RAČUNARSKI FAKULTET
UNIVERZITET UNION

RAZVOJ SEMANTIČKIH SISTEMA ZA PRETRAGU PODATAKA I NJIHOVO OBUČAVANJE

Diplomski rad

Mentor:
Prof dr Nemanja Ilić

Kandidat:
Novak Živanić 23/19 RN

Beograd, 2023.

SADRŽAJ

1. UVOD

Količina podataka koja se nalazi na internetu neprestalno raste, što dovodi do velikog broja prilika i izazova. Korisnici su dobili pristup neograničenom skupu znanja, ali pronalaženje potrebne informacije u moru veb sajtova postaje sve veći izazov.

Sistemi zasnovani na rečima bili su osnova pretraživanja velike količine tekstualnih podataka još od pojave interneta, zbog toga ih ubrajamo u tradicionalne metode pretrage. Međutim, uprkos svojoj popularnosti, oni se oslanjaju na tačna podudaranja između ključnih reči, sto ih čini manje efikasnim u određenim situacijama. Semantička sličnost, kao mera sličnosti u značenju dva teksta, biva totalno zanemarena u tradicionalnim sistemima. Takođe, limitirani su svojom nesposobnošću da razumeju kontekst upita korisnika. Kao odgovor na ove nedostatke, pojavili su se sistemi za prepoznavanje semantičke sličnosti između tekstova koji dopunjuju ili čak u potpunosti zamenjuju tradicionalne sisteme.

Sistemi za prepoznavanje semantičke sličnosti koriste napredne tehnike obrade prirodnog jezika (Natural Language Processing - NLP) i duboko učenje (Deep Learning) kako bi bolje razumeli značenje reči, fraza i rečenica. U cilju efikasnog pronalaženja semantičke reprezentacije, razvijen je niz modela kao što su word2vec, rekurentne neuralne mreže (RNN) i transformer arhitekture. Ono što je zajedničko svim ovim modelima jeste upotreba embedinga i vektorskih reprezentacija kao osnovnih alata za obradu i razumevanje semantike jezika. Transformer arhitektura, sa svojom funkcijom pažnje sa više glava (engl. multihead-self-atention), ističe se u razumevanju semantičkih odnosa i prepoznavanja kontekstualnih nijansi unutar tekstova, i zbog toga su transformeri osnovna komponenta svakog modernog sistema za semantičku pretragu.

Ovaj diplomski rad ima za cilj da istraži i analizira problem pretrage podataka, da istraži tradicionalne, kao i neuralne pristupe rešavanju ovog problema i na praktičan način pokaže njihove prednosti i mane. Oslanjaće se na bogatu literaturu iz oblasti obrade prirodnog jezika i dubokog učenja, sa posebnim fokusom na najnovija istraživanja i prakse u industriji.

Rad se sastoji od pet celina. U drugom poglavlju upoznaćemo se sa problememom pretrage podataka i istražiti osnovne principe rada tradicionalnih sistema za pretragu, dublje razmatrajući njihove prednosti i mane. Treće poglavlje fokusira se na neuralne

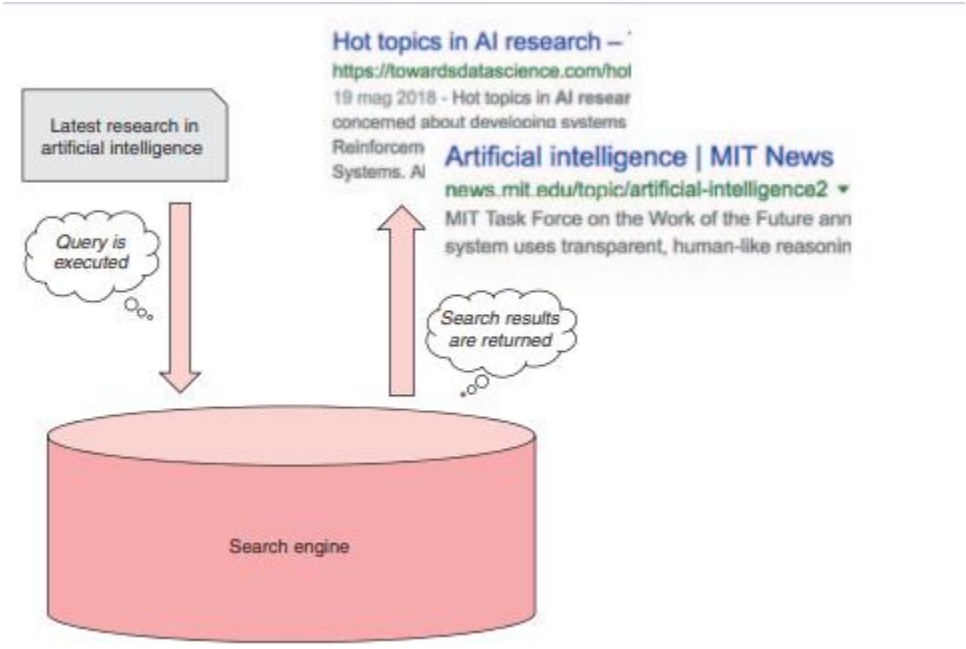
mreže, kako prilagoditi tekst ulaznom sloju neuralne mreže, latentnim prostorima (embedinzima) i najpoznatijim arhitekturama neuralnih mreža koje koristimo za obradu teksta. Četvrto poglavlje zalazi dublje u teoriju iza embedera teksta i semantičkih pretraživača koji su glavni fokus praktičnog dela ovog rada. Peto poglavlje predstavlja praktični deo, treniranje embedera teksta koji se koristi za semantičku pretragu. Ova sekcija nam takođe daje uvid u najnovije tehnike treniranja, funkcije troška, metode za smanjivanje memorijskog otiska, kao i nove metode evaluacije ovakvih sistema za pretragu.

2. PRETRAGA PODATAKA (INFORMATION RETRIEVAL)

Naš fokus biće na pretraživanju teksta, gde korisnik unosi tekstualni upit i sistem mu vraća rangiranu listu rezultata pretrage. Rezultati pretrage mogu biti pasusi teksta ili celi tekstualni dokumenti. Cilj je da rangiramo željene rezultate korisnika na vrh liste. U daljem tekstu ovakve sisteme nazivaćemo **Pretraživači**.

2.1. Pretraživači

Pretraživači su dakle, programi koje ljudi koriste kako bi došli do željenih informacija. Glavna vrednost pretraživača jeste mogućnost da transformiše neobrađene podatke u smislene informacije. Pretraživač se trudi da unese smisao u podatke koje dobije kako bi umeo da vraća smislene odgovore korisnicima. Korisnici retko kad traže mnogo podataka na određenu temu, već traže specifičnu informaciju, i bili bi zadovoljniji samo jednim odgovorom, a ne sa hiljade rezultata koje moraju da pregledaju.



Slika 2.1. Primer postavljanja upita pretraživaču [2]

Osnovne odgovornosti pretraživača su:

- ***Indeksiranje*** – efikasno unošenje i skladištenje podataka u sistem tako da se mogu preuzeti brzo
- ***Postavljanje upita*** – mogućnost postavljanja upita od strane korisnika
- ***Rangiranje*** – prezentovanje i rangiranje rezultata prema određenoj metrici, tako da najbolje zadovolji korisnika

Sledeća sekcija fokusiraće se na unošenje i skladištenje teksta u tradicionalnim pretraživačima. Ovo je krucijalno jer direktno utiče na brzinu pretraživača i njegovu sposobnost dohvatanja relevantnih podataka.

2.1.1. Reči, termini, dokumenti i indeksiranje

Kompjuteri, za razliku od ljudi, jako su dobri u skladištenju velikih količina delova teksta od ulaznih dokumenata. Ovaj proces nazivamo *analiza teksta*, razbijanje dokumenta na njegove sastavne delove, uglavnom reči. Zbog činjenice da se sastavni delovi dokumenta razbijeni tokom analize mogu sastojati od više reči (npr. “Novi Sad”, “New York”, …) ili samo dela reči („don’t“ -> „do“, „not“) ne nazivamo ih reči, nego ***termini*** ili ***tokeni.*** Termini (tokeni) su osnova svakog tradicionalnog sistema za pretragu. Korisnik unosi skup termina koje očekuje u rezultatima i pretraživač vraća listu dokumenata koji sadrže neke ili sve termine iz upita. To je tradicionalni način pretrage, i sistem na kom su zasnovani prvi pretraživači pre nekoliko decenija.

Program za analizu teksta je obično organizovan u pajplajn (engl. *pipeline*): lanac komponenti, gde svaka uzima izlaz prethodne komponente kao ulaz. Takvi pajplajni se uglavnom sastoje od dve vrste komponenata:

- *Tokenizatori* - komponente koje razdvajaju ulazni niz u reči, fraze, simbole, termine (tokene)
- *Token filteri* - komponente koje kao ulaz primaju niz tokena (od tokenizatora ili drugog filtera) i modifikuju ih, brišu ili generišu nove tokene.

Za ulazni dokument, izlaz pajplajna je sekvenca termina:

Slika 2.2. Rezultat prolaska teksta “I like sesarch engines” kroz pajplajn za analizu teksta

Takođe, neki tokeni nisu neophodni za pretraživač, česta je praksa da se izbegava skladištenje termina koji su učestali, i ne nose puno informacija o samom dokumentu. Na primer, u engleskom jeziku termini poput “a”, “an”, “of”, “the”, itd. bivaju izbačeni token filterom. Ovi termini se takođe nazivaju i ***stop reči.*** Ako bismo zadržali stop reči, u slučaju da naš upit sadrži stop reč, drastično se povećava broj dokumenata koje moramo pogledati i oceniti relevantnost sa upitom, što šteti performansama sistema.

2.2. Tipovi pretraživanja

2.2.1. Ad-hoc pretraga

Ad-hoc pretraga kao rezultate vraća listu celih dokumenata iz korpusa. Ovo je klasičan problem koji rešavaju najpoznatiji komercijalni pretraživači poput *Google*, *Bing*, *DuckDuckGo*,... Upiti u ad-hoc sistemima obično su dosta kraći od dužine dokumenata iz korpusa. Pretraživači koriste specijalizovanu *indeks strukturu*[1] za pretragu milijarde dokumenata. Ad-hoc pretraga se smatra jednim od najvažnijih problema u pretrazi podataka. Dnevni broj upita postavljen gugl pretraživaču u proseku iznosi 3.4 milijarde[2], što ukazuje na veliku potražnju za ad-hoc pretraživačima.

2.2.2. Direktno odgovaranje na pitanja (engl. Question answering)

Direktno odgovaranje na pitanja uglavnom vraća listu kraćih isečaka teksta. Problem može biti definisan kao odabir jednog od više ponuđenih odgovora (tipično entiteta), rangiranja isečaka tekstova i pasusa iz korpusa dokumenata, ili može uključiti sintezu tekstualnih odgovora prikupljanjem dokaza iz jednog ili više izvora. Odgovaranje na pitanja predstavlja drugačiji skup izazova od ad-hoc pretrage. Za razliku od dugačkih tekstova, pojedinačne rečenice i kratki isečci teksta se tipično fokusiraju samo na jednu temu. Međutim, odgovori često koriste drugačiji rečnik od onog koji se koristi za postavljanje pitanja. Na primer, tekst sa odgovorom na pitanje “Koje godine je rođen Novak Đoković?” možda neće sadržati termin ‘godine’. Takođe, fraza ‘koje godine’ implicira da tekst odgovora treba sadržati godinu, kao na primer ‘1987’ u ovom slučaju. Odatle, sistemi za odgovaranje na pitanja, u daljem tekstu **QA pretraživači**, moraju modelovati odgovor u odnosu na nameru pitanja. Jedan primer ovakvog sistema je yottaanswers[3].

2.3. Osobine dobrog pretraživača

Pre nego što opišemo bilo koji model pretraživača, važno je napomenuti neke atribute koje očekujemo od dobrog sistema za pretragu. Za svaki pretraživač, relevantnost rezultata koje daje za postavljeni upit je od najveće važnosti. Ali merenje relevantnosti može biti prilagođeno svojstvima sistema koje želimo da postignemo (robusnost, osetljivost i efikasnost). Ovi atributi ne samo da definišu dizajn naših modela, već služe i kao merila za kasnija **poređenja semantičkih i ne-semantičkih pristupa.**

2.3.1. Semantičko razumevanje

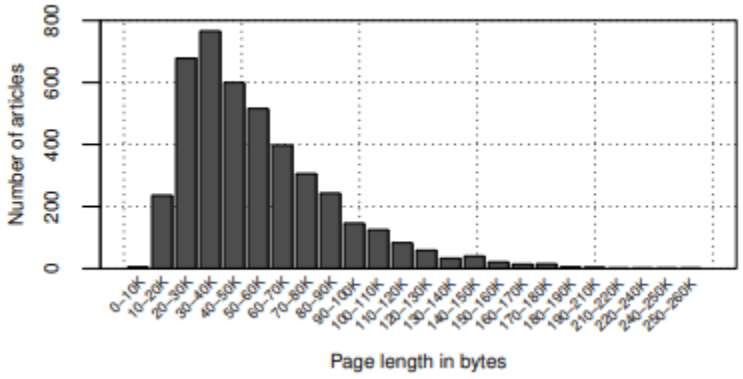
Većina tradicionalnih pristupa ad-hoc pretraživača prebrojava ponavljanja reči iz upita i reči iz dokumenata. Tačno podudaranje reči između upita i teksta dokumenta, iako jednostavno i efikasno, gubi se mogućnost razumevanja teksta.

