SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 1095

Jezični model temeljen na neuronskoj mreži

Florijan Stamenković

Zagreb, lipanj 2015.

SADRŽAJ

1. Uvod

Tema ovog diplomskog rada je jezični model baziran na umjetnoj neuronskoj mreži. Jezični model je računalni sustav korišten u području obrade prirodnog jezika (statističke lingvistike) koji nalazi primjenu primjerice u automatiziranom prevođenju, prepoznavanju govora, i mnogim drugim zadatcima obrade teksta. Umjetne neuronske mreže koriste se kao jedan od pristupa izradi prediktivnih modela koji je u posljednjih desetak godina sve popularniji, zahvaljujući novim tehnikama i odličnim rezultatima. Kako bi se jezični model temeljen na neuronskoj mreži razmotrio u širem kontekstu, u ovom radu je implementirano nekoliko pristupa izgradnji jezičnih modela.

Naglasak je stavljen na objašnjenje kako prediktivne modele koristiti za izgradnju jezičnih modela. Objašnjavaju se prednosti i nedostatci isprobanih pristupa, kao i izazovi implementacije. Modeli su definirani matematički egzaktno, ali se ne ulazi u detalje računalne implementacije pošto su za to dostupni brojni kvalitetni materijali. Pretpostavlja se da čitatelj posjeduje osnovno znanje iz područja strojnog učenja, neuronskih mreža i linearne algebre. Ovaj rad na temelju tih znanja čitatelju može razjasniti osnovne pojmove o izradi i primjeni jezičnih modela, kao i specifičnosti, prednosti i nedostatke razmatranih modela. Bitno je naglasiti da su u ovom radu obrađeni tek neki od brojnih pristupa izradi jezičnih modela, rad nipošto nije iscrpan.

Rad je organiziran na sljedeći način. U drugom poglavlju ("Jezični model") se govori o jezičnim modelima općenito. Razmatra se njihova svrha, moguće primjene, te se iskazuje najčešće korištena matematička definicija jezičnog modela. Sljedeća četiri poglavlja razlažu pristupe izgradnji jezičnog modela koji su razmotreni u ovom radu. U svakom od tih poglavlja je matematički definiran po jedan model. Komentirani su pristupi implementaciji, ali algoritmi treniranja i korištenja nisu ispisani. Za svaki od modela se razmatraju prednosti i nedostatci. Potom je u sedmom poglavlju ("Vrednovanje") opisan način usporedbe jezičnih modela općenito, kao i konkretno vrednovanje provedeno u ovom radu. Razmotreni su rezultati vrednovanja kako bi čitatelj stekao što bolji dojam o primjenjivosti korištenih modela. U konačnici slijede zaključak i reference.

2. Jezični model

Jezični model jedan je od ključnih elemenata računalnog procesiranja prirodnog jezika. Glavni zadatak tom računalnom sustavu je ocjenjivanje pripadnosti teksta u prirodni jezik za koji je sustav rađen. Dodatno, poželjno je da sustav između više ponuđenih može izabrati tekst koji je najviše "u duhu" jezika. Primjerice, model hrvatskog jezika trebao bi biti sposoban ocijeniti da je niz *"dan je lijep"* generalno više u duhu jezika od *"lijepo dana biti"*, a da niz *"it's a nice day"* u dotični jezik ne pripada.

Smislenost niza riječi može se promatrati iz pravopisne, gramatičke i semantičke perspektive. Model treba biti sposoban nizu *"lijep dan"* dati bolju ocjenu nego nizu *"ljep dan"*, jer je drugi niz pravopisno neispravan. Model treba bolje ocijeniti niz *"lijep dan"* od niza *"lijepo dani"*, po kriteriju gramatičke ispravnosti. Konačno, treba bolje ocijeniti niz *"lijep dan"* od niza *"gumeni parket vječno"*, jer je drugi niz semantički skoro sasvim besmislen.

2.1. Primjena

Jezični modeli imaju velik broj primjena. Sustavi za strojno prevođenje koriste ih kako bi odabrali što smisleniji među više potencijalnih prijevoda. Sustavi za prepoznavanje govora koriste ih kako bi prepoznate slogove i riječi pretvorili u konačni tekst. Sustavi za pretraživanje teksta mogu ih koristiti za mjerenje sličnosti između termina pretrage i dokumenta. Ovo su samo neke od mnogih primjena, povećanjem dostupne količine informacija i mogućnosti njihove računalne obrade, moguće primjene i kvaliteta jezičnih modela imaju trend povećanja.

2.2. Probabilistička definicija

Pošto prirodni jezik omogućava praktično neiscrpne mogućnosti kombiniranja riječi u tekst, smisleno je jezični model razmatrati statistički. Na temelju nekog korpusa teksta moguće je napraviti sustav koji za proizvoljni niz označava koliko je sličan tekstu

iz korpusa. Ovo je u skladu s definicijom i iskazanim svojstvima jezičnog modela. Kako bi se računalni statistički sustav izgradio, potrebno je smislenost teksta definirati probabilistički.

Razmatra se niz riječi $w_1, w_2, ..., w_m$. Niz sadrži m riječi, pri čemu w_i označava riječ koja se pojavljuje na i-tom mjestu u nizu. Primjerice, za niz *"Nebo je plavo"*, pojedine riječi se označavaju na sljedeći način: $w_1 = "Nebo"$, $w_2 = "je"$, $w_3 = "plavo"$. U kontekstu statističkog modeliranja jezika potrebno je definirati vjerojatnost pojavljivanja proizvoljnog niza riječi. Oznaka za vjerojatnost niza riječi slijedi.

$$P(\text{niz}) = P(w_1, w_2, ..., w_m)$$

Dakle, razmatra se vjerojatnost da se pojavio niz od m riječi, pri čemu je prva riječ niza w_1, sljedeća w_2, i tako sve do konačne riječi niza w_m. U skladu s teorijom vjerojatnost, izraz za vjerojatnost niza moguće je faktorizirati (izraziti kao umnožak jednostavnije definiranih vjerojatnosti) na sljedeći način.

$$\begin{aligned} P(w_1, w_2, ..., w_m) &= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_m|w_1, w_2, ..., w_{m-1}) \\ &= \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}) \end{aligned}$$

Pri tome je $P(w_i|w_1, ..., w_{i-1})$ uvjetna vjerojatnost pojavljivanja riječi w_i na i-toj poziciji u nizu, ako su joj prethodile riječi w_1, ... w_{i-1} na pozicijama 1, ... , $i - 1$. Primjetimo kako je ovakva faktorizacija u skladu s čovjekovim čitanjem teksta (slijedno po riječima).

