenost trenutnog vrha od početnog (root)
    // T := lista težina bridova
1   dok | C | < broj članaka koje je potrebno izgenerirati
2       v ← nasumičan vrh grafa G
3       lh ← VH[v] + 1
4       VS ← susjedstvo vrha v
5       NS ← VS \ C          // dohvati one koje nismo posjetili
6       ako | NS | == 0
7           nastavi          // izaberi drugi nasumičan vrh
8       T ← ∅
9       za svaki s u NS radi
10          eid ← G.dohvati_id_brida(v, s)
11          et ← G.dohvati_težinu_brida(eid)
12          T.append(et)
13      maxval ← sum(T)
14      za svaki i u range(0, | T |)
15          T[i] ← T[i] / maxval
16          ako i > 0
17              T[i] ← T[i] + T[i - 1]
18      k ← random.uniform(0, 1)
19      ind ← searchSorted(T, k)   // dohvati prvi indeks veći ili jednak broju k
20      nv ← NS[ind]               // izabrani vrh iteracije
21      ako nv nije unutar VH
22          VH[nv] ← lh
23      C.add(nv)
24  vrati C, VH
```

Algoritam 3: Generiranje teških negativnih primjera ponderiranom slučajnom šetnjom

Ulaz: *P := set generiranih pozitivnih primjera iz algoritma 1,*

Izlaz: *set teških negativnih primjera*

*// **HNmax :=** zadani broj teških negativnih za izgenerirati (hiperparametar)*

```
tm ← max(VH.values()) // maksimalna udaljenost šetnje od inicijalnog vrha
minh ← tm + MinHD
maxh ← tm + MaxHD
N ← Ø          // set teških negativnih za vratiti
F ← P
V ← P
dok je | HN | < HNmax
    v ← nasumičan vrh grafa G
    V.add(v)
    lh ← VH[v] + 1
    VS ← susjedstvo vrha v
    NS ← VS \ C          // dohvati one koje nismo posjetili
    F.remove()
    ako | NS | == 0
        nastavi
    T ← Ø
    Za svaki s u NS radi
        ako s nije unutar NH
            NH[s]=lh
        eid ← G.dohvatiIdBrida(v, s)
        et ← G.dohvatiTežinuBrida(eid)
        T.append(et)
    maxfit ← max(T)
    ako lh < minh ili lh > maxh
        nastavi
    za svaki i u range(0, | T |)
        T[i] ← (maxfit – T[i] + eps)
    maxval ← sum(T)
    za svaki i u range(0, | T |)
        T[i] ← T[i] / maxval
        ako i > 0
            T[i] ← T[i] + T[i - 1]
    k ← random.uniform(0, 1)
```

```
34        ind ← searchSorted(T, k)    // dohvati prvi indeks veći ili jednak broju k
35        nv ← NS[ind]
36        HN.add(nv)
37        F.update(NS)      //dodaj nove
38    vrati HN
```

**** Algoritam se u svojoj paralelnoj izvedbi za generiranje 300 000 korisnika sa konfiguracijom šetnje jednakoj onoj konačnoj (vidjeti rezultate) izvodi ispod dvije minute uz procesor 11th Gen Intel® Core™ i7-11700K @ 3.60GHz × 16.

3.4 Priprema dobivenog skupa za treniranje

Postoje razni pristupi smještaja teksta u visokodimenzionalni prostor (engl. *embedding*) u kojemu je cilj smjestiti tekst tako da se poveća sličnost između sličnih riječi ili teksta. U okviru ovoga rada, razmatrati će se dva osnovna tipa već naučenih modela koji obavljaju taj zadatak, statične i dinamičke. Razlika je što statični, za razliku od dinamičnih, ne razmatraju okolni kontekst riječi, pa tako ista riječ u dva različita konteksta ima istu reprezentaciju (isti vektor).

Statički modeli poput Fasttext modela [7]–[11] trenirani su da raspoznaju podriječi (engl. *n-grams*). Model specificira funkciju koja mapira n-torku koju čini riječ v i njene podriječi $(c_1, \dots, c_t)$ u vektor $h \in R^d$ na način [12] :

$$f_{podrijec}: \left(v, (c_1, \dots, c_t)\right) \rightarrow h \qquad (3.1)$$

Dinamički modeli su s druge strane mnogo ekspresivniji, tokom treniranja uzimaju u obzir i okolne riječi tj. kontekst, tako funkcija modela mapira n-torku koju čine riječi teksta w_i u n-torku vektora smještaja tih riječi:

$$f_{kontekst}: (w_1, \dots, w_N) \rightarrow (h_1, \dots, h_N) \qquad (3.2)$$

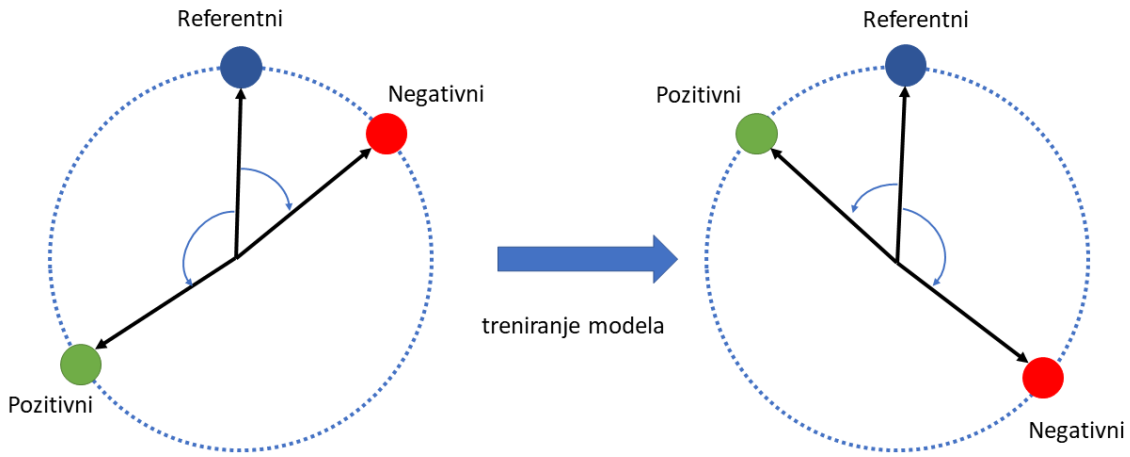
Smještaj teksta poznatim dinamičkim modelima poput ELMo [13], BERT [14] , ULMFiT [15], GPT-3 [16] daje znatno preciznije rezultate, no njihova složenija arhitektura rezultira sporijom brzinom smještaja što ih čini neskalabilnima pa time često nisu opcija u produkcijskim okruženjima.

Za svrhe ovog istraživanja radi usporedbe rezultata i doprinosa predložene metode, izabrana su dva predtrenirana modela, Fasttext i SentenceBERT [17]. Radi brzine ali i zadovoljavajuće preciznosti smještaja kao primaran model izabran je model Fasttext. Ideja je bila iskoristiti najbržu metodu smještaja teksta koja nema nužno najpreciznije rezultate i dalje je poboljšati uz dodatno treniranje WRS metodom razvijenom u sklopu ovog istraživanja. Fasttext je javno dostupna biblioteka koja se koristi za svrhe klasifikacije teksta i učenja reprezentacije teksta. Biblioteka nudi različite modele trenirane nad podriječima, razlikuju se ovisno o broju riječi kao i o izvoru teksta nad kojima su trenirani. Izabrani model treniran je nad korpusom teksta koji je sveukupno sadržavao 600 milijardi riječi. Izlazna dimenzija vektora je 300 no postoji mogućnost smanjivanja.