[1] Obrnuti index(engl. reverse-index) vise u poglavlju X
[2] https://www.internetlivestats.com/google-search-statistics/
[3] https://yottaanswers.com/

Razumevanje semantike, ide dalje od mapiranja reči iz upita na reči iz dokumenata [5]. Dobar pretraživač treba da uzme u obzir da su reč *engl.* “hot” i *engl.* “warm” slične, isto kao i reči *engl.* „dog“ i *engl.* „puppy“, ali mora i da razlikuje da korisnik koji pretražuje *engl.* “hot dog”, nije zainteresovan za *engl.* “warm puppy”. Ovi primeri pokazuju kako sistemi za pretragu treba da sadrže šire razumevanje o pojmovima i osobinama sveta koji nas okružuje. Oni su motivisali pretraživače da koriste latentne reprezentacije namera korisnika i upita, kako bismo mogli da obavljamo “ne-egzaktna“ podudaranja.

2.3.2. Robusnost na varijanse dužina ulaza

Tipični korpus dokumenata sadrži dokumenta različitih dužina (pogledati sliku 2.3). Dobar pretraživač mora biti u mogućnosti da radi sa dokumentima različitih dužina, bez pristrasnosti (engl. *bias*) ka kratkim/dugačkim dokumentima. Relevantni dokumenti mogu sadržati nerelevantan sadržaj. Normalizacija dužine dokumenata je dobro proučavana u pretraživačima [6], i treba je uzeti u obzir tokom dizajniranja pretraživača.



Slika 2.3: Distribucija dužina članakova sa vikipedije (jun 2014) [1]

2.3.3. Robusnost na varijanse korpusa

Zanimljivo je uzeti u obzir koliko dobro pretraživač radi na korpusu čije su distribucije drugačije od korpusa nad kojim je treniran. Pod distribucijom podrazumevamo jezik, vokabular korišćenih reči, prosečnu dužinu tekstova, veličinu korpusa i slično... Klasični modeli pretraživača kao što su BM25[4] [3] imaju jako mali broj hiper-parametara, što im daje dobre performanse bez mnogo truda. Za razliku od njih, za modele zasnovane na neuralnim mrežama [4] koji se sastoje od milione (pa čak i milijarde) parametara je poznata osetljivost na razlike u distribuciji u podacima.

2.3.4. Robusnost na ulazne greške

Svaki pretraživač treba da uzme u obzir korisničke greške. One se najčešće javljaju u upitima ali ih ima i u dokumentima. Greške su uglavnom gramatičke ili nastale usled kucanja na tastaturi (engl. typos). Greške se javljaju u čak 10 do 12 posto *guglovih upita*

[4] Više reči o BM25 u poglavlju 2.6.

[61]. Pretraživači uglavnom imaju posebne komponente zadužene za prepoznavanje ovakvih grešaka.

2.3.5. Senzitivnost na kontekst

Pretraživači mogu iskoristiti mnoge informacije o kontekstu (vreme, mesto, pol korisnika, jezik, lista prethodnih upita...). Upit "vremenska prognoza" može se odnositi na Beograd ili Novi Sad, u zavisnosti od lokacije korisnika. Na upit "crvena zvezda rezultati" je mnogo lakše odgovoriti ako znamo da li je korisnik više zainteresovan za košarku ili fudbal, ili ako je pretraživač upoznat sa najsvežijim sportskim rezultatima. Relevantnost se u velikom broju upita nalazi u kontekstu korisnika i zadatka, što treba uzeti u obzir pri projektovanju sistema za pretragu.

2.3.6. Efikasnost

Efikasnost pretraživača je jedna od ključnih osobina. Kao što smo već spomenuli, komercijalni veb pretraživači uslužuju i do 3.4 milijarde upita dnevno (~40 hiljada po sekundi), pri tome da za svaki upit moraju da pregledaju milijarde tekstualnih dokumenata. Pretraživači tipično imaju arhitekture od više slojeva [7], u svakom sloju eliminišu deo korpusa dok ne ostanu sa setom najkvalitetnijih dokumenata za rangiranje. Početni slojevi prolaze kroz milijarde dokumenata, eliminišući irelevantne dokumente i smeće. Dok kasniji slojevi obrađuju mnogo manje podataka, što im omogućava da budu sofisticiraniji i 'pametniji'.

2.4. Notacija

Značenje	Notacija
Pojedinačni upit	q
pojedinačni dokument	d
Skup upita	Q
Kolekcija dokumenata	D
Termin (token) u upitu q	t_q
Termin (token) u dokumentu d	t_d
Vokabular svih termina	T
Skup vraćenih rezultata za upit q	R_q
Par rezultata (dokument d na poziciji i)	$< i, d >$, gde $< i, d > \in R_q$
Ocena relevantnosti dokumenta d za upit q	$rel_q(d)$
Broj pojavljivanja termina t u dokumentu d	$tf(t, d)$
Broj dokumenata koji sadrže termin t	$df(t)$

Tabela 2.1. Notacija

Tabela 2.1. nije potpuna, ona služi kao referenca za poređenje i razumevanje kompleksnijih izraza korišćenih u Sekciji 2.5. i 2.6.

2.5. Merenje kvaliteta

Merenje kvaliteta sistema za pretragu predstavlja veliki izazov. Veliki broj istraživanja [8][9] pokazao je da korisnici pretraživača većinu svoje pažnje posvećuju samo najbolje rangiranim rezultatima. Prema guglovoj statistici, 27.6% korisnika ne gleda dalje od prvog rezultata[5], dok drugu stranu guglovih rezultata (od 10. do 20. rezultata) posećuje samo 0.6% korisnika[6]. Zbog toga, jako je važno da metrike kvaliteta u obzir uzimaju pozicije rangiranih rezultata.

Pretpostavimo da želimo da rangiramo skup rezultata *R*. Idealna rangiranja zasnovana su na ručno labeliranim primerima i ljudskim presudama. Ove metrike se tipično računaju za svaku poziciju u rangiranju, pa se uzima prosek svih upita u test skupu. Sada ćemo opisati nekoliko takvih metrika.

Preciznost i odziv

Preciznost i odziv, obe mere računaju razliku broja dohvaćenih relevantnih dokumenata za upit q, ali sa ukupnim brojem dokumenata u dohvaćenom skupu rezultata Rq i ukupnim brojem relevantnih dokumenata u kolekciji D, respektivno. Jednostavnije rečeno:

- Preciznost (Precision) nam govori prosečnu relevantnost dokumenata koje smo dohvatili

$$Precision_q = \frac{\sum_{<i,d>\in R_q} Rel_q(d)}{|R_q|} \tag{2.1}$$

- Odziv (Recall) nam govori procenat postignutog rezultata u odnosu na maksimalni rezultat koji smo mogli da ostvarimo

$$Recall_q = \frac{\sum_{<i,d>\in R_q} Rel_q(d)}{\sum_{d\in D} Rel_q(d)} \tag{2.2}$$

Mana ovih mera je ne uzimanje u obzir redosleda u kom su se dokumenti nalazili, podjednaku važnost dajemo svakoj poziciji u rezultatima.

[5] https://backlinko.com/google-ctr-stats
[6] https://keyword.com/blog/how-to-climb-from-page-2-to-page-1-onserps

Mean reciprocal rank (MRR)

Srednji recipročni rang [10] se računa preko binarnih presuda relevantnosti (dokument *d* je relevantan upitu *q* $Rel_q(d) = 1$, nije relevantan $Rel_q(d) = 0$). Računamo ga kao recipročni rang najbolje rangiranog relevantnog dokumenta, usrednjen za sve upite.

$$RR_q = \max_{<i,d>\in R_q} \frac{rel_q}{i} \tag{2.3}$$

Mean average precision (MAP)

Srednja prosečna preciznost [11] za rangiranu listu dokumenata *Rq*

$$AveP_q = \frac{\sum_{<i,d>\in R_q} Precision_{q,i} \times Rel_q(d)}{\sum_{d\in D} Rel_q(d)} \tag{2.4}$$

gde, $Precision_{q,i}$ predstavlja preciznost izračunatu do ranga i za upit q. Prosečna preciznost se generalno koristi kada su ocene relevantnosti binarne, ali postoje varijante ove metrike [12] gde se koristi i skala ocena kao mera relevantnosti. Srednja vrednost prosečne preciznosti nad svim upitima se koristi kao MAP mera kvaliteta.

Normalized discounted cumulative gain (nDCG)

Postoji nekoliko varijanti DCG [13] metrike. Ona se najčešće koristi kada imamo skalu ocena kao meru relevantnosti. Skala je tipično od nula do četiri. Jedan oblik ove metrike je:

$$DCG_q = \sum_{<i,d>\in R_q} \frac{2^{Rel_q(d)}-1}{log_2(i+1)} \tag{2.5}$$

Idealni DCG za upit q ($IDCG_q$) se računa na isti način kao DCG_q, ali sa pretpostavkom da su vraćeni dokumenti idealno rangirani. Normalizovani DCG ($nDCG_q$) se onda računa na sledeći način:

$$nDCG_q = \frac{DCG_q}{IDCG_q} \tag{2.6}$$

2.6. Tradicionalni modeli

U ovoj sekciji proćićemo kroz način rada tradicionalnih modela za pretragu, baziranih na tačnom podudaranju ključnih reči i statističkim metodama. Decenije istraživanja ovakvih pristupa je značajno uticalo na razvoj današnjih semantičkih pretraživača. Ovi modeli takođe služe kao osnova za poređenje kvaliteta novih modela.

2.6.1. Tf-idf (term frequency-inverse document frequency)

Tf-idf je mera važnosti specifične reči (termina) za dokument u nekom korpusu, prilagođena tome da se neke reči češće javljaju od drugih. Tf-idf se sastoji od dve mere, učestalost termina (tf) i inverzne učestalosti dokumenta (idf). Postoji nekoliko varijanti računanja obe mere.

Učestalost Termina (tf), tf(t,d)**,** predstavlja relativan broj pojavljivanja termina t u dokumentu d.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.7}$$

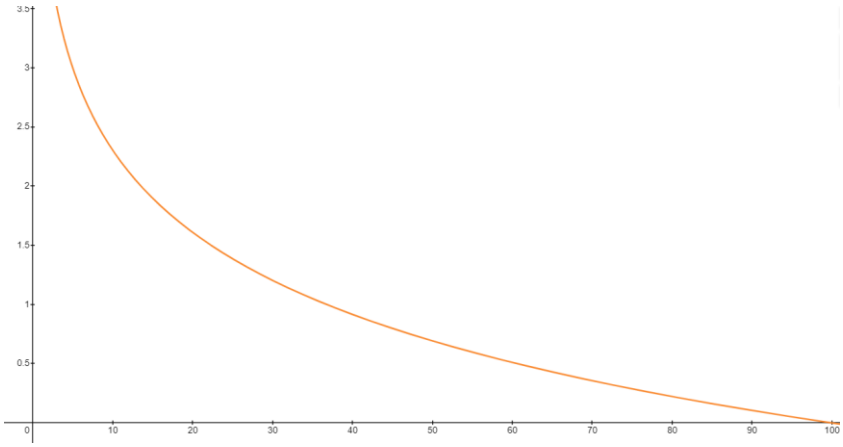
gde je $f_{t,d}$ broj pojavljivanja termina u dokumentu. Obratiti pažnju da je delilac samo broj termina u dokumentu d. Postoji nekoliko dodatnih varijacija za računanje učestalosti termina:

- Bulova "frekfencija" [58], tf(t,d) =1 ako se termin nalazi u dokumentu, tf(t,d)=0 u suprotnom
- Logaritamski skalirana [59], tf(t,d) = log(1+$f_{t,d}$)

U 1972, Karen Sparck Jones osmislio je statističku interpretaciju specifičnosti termina pod nazivom **Inverzna učestalost dokumenta** (idf), koja je postala kamen temeljac za dodeljivanje težine terminima. Idf predstavlja količinu informacije koju nosi određeni termin (da li je redak/čest u korpusu). Idf je logaritamski skaliran inverz procenta dokumenata koji sadrže termin t.

$$idf(t,D) = ln\frac{|D|}{|\{d \in D : t \in d\}|} \tag{2.8}$$

- $|\{d \in D : t \in d\}|$ - broj dokumenata u kojima se termin t pojavljuje (broj dokumenata za koje važi tf(t,d) != 0)



Slika 2.4. Vizualan prikaz idf metrike, x osa predstavlja $|\{d \in D : t \in d\}|$, y osa predstavlja idf vrednost (|D|=100)

Tada tf-idf možemo računati kao

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \tag{2.9}$$

Visoka težina tf-idf se postiže velikom frekfencijom tokena (u datom dokumentu) i niskom učestalosti tokena u dokumentima iz korpusa. Česti tokeni bivaju filtrirani malim težinama. Zbog logaritamskog svojstva idf (Slika 2.4) što je token češći u dokumentima, njegov idf i tf-idf će težiti nuli.