Slijedi primjer izraza vjerojatnosti i faktorizacije, za već spomenuti niz *"Nebo je plavo"*.

$$\begin{aligned} P(\textit{"Nebo je plavo"}) =& P(w_1 = \textit{"Nebo"}, w_2 = \textit{"je"}, w_3 = \textit{"plavo"}) \\ =& P(w_1 = \textit{"Nebo"})P(w_2 = \textit{"je"}|w_1 = \textit{"Nebo"}) \\ & P(w_3 = \textit{"plavo"}|w_1 = \textit{"Nebo"}, w_2 = \textit{"je"}) \end{aligned}$$

Česće se u literaturi može vidjeti kraći zapis iste faktorizacije.

$$\begin{aligned} P(\textit{"Nebo je plavo"}) =& P(\textit{"Nebo"}, \textit{"je"}, \textit{"plavo"}) \\ =& P(\textit{"Nebo"})P(\textit{"je"}|\textit{"Nebo"})P(\textit{"plavo"}|\textit{"Nebo"}, \textit{"je"}) \end{aligned}$$

Iako egzaktna, iskazana faktorizacija vjerojatnosti niza praktična za dugačke nizove jer je vjerojatnost riječi na poziciji i uvjetovana svim prethodnim riječima. Stoga se u jezičnim modelima najčešće koristi kontekst ograničene duljine od n riječi. Primjerice, za kontektst duljine $n = 3$, svaka riječ se razmatra samo s obzirom na prethodne dvije.

$$\begin{aligned} P(w_1, w_2, ..., w_m) &\approx P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_m|w_{m-2}, w_{m-1}) \\ &\approx \prod_{i=1}^{m} P(w_i|w_{i-2}, w_{i-1}) \end{aligned}$$

Općenito, za proizvoljnu duljinu konteksta *n* faktorizacija je sljedeća.

$$\begin{aligned} P(w_1, w_2, ..., w_m) &\approx P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)...P(w_m|w_{m-(n-1)}, ..., w_{m-1}) \\ &\approx \prod_{i=1}^{m} P(w_i|w_{i-(n-1)}, ..., w_{i-1}) \end{aligned}$$

Tako definiran jezični model više nije probabilistični egzaktan, ali se može praktično implementirati kao računalni sustav.

U kontekstu obrade prirodnog jezika, standardna oznaka za niz od *n* riječi je "*n*-gram". Za $n = 1, 2, 3$ govorimo o "unigramu", "bigramu", odnosno "trigramu", dok za $n = 4$ o "4-gramu" itd. U tom smislu niz *"Nebo je plavo"* sadrži unigrame {*"Nebo"*, *"je"*, *"plavo"*}, bigrame {*"Nebo je"*, *"je plavo"*} i trigram {*"Nebo je plavo"*}.

Kada se koristi ograničeni kontekst, javlja se pitanje optimalne duljine konteksta, odnosno vrijednost parametra n. Pošto na konačno značenje niza mogu utjecati riječi koje su od promatrane riječi proizvoljno daleko, može se pretpostaviti da je veći kontekst bolji od manjeg. Iako ovo u idealnom slučaju vrijedi, u praksi se koriste relativno male duljine konteksta. Razlog za to je rijetkost pojavljivanja dugih nizova. Vjerojatnost pojavljivanja nekog konkretnog niza riječi duljine n drastično pada kako n raste. Istovremeno, jezični model se bazira na nekom konkretnom korpusu koji ima konačnu količinu teksta. Čak i ako je korpus vrlo velik (što je poželjno), vjerojatnost pojavljivanja nekon proizvoljnog, smislenog niza duljine primjerice 100 je vrlo mala. U drugu ruku, ako je korpus dovoljno velik, smislene kombinacije od tri ili četiri riječi pojavit će se često. Stoga je moguće napraviti sustav koji na temelju statistike pojavljivanja trigrama ili 4-grama vrednuje smislenost neviđenog teksta, ali je praktički nemoguće napraviti sustav koji bi to činio na temelju pojavljivanja 100-grama. U praksi se stoga često koriste konteksti duljine oko četiri riječi, ovisno o primjeni. Neke implementacije

modela baziranih na velikom korpusu teksta mogu povećati kontekst do desetak riječi, rijetko više od toga.

Alternativni pristup ograničavanju veličine konteksta je kompresija odnosno aproksimacija konteksta veće (ponekad pune) duljine. Konkretno, ako kontekst prikažemo kao funkciju prethodnih riječi, moguća je sljedeća faktorizacija.

$$\begin{aligned} P(w_1, w_2, ..., w_m) &\approx P(w_1)P(w_2|f(w_1))P(w_3|f(w_1, w_2))...P(w_m|f(w_1, w_2, ..., w_{n-1})) \\ &\approx \prod_{i=1}^{n} P(w_i|f(w_1, ..., w_{i-1})) \end{aligned}$$

Pri tome funkcija $f(...)$ preslikava kontekst proizvoljne duljine u zapis fiksne duljine. Ovakva formulacija koristi se primjerice u izvedbi jezičnih modela korištenjem rekurzivnih neuronskih mreža. Pošto se u ovom radu ne koriste implementacije bazirane na toj formulaciji, onoe se neće u nastavku spominjati.

Postoje varijante jezičnih modela koje kao kontekst ne promatraju riječi koje doslovno prethode promatranoj, već riječi iz šire okoline. Jasno je da su takvi modeli primjenjvi samo u situacijama kada je cijeli niz poznat unaprijed, ali takve situacije su dovoljno česte da bi dotični modeli bili primjenjivi. Neki modeli čak razmatraju "koncepte", uzorke sačinjene od nekoliko specifičnih riječi koje se mogu pojaviti bilo gdje u tekstu. Ovakvi modeli se neće razmatrati u nastavku.

Konačno, bitno je spomenuti jezične modele bazirane na manjim lingvističkim jedinicama. Jezični modeli bazirani na morfemima i znakovima su od nedavno, pogotovo sa sve većom primjenom dubokih neuronskih mreža, postali uspooredivi i čak u nekim primjenama bolji od leksičkih modela [14].

2.3. Načini izvedbe

Postoje brojne implementacije jezičnih modela. Većina njih izvedena je iz probabilističke formulacije modela, ili tu formulaciju aproksimiraju. Tehnike korištene za izradu modela variraju. U ovom diplomskom radu biti će uspoređene izvedbe prebrojavanjem *n*-grama, jezični model temeljen na jednostavnoj neuronskoj mreži i log-bilinearni probabilistički model. Njihova definicija i opis slijede.

3. Prebrojavanje *n*-grama

Najjednostavniji pristup implementaciji jezičnog modela bazira se na prebrojavanju odnosno frekvenciji pojavljivanja *n*-grama. Na primjeru trigrama, vjerojatnost pojavljivanja riječi w_i na i-tom mjestu u tekstu je sljedeća.