SentenceBERT (SBERT) je dodatno istrenirani BERT duboki model koji je jednake preciznosti kao BERT ali namijenjen za bržu usporedbu teksta. Smatra se jednim od najsuvremenijim modela za smještaj teksta. SBERT je među ostalim treniran i nad znanstvenom literaturom. SBERT je prvenstveno izabran kako bi se usporedio sa Fasttext modelom nadograđenog WRS metodom. Fasttext sam za sebe daje smještaj riječi manje kvalitete ali je zato znatno brži od dinamičkih modela poput SBERTa.

Prije nego što se tekst mogao smjestiti uz odabrane modele bilo je potrebno svaki članak prvo pretprocesirati sa bibliotekom Spacy [18]. Spacy je popularna biblioteka namijenjena za svrhe obrade prirodnog jezika u pythonu. Proces pretprocesiranja je se sveo na konkatenaciju sažetka članaka, zajedno sa naslovom i kategorijom te uz to svođenje teksta na mala slova, tokenizaciju riječi, uklanjanje specijalnih znakova i uklanjanje zaustavnih riječi.
Nakon faze predprocesiranja , tekst je smješten u vektorski prostor uz poziv Fasttext metode koja prima riječ i vraća vektor smještaja. Sumirane vektore riječi teksta se naknadno podijelilo sa brojem riječi. Slično je napravljeno i za smještaj uz SBERT koji radi sa odlomcima teksta ne dugima od 256 riječi, što odgovara prosječnoj duljini sažetaka *dblpv13* baze članaka na raspolaganju. Izlaz SBERT modela su pritom vektori fiksne dimenzije 384.

3.5 Arhitektura duboke neuronske mreže

Imajući na umu cilj samonadziranog kontrastnog učenja, kao prirodno rješenje postavila se sijamska arhitektura koja se može naći [19] i [17] kao i u brojnim drugim radovima koji se bave sustavima preporuke [20]–[22]. Sijamsko učenje koristi neku od varijanti kontrastnog gubitka koji ovisi o tome jesmo li na ulaze sijamskih modela doveli primjerke iste klase. Cilj učenja uz sijamskog modela je učenje smještaja ulaznih podataka u prostor gdje zadana metrika odražava sličnost tj. smještaj podataka. Parovi podataka prolaze kroz dva primjerka modela. Cilj je približiti pozitivne primjere i udaljiti negativne bilo to gledajući euklidsku distancu, kosinusnu sličnost ili neku drugu metriku udaljenosti. Izabrana metrika udaljenosti je kosinusna sličnost () .



Slika 3.2 Cilj treniranja neuronske mreže. Povećavanje kosinusne sličnosti između pozitivnih i referentnih primjera i udaljavanje referentnih i negativnih.

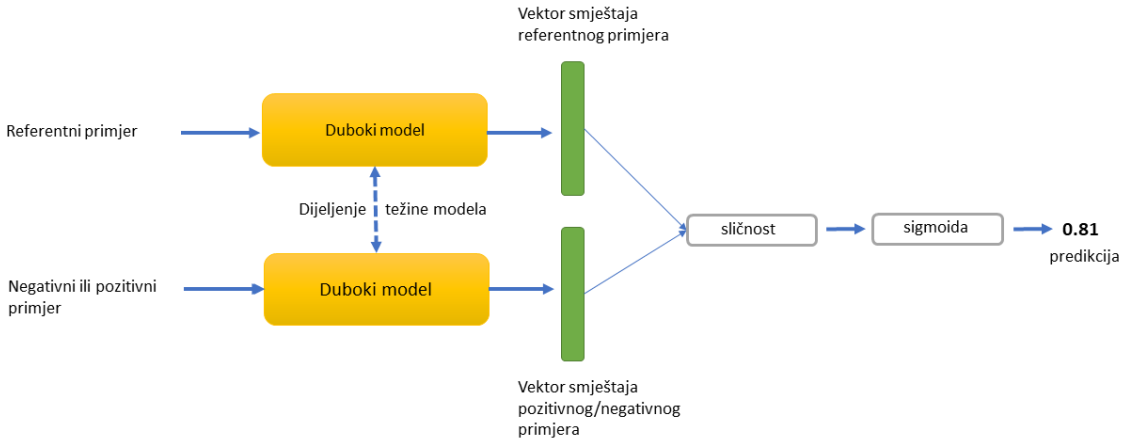
Arhitektura neuronske mreže je izgrađena uz pomoć biblioteke Keras [23] i Tensorflow [24]. Sav kôd, uključujući onaj prethodnih kao i narednih koraka može se pronaći na githubu [25].

U implementiranoj izvedbi (Slika *3.3*) jedan primjerak modela primao je reprezentacije preferencija korisnika predstavljenih referentnim primjerkom a drugi reprezentacije naizmjenice pozitivnih i (udruženih) negativnih primjera. Ulazni podatak tako je činio par (referentni, pozitivni) ili par (referentni, negativni). Oznake pozitivnih su pritom jedinice a negativnih nule. Navedena dva primjerka modela pritom dijele parametre težina, a gradijenti odgovarajućih parametara dviju grana se akumuliraju. Ulazni podatci se prenose kroz slojevitu

mrežu, u dodatnom sloju izlazi dva skalarno množe te time dobiva predikcija, koja se uz sigmoidu postavlja na vrijednosti između 0 i 1 koje označavaju pripadnost klasi. Od funkcije gubitaka valja spomenuti gubitak kosinusnog smještaja (engl. *cosine embedding loss)* (3 .1) koji je od testiranih davao najbolje rezultate.

$$L(x,y) = \begin{cases} 1 - cos(x_1, x_2), akoy = 1 \\ max(0, cos(x_1, x_2) - \alpha), akoy = 0 \end{cases} \quad (3.1)$$

Navedena sijamska arhitektura postizala je točnost do 95% nad testnim skupom podataka izdvojenim iz generiranog skupa podataka nad grafom. Gdje je predikcija bila točna ukoliko je izlaz bio veći od 0.5 ukoliko se radilo o pozitivnom primjeru, ili u slučaju negativnog manji od 0.5. Iako je model naizgled bio dobar nad testnim skupom, nad definiranom twitter metrikom (definirana u **3.6**) davao je lošije rezultate od čistog Fasttext i SBERT modela. Iz tog razloga zamijenjena je modelom trojnog učenja koji se naposlijetku pokazao boljim rješenjem.



Slika 3.3 Sijamska arhitektura kao klasifikacijski model

Model trojnog učenja (Slika *3.4*) je konačni model uz pomoć kojega su dobivena uočljiva poboljšanja u kvaliteti preporuka generiranim predloženom metodom.

U sijamskom učenju jedan te isti podatak uspoređujemo i s negativnim i s pozitivnim primjerima. Pri tome ugrađivanje promatranog primjera trebamo izračunati dva puta.