2.6.2. BM25

Okapi BM25 [3], gde se *BM* odnosi na najbolje podudaranje *(engl. Best Matching)*, je jedna od najpoznatijih funkcija za rangiranje zasnovanih na tf-idf meri važnosti. Ona rangira skup dokumenata u odnosu na termine koji se javljaju u upitu korisnika.

$$BM25(q,d) = \sum_{t_q \in q} idf(t_q) * \frac{tf(t_q,d)*(k_1+1)}{tf(t_q,d)+k_1*(1-b+b*\frac{|d|}{avgdl})} \quad (2.10)$$

gde *avgdl* predstavlja prosečnu dužinu dokumenta u korpusu D, k_1 i *b* predstavljaju hiper parametre koji se obično podešavaju na validacionom skupu podataka. U praksi, k_1 se postavlja na podrazumevanu vrednost u opsegu [1.2, 2.0], i *b* kao 0.75.

2.6.3. Indeksiranje

Iako pretraživači dele ulazni dokument na termine zarad bržeg davanja rezultata, krajnji korisnici očekuju da rezultat bude lista dokumenata. Uzmimo primer pretrage upita "besplatne knjige" nad tradicionalnim pretraživačem. Da bismo došli do rezultata efikasno, potrebno je izračunati funkciju rangiranja nad svim dokumentima koji u sebi imaju termin "besplatne" ili termin "knjige". Zbog toga, potrebno je analizirati i prethodno obraditi (engl. *preprocess*) dokumenta za efikasnu pretragu. Ovaj proces analiziranja niza tokena i njihovo skladištenje (zajedno sa dokumentima iz kog dolaze) nazivamo **indeksiranje.** Naziv indeksiranje je potekao od strukture podataka koja se koristi, **invertovani indeks** (engl. *Inverted index*). To je struktura podataka koja mapira termine na dokumenta u kojima se nalaze. Za svaki termin, čuvamo listu dokumenata (njihovih identifikacionih brojeva) koji sadrže taj termin. Tokom postavljanja upita, možemo minimizovati broj dokumenata za koje računamo funkciju rangiranja time što ćemo se fokusirati samo na dokumente koji sadrže termine iz upita. Probajmo da vizualizujemo ovu strukturu sa primerom korpusa od dva dokumenta:

- Braon lisica je preskočila žutog psa (Dokument 1)
- Brza braon lisica preskače žutog psa (Dokument 2)

termin	id dokumenata
braon	1,2
brza	2
preskočila	1
preskače	2
psa	1,2
lisica	1,2
žutog	1,2

Tabela 2.1. Invertovani indeks korpusa sa dva dokumenta

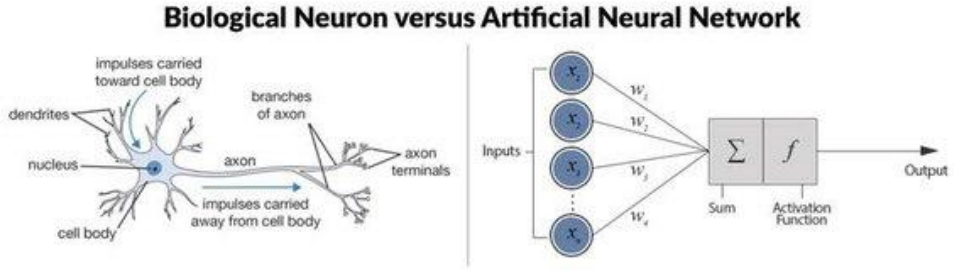
Termini iz korpusa su prošli kroz pajplajn u kom su uklonjena velika slova. Takođe, termin „**je**" ubrajamo u stop reč i gubi se u token filteru stop reči.

3. NEURALNE MREŽE ZA OBRADU TEKSTA

Kao što smo već spomenuli, pretraživači u pokušaju što boljeg razumevanja semantičkog značenja teksta i namere korisnika, koriste napredne tehnike obrade prirodnog jezika (Natural Language Processing - NLP). Poslednjih godina, obrada prirodnog jezika je doživela transformativnu revoluciju, uglavnom vođenu pojavom veštačkih neuralnih mreža. U ovom poglavlju upoznaćemo se sa radom neuralnih mreža i specijalnim arhitekturama posvećenim obradi teksta.

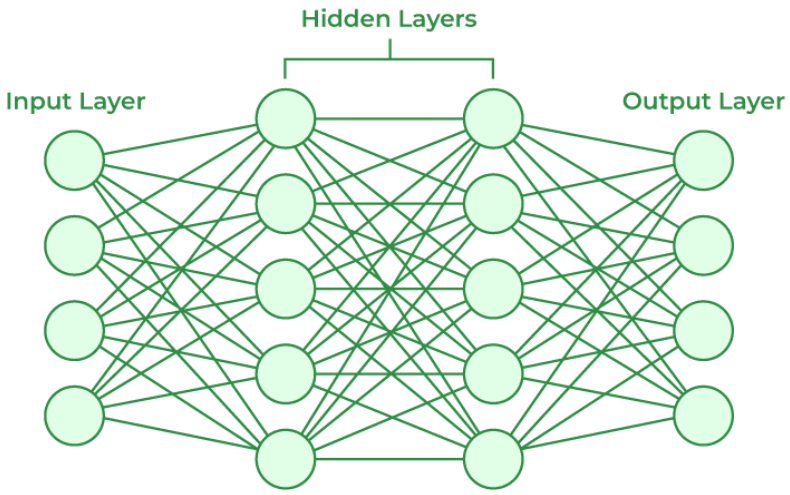
3.1. Neuralna mreža

Veštačke neuralne mreže, poznate kao neuralne mreže, inspirisane su radom ljudskog mozga i njegovom imitacijom. Ljudski mozak računa na drugačiji način od konvencijalnog digitalnog kompjutera. Možemo ga zamisliti kao izuzezno kompleksan, nelinearni, paralelni računar koji uči tokom vremena kroz iskustvo i organizuje svoje međusobno povezane gradivne elemente, neurone, da bi što bolje izvršavao predviđene zadatke. [4] Neuralne mreže pokušavaju da oponašaju mozak i interakciju neurona, gde se procesiranje informacija dešava u zamršenim mrežama.



Slika 3.1. Poređenje biološke neuralne mreže sa veštačkom

Neuralne mreže se sastoje od neurona, organizovanih u slojeve. Ulazni sloj prima podatke, unutrašnji (skriveni) slojevi ih obrađuju i izlazni sloj proizvodi finalni rezultat. Mreže sa velikim brojem skrivenih slojeva nazivamo dubokim neuralnim mrežama. Konekcije između neurona su poznate kao težine, i one određuju količinu propuštene informacije u sledeći sloj. Tačna struktura povezanosti i način na koji vrše izračunavanje definiše arhitekturu te neuralne mreže. Primer najjednostavnije arhitekture neuralne mreže je potpuno povezana neuralna mreža, u kojoj je svaki neuron iz prethodnog sloja povezan sa svakim neuronom iz sledećeg sloja.



Slika 3.2. Primer potpuno povezane neuralne mreže

Neuralne mreže spadaju u kategoriju algoritama nadgledanog mašinskog učenja koja se najbolje može razumeti u kontekstu aproksimacije funkcija. Uzimajući u obzir ulazne podatke i pripadajuće izlazne vrednosti, ove mreže uče složene veze i obrasce, čime omogućavaju generalizaciju na nepoznate ili nove podatke. Ovaj proces učenja je zasnovan na optimizaciji težina veza između neurona kako bismo smanjili razliku između stvarnih i predviđenih izlaza.

Zbog već postojeće obimne literature na temu rada neuralnih mreža [14] [15], ovaj rad će podrazumevati određeno znanje iz ove oblasti i odmah se fokusirati na kompleksnije koncepte i arhitekture neuralnih mreža.

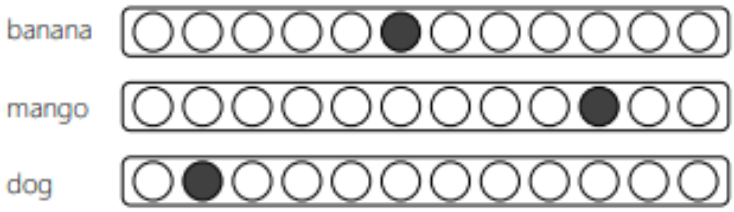
3.2. Vektorska reprezentacija reči i termina

Prvi izazov koji se javlja pri primeni neuralnih mreža u kontekestu obrade teksta odnosi se na prilagođavanje ulaznih podataka neuralnoj mreži. Neuralne mreže zahtevaju ulaz u formi numeričkih vrednosti i vektora. U pretrazi podataka i obradi teksta, najmanja jedinica koju posmatramo su najčešće termini (tokeni) i u ovoj sekciji razmatraćemo njihovo predstavljanje u vektorskom obliku, kako bismo mogli da ih procesiramo algoritmima dubokog učenja. Neke metode za svaki različiti termin imaju različitu vektorsku reprezentaciju, dok neke uče da indentifikuju slične atribute između termina. Različite metode reprezentacije imaju različit osećaj za semantičku sličnost između termina. Neke metode reprezentacije mogu biti proširene i na veće jedinice od tokena (rečenice, pasusi, dokumenta). Biranje metode za vektorsku reprezentaciju ima veliki uticaj na kasnije performanse i kvalitet tokom obrade teksta neuralnim mrežama.

3.2.1. Lokalna reprezentacija

Lokalna reprezentacija ili (engl. *one-hot*) enkodovanje, podrazumeva da svaki token iz vokabulara fiksne dužine *T* predstavljamo binarnim vektorom $\vec{v} = \{0,1\}^{|T|}$ gde je samo

jedna od vrednosti jedan i sve ostale su postavljene na nulu. Svaka pozicija u vektoru $\vec{v}$ odgovara jednom tokenu. Token „banana“, predstavili bismo vektorom koji ima vrednost jedan na poziciji koja odgovara tokenu “banana” i nule na ostalim pozicijama. Slika 3.3 pokazuje različite reprezentacije za različite termine iz vokabulara. Termini koji se ne nalaze u vokabularu obično ili nemaju reprezentaciju i bivaju zanemareni, ili su svi predstavljeni pod istim terminom “UNK”.

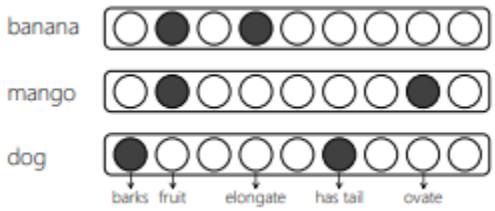


Slika 3.3 Lokalna vektorska reprezentacija termina

3.2.2. Distribuirana reprezentacija

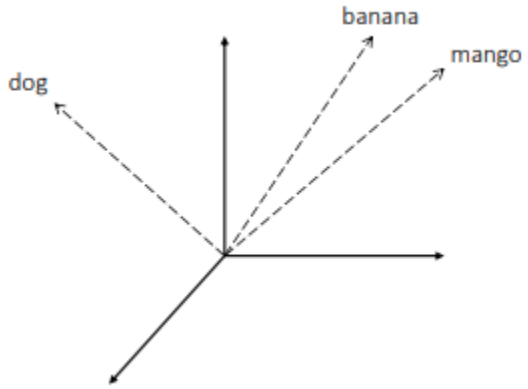
Kod distribuirane reprezenacije svaki termin je definisan vektorom $\vec{v} = \mathbb{R}^{|k|}$. $\vec{v}$ može biti retki (*engl. sparse*) ili gusti (*engl. dense*) vektor, vektor ručno odabranih karakteristika. Glavna hipoteza svake distribuirane reprezenzacije, jeste da vektorska reprezentacija termina omogućava definisanje neke mere sličnosti između termina.

Na primer, na Slici 3.4 “banana” je sličnija terminu „mango“ nego „dog“ zato što su oba termina voćke, ali se i dalje razlikuju u osobinama koje im nisu zajedničke. U ovom slučaju, kod lokalne reprezentacije termina svaki termin je različit i ne možemo odrediti njihovu sličnost, dok u distribuiranoj reprezentaciji karakteristika možemo odrediti sličnost preko zajedničkih karakteristika.



Slika 3.4 Distribuirana reprezentacija termina

Određivanje karakteristika je ključan deo kod distribuiranih reprezentacija. Jedan od pristupa je razmatranje distribucija pojavljivanja kao karakteristika. U hipotezi distribucija [17] tvrdi se da termini koji se koriste u sličnim kontekstima teže da budu semantički slični. John Rupert Firth (1957) [16] je predložio čuvenu ideju o distributivnoj semantici rekavši “reči karakterišu društva reči u kojima se nalaze”. Na ovoj tezi se zasniva veliki deo modela za obradu prirodnog jezika, o njoj će biti više u Sekciji 3.2.3.