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})}$$

Pri tome je $C(w_a, ..., w_b)$ funkcija prebrojavanja koja za niz $w_a, ..., w_b$ daje broj njegovih pojavljivanja u korpusu teksta nad kojim se model gradi. Uvjetna vjerojatnost pojavljivanja riječi, s obzirom na prethodne dvije, je dakle broj pojavljivanja relevantnog trigrama, podjeljeno s brojem pojavljivanja bigrama koji joj prethode.

Intuicija je sljedeća. Ako se u korpusu za učenje jezičnog modela trigram *"dan je lijep"* pojavljuje 10 puta, a bigram *"dan je"* 42 puta, tada je vjerojatnost da se nakon konteksta *"dan je"* pojavi riječ *"lijep"* jednaka $P(\text{"}lijep\text{"}|\text{"}dan\text{"}, \text{"}je\text{"}) = 10/42$.

Ovako definiran procijenitelj vjerojatnosti zapravo je procijenitelj najveće izglednosti (ML-procijenitelj, engl. *maximum likelihood*). Općenito razmatranje statističkih procjenitelja je izvan opsega ovog diplomskog rada, više informacija na temu može se naći u materijalima na temu statistike i strojnog učenja.

Definirana je uvjetna vjerojatnost za kontekst duljine $n = 3$. Formulaciju je potrebno poopćiti na proizvoljnu vrijednost parametra n. Uvjetna vjerojatnost riječi s obzirom na kontekst proizvoljne duljine n je sljedeća.

$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{C(w_{i-(n-1)}, ..., w_{i-1}, w_i)}{C(w_{i-(n-1)}, ..., w_{i-1})} \tag{3.1}$$

3.1. Zaglađivanje

Već je spomenuto kako je otežavajuć faktor izgradnje jezičnih modela rijetkost pojavljivanja duljih *n*-grama u konačnom korpusu teksta. Ovaj problem zapravo nije ograničen samo na modele koji koriste veliki kontekst. Čak i za male vrijednost n,

primjerice 3, lako se može desiti da neki smisleni trigram ne bude prisutan u korpusu na kojem se model bazira. Ovo nije začuđujuće, s obzirom da je broj mogućih trigrama $|V|^3$, gdje je V skup svih riječi jezika (vokabular) za koji se model gradi. Primjerice, za engleski se jezik procjenjuje da sadrži oko $300,000$ riječi[1], što znači da je broj mogućih trigrama oko $27 \cdot 10^{15}$. Jasno je da se mnogi smisleni trigrami među njima nikada ne pojavljuju u korpusu za učenje, bez obzira na njegovu veličinu. U slučaju da se pri korištenju modela evaluira vjerojatnost jednog od njih, po definiciji 3.1 ona će biti 0. Numerički i intuitivno vjerojatnost 0 govori kako je nešto nemoguće, što se svakako želi izbjeći, pogotovo ako se radi smislenom *n*-gramu.

U jezičnim modelima koji se temelje na prebrojavanju *n*-grama se problem rijetkosti rješava zaglađivanjem (engl. *smoothing*). Ideja je da se u račun vjerojatnosti unese pretpostavka kako je korpus na kojem se model bazira samo uzorak jezika. Pretpostavivši da postoje mnogi *n*-grami koji nisu prisutni u korpusu, ali zavređuju određenu vjerojatnost, njen račun se modificira kako bi se vjerojatnosna masa šire rasporedila. Time se disproporcije u vjerojatnostima *n*-grama smanjuju, odnosno zaglađuju.

Postoji više pristupa zaglađivanju, neki od popularnijih su aditivno (Laplace), Good-Turing i Knesser-Ney zaglađivanje. Aditivno zaglađivanje u pravilu daje lošije rezultate, koristi se pretežno u edukativne svrhe. Zaglađi vanje Kneser-Ney metodom je manje intuitivno, ali daje odlične rezultate. U ovom radu implementirano je aditivno zaglađivanje i zaglađivanje Kneser-Ney metodom.

3.1.1. Aditivno zaglađivanje

Ideja aditivnog zaglađivanja [6] je da se svakom mogućem *n*-gramu pridjeli neka mala vjerojatnost, makar se taj *n*-gram nije pojavio u korpusu za treniranje. U izrazu vjerojatnosti mora se voditi računa da suma vjerojatnosti svih mogućih *n*-grama bude 1. Uzevši to u obzir dobiva se sljedeći izraz uvjetne vjerojatnosti riječi.

$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{C(w_{i-(n-1)}, ..., w_{i-1}, w_i) + \alpha}{C(w_{i-(n-1)}, ..., w_{i-1}) + \alpha d}$$

Pri tome je α proizvoljan broj, tipično 1 ili manje, a d broj svih mogućih *n*-grama. Koristeći aditivno zaglađivanje na jednostavan se način izbjegava da vjerojatnost bilo kojeg *n*-grama bude 0. Nedostatak ovog pristupa je to što svim *n*-gramima umjetno povećava broj pojavljivanja za isti broj α, bez obzira na njihovu relativnu smislenost.

[1] Oxford English Dictionary, http://www.oed.com

Modificiranjem vjerojatnosti kvalitetnijim pretpostavkama drugi oblici zaglađivanja u pravilu postižu bolje rezultate.

3.1.2. Zaglađivanje Kneser-Ney metodom

Prezentirano u radu [5], zaglađivanje Kneser-Ney metodom jedan je od načina zaglađivanja oduzimanjem fiksne vrijednost od vjerojatnosti *n*-grama (engl. *absolute value discounting*). Ideja je da se oduzimanjem od vjerojatnosti *n*-grama dobije "višak" vjerojatnosne mase, koja se potom raspodjeljuje svim *n*-gramima, uključujući one manje zapostavljene u korpusu za treniranje. Generiranje viška vjerojatnosne mase, na primjeru trigrama, se u Kneser-Ney metodi vrši na sljedeći način.

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{max\left(C(w_{i-2}, w_{i-1}, w_i) - \delta, 0\right)}{C(w_{i-2}, w_{i-1})} + ...$$

Pri tome je δ vrijednost koja se oduzima od prebrojavanja, u rasponu $\delta \in (0, 1)$. S obzirom da nije dozvoljeno da vjerojatnost postane negativna, potrebno je koristiti funkciju $max(x, 0)$, za slučaj kada je $C(w_{i-2}, w_{i-1}, w_i) < \delta$, odnosno kada je $C(...) = 0$.