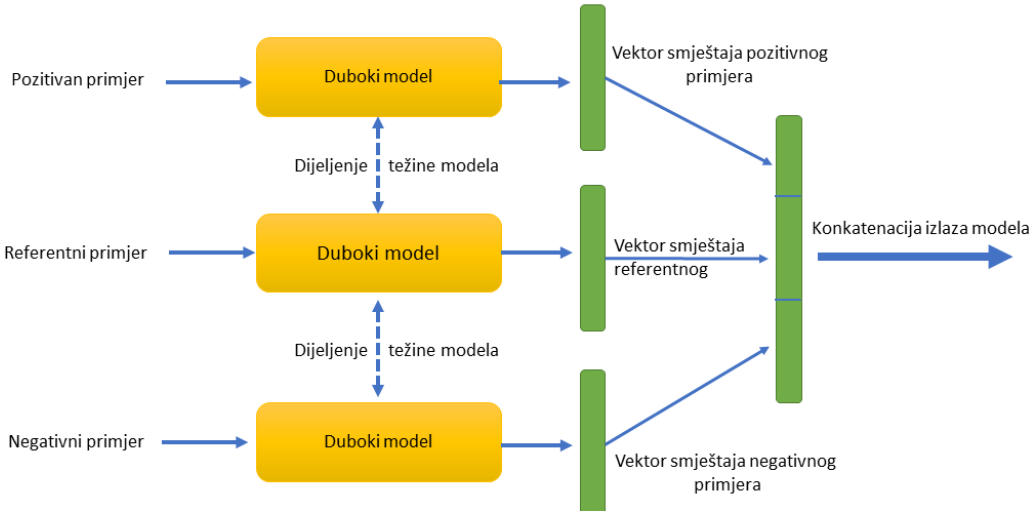
Taj problem adresira trojno učenje u kojem referentno ugrađivanje uspoređujemo s pozitivnim i negativnim primjerom istovremeno. Kao što je to slučaj u sijamskoj, tri primjerka modela i u ovoj arhitekturi dijele parametre i ravnopravno doprinose gradijentima gubitka. Trojno učenje

se tipično kombinira uz trojni gubitak [26]. Trojni gubitak spaja obje komponente kontrastnog gubitka u jedan izraz:

$$L = max(0, D(a,p) - D(a,n) + \alpha), \qquad (3.2)$$

gdje je D udaljenost između referentnog primjera a i pozitivnog p ili negativnog n primjera. α je važan hiperparametar koji označava marginu. Gradijenti potiču približavanje pozitiva i udaljavanje negativa samo ako je negativ bliže od udaljenosti pozitiva uvećane za marginu α. Malim izmjenama ovog hiperparametra dobivaju se drastično drugačiji rezultati. α je postavljen fiksno na vrijednost 0.1 koja se pokazala najboljom.

Iako je poznata pretpostavka kako više slojeva čini bolje rezultate [27], zbog ograničenja koje predstavlja dimenzija ulaza odlučeno je da će svaki toranj mreže sastojati 3 skrivena slojeva, uz toliko slojeva mreža raspolaže sa dovoljno slojeva da nauči dovoljno dobro generalizirati nad podatcima visoke apstrakcije.
Svaki primjerak modela čini: ulazni sloj dimenzije 300 ili 384 ovisno da li je za smještaj korišten Fasttext ili SBERT, zatim skriveni sloj dimenzije 250-200-150 te izlaz. Izlazi se spajaju u jedan vektor .



Slika 3.4 Trojna arhitektura. Za razliku od prikazane sijamske (Slika 3.3), svaki element trojke ima svoj pripadajući primjerak modela.

Zanimljivo je razmatrati moguće razloge zašto se trojno učenje pokazalo bolje u ovoj primjeni od sijamskog. Tako se može uočiti kako trojni gubitak (3 .2) kao posljedicu nema

smještaj pozitivnog primjera i referentnog u istu točku u prostoru kao što je to kod kontrastnog gubitka korištenog u sijamskoj arhitekturi. Ovo svojstvo trojnog gubitka omogućava protezanje pozitivnih primjera u prostoru, ali i dalje osiguravajući zadanu marginu prema negativnim primjerima. Nadalje, trojni gubitak je manje pohlepan jer primarno osigurava poštivanje margine i ne uvodi promjene u smještaju ukoliko je ta margina osigurana. Kontrastni gubitak s druge strane uzima u obzir marginu samo kada uspoređuje negativne primjere i ne gleda gdje su u tom trenutku pozitivni primjeri. Ovo bi moglo značiti da kontrastni gubitak prije nalazi lokalni minimum dok trojni gubitak potiče daljnje uređivanje smještaja primjera u stabilnije stanje.

3.6 Twitter skup podataka

Za svrhe testiranja predložene WRS metode, uzimanje gotovog skupa podataka o korisniku i njegovim preferencijama za znanstvene članke nije bila opcija pošto takav skup ne postoji. Kako bi se testirala uspješnost metode, podaci za testiranje su formirani uz pomoć Twittera. Twitter kao najpopularnija društvena mreža nudi API za dohvat tvitova koji se vrlo često koristi za svrhe istraživanja.

Možemo pretpostaviti da korisnici tvitaju sadržaj na teme koje su im od interesa. Dalje možemo deducirati da korisnici tvitaju znanstvene članke koji su im od interesa. Uz ovu pretpostavku odlučeno je prikupiti tvitove znanstvenih članaka i povezati ih sa korisnikom koji ih je objavio u svrhu dobivanja skupa podataka koji reflektira prave korisničke preferencije za pojedinim znanstvenim člankom.

Tvitovi su filtrirani na način da je svaki tvit morao sadržavati link na neki članak od onih izdavačkih kuća koji i sami imaju API za dohvat sažetaka, naslova i kategorija članaka. Izdavačke kuče koje nude ove APIje su IEEE, Elsavier i arXiv.
Osim što je svaki tvit morao imati link na članak od navedenih izdavača, uključeni su dodatni filtri koji: 1) izbacuju iz rezultata teme vezane na covid zbog njihove prevelike zastupljenosti, 2) isključuju tvitove koje sadrže ključne riječi poput „our“ ili “ congratulations“ koje bi mogle indicirati na eventualnu promociju vlastitih ili suradničkih radova. Iako takvi tvitovi nisu direktno u konfliktu sa ciljem, izbačeni su radi dodatne objektivnosti.

Iz razloga što Twitter ima ograničenje na vremenski period tvitova koji se mogu dohvatit, prikupljeni podaci se odnose na tvitove stare do 2 tjedna.

Jednom dobivena lista tvitova sa linkovima morala je biti dodatno procesirana. Iz razloga što pojedini korisnici koriste Twitter kao platformu za promoviranje vlastitih

znanstvenih radova ali i svojih suradnika i prijatelja, osim inicijalne filtracije obratila se pozornost i na ponavljanje identifikacijske oznake korisnika i znanstvenog članka gledajući se pritom DOI ili u slučaju IEEE broj znanstvenog rada. Za neke korisnike je utvrđeno da koriste Twitter za spremanje njima korisnih članaka te samim time imaju veći broj tjedno tvitanih linkova. Imajući na umu takve korisnike ali također i eventualne botove, manji dio korisnika koji su imali više od 100 tvitanih članaka su odbačeni.

Nakon što se osigurala nepristranost podataka, od korisničkih članaka bilo je potrebno izdvojiti dio članaka za referentni primjer i dio članaka za pozitivne primjere. Referentni primjer twitter korisnika čini konkatenacija dva do tri nasumična članaka koje je tvitao. Kao što je već navedeno, referentni primjeri predstavljaju korisničke preferencije. Osigurano je da svaki korisnik ima barem 5 članaka sveukupno.

Nakon svih postavljenih uvjeta za nepristranost podataka kao i ograničenja Twitterovog APIja dobiveno je sveukupno 413 korisnika i njihovih pripadnih 4702 tvitanih članaka.