Slika 3.5. Vektorske reprezentacije sa slike 3.4. reč banana je bliža reči mango zbog semantičke sličnosti

Kada su vektori visoko dimenzionalni, retki, i bazirani na vidljivim karakteristikama, njih nazivamo eksplicitne vektorske reprezentacije. Kada su vektori gusti, mali (k << |T|), i naučeni iz podataka, njih oslovljavamo sa latentni vektori, iliti **embedinzi**. U oba slučaja nekoliko metrika za distancu može biti korišćeno za definisanje sličnosti između termina, najčešće korišćena je kosinusna sličnost.

$$sim(\vec{v}, \vec{u}) = \frac{\vec{v^t} * \vec{u}}{\|\vec{u}\| * \|\vec{v}\|} \tag{3.1}$$

3.2.3. Latentne reprezentacije (embedinzi)

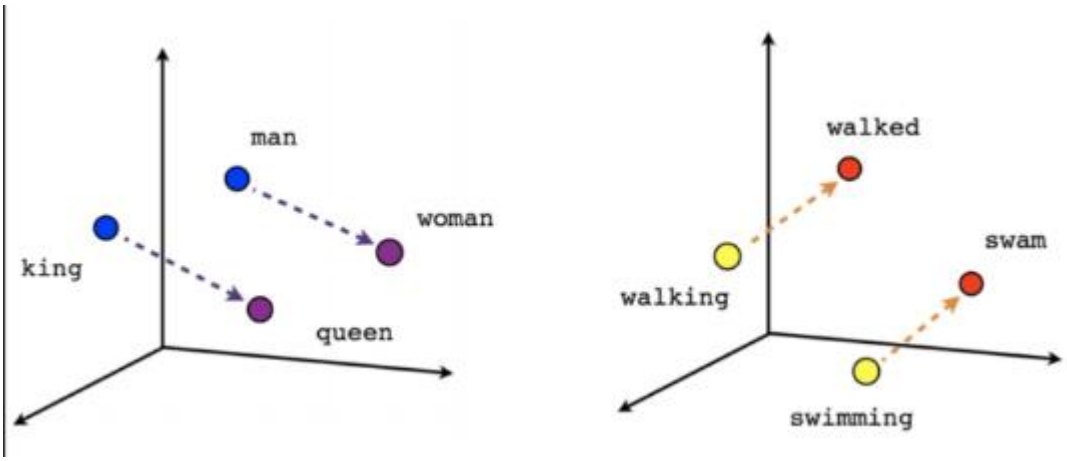
Iako eksplicitne vektorske reprezentacije mogu uhvatiti zanimljive relacije između termina (kao na Slici 3.4.), imaju jednu veliku manu – vektori su retko raspoređeni u prostoru i visokodimenzionalni. Broj dimenzija može biti isti kao i veličina vokabulara. Alternativan pristup su embedinzi, da naučimo vektore nižih dimenzija koji zadržavaju korisne atribute i odnose sličnosti između termina.

Tipičan pristup učenja embedinga uključuje metode zasnovane na gradijentnom spustu koje pokušavaju da predvide karakteristike datog termina [18]. Embedinzi su postali standardan način predstavljanja teksta kao ulaz u neuralne mreže zbog svojih performansi i kvaliteta. U daljem tekstu upoznaćemo se sa jednim algoritmom za njihovo učenje, word2vec.

3.2.4. Word2vec

Word2vec [19] je razvijen od strane Gugla u 2013. godini, i od tad je postao standard za treniranje embedinga termina. Dodatno, rad je uveo zanimljivu ideju matematičkih operacija nad vektorskim reprezentacijama. Na primer, oduzimanje “man-ness” od reči “King” i dodavanje “woman-ness” proizvodi vektor blizak vektoru “Queen”.

$$\vec{v}_{king} - \vec{v}_{man} + \vec{v}_{woman} \approx \vec{v}_{queen} \tag{3.2}$$



Slika 3.6. Prikaz zajedničkih odnosa između embedinga

Word2Vec algoritam karakteristike termina uči na osnovu njegovih suseda unutar prozora fiksne veličine. Predstavljene su dve arhitekture, Skip-gram i CBOW.

Skip-gram arhitektura je neuralna mreža sa jednim skrivenim slojem, dok su ulaz i izlaz neuralne mreže lokalni(*one-hot*) vektori termina. Motivacija skip-gram arhitekture je da model na osnovu trenutnog termina, predvidi njegove susedne termine.

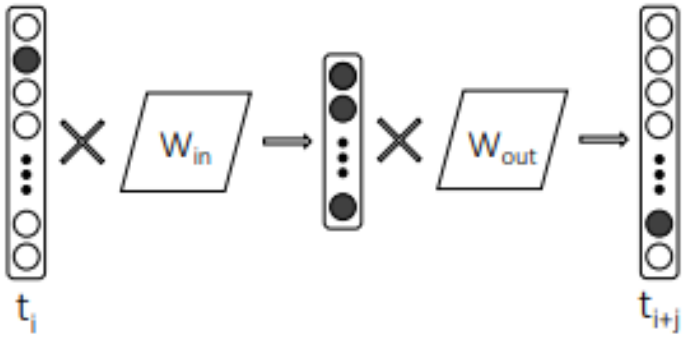
Funkcija troška je definisana na sledeći način:

$$\mathcal{L}_{skip-gram} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-c\leq j\leq c, j\neq i}\log\left(p(t_{i+j}|t_i)\right) \tag{3.3}$$

gde važi

$$p\left(t_{i+j}\middle|t_i\right) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^T(W_{in}\overrightarrow{v_{t_i}})\right)}{\sum_{k=1}^{|T|}\exp\left((W_{out}\vec{v}_{t_k})^T(W_{in}\overrightarrow{v_{t_i}})\right)} \tag{3.4}$$

S predstavlja skup svih pomerajućih prozora dok *c* predstavlja broj termina koji želimo da predvidimo, sa obe strane t_i.

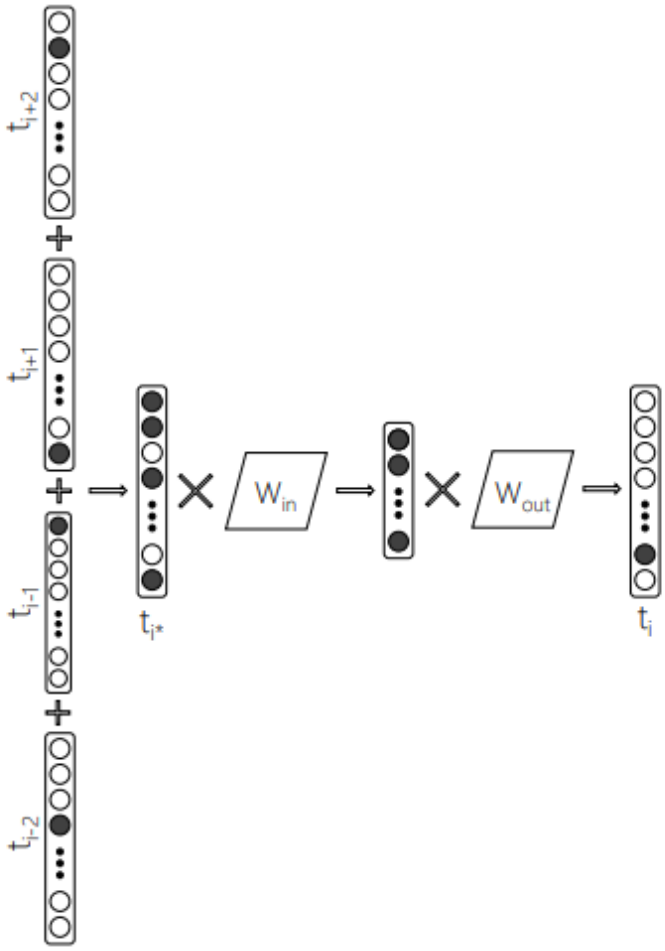


Slika 3.7. Word2Vec, skip-gram arhitektura

Na Slici 3.7. možemo uočiti dva skupa težina, W_{in} i W_{out} koji služe kao obučivi parametri modela. Generalno, nakon obučavanja nad korpusom teksta, W_{in} parametri se zadržavaju i koriste kao embedinzi termina dok W_{out} parametre odbacujemo.

CBOW (*engl.* Continuous bag-of-words) je takođe neuralna mreža sa jednim skrivenim slojem. Ona uči latentne reprezentacije termina tako što na osnovu sesednih reči pokušava da predvidi trenutni termin. Na kraju obučavanja, W_{out} težine se koriste kao embedinzi termina. Funkcija troška kod ovog modela je sledeća:

$$\mathcal{L}_{CBOW} = -\frac{1}{|S|}\sum_{i=1}^{|S|} \log\left(p(t_i|t_{i-c},\dots,t_{i-1},t_{i+1},t_{i+2},\dots,t_{i+c})\right) \tag{3.5}$$

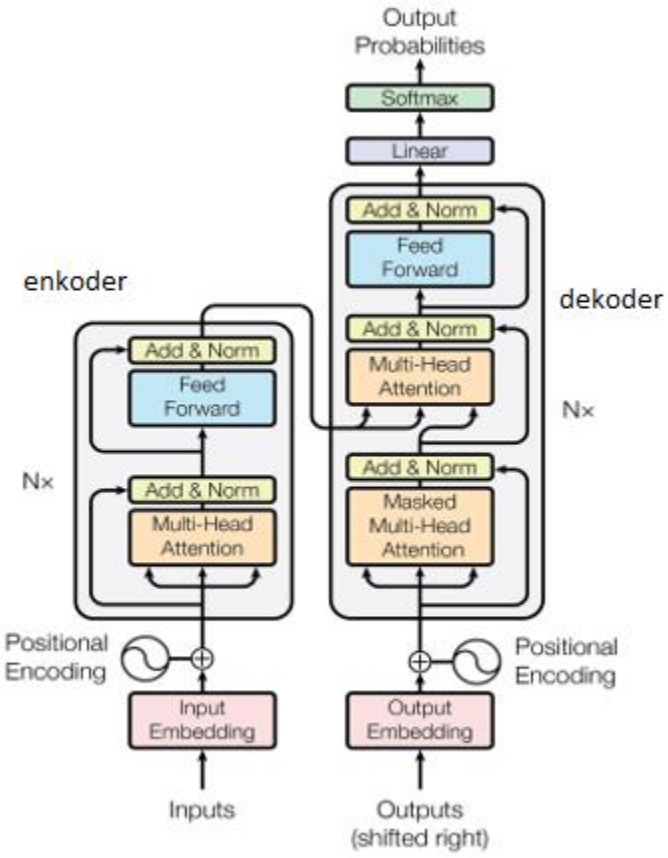


Slika 3.8. Word2Vec, CBOW arhitektura

3.3. Transformer arhitektura

Sada kada smo se upoznali sa neuralnim mrežama i načinima na koje možemo prilagoditi tokene teksta u adekvatne ulaze u neuralnu mrežu, predstavićemo jednu od najpoznatijih arhitektura za obradu teksta. ***Transformer arhitektura*** predstavljena je u poznatom naučnom radu sa naslovom "Attention is all you need" [20] 2017. godine. Ova arhitektura je rešila probleme efikasnosti, nestajućeg gradienta i dalekih zavisnosti između termina koje prethodne arhitekture poput rekurentnih neuralnih mreža (RNN) [21] nisu mogle da reše.

Prvi modeli sastojali su se od enkoder-dekoder arhitekture i bili su namenjeni mašinskom prevođenju teksta. Današnje arhitekture transformera prilagođavaju se u zavisnosti od zadatka koji rešavaju. Naš fokus u daljem tekstu biće na arhitekturi transformera koja se sastoje samo od enkoder komponente (poput BERT [22], RoBERTa [23], ...), sličnu arhitekturu ćemo koristiti kasnije za obučavanje semantičkog pretraživača.



Slika 3.9. Enkoder-dekoder arhitektura transformera

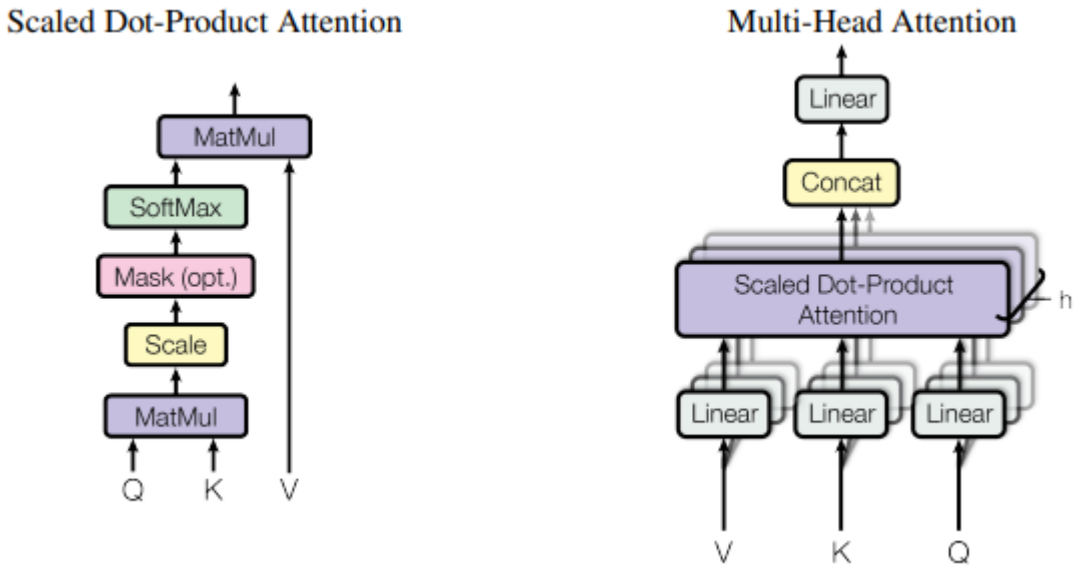
3.3.1. Enkoder komponenta

Enkoder se sastoji od N identičnih slojeva. Svaki sloj se sastoji od dva pod-sloja. Prvi je Pažnja sa više glava, dok je drugi jednostavna potpuno povezana mreža. Između njih se nalazi rezidualna konekcija [24] sa normalizacijom po slojevima [25]. Modeli sa samo enkoderom se koriste kada izlaz neuralne mreže nije sekvencijalan. Izlaz ovakvih modela je vektor kontekstualnih reprezentacija i može se koristiti za razne vrste poslova poput klasifikacije, regresije ili u našem slučaju, pravljenje embeding-a dokumenata koje koristimo za semantičku pretragu.