Dobiveni višak vjerojatnosne mase Kneser-Ney metoda raspodjeljuje na temelju prebrojavanja *n*-grama nižeg reda (manjeg parametra *n*). Stoga, u slučaju kada se model bazira na trigramima, izraz se nastavlja dodavanjem vjerojatnosti bazirane na bigramima.

$$P(w_i|w_{i-2}, w_{i-1}) = \frac{max\left(C(w_{i-2}, w_{i-1}, w_i) - \delta, 0\right)}{C(w_{i-2}, w_{i-1})} + \lambda P(w_i|w_{i-1})$$

Pri tome je λ višak vjerojatnosti dobiven oduzimanjem fiksne vrijednosti δ. Moglo bi se učiniti kako je $\lambda = \delta$, pošto je upravo δ količina oduzete vjerojatnosti, ali to bi bilo pogrešno. Količina δ je oduzeta samo u slučajevima kada je $C(w_{i-2}, w_{i-1}, w_i) >= 1$. U suprotnom, kada je $C(...) = 0$, funkcija $max(x, 0)$ poništava to oduzimanje. Broj slučajeva kada se oduzimanje provelo je broj slučajeva kada je $C(w_{i-2}, w_{i-1}, w_i) >= 1$. Stoga je λ definiran na sljedeći način.

$$\lambda = |\{w : C(w_{i-2}, w_{i-1}, w) > 0\}| \frac{\delta}{C(w_{i-2}, w_{i-1})}$$

Prvi dio izraza (sve što prethodi razlomku) prestavlja broj slučajeva kada se oduzimanje provelo. Konkretno, definiran je skup riječi w za koje je broj pojavljivanja trigrama $C(w_{i-2}, w_{i-1}, w)$ veći od 0 (što povlači da se oduzimanje provodi), te je potom uzeta veličina tog skupa, odnosno broj provedenih oduzimanja. Drugi dio izraza

sačinjen je od razlomka. U brojniku je δ, što broj provedenih oduzimanja pretvara u konkretnu vrijednost svih oduzimanja. Nazivnik je isti kao u osnovnom izrazu prebrojavanja, kako bi vjerojatnost ostala normalizirana na sumu od 1.

Time je objašnjeno kako se u zaglađvanju Kneser-Ney metodom dio vjerojatnosne mase prenosi na *n*-grame nižeg reda, ali stvar tu ne završava. Kako bi se kvaliteta zaglađivanja dodatno poboljšala, procijena vjerojatnosti *n*-grama nižeg reda bazirana je na "vjerojatnosti nastavljanja" (engl. *continuation probability*). Pitanje koje vjerojatnost nastavljanja formulira je sljedeće: koliko je vjerojatno da se riječ w pojavila nakon prethodnih, bez obzira kojih? Odnosno, koliko je riječ w dobar "nastavljač"? Ta vjerojatnost smatra se proporcionalnom broju različitih prehodnih konteksta nakon kojih se pojavljuje.

Intuicija vjerojatnosti nastavljanja se često objašnjava na primjeru bigrama *"San Francisco"*. Pretpostavivši da se u korpusu teksta taj bigram pojavljuje često, vjerojatnost unigrama *"Francisco"* biti će proporcionalno velika, ali je njena vjerojatnost nastavljanja mala jer se javlja isključivo nakon riječi *"San"*. Ako je na temelju tog korpusa potrebno dovršiti rečenicu *"Cipele popravlja _."*, jednostavan model bi mogao na temelju vjerojatnosti unigrama preferirati riječ *"Francisco"*, jer se riječ *"postolar"* u korpusu pojavljuje rijeđe. Kneser-Ney metoda korištenjem vjerojatnosti nastavljanja ne preferira riječ *"Francisco"*, koja se javlja isključivo nakon riječi *"San"*, jer se riječ *"postolar"* javlja kao nastavak većeg broja riječi, iako je u korpusu rjeđa.

Matematički, vjerojatnost nastavljanja bigrama P_{cont} definirana je na sljedeći način.

$$P_{\text{cont}}(w_i|w_{i-1}) = \frac{|\{w_{i-1} : C(w_{i-1}, w_i) > 0\}|}{\sum_{w_j \in V} |\{w_{j-1} : C(w_{j-1}, w_j) > 0\}|}$$

U brojniku se nalazi broj različitih riječi nakon koji se javlja w_i. Na temelju bigrama *"San Francisco"*, za riječ *"Francisco"* će brojnik biti samo 1. Za riječ *"postolar"* će brojnik biti veći, jer se ta riječ javlja kao nastavak većeg broj riječi, primjerice u *"dobar postolar"* i *"otac mu je postolar"*. U nazivniku se nalazi normalizacijska suma koja je jednaka za obje riječi iz primjera.

Zaglađivanje Kneser-Ney metodom se koristi tako da se za *n*-grame najvišeg reda koriste obične vjerojatnosti bazirane na prebrojavanju, od kojih se oduzimaju fiksne vrijednosti. Dobivenom se dodaje vjerojatnost nastavljanja *n*-grama jednog reda niže, množeno faktorom λ. Pri tome se vjerojatnost nastavljanja isto tako zaglađuje Kneser-Ney metodom, čime je definirana rekurzivna jednadžba. Vjerojatnost najnižeg reda (unigrama), procjenjuje se njihovom frekvencijom. Rekurzivno definirano zaglađivanje Kneser-Ney metodom moguće je koristiti za proizvoljnu duljinu konteksta n.

3.2. Implementacija

Praktična izvedba prebrojavanja *n*-grama je konceptualno vrlo jednostavna. Tekst treba pretvoriti u *n*-grame i za svaki *n*-gram ustanoviti broj pojavljivanja u korpusu. Kompleksnije metode zaglađivanja, poput Kneser-Ney metode, koriste dodatna prebrojavanja koja su tek nešto kompliciranija za izvedbu. Nakon što se shvati što je točno potrebno prebrojati, implementacija algoritma nije zahtjevna. Jedini problem prebrojavanja *n*-grama je njihova brojnost.

Već je spomenuto kako je za engleski jezik broj mogućih trigrama okvirno $27 \cdot 10^{15}$. Za pohranu broja pojavljivanja svakog od njih korištenjem samo dva okteta računalne memorije bilo bi potrebno oko 50 tisuća *terabyte*-a memorije. Očigledno je ovo s današnjim računalima neizvedivo. Problem je potrebno rješiti drukčije. Mora se uzeti u obzir da je većina tih trigrama iznimno malo vjerojatno, te se oni nikada ne javljaju u tekstu. Stoga je nepotrebno trošiti memoriju na njih. Zato se za pohranu broja pojavljivanja *n*-grama koriste rijetke (engl. *sparse*) podatkovne strukture. Pamte se brojevi pojavljivanja samo onih *n*-grama koji su se barem jednom pojavili. Ovaj pristup rješava problem memorijske zahtjevnosti, ali je izvedba kompliciranija. Rijetke strukture podataka mogu se implementirati na različite načine, o čemu im ovise memorijske i brzinske performanse. Kvalitetno upoznavanje rijetkih struktura podataka preporučeno je pri pokušaju implementacije jezičnih modela baziranih na prebrojavanju *n-grama*.