Za negativne primjere, za svakog korisnika izabrano je nasumično 1000 članaka iz dblpv13 baze članaka, dakle one s kojom je izgrađen graf citata. Dblpv13 skup sadrži 3.2 milijuna članaka što je dovoljno da nasumično izabrani skup sadrži većinski korisniku irelevantan sadržaj. Broj 1000 je proizvoljno izabrani broj koji je s jedne strane dovoljno veliki da uvede šum među pozitivne primjere i s druge strane dovoljno mali da je vjerojatnost da su nasumičnim odabirom izabrani korisniku većinski relevantni članci mala.

3.7 Metrike uspješnosti preporuka

Najpopularnije metrike za mjerenje kvalitete sustava preporuke su nDCG@k i P@k. P@k (engl. *Precision at k*) računa koliko je relevantnih, u ovome slučaju pozitivnih članaka spalo unutar top *k* mjesta.

$$P@k = \frac{brojpozitivnihčlanakaunutark}{min\ (brojpozitivnihčlanakakorisnika, k)} \qquad (3.3)$$

nDCG se može predočiti kao ponderirani P@k, gdje se pozitivni primjeri ranga niža od prvog kažnjavaju. Formula glasi:

$$nDCG = \frac{DCG}{IDCG} \qquad (3.4)$$

Idealan DCG (IDCG) iznosi:

$$\sum_{i=1}^{P} \frac{rel_i}{log_2(i+1)} = \sum_{i=1}^{P} \frac{1}{log_2(i+1)}, \qquad (3.5)$$

gdje je *P* broj pozitivnih primjera twitter korisnika. Ocjena relevantnosti *rel* iznosi 1 ukoliko je pozitivan primjer i 0 ukoliko je negativan, čime se efektivno isključuje doprinos negativnih. Prethodno se jednostavnije može izjasniti kao „*P* pozitivnih članaka na prvih *P* mjesta“.
DCG (engl. *Discontinued cumulative gain*) razmatra rang r_i pojedinog pozitivnog članka *i* u skupu svih članaka, pozitivnih i negativnih.

$$DCG = \sum_{r=r_1}^{r_P} \frac{1}{log_2(r+1)} \qquad (3.6)$$

nDCG@k tako računa koliko je pozitivnih članaka rangirano unutar top *k,* gdje je *P* u (3 .5) jednak *min(broj pozitivnih članaka korisnika, k)* .

Kao što je već spomenuto, za smještaj korisnika i članaka koristi se Fasttext i SBERT. Gleda se koliko je poboljšanje preporuka uz dodanu WRS metodu. Prvo se korisnike i njihove pozitivne i negativne znanstvene radove smjesti u vektorski prostor uz Fasttext, time se dobiva skup koji sadrži vektore twitter korisnika i skup koji sadrži vektore njihovih pozitivnih i negativnih primjeraka radova. Dalje se za svakog korisnika gleda kolika je sličnost između njega kao referentne točke i njegovih pozitivnih i negativnih primjera. Idealna situacija je ona u kojoj su pozitivni što sličniji korisniku (kosinusna sličnost) pa time i bliže u prostoru, a negativni što manje slični i što dalje u prostoru (Pogledati sliku *3.2*). Sve članke je dalje

potrebno sortirati po sličnosti s korisnikom i zabilježiti pozicije pozitivnih. Što je više pozitivnih primjera na višim pozicijama, to je model kvalitetniji.
Isti koraci napravljeni su za SBERT.

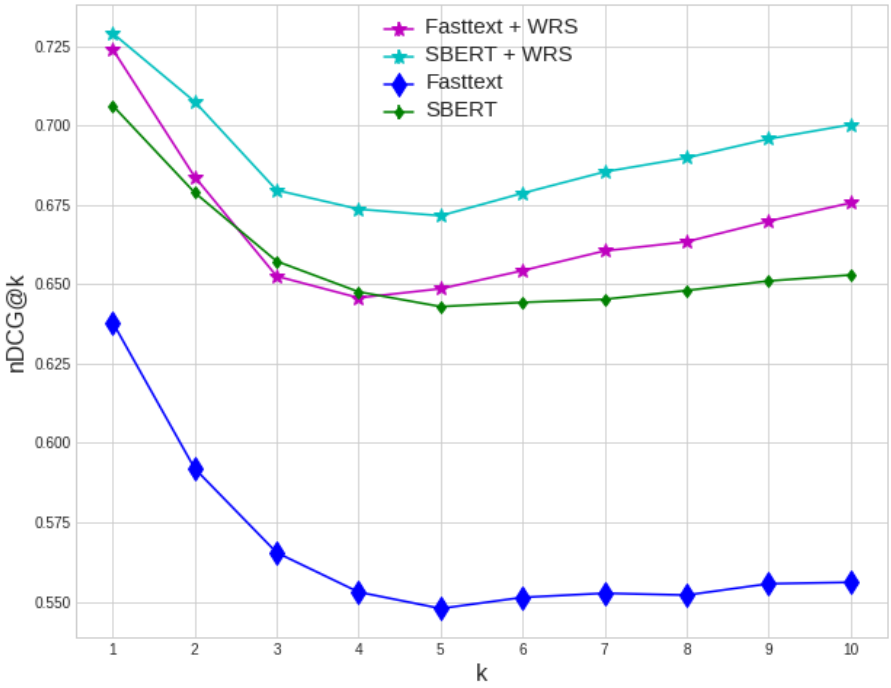
Za WRS metodu postoje modeli trenirani nad podacima umjetnih korisnika smještenih uz Fasttext i SBERT. Modeli su istih konfiguracija, izuzev ulaznog sloja modela zbog različitih dimenzija smještaja. Twitter korisnike smještamo koristeći istrenirane modele i na isti način kao što je već navedeno rangiramo po sličnosti.
Na opisani način je moguće točno vidjeti koji modeli su uspješnije rangirali pozitivne članke twitter korisnika.

4 Rezultati

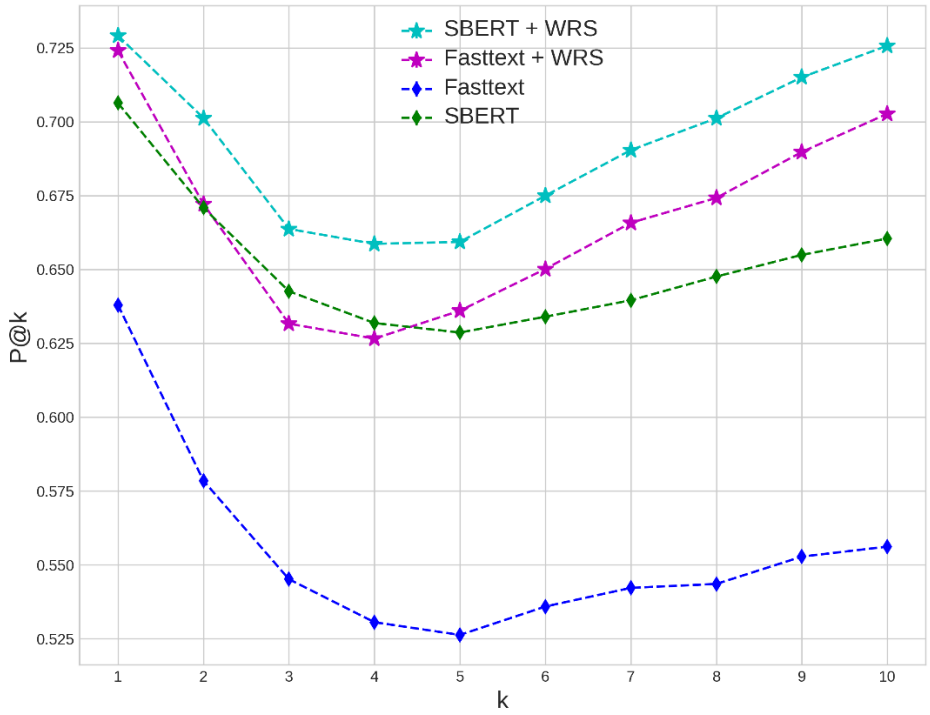
Konačna konfiguracija šetnji koja se pokazala najboljom generirala je 200000 umjetnih korisnika gdje je svakom korisniku pripalo od 6 do 10 članaka, od kojih su dva do tri spojena pod referentni primjer a ostalo je sačuvano za pozitivne primjere.