3.3.2. Mehanizam pažnje

U srži arhitekture transformera je mehanizam pažnje koji omogućava modelu da se fokusira na različite delove ulazne sekvence sa različitim stepenom pažnje. Mehanizam pažnje uključuje izračunavanje rezultata pažnje za svaki element u ulaznoj sekvenci u odnosu na svaki drugi element, određujući važnost odnosa između svaka dva elementa. Ovaj proces se ponavlja više puta kroz slojeve transformera, omogućavajući modelu da iterativno precizira svoje razumevanje ulaznih podataka.

Funkcija pažnje može se opisati kao mapa upita i skupa ključ-vrednost parova na izlaz, gde su upit, ključevi, vrednost i izlaz sve vektori. Izlaz se računa kao ponderisana suma vrednosti, gde su težine izračunate uz pomoć odnosa upita i ključa.



Slika 3.10. Pažnja sa skalarnim proizvodom (levo) i pažnja sa više glava (desno)

Pažnja sa skalarnim proizvodom je funkcija pažnje najčešće korišćena u transformerima. Ulaz se sastoji od upita Q i ključeva K dimenzije d_k. Nakon računanja skalarnog proizvoda vektora između upita i ključeva, delimo ih sa $\sqrt{d_k}$ i primenjujemo softmaks kako bismo dobili ponderisane težine pažnje koje množimo sa vrednostima.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.6)$$

Pored pažnje sa skalarnim proizvodom, može se koristiti i pažnja sa sabiranjem. Iako teoretski imaju istu kompleksnost, množenje je mnogo brže i efikasnije u praksi, jer se može primeniti visoko optimizovano množenje matrica.

Za razliku od obične funkcije pažnje gde se pažnja računa sa d_k dimenzionalnim vektorima, **Pažnja sa više glava (***engl. Multi-Head attention***)**, koju koriste transformeri, računa funkciju pažnje nad *h* različitih linearnih projekcija upita, vrednosti i ključeva. Svaka projekcija može se računati paralelno, i daje d_k/h dimenzionalne izlaze. Sve projekcije se na kraju ponovo spajaju, kako što je prikazano na slici 3.10. Cena računanja pažnje sa više glava je slična ceni računanja jedne funkcije pažnje nad svim dimenzijama.

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O \quad (3.7)$$

gde je $$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Obučivi parametri projekcija su matrice $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}, W_i^K \in \mathbb{R}^{d_{model} \times d_k}, W_i^V \in \mathbb{R}^{d_{model} \times d_k}$ and $W^O \in \mathbb{R}^{hd_k \times d_{model}}$, gde važi $d_k = \frac{d_{model}}{h}$.

3.3.3. Poziciono enkodovanje

Kako transformeri nemaju iteracije ili konvolucije, moramo im na neki način ubaciti podatak o redosledu elemenata u sekvenci koju obrađuju. Informacija se prosleđuje relativno ili apsolutno u odnosu na poziciju tokena u sekvenci. Poziciono enkodovanje se dodaje samo na ulazne embedinge tokena kao "šum". Vektor pozicionog enkodovanja ima iste dimenzije kao i vektor ulaznih embedinga, tako da se dva vektora mogu sabrati u jedan. Vektor pozicionog enkodovanja je najčešće sinusna i kosinusna funkcija različitih frekfencija:

$$PE_{(pos,2i)} = \sin\left(pos/10000^{2i/d_{model}}\right) \quad (3.8)$$

$$PE_{(pos,2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right) \quad (3.9)$$

gde je *pos* pozicija, a *i* trenutna dimenzija, tako da će svaka dimenzija pozicionog enkodovanja odgovarati sinusoidi.

3.4. Neuralni pristup u tradicionalnim pretraživačima

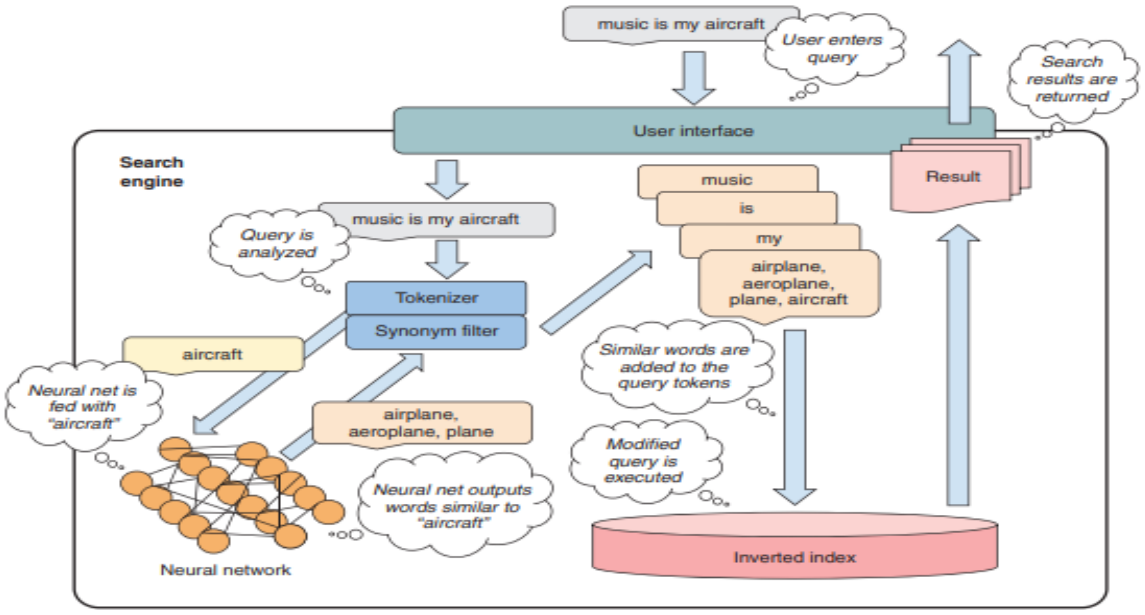
Nakon popularizacije neuralnih mreža, počela je i njihova primena u problemu pretrage podataka. Prvi neuralni pristupi u pretraživačima služili su kao dopuna tradicionalnim modelima baziranim na podudaranju termina. Neuralne mreže bi uticale na predstavljanje upita, predstavljanje dokumenata, funkcija za rangiranje ili kao model za pretpostavljanje sličnosti. U ovom poglavlju uradićemo kratak pregled nekoliko načina na koje možemo upotrebiti neuralne mreže da poboljšamo tradicionalne metode, a onda

ćemo u Sekciji 4 predstaviti nov pristup korišćenja neuralnih mreža koji zamenjuje tradicionalne sisteme, ***neuralne pretraživače***.

3.4.1. Proširivanje upita

Najjednostavniji način da poboljšamo tradicionalne sisteme jeste da im omogućimo "ne-egzaktno" podudaranje. Proširivanje upita je tehnika koja pravi izmene nad upitom korisnika i ubacuje nove termine, kako bismo maksimizovala broj relevantnih rezultata za krajnjeg korisnika.

Jedan primer proširenja upita je generisanje sinonima.



Slika 4.1. Primer proširivanja reči "airplane" sinonimima "aeroplane", "plane" i "aircraft"

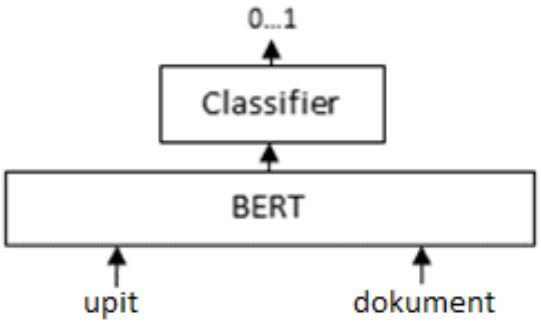
Glavni cilj proširivanja upita je da:

- Minimizujemo upite bez rezultata
- Povećamo preciznost, davanjem prednosti dokumentima koji odgovaraju i upitu i proširenju
- Povećamo odziv (engl. *recall*), ubacivajući tokene koje bismo inače propustili

Proširivanje upita neuralnim metodama možemo postići na više načina. Jedan je da koristimo embedinge reči, i nekom metrikom sličnosti odlučimo kako da proširimo termine iz upita njima bliskim terminima. Druga popularna metoda je korišćenje neuralnih mreža za generisanje teksta (*engl. Seq2seq*) koje generišu proširenje upita ili slične upite. Ovakve modele trenirali bismo na istoriji upita iz logova sistema, kao i ručno labeliranim podacima. Takođe, korišćenjem bim pretrage (engl. *beam search*) [26] možemo efikasno generisati više raznovrsnih upita odjednom.

3.4.2. Rangiranje

Neuralne mreže su dobre u učenju funkcija za rangiranje. Dostupnost velike količine korisničkih podataka o klikovima, kao i dostupnih ručno labeliranih podataka im omogućava da dobro nauče preference korisnika i "reorganizuju" rezultate vraćene od strane pretraživača, tako da najbolje odgovoraju korisniku. Ovi modeli uglavnom kao izlaz vraćaju meru sličnosti između upita korisnika i određenog dokumenta. Nazivamo Ih i kros enkoderi (engl. Cross-Encoder).



Slika 4.2. Primer transformer arhitekture kros enkodera

4. SEMANTIČKI PRETRAŽIVAČI

Verovatno najveća korist koju donose moderne tehnike dubokog učenja za pretragu tekstova je udaljavanje od oskudnih signala ograničenih na egzaktna podudaranja, ka kontinualnim gustim reprezentacijama prostora koje su u stanju da uhvate semantička podudaranja radi bolje relevantnosti modela. Takozvani semantički pretraživači, tema ovog rada, nam omogućavaju da pretragu i rangiranje obavimo direktno na vektorskim reprezentacijama dokumenata (generisanih od strane transformera). Ovaj pristup ima potencijal da prevaziđe problem neusklađenosti vokabulara time što direktno poredi embedinge dokumenta koji u sebi "čuvaju smisao" umesto rerangiranja rezultata vraćenih od strane tradicionalnih metoda (metoda spomenuta u 3.4.2).

Kako je semantička pretraga preko gustih reprezentacija teksta u prostoru nova oblast u usponu, u literaturi ne postoji konzistentna terminologija na koju se možemo osloniti. Sa tim na umu, definisaćemo nekoliko termina kojima ćemo se voditi u ovom poglavlju:

- Terminom ***tekstualni embederi (enkoderi)*** oslanjaćemo se na neuralne arhitekture koje prave guste vektorske reprezentacije tekstova, dokumenata i upita.
- **Semantički *pretraživači*** su pretraživači zasnovani na tekstualnim embederima, koriste njihove vektorske reprezentacije teksta kao glavnu metriku relevantnosti.

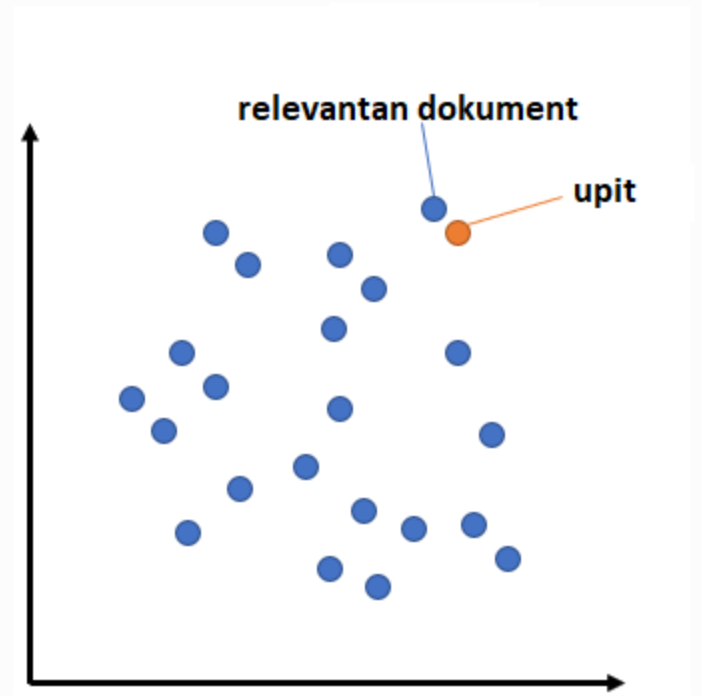
4.1. Definisanje problema

Glavni problem rangiranja dokumenata ostaje isti kao u Sekciji 2, Za dati upit *q,* zadatak je da vratimo listu sa *k* najbolje rangiranih dokumenata iz korpusa. Tekstualni embederi pokušavaju da nauče transformaciju

$$\eta : [t_1 t_2 \dots t_n] \rightarrow \mathbb{R}^m \tag{4.1}$$

nad upitima i dokumentima iz korpusa, sa oznakama $\eta_q(*)$ i $\eta_d(*)$ respektivno, tako da sličnost između $\eta_q(*)$ i $\eta_d(*)$ bude maksimizovana ako je tekst dokumenta d relevantan upitu q i da sličnost između $\eta_q(*)$ i $\eta_d(*)$ bude minimizovana ako dukument i upit nisu relevantni, za datu funkciju sličnosti ϕ.