4. Neuronska mreža

Umjetne neuronske mreže jedan su od često korištenih pristupa izgradnji prediktivnih modela. U ovom radu se pod "neuronska mreža" podrazumijeva najjednostavniji oblik mreže: unaprijedna slojevita mreža. Jezične modele moguće je implementirati i korištenjem drukčijih arhitektura neuronskih mreža [7]. Osnove korištenja neuronskih mreža i njihova tipologija izlaze izvan okvira ovog rada, kao uvod u područje može se konzultirati [15].

Jezični model može se smatrati svojevrsnim klasifikatorom. Za viđeni niz prethodnih riječi, model treba predvidjeti sljedeću riječ. Ako je kontekst baziran na *n*-gramima, model na temelju $(n-1)$ prethodne riječi treba predvidjeti sljedeću. Ovakvu interpretaciju jezičnog modela jednostavno je preslikati u umjetnu neuronsku mrežu. Pošto u prirodnom jeziku više različitih riječi može smisleno nastaviti prethodne, prikladno je mrežu definirati kao "meki" klasifikator koji ne predviđa točno koja riječ slijedi, već za više riječi (moguće cijeli vokabular) računa relativnu smislenost.

Izlaz iz mreže je u tom smislu jasan: poželjno je dobiti relativnu smislenost da nakon prethodnog konteksta slijedi jedna od proizvoljnog skupa riječi. Ako se za taj skup koristi cijeli vokabular, može se reći da je izlaz iz mreže vjerojatnost svake riječi vokabulara, za zadani kotekst. Iako ima više načina na koji je ovo moguće izvesti, koncept je isti.

Kompliciranije je pitanje kako pojedine riječi predočiti mreži.

4.1. Ulaz u mrežu

Slojevite unaprijedne neuronske mreže kao ulaznu vrijednost primaju vektor brojeva. Ulaz u jezični model je ograničeni niz riječi. Potrebno je stoga niz riječi predočiti kao vektor brojeva.

4.1.1. Riječ kao redni broj

Riječ je jedan element iz skupa svih riječi, vokabulara. Ako se svakoj riječi dodijeli redni broj unutar vokabulara, dobiven je jedinstveni brojevni zapis za riječ. Potrebno je razmotriti da li je ovakva konverzija riječi u broj prikladna za korištenje s neuronskom mrežom. Izlaz mreže je zapravo matematička funkcija nad ulaznim varijablama. U tom smislu se izlaz mreže mjenja ovisno o ulazu te generalno vrijedi da slične vrijednosti ulaza daju slične vrijednosti izlaza, i obrnuto. Dakle, mreža bi generalno govoreći za ulaze $(15, 3, 56)$ i $(15, 3, 57)$ trebala dati sličan izlaz, jer su ti ulazi numerički slični. Istovremeno bi za ulaze $(15, 3, 56)$ i $(0, 88, 36)$ vjerojatno dala različit izlaz jer su numerički različiti. Lako je uočiti probleme korištenja ove formulacije u jezičnom modelu. Primjerice, nizovi *"sutra idem u"* i *"sutra idem kući"* mogu numerički biti vrlo slični, ako su redni brojevi riječi *"u"* i *"kući"* bliski, ali je značenje tih nizova sasvim različito. U drugu ruku, nizovi *"ogromno stablo"* i *"veliko drvo"* imaju vrlo slično značenje, ali mogu biti predočeni sasvim različitim rednim brojevima vokabulara.

Problem je dakle to što kompliciranе odnose značenja riječi nije moguće preslikati u jednostavan poredak cijelih brojeva. Potrebno je potražiti bolji način predočenja riječi neuronskoj mreži.

4.1.2. Binarni vektor

Elementi proizvoljnog skupa koji nemaju numerički poredak mogu se pretočiti binarnim vektorom. Ovaj koncept je primjenjiv na riječi u vokabularu. Svakoj riječi se dodijeli jedinstveni redni broj vokabulara. Potom se za tu riječ definira binarni vektor vektorom koji ima broj elemenata jednak veličini vokabulara. Svi elementi vektora imaju vrijednost 0, osim elementa na poziciji koja odgovara rednom broju riječi, koji ima vrijednost 1. Ovakav zapis jednog elementa skupa se često naziva *"one-hot encoding"*.

Binarni vektor je prikladniji oblik zapisa riječi za ulaz za neuronsku mrežu od rednog broja riječi. Mreža svaki od ulaza (elemenata vektora) vidi kao nezavisnu varijablu. Korištenjem binarnog vektora za zapis riječi, zapravo se mreži za svaku riječ odvojeno dovodi informacija da li se nalazi u kontekstu ili ne. Redni brojevi riječi više ne igraju ulogu.

Postoji nekoliko problema sa zapisom riječi u obliku binarnog vektora. Veličina ulaza u mrežu ovisi o veličini vokabulara i veličini konteksta. S obzirom na to da vokabular može sadržavati desetke tisuća riječi, ulaz u mrežu neminovno bi bio sukladno velik. To implicira velik broj parametara mreže, dugo treniranje i potrebu za ogromnim

skupom za učenje. Nadalje, u zapisu oblika binarnog vektora ne postoji veza između sličnih riječi, pošto je svaka riječ interpretirana kao nezavisan element vokabulara. U tom smislu se riječ *"automobil"* smatra jednako različitom od riječi *"motocikl"* i *"leptir"*, što za jezični model nije pogodno. Poželjno je da jezični model sadrži znanje o sličnosti riječi jer to povećava njegovu sposobnost generalizacije. Ako je model uspješno naučio u kojim kontekstima se javlja riječ *"automobil"*, poželjno je da u sličnim kontekstima smatra pojavljivanje riječi *"motocikl"* vjerojatnijim od pojavljivanja riječi *"leptir"*.

4.1.3. Distribuirane reprezentacije

Navedeni problem numeričkog zapisa riječi prisutan je i u drugim područjima obrade prirodnog jezika, primjerice u pretraživanju. Čest pristup rješavanju ovog problema je korištenje "distribuiranih reprezentacija" (engl. *distributed representations*) riječi [10]. U ovom pristupu se svaka riječ predočava vektorom realnih brojeva. Ideja je da svaki element vektora predstavlja neki apstraktni pojam, a vrijednost tog elementa za pojedinu riječ predstavlja koliko su pojam i riječ podudarni. Svaka riječ je u tom smislu definirana putem podudarnosti sa svakim od korištenih pojmova. Njeno značenje je "distribuirano" po pojmovima. Ovisno o primjeni vektor može imati nekoliko desetaka do nekoliko stotina elemenata.