Svakom korisniku je pripalo tri teška i tri laka negativna članka, gdje minimalna dodatna udaljenost teških negativnih od pozitivnih iznosi dva koraka, a maksimalna 6.
Povećanje generiranih korisnika za dodatnih 100000 korisnika nije imalo preveliki utjecaj na rezultat no dodatno je produljio treniranje modela, stoga je zaključno ostalo na 200000.
Generirani korisnici su šetnjama pokrili oko 56% ukupnih, dostupnih članaka u grafu.

Treniranje trojnog modela završeno je sa validacijskim gubitkom 0.0312 za model sa Fasttext bazom i 0.0284 za model sa SBERT bazom. Klasične metrike poput točnosti ovdje nemaju značaj iz razloga što ne postoji točno definirani klasifikacijski zadatak sa znanim oznakama. Zadatak modela je postavljen na maksimiziranje udaljenosti negativnih primjeraka te ga ima smisla testirati nad ciljanim zadatkom, a to je preporuka.

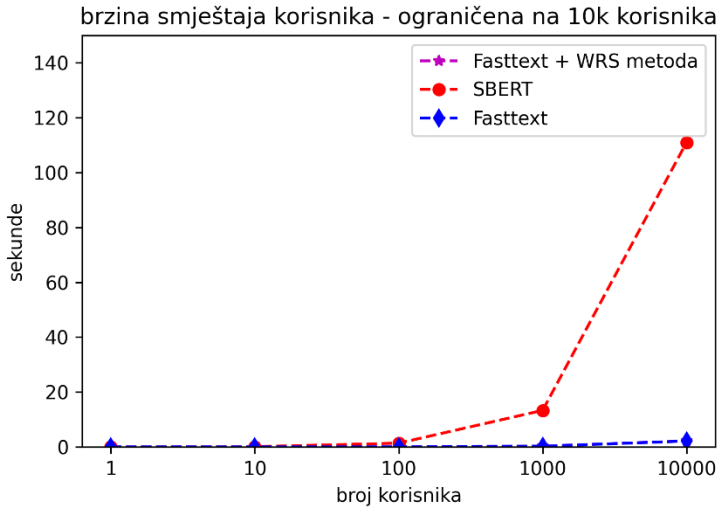
U nastavku slijede rezultati preporuka znanstvenih radova twitter korisnicima mjereni metrikom nDCG@k i P@k.



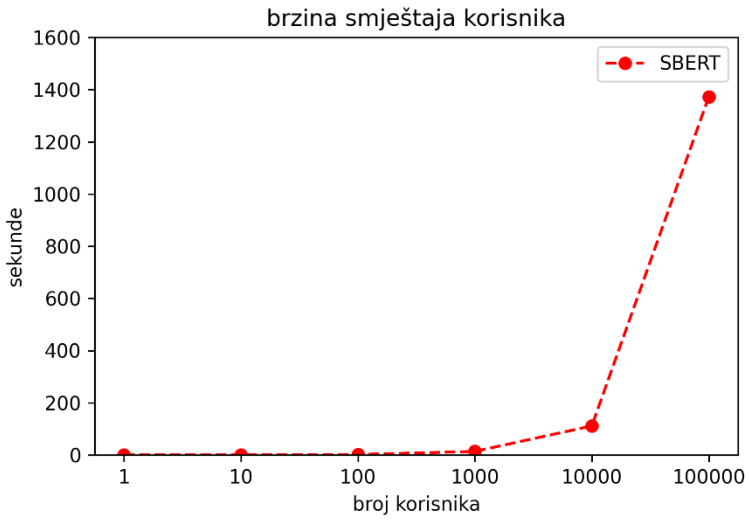
Slika 4.5 nDCG@k metrika nas svim verzijama modela



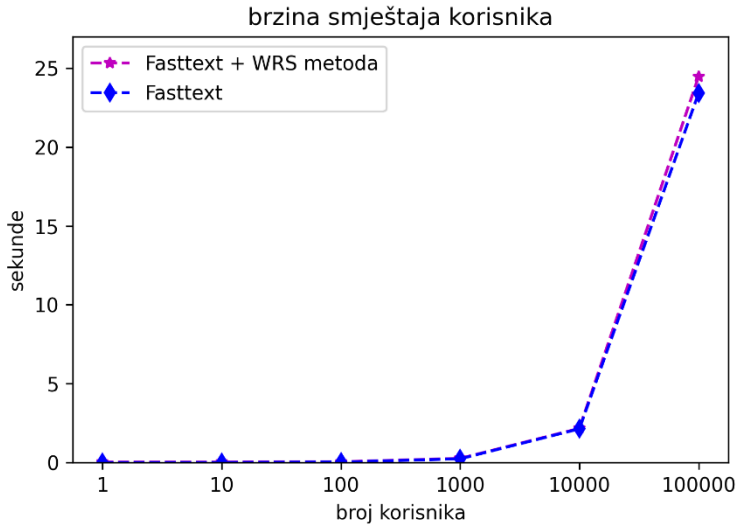
Slika 4.6 P@k nad svim verzijama modela



Slika 4.7 Brzina smještaja ovisno o broju korisnika koje je potrebno smjestiti u vektorski prostor. Ograničeno na 10k korisnika radi prevelikog odstupanja SBERT modela



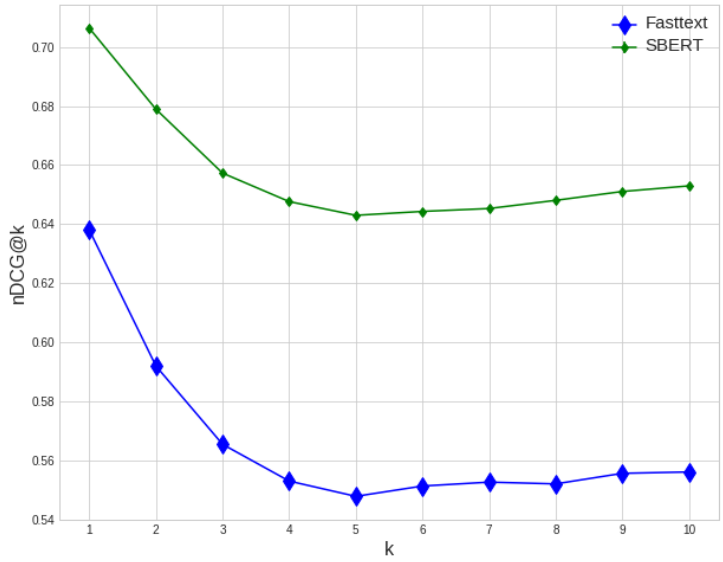
Slika 4.8 Brzina smještaja korisnika uz SBERT model. Obratiti pozornost na y os



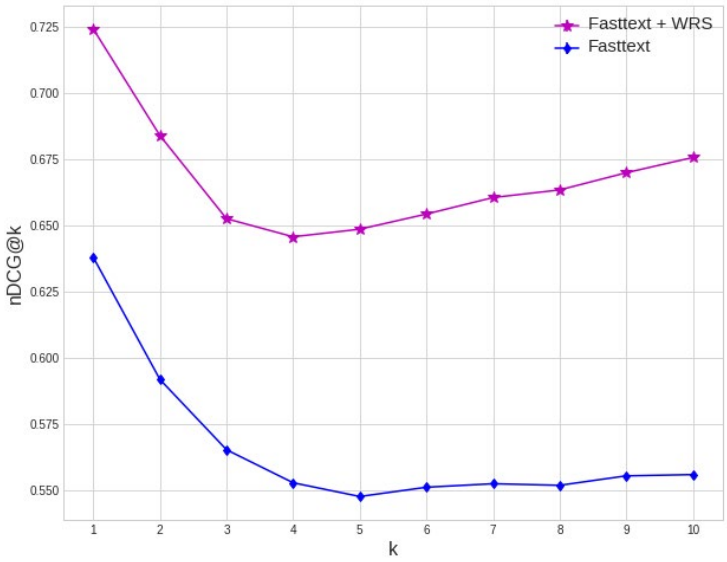
Slika 4.9 Brzina smještaja korisnika uz Fasttext bazni model i WRS model sa Fasttext bazom

Tablica 1: Brzina smještaja korisnika u sekundama

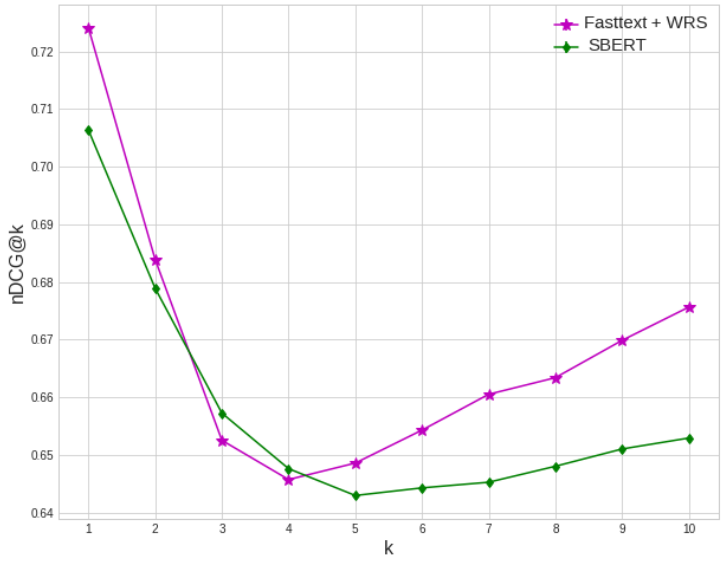
Broj korisnika	*SBERT*	*Fasttext*	*Fasttext + WRS*
1	0.025788	0.000434	0.020323
10	0.073306	0.001485	0.019057
100	1.360047	0.025728	0.044321
1000	13.30029	0.235429	0.258844
10000	110.9887	2.134885	2.161189
100000	1372.032	23.42448	24.4719



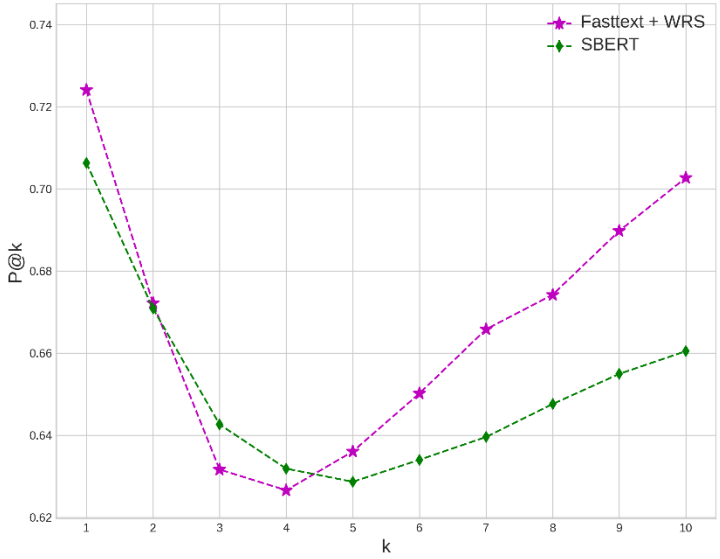
Slika 4.10 Zasebna usporedba Fasttext i SBERT modela nad nDCG@k metrikom



Slika 4.11 Zasebna usporedba metrike nDCG@k Fasttext modela sa WRS modelom sa Fasttext bazom



Slika 4.12 Zasebna usporedba metrike nDCG@k nad SBERT modelom i WRS modelom sa Fasttext bazom



Slika 4.13 Zasebna usporedba metrike P@k SBERT i WRS modela sa Fasttext bazom

Tablica 2: P@k modela i razlike (Δ) u vrijednostima

	P@1	P@2	P@3	P@4	P@5	P@6	P@7
FASTTEXT *	0.637975	0.578481	0.545148	0.530591	0.526287	0.535907	0.542218
FASTTEXT+WRS */***	0.724051	0.672152	0.631646	0.626582	0.636034	0.650169	0.665799
Δ *	**0.086076**	**0.093671**	**0.086498**	**0.095992**	**0.109747**	**0.114262**	**0.12358**
SBERT **/***	0.706329	0.670886	0.642616	0.631857	0.62865	0.634008	0.639566
SBERT + WRS **	0.729114	0.701266	0.663713	0.65865	0.659325	0.674979	0.69041
Δ **	**0.022785**	**0.03038**	**0.021097**	**0.026793**	**0.030675**	**0.04097**	**0.050844**
Δ ***	**0.017722**	**0.001266**	-0.01097	-0.00527	**0.007384**	**0.01616**	**0.026233**

Tablica 3: Nastavak Tablice 2

	P@8	P@9	P@10	50	100
FASTTEXT *	0.543541	0.552794	0.556176	0.686761	0.774583
FASTTEXT+WRS */***	0.67421	0.689761	0.702709	0.859541	0.919117
Δ *	**0.130669**	**0.136967**	**0.146533**	**0.17278**	**0.144533**
SBERT **/***	0.647592	0.654931	0.660471	0.706329	0.670886
SBERT + WRS **	0.701193	0.715065	0.725727	0.888212	0.934501
Δ **	**0.053602**	**0.060135**	**0.065256**	**0.181883**	**0.263615**