Za vreme pretrage (postavljanja upita), za dati upit q, pokušavamo da vratimo k dokumenata iz korpusa sa najvećom sličnošću koristeći enkodere η_q i η_d i funkciju sličnosti ϕ. U većini slučajeva, za funkciju ϕ biramo jeftinu, skalabilnu operaciju poput kosinusne distance ili skalarnog proizvoda.



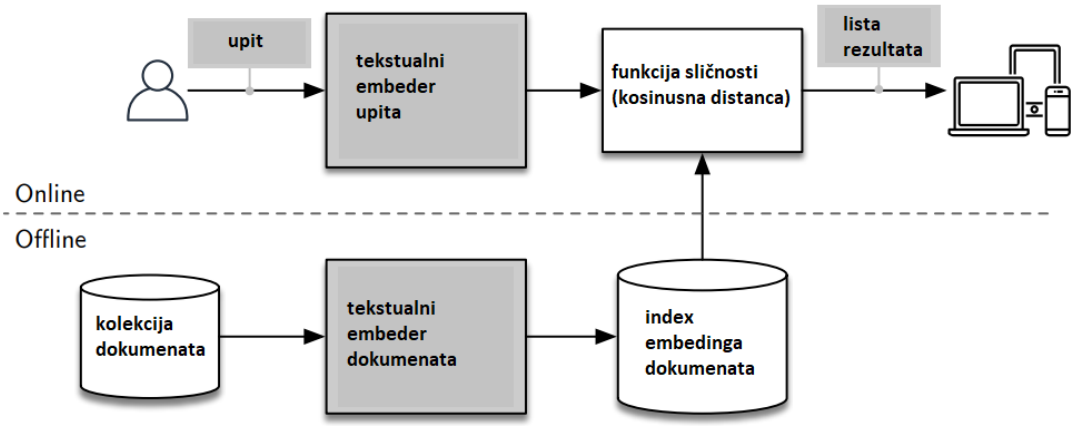
Slika 4.3. Pojednostavljen grafički prikaz dvodimenzionalnog latentnog prostora tekstualnog embedera [60]

Transformacija sa oznakom η (eta) je transformacija koja predstavlja „enkoder“. Enkoderi za upite i dokumente mogu koristiti istu ili različitu neuralnu mrežu. Izlaz enkodera je gusta reprezenacija (vektor fiksne dužine). Jedan od intuitivnih načina da se razmišlja o njima jeste “kao embeding za reči, samo za sekvence tokena”. Vektorske reprezentacije uglavnom imaju stotine dimenzija, za razliku od lokalnih reprezentacija čiji vektori imaju dimenziju veličine vokabulara.

Još jedna velika prednost semantičkih pretraživača jeste mogućnost da izračunamo unapred (engl. *precompute*) tekstualne embedinge dokumenata sa naučenim embederom. Za svaki korisnički upit, onlajn računamo samo embeding upita i poredimo ga sa dokumentima iz korpusa (Slika 4.4.).

Svaki semantički pretraživač se susreće sa dve prepreke:

- **Reprezentacija**, ili dizajn tekstualnog embedera za upite i dokumente, kako bi precizno odredio “značenje” upita i dokumenata iz korpusa.
- **Poređenje**, koje uključuje efikasno računanje sličnosti između upita i velikog korpusa embedinga dokumenata.



Slika 4.4. Šema semantičkog pretraživača [27]

4.2. Traženje najbližih suseda (engl. Nearest Neighbor Search)

Unapred izračunati embedinzi dokumenata se čuvaju u specijalne strukture podataka pod nazivom *indeksi*. U najjednostavnijoj formi, ovi indeksi čuvaju embedinge dokumenata i obezbeđuju algoritam za pretragu takav da, za dati upit, efikasno pronađu dokumenta sa najvećom sličnošću, ili u našem slučaju, sa najmanjom kosinusnom distancom. Problem možemo reformulisati kao problem traženja najbližih suseda (dokumenata) vektorskoj reprezentaciji upita *q*.

Naivan pristup ovom problemu je da koristimo *ravan index (engl. Flat Index)*, koji prilikom upita, računa distancu između svakog dokumenta i upita. Kompleksnost ovog pristupa je O($n * l$) gde je n broj dokumenata a l broj dimenzija.

Pojava semantičkih pretraživača dovela je do razvoja velikog broja algoritama u ovoj oblasti. Moderna efikasna i skalabilna rešenja se zasnivaju na aproksimativnim metodama (*engl. Approximate nearest neighbor (***ANN***) search*). Aproksimativna rešenja su u ovom slučaju prihvatljiva jer je i samo računanje sličnosti aproksimacija.

Prva rešenja aproksimativne pretrage najbližih suseda zasnivala su se na lokalno-senzitivnom heširanju (*engl. locality-sensitive hashing*) [29], ali danas, grafovske metode su se pokazale kao najbolji pristup. Metode bazirane na hiararhiskim navigacionim malim svetovima (*engl. Hierarchical navigable small world graphs*, **HNSW**) [28] su se najbolje pokazale i daju najbolje rezultate po poznatim testovima [30]. Popularna open-source bibilioteka za **ANN** pretragu je ***faiss***[7].

Ovi algoritmi, zajedno sa tehnikama kvantizacije vektora nam omogućavaju da pretražimo indekse sa milijardama dokumenata u desetinama milisekundi.

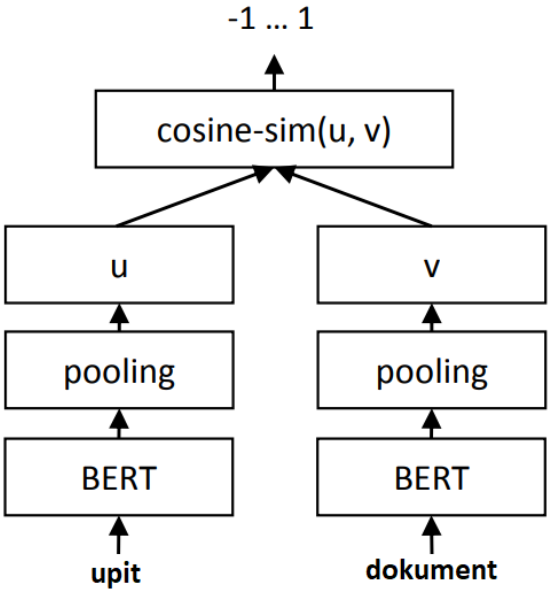
[7] https://github.com/facebookresearch/faiss

4.3. Tekstualni embederi

Podsetimo se definisanja problema iz Sekcije 4.1, za dati enkoder upita (η_q), enkoder dokumenata (η_d) iz korpusa i funkcije sličnosti ϕ(fi), pretraživač uključuje pravljenje sledeće predikcije nad korpusom *C:*

$$P(Relevant = 1|d_i, q) \triangleq \phi(\eta_q(q), \eta_d(d_i))$$

U našem slučaju, podrazumevaćemo da važi $\eta_q = \eta_d = \eta$ (koristimo isti enkoder za upite i dokumenta). Enkoder η zasnovan je na transformer arhitekturi. Ovakav dizajn nazivamo *engl. "Bi-encoder"* dizajn. Termin je predstavljen[31] i skiciran na Slici 4.5.



Slika 4.5. Bi-enkoder arhitektura

Jedan od najpoznatijih bi-enkodera je ***Sentence-BERT***, dizajniran da generiše embedinge sa semantičkim značenjem rečenica. Korišćen je za masivna poređenja sličnosti rečenica, ali ne i za semantičku pretragu. U sledećem poglavlju pričaćemo više o obučavanju bi-enkodera za semantičku pretragu, funkciji troška, evaluaciji i podacima neophodnim za treniranje.

5. PROJEKAT - OBUČAVANJE EMBEDERA ZA SEMANTIČKU PRETRAGU

Projekat se zasniva na obučavanju embedera teksta za semantičku pretragu, prateći tehnike korišćene iz najprestižnijih naučnih radova iz ove oblasti [32][33][34][35]. Pokušaćemo da se što više približimo najboljim (*engl. StateOfTheArt)* modelima poput sbert, gte, i e5, sa ograničenim resursima i podacima. Takođe, uporedićemo rezultate sa tradicionalnim metodama poput BM25.

Za reprodukovanje najboljih (*engl. StateOfTheArt)* rezultata potrebni su klasteri sa hiljade gigabajta video memorije kojima nemamo pristup za ovaj projekat, zbog čega je očekivano da dobijemo lošije ili podjednako dobre rezulate.

5.1. Funkcija troška

Za obučavanje modela koristimo kontrastivno učenje (*engl. Contrastive Learning*). Kontrastivno učenje je tehnika pri kojoj poredimo primere jedne sa drugima kako bismo utvrdili semantički slične parove od onih koji nisu semantički slični. Ova trening procedura zahteva pozitivne i negativne parove kao primere. Za upit q, potrebni su nam pozitivni primeri dokumenata d^+ za koje želimo da model predviđa veliku kosinusnu sličnost, i skup negativnih primera d^- za koje očekujemo da model predvidi manju bliskost u odnosu na d^+. Od modela se očekuje da se svaki element d^- iz skupa negativnih primera nalazi dalje u prostoru od pozitivnog primera d^+.

Poznata funkcija troška za ovu vrstu obučavanje je InfoNCE [36]

$$L_{cl} = -log \frac{e^{s(q,d^+)/t}}{e^{s(q,d^+)/t} + \sum_{i=1}^{n} e^{s(q,d_i^-)/t}}$$

gde je funkcija $s(q,d)$ procenjena mera sličnosti između upita i dokumenta preko njihovih vektorskih reprezentacija. Promenljiva t je hiper parametar, koji u ovom eksperimentu postavljamo na $t = 0.01$. Pokazalo se da što je više negativnih primera u ovoj funkciji troška, ona radi bolje. Zbog toga običaj je da se recikliraju ostali dokumenti iz serije (engl. *batch*) kao negativni kandidati [37]. U ovom radu ćemo koristiti funkciju troška korišćenu u gte naučnom radu [34], koja je poboljšana verzija InfoNCE funkcije troška, bi-direkciona i povećava skup negativnih primera, reciklirajući i ostale upite iz serije.

Razmotrimo da za uneti tekst skup pozitivnih parova

$$B = \{(q_1, d_1), (q_2, d_2), \dots, (q_n, d_n), \}$$

koristimo poboljšanu kontrastivnu funkciju troška

$$L_{icl} = -\frac{1}{n}\sum_{i=1}^{n} log\frac{e^{s(q_i,d_i)/t}}{Z} \quad (5.1)$$

$$Z = \sum_j e^{s(q_i,d_j)/t} + \sum_{j\neq i} e^{s(q_i,q_j)/t} + \sum_j e^{s(q_j,d_i)/t} + \sum_{j\neq i} e^{s(d_i,d_j)/t} \quad (5.2)$$

U ovom radu kao funkciju sličnosti koristimo kosinusnu sličnost:

$$sim(q, \mathrm{d}) = \frac{\vec{\mathrm{q}}*\vec{d}}{\|\vec{q}\|*\|\vec{d}\|} \quad (5.3)$$

5.2. Podaci

Za treniranje modela uz izabranu funkciju troška očekuju se podaci u obliku pozitivnih parova teksta, dok za negativne parove, kao što je spomenuto u prethodnoj sekciji, koristimo ostale upite i dokumenta iz serije.

Korišćeno je više različitih skupova podataka, svi su navedeni u Tabeli 5.1. Skup svih podataka sastoji se iz približno 130 miliona parova.

Ime skupa podataka	*Veličina (broj parova)*
SQUAD	0.1M
NAQ-train	0.1M
PAQ-train	64.3M
S2ORC	41.7M
Amazon QA	4M
Eli5	0.3M
Wikihow	0.1M
Quora Duplicates	0.1M
Gooaq	3M
MSMARCO-train	17.5M
Ukupno	131.2M

Tabela 5.1. Spsak skupova podataka korišćenih za treniranje

Raznovrsnost ulaznih podataka je ključna stvar u projektovanju embedera za pretragu, modeli trenirani na većoj količini podataka daju bolje rezultate. Zbog toga, iako neki skupovi podata predstavljaju mali procenat ukupnog broja parova, značajno utiču na povećanje kvaliteta embedera. Najmoderniji embederi koristili su skupove podataka veličine sa >1 Milijarde parova (bge model[8]) i 800 Miliona parova (gte model [34]). Projekat se ograničava na navedene skupove podataka zbog ograničenih dostupnih komputacionih resursa.