Dvije su dobre posljedice korištenja ovog pristupa. Prvo, veličina reprezentacije riječi ne ovisi linearno o veličini vokabulara. Drugo, riječi sličnog značenja imaju slične distribuirane reprezentacije. Obje posljedice vrlo su pogodne za izgradnju jezičnog modela temeljenog na neuronskoj mreži. Problem je pronalazak prikladnih distribuiranih reprezentacija svih riječi.

Distribuirane reprezentacije moguće je izgraditi korištenjem Latentne Semantičke Analize (LSA) [1], koja se bazira na linearnoj dekompoziciji matrice pojavljivanja riječi unutar dokumenta. LSA se uspješno koristi za pretraživanje teksta, gdje je za termin pretrage potrebno pronaći relevantne dokumente. Problem pri korištenju LSA pristupa je memorijska zahtjevnost linearne dekompozicije s obzirom na količinu primjera koji se obrađuju. Kako se jezični modeli grade korištenjem što većeg korpusa, LSA nije prikladan alat. Reprezentacije je također moguće izgraditi korištenjem neuronskih mreža specijaliziranih za to [8]. S obzirom na iterativnu prirodu učenja neuronskih mreža, memorijska složenost lakše se kontrolira. Nadalje, mreže se mogu formulirati na mnogo načina, što omogućava veću kontrolu nad izlučenim reprezentacijama. Konačno, moguće je učenje reprezentacija ugraditi u neuronsku mrežu koja ima neku

drugu primjenu. Tada su dobivene reprezentacije u pravilu specijalizirane za dotičnu primjenu, te je njihovo korištenje u nekom drugom kontekstu vjerojatno neprikladno.

U ovom radu se za ulaz u mrežu koriste distribuirane reprezentacije, zbog navedenih prednosti tog pristupa. Reprezenacije se uče istovremeno s jezičnim modelom.

4.2. Definicija mreže

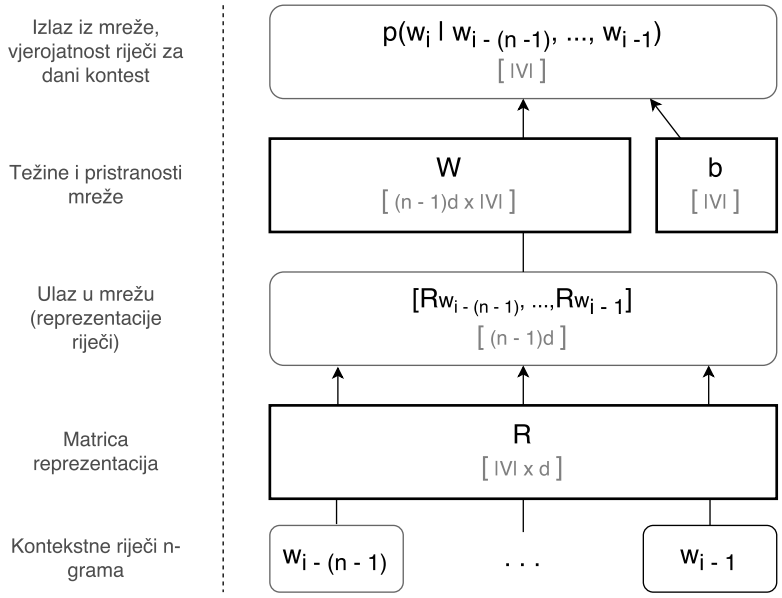
Jezični model temeljen na neuronskoj mreži je definiran na sljedeći način. Ulazni sloj mreže sastoji se od $(n-1)d$ neurona, pri čemu je d veličina distribuirane reprezentacije riječi. Vrijednosti koje se dovode na ulaz mreže su distribuirane reprezentacije prvih $(n-1)$ riječi *n*-grama. Mreža nema skrivenih slojeva. Izlazni sloj mreže ima po jedan neuron za svaku riječ vokabulara, dakle sveukupno $|V|$ neurona. Izlazni sloj definiran je kao *softmax* funkcija. Stoga su izlazne vrijednosti mreže distribucija vjerojatnosti posljednje riječi *n*-grama, s obzirom na prethodnih $(n-1)$ riječi. Slijedi matematična definicija opisane mreže.

$$\begin{aligned} P(w_i|w_{i-(n-1)},...,w_{i-1}) &= \text{softmax}_{w_i}\left(\left[R_{w_{i-(n-1)}},...,R_{w_{i-1}}\right]W+b\right) \\ &= \frac{\exp\left(\left[R_{w_{i-(n-1)}},...,R_{w_{i-1}}\right]W_{w_i}+b_{w_i}\right)}{\sum_{w\in V}\exp\left(\left[R_{w_{i-(n-1)}},...,R_{w_{i-1}}\right]W_w+b_w\right)} \end{aligned} \tag{4.1}$$

Pri tome R označava matricu koja distribuiranih reprezentacija riječi, koja je dimenzija $|V|\times d$, dakle svaki redak joj sadrži distribuiranu reprezentaciju jedne riječi. R_w označava distribuiranu reprezentaciju riječi w, a $\left[R_{w_j},...,R_{w_k}\right]$ označava vektor koji se dobije spajanjem distribuiranih reprezentacija riječi $w_j,...,w_k$ u jedan vektor-redak (ulaz u mrežu). W označava matricu težina (engl. *weight*) mreže između $(n-1)$ reprezentacije na ulazu i izlaznog sloja, dimenzija $(n-1)d\times|V|$. W_w označava vektor težina mreže između $(n-1)$ reprezentacije na ulazu i izlaznog neurona za riječ w. Vektor b su pristranosti (engl. *bias*) svih neurona izlaznog sloja, a b_w pristranost neurona izlaznog sloja koji odgovara riječi w.

Može se primjetiti kako se za sve ulazne riječi koristi ista matrica distribuiranih vektora. Time se smanjuje broj parametara mreže i pospješuje učenje reprezentacija (riječi sličnog značenja dobivaju slične reprezentacije).

Mreža je prikazana na slici 4.1.



Slika 4.1: Arhitektura neuronskog jezičnog modela. Pravokutnici s debljim rubovima su parametri mreže. Pravokutnici sa zaobljenim rubovima su podatci koji kroz mrežu prolaze.