Δ ***	**0.026618**	**0.03483**	**0.042238**	**0.153212**	**0.248231**

Tablica 4: nDCG@k modela i razlike (Δ) u vrijednostima

	NDCG@**1**	NDCG@**2**	NDCG@**3**	NDCG@**4**	NDCG@**5**	NDCG@**6**	NDCG@**7**
FASTTEXT *	0.637975	0.591944	0.565456	0.553014	0.547842	0.551338	0.552654
FASTTEXT+WRS*/* **	0.724051	0.683896	0.65256	0.645755	0.648613	0.654329	0.660544
Δ*	**0.086076**	**0.091952**	**0.087105**	**0.092741**	**0.100771**	**0.102991**	**0.10789**
SBERT **/* **	0.706329	0.678907	0.657279	0.647641	0.642989	0.644312	0.64529
SBERT + WRS **	0.729114	0.707568	0.679674	0.673703	0.671628	0.678726	0.685457
Δ**	**0.022785**	**0.028661**	**0.022396**	**0.026062**	**0.028639**	**0.034413**	**0.040167**
Δ***	**0.017722**	**0.00499**	-0.00472	-0.00189	**0.005624**	**0.010016**	**0.015254**

Tablica 5: Nastavak Tablice 4

	NDCG@**8**	NDCG@**9**	NDCG@**10**	NDCG@**50**	NDCG@**100**
FASTTEXT *	0.552063	0.555634	0.556097	0.595585	0.623943
FASTTEXT + WRS */* **	0.663423	0.669896	0.675664	0.731128	0.751984
Δ *	**0.11136**	**0.114262**	**0.119567**	**0.135543**	**0.128041**
SBERT **/* **	0.64806	0.651058	0.652955	0.703304	0.721652
SBERT + WRS **	0.689865	0.695826	0.700328	0.75997	0.776774
Δ **	**0.041805**	**0.044768**	**0.047373**	**0.056666**	**0.055122**
Δ ***	**0.015363**	**0.018838**	**0.022709**	**0.027824**	**0.030332**

5 Rasprava

Iz rezultata možemo uočiti poboljšanje baznih modela uz WRS metodu. Na slici (*Slika 4.10*) možemo jasno vidjeti razliku u kvaliteti dinamičkog SBERT modela od statičkog Fasttext modela. Ovaj rezultat nije iznenađujući imajući na umu način na koji su modeli trenirani i njihovu kompleksnost. Fasttext sam za sebe nema kapacitet kao SBERT no uz WRS metodu Fasttext postaje vidljivo bolji. Na slikama (*Slika 4.12*) i (*Slika 4.13*) vidimo usporedbu WRS modela sa Fasttext baznim modelom i SBERT-a. Za P@1 i nDCG@1 može se vidjeti jasno poboljšanje, za P@2 i nDCG@2 manje, nakon čega vidimo kratko pogoršanje tokom naredna dva ranga. Razlog obliku nastalih krivulja može se naći u generiranim twitter podacima. Srednja vrijednost broja pozitivnih članaka po korisniku iznosila je 11, dok je medijan iznosio 5. Ovo dalje znači da korisnik koji ima 5 članaka, uz jasno raspoznavanje negativnih od pozitivnih tokom kvalitetnog rangiranja, u teoriji nakon pete pozicije svi njegovi članci mogu već biti pokriveni, kako rang raste tako raste i broj korisnika kojemu su svi članci već rangirani nad nižim pozicijama. Uz *P* iz (3.5) jednakim *min(broj pozitivnih članaka korisnika, k)* nDCG@k kao i P@k vrijednost će dalje rasti.

Nadalje, kod sustava preporuke je od velike važnosti brzina. Jedan od ciljeva ovoga rada bio je pokazati korist i snagu samonadziranog učenja. Uz razvijenu metodu cilj je bio poboljšati brzi i model cilj ovog istraživanja je nadogradnja brzog modela smještaja poput Fasttext-a, na način da mu se poveća preciznost preporuke bez prevelikog usporavanja smještaja. Na slikama (*Slika 4.7*), (*Slika 4.8*), (*Slika 4.9*) i tablici (*Tablica 1*) se može vidjeti da je Fasttext utreniran predloženom WRS metodom i dalje znatno brži od SBERT modela. Dok na slici (*Slika 4.9*) vidimo da je i do 100 000 korisnika brzina ostala skoro jednaka originalnoj.

Uz samu brzinu bitno je sagledati i utrošak memorije vektora reprezentacija korisnika i članaka. Najuspješniji modeli, poput BERT-a ili njegovih nadogradnji imaju smještaj vrlo visoke dimenzionalnosti, često do sedam ili osam puta veća od one korištene u WRS metodi. Ovo kao posljedicu ima veći kapacitet smještaja ali i veći utrošak memorije i manju brzinu tokom računanja. Inicijalna dimenzija 300 u slučaju Fasttexta i 384 u slučaju SBERT-a je uspješno snižena na dimenziju 100 i to uz dodatno poboljšanje smještaja.