NAQ [45]**, SQAD** [47]**, PAQ** [46]**, Eli5** [49]**, Gooaq** [50]**, Amazon QA** [51]**, MSMARCO** [48] su skupovi pozitivnih parova **(pitanje, pasus sa odgovorom)**.

pitanje	paragraf sa odgovorom
How often is Notre Dame's the Juggler published?	As at most other universities, Notre Dame's students run a number of news media outlets. The nine student-run outlets include three newspapers, both a radio and television station, and several magazines and journals. Begun as a one-page journal in September 1876, the Scholastic magazine is issued twice monthly and claims to be the oldest continuous collegiate publication in the United States. The other magazine, The Juggler, is released twice a year and focuses on student literature and artwork. The Dome yearbook is published annually. The newspapers have varying publication interests, with The Observer published daily and mainly reporting university and other news, and staffed by students from both Notre Dame and Saint Mary's College. Unlike Scholastic and The Dome, The Observer is an independent publication and does not have a faculty advisor or any editorial oversight from the University. In 1987, when some students believed that The Observer began to show a conservative bias, a liberal newspaper, Common Sense was published. Likewise, in 2003, when other students believed that the paper showed a liberal bias, the conservative paper Irish Rover went into production. Neither paper is published as often as The Observer; however, all three are distributed to all students. Finally, in Spring 2008 an undergraduate journal for political science research, Beyond Politics, made its debut.
who produces the most wool in the world	Wool Global wool production is about 2 million tonnes per year, of which 60% goes into apparel. Wool comprises ca 3% of the global textile market, but its value is higher owing to dying and other modifications of the material.[1] Australia is a leading producer of wool which is mostly from Merino sheep but has been eclipsed by China in terms of total weight. [30] New Zealand (2016) is the third-largest producer of wool, and the largest producer of crossbred wool. Breeds such as Lincoln, Romney, Drysdale, and Elliotdale produce coarser fibers, and wool from these sheep is usually used for making carpets.
what part of the country are you likely to find the majority of the mollisols	Mollisol Mollisols occur in savannahs and mountain valleys (such as Central Asia, or the North American Great Plains). These environments have historically been strongly influenced by fire and abundant pedoturbation from organisms such as ants and earthworms. It was estimated that in 2003, only 14 to 26 percent of grassland ecosystems still remained in a relatively natural state (that is, they were not used for agriculture due to the fertility of the A horizon). Globally, they represent ~7% of ice-free land area. As the world's most agriculturally productive soil order, the Mollisols represent one of the more economically important soil orders.

Tabela 5.2. Primeri parova (pitanje, pasus sa odgovorom) iz više različitih skupova podataka

S2ORC[52] je skup podataka koji se sastoji od pozitivnih parova (naslov naučnog rada, abstrakt naučnog rada).

naslov naučnog rada	abstrakt naučnog rada
Some unsolved problems in the science of nylon and polyester fibres	Despite the commercial importance of melt-spun fibres there are many aspects of their science that are not well understood. A common working model of fine structure is a pseudo-fibrillar assembly of crystallites linked by tie molecules in amorphous regions, but other possibilities are suggested . Heat-setting is an accepted industrial practice, but the science is poorly under stood: there is a lack of good experimental data and no adequate theory. Even the nature of the basic mechanisms is not certain. There has been little attempt to analyze the mechanics of the material. Composite models a re inadequate, but network models show more promise. There are a number of differences between nylon and polyester fibres, which are unexplained. It will be necessary to solve these problems to maximize the industrial potential of nylon and polyester fibres .
Traffic-Aware Energy-Efficient Adaptive Cell Sectorization for Future Wireless Networks	Mobile data traffic has been increasing over the past few years due to the rise of both smart phones and tablets. To meet these traffic demands, the wireless infrastructure is expanding resulting in higher energy consumption and consequently higher operational costs. In this paper, an algorithm is proposed to minimize the energy consumption in future wireless networks. The algorithm utilizes adaptive cell sectorization, which is very efficient in cases of non-uniform distribution of the users within the cell. Specifically, the angular angle of a sector is changing dynamically according to the number of users in its coverage area, such that the sector will be switched off if it has no users to save energy. The simulation results show that the algorithm proposed is capable of saving the power up to 76% per sector.

Tabela 5.3. Primeri iz S2ORC skupa podataka

[8] https://github.com/FlagOpen/FlagEmbedding

Wikihow[44] je skup parova sličnih rečenica (rečenica 1, rečenica 2).

rečenica 1	rečenica 2
Find the Nearest Casino	Finding the nearest casino is challenging, especially if casino gambling is not legal in your area.
Be Healthy	Many people think that being healthy is a difficult task, involving months of dieting accompanied by hours at the gym.
Sell Fine Art Online	So you're a new or aspiring artist and your creativity has spawned something unique and interesting.

Tabela 5.4. Primeri iz Wikihow skupa podataka

Quora[53] je skup podataka sličnih pitanja, koja su na kvori (engl. *quora.com*) obeležena kao duplikati (pitanje 1, pitanje 2).

question1	question2
How do you start a bakery?	How can one start a bakery business?
Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?

Tabela 5.5. Primeri duplikata iz Quora skupa podataka

5.3. Arhitektura modela

Okosnica našeg embeding modela je transformer enkoder arhitektura, Majkrosoftov MiniLM-L12 [38]. Naš model se sastoji od 30 miliona parametara, i prati vanila bi-enkoder arhitekturu opisanu u Sekciji 4, sa istim enkoderom za upit i dokumenta. Za puling sloj se koristi "*mean pooling*" na izlazne kontekstualne reprezentacije tokena. Izlazni modeli vektora imaju 384 dimenzije, i normalizovani su na dužinu 1.

5.4. Evaluacija

Evaluaciju i kvalitet ovakvih modela je teško uraditi ručno, jer očekujemo da model dobro radi na raznim distribucijama tekstova, kao i specijalizovanim temama. Zbog toga je evaluacija nad velikim skupovima podataka ključna za merenje kvaliteta pretraživača. To nam omogućava **BEIR** benčmark (*engl.* ***Be****nchmarking* ***I****nformation* ***R****etrieval*) [39]. Benčmark se sastoji od nekoliko skupova podataka, namenjen je testiranju tradicionalnih i semantičkih pretraživača. Svaki od skupova podataka se sastoji od korpusa dokumenata (nekoliko miliona dokumenata) i liste upita. Za svaki upit, dat je podskup dokumenata koji se smatra kao relevantan datom upitu, a kao metriku kvaliteta koristimo nDCG@10 (*normalized discounted cumulative gain* nad 10 najboljih rezultata), o njoj smo pričali u Sekciji 2.5.

Kako se evaluacija obavlja nad velikim skupovima sa milionima dokumenata, ne možemo priuštiti da je obavljamo često. Evaluaciju smo radili 4 puta, nakon 5,7,9 i 14

hiljada trening koraka. Isto tako, zbog velike komputacije potrebne za evaluaciju, korišćen je podskup BEIR benčmarka:

Skup podataka	**Upit**	**Opis**	**Korpus**
NQ [45]	Pitanje	Naći pasus koji sadrži odgovor na pitanje.	~2.7 miliona pasusa
ArguAna [56]	Medicinski članak	Za dati upit, Pronaći slične članke	~10 hiljada medicinskih članaka
FiQa2018 [55]	Finansijska pitanja	Naći pasus koji sadrži odgovor na pitanje. Pitanja i pasusi su iz oblasti finansija.	~58 hiljada pasusa
CQADupStack [57]	Pitanje	Naći pasus koji sadrži odgovor na zadato pitanje. Pitanja su iz 10 različitih oblasti (statistika, engleski, programiranje, svakodnevna...)	~100 hiljada pasusa
Quora [53]	Pitanje	Za dato pitanje, nađi ostale duplikate iz korpusa	~530 hiljada pitanja
MS MARCO [48]	Pitanje	Naći pasus koji sadrži odgovor na pitanje.	~8.8 miliona pasusa
hotpotQA [54]	Pitanje	Naći pasus koji sadrži odgovor na pitanje.	~5.2 miliona pasusa

Tabela 5.6. Lista i opis evaluacionih skupova

5.5. Trening

Za treniranje modela je korišćen server sa četiri Nvidia Tesla V100 32GB grafičke kartice i 128GB RAM memorije (Slika 5.1.). Za implementaciju modela korišćeni su programski jezik *python* i biblioteke *pytorch* i *huggingface*. Za vizualizaciju, praćenje treninga i čuvanje modela korišćena je biblioteka *weights and biases*.

Model je treniran sa 14 hiljada trening koraka, malo više od jedne epohe i trajao je oko 40 sati. Za optimizaciju je korišćen AdamW [40] optimizator, sa stepenom učenja $lr = 3 * 10^{-4}$ i periodom zagrevanja od 10%. Za regularizaciju je korišćena L2 regularizacija sa $wd = 0.0001$. Temperatura t, hiperparametar iz funkcije troška, postavljena je na $t = 0.01$.

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.223.02   Driver Version: 470.223.02   CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  Off  | 00000000:1A:00.0 Off |                    0 |
| N/A   39C    P0    63W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  Off  | 00000000:1C:00.0 Off |                    0 |
| N/A   35C    P0    56W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  Off  | 00000000:1D:00.0 Off |                    0 |
| N/A   34C    P0    55W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  Off  | 00000000:1E:00.0 Off |                    0 |
| N/A   40C    P0    57W / 300W |      0MiB / 32510MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

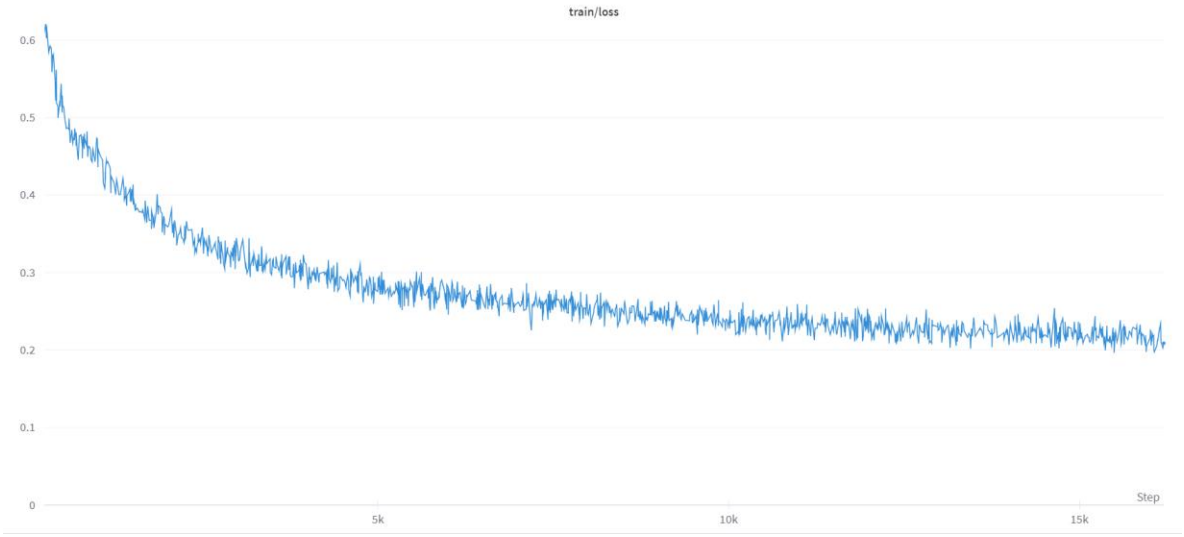
Slika 5.1. Pregled grafičkih kartica korišćenih za treniranje modela

Kao što smo već spomenuli tokom objašnjavanja funkcije troška, velika veličina serije (engl. *batch size*) je ključna u boljoj generalizaciji modela i dostizanju bolje aproksimacije cilja obučavanja. Zbog ograničenih resursa video memorije, korišćeno je nekoliko tehnika kako bismo minimizovali memorijski otisak i maksimizovali veličinu serije (engl. *batch size*):

- **Ograničavanje ulaznih sekvenci dokumenata sa 512 na 128 tokena, upita sa 512 na 64 tokena** – nakon istraživanja trening i evaluacionih skupova podataka, zaključeno je da >90% dokumenata ima dužinu <128 tokena i >95% upita dužinu <64 tokena. Ova metoda nam omogućava da smanjimo memorijski otisak za >4 puta, bez velikog gubitka performansi.

- **Kastovanje modela, aktivacija i gradijenata na fp16** – smanjivanjem preciznosti takođe gubimo nad performansama, ali nam omogućava duplo manji memorijski otisak.
- **Prebacivanje optimizatora na RAM** – deepspeed[9] bibilioteka nam omogućava ZeRO [41] optimizaciju kojom gradijente i pomerajuće proseke neophodne za AdamW optimizator čuvamo u RAM memoriji i računamo koristeći CPU.
- **Kontrolne Tačke gradijenata** (Engl. Gradient Checkpointing [42]) – Nakon primena svih gorenavedenih metoda i korišćenja malog modela (model je veličine oko 80MB), većinu zauzete video memorije čine aktivacije koje se računaju za propagaciju unazad. Ova tehnika nam omogućava da ne čuvamo sve aktivacije, već samo aktivacije na specificiranim kontrolnim tačkama, a ostatak aktivacija ponovo računamo tokom propagacije unazad. Ovom metodom usporavamo brzinu treniranja, ali memorijski otisak se smanjuje nekoliko puta.

Primena svih ovih tehnika nam je omogućila da povećamo veličinu serije sa 1000 na 16384. Ova veličina serije (engl. *bath size*) srazmerna je veličinama serija iz najpoznatijih naučnih radova (gte model je koristio 16384).