4.3. Učenje

Algoritam propagacije greške unatrag bazira se na gradijentnom spustu s obzirom na funkciju greške mreže. S obzirom na broj primjera koji se koristi za izračun gradijenta u svakom koraku gradijentnog spusta, postoji par inačica učenja. Gradijent se može računati na temelju svih primjera za učenje, što se naziva grupno (engl. *batch*) učenje. U praksi se pokazalo kako je efikasnije podijeliti skup za učenje na mini-grupe (engl. *minibatch*) i koristiti njih za izračun gradijenta. Time se parametri mreže promjene više puta u svakoj epohi učenja (prolasku kroz sve primjere za učenje), što ubrzava učenje. Istovremeno, gradijent nije isti za svaku mini-grupu, što povećava otpornost na zaglavljivanje gradijentnog spusta u lokalnim minimumima. Ekstrem korištenja mini-grupa je podešavanje parametara mreže nakon evaluacije samo jednog primjera za učenje, što se naziva pojedinačno (engl. *online*) učenje. Ovaj ekstremni pristup rezultira nestabilnim gradijentom, zbog čega se mora smanjiti stopa učenja, i manjom efikasnošću korištenja računalnih resursa uslijed otežane paralelizacije. Stoga se najčešće koriste mini-grupe. Konkretna veličina mini-grupe ovisi domeni primjene, arhitekturi mreže i načinu učenja. Ne postoji univerzalno prikladna veličina.

Osim "običnog" gradijentnog spusta, koji mjenja parametre mreže za vrijednost negativnog gradijenta pomnoženog sa stopom učenja, postoje i modificirane inačice. One modificraju smjer ili veličinu gradijenta kako bi ubrzale učenje. Ta modifikacija

se tipično bazira na gradijentu prethodnih koraka učenja. Najjednostavniji primjer modifikacije gradijenta je korištenje "momenta", pri čemu se gradijent kroz više koraka akumulira u "moment" kretanja (analogno momentu kretanja kuglice određene mase koja se kotrlja u smjeru nagiba, ali i u smjeru trenutne brzine). U ovom radu korištena je inačica "RMSprop" (engl. *resilient mean square propagation*) [13]. Radi se o obliku gradijentnog spusta koji od izračunatog gradijenta uzima samo smjer, bez iznosa. Količinu promjene gradijenta određuje automatski, na temelju iznosa gradijenta prethodnih koraka.

Specifičnost jezičnog modela baziranog na neuronskoj mreži, u odnosu na standardne unaprijedne slojevite mreže, je korištenje matrice reprezentacija R. Odabir reprezentacije specifične riječi (retka matrice R) nije standardna derivabilna matematička operacija. Može se postaviti pitanje kako matricu R optimirati gradijentnim spustom. Rješenje je u tome da se matrica ne optimira u cjelosti, već se optimiraju reprezentacije riječi korištene u trenutnom koraku gradijentnog spusta, koje se tretiraju kao samostalni parametar, neovisan o matrici R.

4.4. Složenost mreže

Umjetne neuronske mreže treniraju se kroz mnogo iteracija (prolazak kroz skup podataka za treniranje), u rasponu od nekoliko desetaka do nekoliko tisuća. U svakoj od njih se u svakom koraku gradijentnog spusta (kojih može biti i do nekoliko tisuća) mjenjaju svi parametri mreže. Stoga je bitno obratiti pozornost na vremensku i memorijsku složenost treniranja mreže, koje su u pravilu usko vezane uz broj parametara mreže. Stoga je potrebno razmotriti broj parametara jezičnog modela temeljenog na neuronskoj mreži.

Matrica distribuiranih reprezentacija R sadrži $|V|d$ parametara, matrica težina W sadrži $(n-1)d|V|$ parametara, a vektor pristranosti $|V|$ parametara. Sveukupno, mreža sadrži $(nd+1)|V|$ parametara. Njena memorijska složenost stoga je $O(nd|V|)$. Vremenska složenost treniranja neuronske mreže ovisi o mnogim čimbenicima, ali se u ovom slučaju zbog jednostavnosti mreže može aproksimirati kao $O(nd|V|)$. Linearna složenost s obzirom na često vrlo veliki $|V|$ nije pogodna. Za tipične vrijednost $n = 4$ i $d = 100$, te vokabular veličine $|V| = 20,000$, mreža sadrži 8 milijuna. Za neuronsku mrežu je to vrlo velik broj parametara, treniranje mreže te veličine postalo je moguće tek u posljednjih desetak godina.

4.5. Implementacija

Za implementaciju neuronske mreže dostupni su mnogi alati komercijalni, akademski i besplatni alati. Mnogi su visoko optimizirani kako bi što učinkovitije koristili dostupne računalne resurse. Višeprocesorska i višejezgrena računala koriste se za paralelizaciju treniranja mreže, a sve dostupnija je i podrška za treniranje mreža koristeći visoki stupanj paralelizma grafičkih procesora. Nadalje, alati često podržavaju definiciju mreže na visokom stupnju apstrakcije (na razini matematičke definicije) i automatizirani izračun gradijenata, što olakšava implementaciju, smanjuje količinu grešaka u programskom kodu i povećava optimalnost korištenja računalne snage.

U ovom radu je za izradu jezičnog modela temeljenog na neuronskoj mreži korištena *open-source* knjižnica Theano[1], čije sučelje je dostupno kroz programski jezik Python[2]. Odabrana je zbog fleksibilnosti definiranja modela (lako se koristi s visoke razine apstrakcije, ali omogućava i rad na nižim razinama), automatiziranog izračuna gradijenata i mogućnosti izvođenja istog koda na centralnim i grafičkim procesorima.

[1] http://deeplearning.net/software/theano/

[2] https://www.python.org/

5. Log-bilinearni model

U radu [9] 3-new-models-Mnih predložene su tri implementacije jezičnih modela. Najuspješniji među njima je log-bilinearni model.

Log-linearni prediktivni modeli su klasa modela u kojima je vjerojatnost uzorka proporcionalna eksponenciranoj linearnoj funkciji tog uzorka.

$$P(a) \propto \exp(f_{lin}(a))$$

Jednostavan primjer takvog modela je već korištena *softmax* funkcija. Prikladni su za modeliranje vjerojatnosti jer eksponencijalna funkcija preslikava iz domene $\mathbb{R}$ u domenu $\mathbb{R}_{\geq 0}$, koja se lako pretvori u vjerojatnost. To se tipično radi na sljedeći način.

$$P(a) = \frac{\exp(f_{lin}(a))}{\sum_b \exp(f_{lin}(b))} = \frac{\exp(f_{lin}(a))}{Z}$$

Pri tome je uobičajena Z oznaka za "particijsku" funkciju, kojoj je svrha normalizirati vrijednost iz brojnika u vjerojatnosnu distribuciju, tako da je zbroj vjerojatnosti svih mogućih primjera jednaka 1. U skladu s tim je particijska funkcija upravo zbroj vrijednosti brojnika, po svim mogućim primjerima.