6 Zaključak

Metoda razvijena unutar ovoga istraživačkog rada pokazuje mogućnosti samonadziranog učenja za svrhe rješavanja problema hladnog starta. Hladni start se javlja u produkcijskoj fazi kada u sustavu još nema dovoljno velik broj korisnika a time i zapisa o preferencijama korisnika.

Razvijena WRS (engl. *Weighted Random Surprise*) metoda pokazala je veliki potencijal za produkcijsku uporabu zbog skalabilnosti i modularnosti. WRS metoda sa bazom statičkog modela smještaja Fasttext pokazala je većinski bolje rezultate od znatno kompleksnijeg dinamičkog modela SBERT, dimenzionalnosti smještaja i više od trostruko veće nego WRS.

Unutar ovoga rada za bazu smještaja izabrani su Fasttext model i SentenceBERT (SBERT). Postoje dinamički modeli koji daju precizniji smještaj znanstvenog teksta kao SPECTER [28] ili SciBERT [29] s kojim bi bilo korisno usporediti rezultate. Radi se o kompleksnim modelima koji daju smještaj teksta dimenzionalnosti skoro dvostruko veće od SentenceBERT-a. Većinski se koriste kao baza za dodatnu specijalizaciju nad nizvodnim zadacima. Iako se ističu u smještaju znanstvenog teksta, imajući na umu broj članaka *dblpv13* baze kao i dimenzionalnost i sporiji smještaj navedenih modela, odlučeno je za referentni dinamički model uzeti SBERT.

Razvijena metoda idealna je za skupove podataka koje je moguće prikazati u obliku grafa na kojima je moguće izvršiti šetnje. Znanstveni radovi međusobno povezuju citati zbog kojih je bilo moguće kreirati graf, no u slučaju kada takve jasne veze ne postoje može se iskoristiti direktni smještaj sadržaja u vektorski prostor sa već istreniranim modelima i u sklopu samonadziranog učenja na isti način iskoristiti smještaj u tom prostoru za grupiranje uzoraka za učenje neuronske mreže [30].

Ukoliko bi se ovakav model preporuke znanstvenih članaka htio iskoristiti u produkcijskoj fazi tokom hladnog starta, korisnik bi mogao dati niz ključnih riječi, ili sažetak/sažetke radova od interesa, nakon čega bi mu bili preporučeni članci od interesa, bez da sustav ima ikakvu povijest interakcije tog korisnika.

Nakon što sustav preporuke zabilježi veći broj interakcije korisnika sa sadržajem, za kvalitetnije preporuke potrebno je ponovno utrenirati model sa novim podacima.

7 Zahvale

Veliko hvala mentoru doc. dr. sc. Mariju Brčiću koji mi je ovim istraživanjem uz povjerenje i svoje uloženo vrijeme omogućio produbljivanje znanja i stjecanje vještina potrebnih za daljnji znanstveni i profesionalni rad.

Također veliko hvala mojoj obitelji na podršci i razumijevanju, a posebno hvala bratu Luki na sponzoriranju istraživanja u obliku računalnih resursa bez kojega bi ovo istraživanje bilo gotovo nemoguće.

Za kraj hvala i kolegici Andrei Rakociji i kolegama Marinu Stojsavljeviću i Filipu Nekiću na iskazanoj podršci.

8 Popis literature

[1] M. Fire i C. Guestrin, „Over-optimization of academic publishing metrics: Observing Goodhart's Law in action", *GigaScience*, sv. 8, lip. 2019, doi: 10.1093/gigascience/giz053.

[2] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, i F. Makedon, „A Survey on Contrastive Self-Supervised Learning", *Technologies*, sv. 9, izd. 1, 2021, doi: 10.3390/technologies9010002.