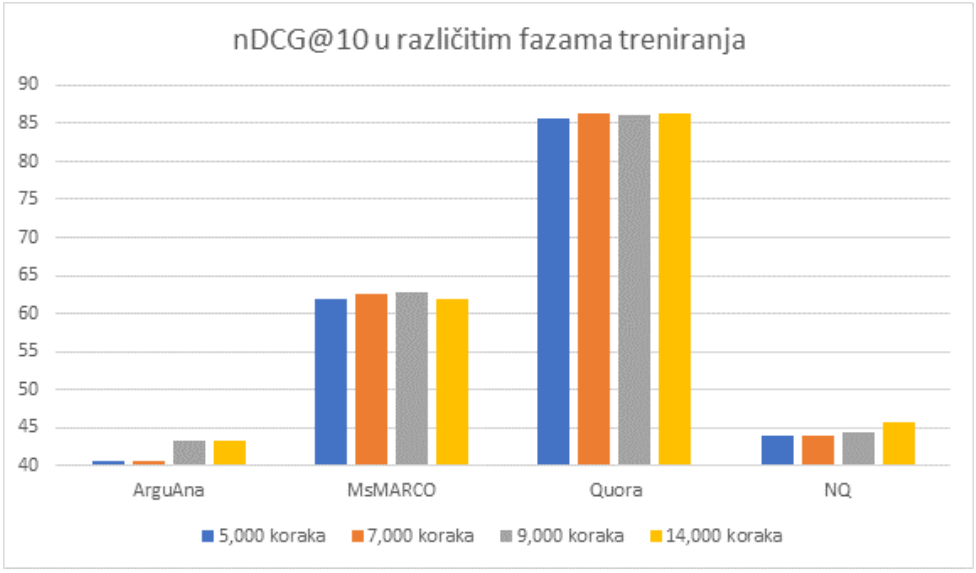
Slika 5.2. Grafikon funkcije troška tokom obučavanja

[9] https://github.com/microsoft/DeepSpeed

5.6. Rezultati

Slika 5.3. Prikaz evaluacije nad NQ (engl. Natural Questions) skupom podataka tokom obučavanja

Kao što što je već spomenuto u Sekciji 5.4 evaluacija je bila skupa i rađena je samo 4 puta. Na Slici 5.3. možemo videti poboljšanje nDCG metrike za evaluacioni skup prirodnih pitanja (NQ). Nad nekim skupovima podataka, model je pokazao podjednako dobre rezultate na 5 i 14 hiljada trening koraka (poput MSMARCO i Quora), dok je za neke skupove (poput NQ) povećavanje broja trening koraka konstatno doprinosilo boljoj metrici (slika 5.4.).



Slika 5.4. Prikaz nDCG@10 metrike nad evaluacijonim skupovima tokom obučavanja

Na kraju treniranja, posle 14 hiljada trening koraka urađena je opširnija evaluacija nad više skupova podataka (Tabela 5.2.). Na njoj možemo uporediti rezultate našeg modela u odnosu na tradicionalnu metodu (BM25), kao i dva poznata modela iste veličine (~30 miliona parmetara). Naš model se na 6/7 skupova podataka pokazao značajno bolji u

odnosu na tradicionalne metode, dok je na 4/7 skupova bio bolji i od postojećih javnih modela. Smatramo da je na ovakve rezultate najviše uticaja imao trening skup nad kojim je model obučavan. Kako je trening skup MSMARCO i NQ skupa podataka bio veliki deo našeg trening skupa, to je omogućilo modelu dobru reprezentaciju sličnih tekstova koji se javljaju u test skupu podataka.

Isto tako, HotpotQA ima mnogo veću raznovrsnost tema i koristi kompleksnije reči koje se ređe javljaju na internetu (a samim tim i u našem trening skupu), što zahteva veći skup podataka za obučavanje kako bismo dostigli pristojne rezultate nad ovim skupom. Takva distribucija termina ne predstavlja problem BM25 algoritmu rangiranja i tradicionalnim metodama, što mu daje prednost nad ovim skupom podataka.

dataset	**BM25**	$E5_{small}$	GTE_{small}	**Naš model**
MS MARCO	22.8	25.4	31.3	61.8
NQ	32.9	37.3	32.0	45.6
ArguAna	31.5	42.5	41.6	43.2
FiQa2018	23.6	38.3	37.0	32.7
CQADupStack	29.9	35.0	38.1	34.0
Quora	78.9	85.8	86.1	86.3
hotpotQA	60.3	46.0	49.3	28.6

Tabela 5.7. nDCG@10 rezultati nad različitim BEIR skupovima podataka, rezultati BM25, E5 i GTE su pozajmljeni iz GTE naučnog rada [34]

6. ZAKLJUČAK

U ovom radu, upoznali smo i pokazali važnost problema pretrage podataka. Prošli smo kroz tradicionalne metode i ukazali na njihove prednosti i mane. Posle toga smo stekli neophodno znanje iz neuralnih mreža potrebno za razumevanje glavnog fokusa rada, tekstualnih embedera i semantičkih pretraživača.

Na praktičan način smo pokazali superiornost semantičkih metoda pretraživanja u odnosu na tradicionalne metode. Treba napomenuti, da su skupovi podataka kreirani za evaluaciju kreirani koristeći tradicionalne pretraživače (ekstrakcija guglovih rezultata). Samim tim, labelirani dokumenti daju prednost tradicionalnim metodama. Pokazali smo da treniranjem na ograničenim resursima možemo da postignemo rezultate svetske klase.

Kako su semantički pretraživači nova tema istraživanja, postoji još mnogo metoda koje nismo stigli da istražimo, a imaju potencijala da poboljšaju naš sistem. Dalji rad bi uključivao treniranje na hardveru sa više video memorije (više grafičkih kartica), treniranje na većem i raznovrsnijem skupu podataka, treniranje većih modela, treniranje bez metoda uštede video memorije koje su uticale na performansu modela, korišćenje Maskiranih auto enkodera [43].

LITERATURA

[1] Bhaskar Mitra and Nick Craswell (2018), "An Introduction to Neural Information Retrieval ", Foundations and Trends® in Information Retrieval: Vol. 13: No. 1, pp 1-126. http://dx.doi.org/10.1561/1500000061

[2] Deep Learning for Search Book, Tommaso Teofili (ISBN: 9781617294792)

[3] Robertson, Stephen, and Hugo Zaragoza. "The probabilistic relevance framework: BM25 and beyond." Foundations and Trends® in Information Retrieval 3.4 (2009): 333-389.

[4] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, (1998)

[5] Li, Hang, and Jun Xu. "Semantic matching in search." Foundations and Trends® in Information Retrieval 7.5 (2014): 343-469.

[6] Singhal, Amit, Chris Buckley, and Manclar Mitra. "Pivoted document length normalization." Acm sigir forum. Vol. 51. No. 2. New York, NY, USA: ACM, 2017.

[7] Matveeva, I., C. Burges, T. Burkard, A. Laucius, and L. Wong. 2006. “High accuracy retrieval with multiple nested ranker”. In: Proc. SIGIR. ACM. 437–444

[8] Granka, L. A., T. Joachims, and G. Gay. 2004. “Eye-tracking analysis of user behavior in WWW search”. In: Proc. SIGIR. ACM. 478–479

[9] Joachims, T., L. Granka, B. Pan, H. Hembrooke, and G. Gay. 2005. “Accurately interpreting clickthrough data as implicit feedback”. In: Proc. SIGIR. Acm. 154–161

[10] Craswell, N. 2009. “Mean reciprocal rank”. In: Encyclopedia of Database Systems. Springer. 1703–1703

[11] Zhu, M. 2004. “Recall, precision and average precision”. Department of Statistics and Actuarial Science, University of Waterloo, Waterloo. 2: 30.

[12] Robertson, S. E., E. Kanoulas, and E. Yilmaz. 2010. “Extending average precision to graded relevance judgments”. In: Proc. SIGIR. ACM. 603–610.

[13] Järvelin, K. and J. Kekäläinen. 2002. “Cumulated gain-based evaluation of IR techniques”. ACM Transactions on Information Systems (TOIS). 20(4): 422–446.

[14] M. Nikolić, P. Janičić, *Veštačka inteligencija*, (2020)

[15] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature 521 (7553) (2015) 436–444

[16] Firth, J. R. 1957. “A synopsis of linguistic theory, 1930-1955”.

[17] Harris, Z. S. 1954. “Distributional structure”. Word. 10(2-3): 146–162.

[18] Mikolov, T., K. Chen, G. Corrado, and J. Dean. 2013a. “Efficient estimation of word representations in vector space”. arXiv preprint arXiv:1301.3781

[19] Goldberg, Y. and O. Levy. 2014. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. arXiv preprint arXiv:1402.3722.

[20] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[21] Medsker, Larry R., and L. C. Jain. "Recurrent neural networks." Design and Applications 5.64-67 (2001): 2.

[22] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[23] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[25] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[26] Defense Technical Information Center (1977-08-01). DTIC ADA049288: Speech Understanding Systems. Summary of Results of the Five-Year Research Effort at Carnegie-Mellon University.

[27] Yates, Andrew, Rodrigo Nogueira, and Jimmy Lin. "Pretrained transformers for text ranking: BERT and beyond." Proceedings of the 14th ACM International Conference on web search and data mining. 2021.

[28] Malkov, Yu A., and Dmitry A. Yashunin. "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs." IEEE transactions on pattern analysis and machine intelligence 42.4 (2018): 824-836.

[29] Buhler, Jeremy. "Efficient large-scale sequence comparison by locality-sensitive hashing." Bioinformatics 17.5 (2001): 419-428.

[30] Aumüller, Martin, Erik Bernhardsson, and Alexander Faithfull. "Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms." International conference on similarity search and applications. Cham: Springer International Publishing, 2017.

[31] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston. Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. arXiv:1905.01969v1, 2019.

[32] Reimers, Nils, and Iryna Gurevych. "Sentence-bert: Sentence embeddings using siamese bert-networks." arXiv preprint arXiv:1908.10084 (2019).

[33] Wang, Kexin, Nils Reimers, and Iryna Gurevych. "Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning." arXiv preprint arXiv:2104.06979 (2021).

[34] Li, Zehan, et al. "Towards general text embeddings with multi-stage contrastive learning." arXiv preprint arXiv:2308.03281 (2023).

[35] Wang, Liang, et al. "Text embeddings by weakly-supervised contrastive pre-training." arXiv preprint arXiv:2212.03533 (2022).

[36] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. CoRR, abs/1807.03748.

[37] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for opendomain question answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 6769–6781, Online. Association for Computational Linguistics

[38] Wang, Wenhui, et al. "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers." Advances in Neural Information Processing Systems 33 (2020): 5776-5788.

[39] Thakur, Nandan, et al. "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models." arXiv preprint arXiv:2104.08663 (2021).

[40] Loshchilov, Ilya, and Frank Hutter. "Fixing weight decay regularization in adam." (2018).

[41] Rajbhandari, Samyam, et al. "Zero: Memory optimizations toward training trillion parameter models." SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE, 2020.

[42] https://github.com/cybertronai/gradient-checkpointing

[43] Liu, Zheng, and Yingxia Shao. "Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder." arXiv preprint arXiv:2205.12035 (2022).

[44] Koupaee, Mahnaz, and William Yang Wang. "Wikihow: A large scale text summarization dataset." arXiv preprint arXiv:1810.09305 (2018).

[45] Kwiatkowski, Tom, et al. "Natural questions: a benchmark for question answering research." Transactions of the Association for Computational Linguistics 7 (2019): 453-466.

[46] Lewis, Patrick, et al. "Paq: 65 million probably-asked questions and what you can do with them." Transactions of the Association for Computational Linguistics 9 (2021): 1098-1115.

[47] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." arXiv preprint arXiv:1606.05250 (2016).

[48] Nguyen, Tri, et al. "MS MARCO: A human generated machine reading comprehension dataset." choice 2640 (2016): 660.

[49] Fan, Angela, et al. "ELI5: Long form question answering." arXiv preprint arXiv:1907.09190 (2019).

[50] Khashabi, Daniel, et al. "GooAQ: Open question answering with diverse answer types." arXiv preprint arXiv:2104.08727 (2021).

[51] Gupta, Mansi, et al. "Amazonqa: A review-based question answering task." arXiv preprint arXiv:1908.04364 (2019).

[52] Lo, Kyle, et al. "S2ORC: The semantic scholar open research corpus." arXiv preprint arXiv:1911.02782 (2019).

[53] https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

[54] Yang, Zhilin, et al. "HotpotQA: A dataset for diverse, explainable multi-hop question answering." arXiv preprint arXiv:1809.09600 (2018).

[55] https://sites.google.com/view/fiqa/

[56] http://argumentation.bplaced.net/arguana/data

[57] Hoogeveen, Doris, Karin M. Verspoor, and Timothy Baldwin. "CQADupStack: A benchmark data set for community question-answering research." Proceedings of the 20th Australasian document computing symposium. 2015.

[58] Schütze, Hinrich, Christopher D. Manning, and Prabhakar Raghavan. Introduction to information retrieval. Vol. 39. Cambridge: Cambridge University Press, 2008.

[59] https://jmotif.github.io/sax-vsm_site/morea/algorithm/TFIDF.html

[60] https://www.sbert.net/examples/applications/semantic-search/README.html

[61] Martins, Bruno, and Mário J. Silva. "Spelling correction for search engine queries." Advances in Natural Language Processing: 4th International Conference, EsTAL 2004, Alicante, Spain, October 20-22, 2004. Proceedings 4. Springer Berlin Heidelberg, 2004.

SPISAK SLIKA

SPISAK TABELA