[3] P. Khosla *i ostali*, „Supervised Contrastive Learning". arXiv, 2020. doi: 10.48550/ARXIV.2004.11362.

[4] X. Liu *i ostali*, „Self-supervised Learning: Generative or Contrastive", *IEEE Transactions on Knowledge and Data Engineering*, str. 1–1, 2021, doi: 10.1109/TKDE.2021.3090866.

[5] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, i Z. Su, „ArnetMiner: Extraction and Mining of Academic Social Networks", u *KDD'08*, 2008, str. 990–998.

[6] G. Csardi i T. Nepusz, „The igraph software package for complex network research", *InterJournal*, sv. Complex Systems, str. 1695, 2006.

[7] L. Mouselimis, *fastText: Efficient Learning of Word Representations and Sentence Classification using R*. 2022. [Na internetu]. Dostupno na: https://CRAN.R-project.org/package=fastText

[8] I. Facebook, *fastText: Library for fast text representation and classification*. 2016. [Na internetu]. Dostupno na: https://github.com/facebookresearch/fastText

[9] P. Bojanowski, E. Grave, A. Joulin, i T. Mikolov, „Enriching Word Vectors with Subword Information", *Transactions of the Association for Computational Linguistics*, sv. 5, str. 135–146, 2017, doi: 10.1162/tacl_a_00051.

[10] P. Bojanowski, E. Grave, A. Joulin, i T. Mikolov, „Bag of Tricks for Efficient Text Classification", u *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 2017, str. 427–431.

[11] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, M. Douze, i H. Jegou, „FastText.zip: Compressing text classification models", *arXiv preprint arXiv:1612.03651*, 2016.

[12] John Hewitt, *Finding Syntax with Structural Probes*. https://nlp.stanford.edu//~johnhew//structural-probe.html?utm_source=quora&utm_medium=referral#the-structural-probe

[13] M. E. Peters *i ostali*, *Deep contextualized word representations*. 2018.

[14] J. Devlin, M.-W. Chang, K. Lee, i K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019.

[15] J. Howard i S. Ruder, „Universal Language Model Fine-tuning for Text Classification". arXiv, 2018. doi: 10.48550/ARXIV.1801.06146.

[16] T. B. Brown *i ostali*, „Language Models are Few-Shot Learners“. arXiv, 2020. doi: 10.48550/ARXIV.2005.14165.

[17] N. Reimers i I. Gurevych, „Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks“. arXiv, 2019. doi: 10.48550/ARXIV.1908.10084.

[18] M. Honnibal i I. Montani, „spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing“, 2017.

[19] P. Covington, J. Adams, i E. Sargin, „Deep Neural Networks for YouTube Recommendations“, u *Proceedings of the 10th ACM Conference on Recommender Systems*, New York, NY, USA, 2016, str. 191–198. doi: 10.1145/2959100.2959190.

[20] D. Park, K. Kim, Y. Park, J. Shin, i J. Kang, „KitcheNette: Predicting and Ranking Food Ingredient Pairings using Siamese Neural Network“, kol. 2019. doi: 10.24963/ijcai.2019/822.

[21] M. Pulis i J. Bajada, „Siamese Neural Networks for Content-Based Cold-Start Music Recommendation.“, u *Fifteenth ACM Conference on Recommender Systems*, New York, NY, USA: Association for Computing Machinery, 2021, str. 719–723. [Na internetu]. Dostupno na: https://doi.org/10.1145/3460231.3478847

[22] H. Yuan, G. Liu, H. Li, i L. Wang, „Matching Recommendations Based on Siamese Network and Metric Learning“, u *2018 15th International Conference on Service Systems and Service Management (ICSSSM)*, 2018, str. 1–6. doi: 10.1109/ICSSSM.2018.8464999.

[23] F. Chollet i others, „Keras“. GitHub, 2015. [Na internetu]. Dostupno na: https://github.com/fchollet/keras

[24] Martín Abadi *i ostali*, „TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems“. 2015. [Na internetu]. Dostupno na: https://www.tensorflow.org/

[25] D. Pavelić, „Metoda samonadziranog učenja za hladni start sustava preporuke znanstvenih članaka temeljena na slučajnim šetnjama“. lipanj 2022. [Na internetu]. Dostupno na: https://github.com/Audida/WRS-paper-recommender

[26] F. Schroff, D. Kalenichenko, i J. Philbin, „FaceNet: A unified embedding for face recognition and clustering“, lip. 2015. doi: 10.1109/cvpr.2015.7298682.

[27] François Chollet, *Deep learning with Python*, 1. Shelter Island, NY : Manning Publications Co., 2018.

[28] A. Cohan, S. Feldman, I. Beltagy, D. Downey, i D. S. Weld, „SPECTER: Document-level Representation Learning using Citation-informed Transformers“. arXiv, 2020. doi: 10.48550/ARXIV.2004.07180.

[29] I. Beltagy, K. Lo, i A. Cohan, „SciBERT: A Pretrained Language Model for Scientific Text“, 2019, doi: 10.48550/ARXIV.1903.10676.

[30] M. Ostendorff, N. Rethmeier, I. Augenstein, B. Gipp, i G. Rehm, „Neighborhood Contrastive Learning for Scientific Document Representations with Citation Embeddings“. arXiv, 2022. doi: 10.48550/ARXIV.2202.06671.

9 Sažetak

Metoda samonadziranog učenja za hladni start sustava preporuke znanstvenih članaka temeljena na slučajnim šetnjama

Dora Pavelić
Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva
Unska 3, 10000 Zagreb, Hrvatska
dora.pavelic2@fer.hr

Poznati izazov u sustavima preporuke jest tzv. problem hladnog starta koji nastaje u ranoj fazi produkcije kada još nemamo zabilježene implicitne niti eksplicitne korisničke interakcije na temelju kojih bi sustav mogao generirati preporuke. U okviru rada, predložena je metoda WRS (engl. *Weighted Random Surprise*) koja koristi tehnike samonadziranog učenja kojima je neuronska mreža sposobna učiti reprezentacije korisnika i sadržaja. Naučene reprezentacije koriste se za generiranje preporuka. Efikasnost i preciznost razvijene metode pokazana je na temelju pravih preferencija korisnika dobivenih uz pomoć Twittera. Elementi razvijene metode mogu poslužiti za razvoj drugih sustava pa tako biti od koristi i za komercijalne svrhe zbog malog utroška memorije, velike brzine izračuna preporuka i zadovoljavajuće preciznosti naučenog dubokog modela.

Ključne riječi: *Samonadzirano učenje; Sustavi preporuke; Hladan start; Obrada prirodnog jezika; Slučajne šetnje*

10 Summary

A self-supervised method for cold-start scientific article recommendation based on random walks

Dora Pavelić
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, 10000 Zagreb, Croatia
dora.pavelic2@fer.hr

A well-known challenge in recommendation systems is the so-called cold start problem that arises in the early stage of production when we do not yet have recorded implicit or explicit user interactions based on which the system could generate recommendations. As a part of the work, the WRS (Weighted Random Surprise) method is proposed. WRS methos uses self-supervised learning techniques with which the neural network is capable of learning user and content representations. This learned representations are used to generate recommendations. The effectiveness and precision of the proposed method was demonstrated on the basis of real user preferences obtained with the help of Twitter. The elements of the developed method can be used for the development of some other recommender systems and thus be useful for commercial purposes due to the low memory consumption, the high speed of recommendation calculation and the satisfactory precision of the learned representations.

Keywords: *Self-supervised Learning; Recommendation systems; Cold start; Natural language processing; Random walk*