Logaritmiranje brojnika log-linearne funkcije rezultira linearnom funkcijom (što objašnjava ime "log-linearni modeli"), koja je pogodna za analizu i optimizaciju. Više informacija na temu log-linearnih modela može se naći u literaturi iz područja strojnog učenja.

Bilinearna funkcija je funkcija dvije varijable koja je s obzirom na njih obje linearna. U skladu s tim, log-bilinearni probabilistički model definira vjerojatnost proporcionalnu eksponenciranoj bilinearnoj funkciji.

$$P(a, b) \propto \exp(f_{lin}(a, b))$$

5.1. Log-bilinearni jezični model

Log-bilinearna funkcija može se primjeniti za definiciju uvjetne vjerojatnosti riječi s obzirom na prethodne. Razmatranja o načinu predstavljanja riječi iz poglavlja 4.1 o jezičnom modelu temeljenom na neuronskoj mreži vrijede i za log-bilinearni model. Stoga se i u ovom modelu koriste distribuirane reprezentacije. Za kontekst baziran na *n*-gramima, kakav je i dosad korišten u ovom radu, uvjetna vjerojatnost proporcionalna je log-bilinearnoj funkciji.

$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) \propto \exp\left(\left[R_{w_{i-(n-1)}}, ..., R_{w_{i-1}}\right] W R_{w_i}^T + b_{w_i}\right) \tag{5.1}$$

W označava matricu preslikavanja $(n-1)$ d-dimenzionalne reprezentacije na ulazu u jedan d-dimenzionalni vektor. Nakon što je time ulazni vektor reprezentacija reduciran s $(n-1)$ d na d dimenzija, računa se skalarni produkt s reprezentacijom uvjetovane riječi, te se dodaje njena pristranost. Navedene operacije su linearne, a rezultiraju skalarom. Nazivaju se bilinearnom funkcijom jer se prethodni kontekst i uvjetovana riječ razmatraju kao zasebne varijable. Rezultirajući skalar se eksponencira, a rezultat je proporcionalan vjerojatnosti, što definira log-bilinearni model. Kako bi proporcionalnost postala jednakost, potrebno je normalizirati ju particijskom funkcijom.

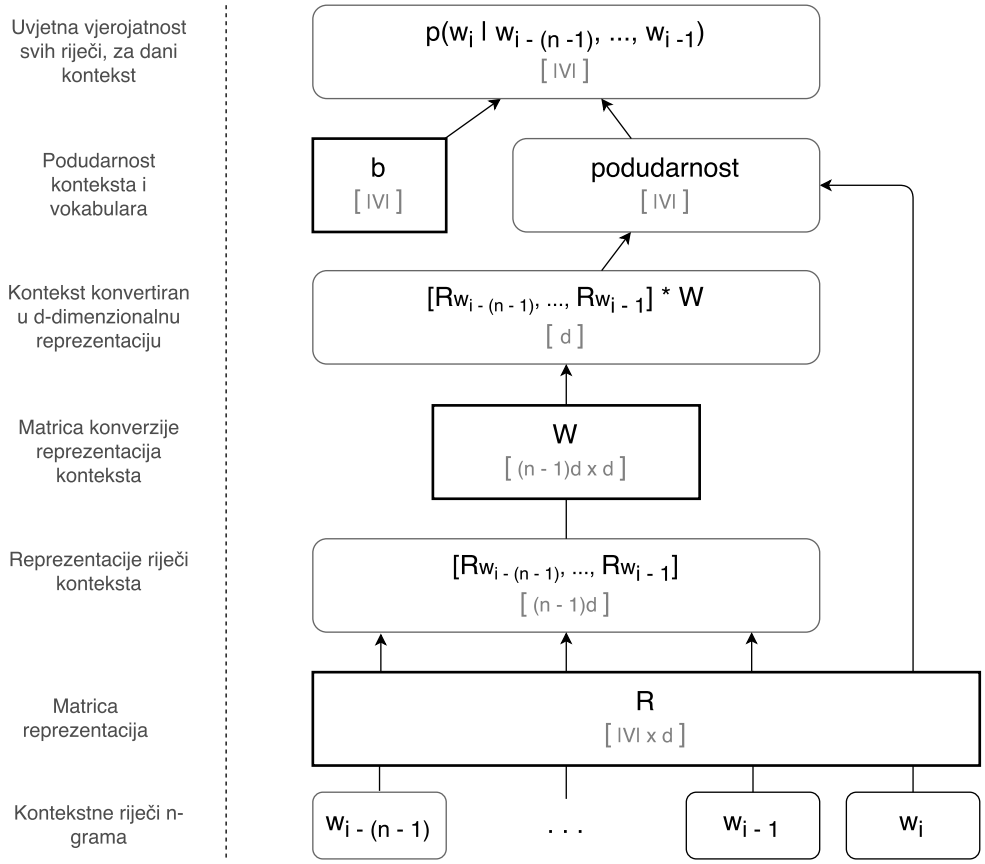
$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{\exp\left(\left[R_{w_{i-(n-1)}}, ..., R_{w_{i-1}}\right] W R_{w_i}^T + b_{w_i}\right)}{\sum_{w \in V} \exp\left(\left[R_{w_{i-(n-1)}}, ..., R_{w_{i-1}}\right] W R_w^T + b_w\right)} \tag{5.2}$$

Iako je ova definicija vrlo slična matematičkoj definiciji jezičnog modela temeljenog na neuronskoj mreži 4.1, bitno je primjetiti da se ovdje ne radi o mreži. Umjesto da se na temelju ulaznih riječi (konteksta) računa funkcija izlaza za $|V|$ izlaznih neurona, računa se mjera podudarnosti sa svakom riječi vokabulara. Ne postoje neuroni koji bi imali svoj ulaz, aktivacijsku funkciju i izlaz. Ovdje se radi o kompoziciji $(n-1)$ riječi u d-dimenzionalni vektor te potom računanju podudarnosti te kompozicije s vektorom koji predstavlja uvjetovanu riječ.

Log-bilinearni model je prikazan na slici 5.1.

5.2. Učenje

Log-linearni modeli se, kao i unaprijedne neuronske mreže, u pravilu treniraju iterativno, gradijentnim smanjenjem funkcije greške. Isti princip učenja primjenjiv je



Slika 5.1: Arhitektura log-bilinearnog jezičnog modela. Pravokutnici s debljim rubovima su parametri modela. Pravokutnici sa zaobljenim rubovima su podatci koji prolaze kroz model.

i na log-bilinearni model. Sve navedeno u poglavlju 4.3 vrijedi i za treniranje log-bilinearnog modela, a dostupni su i brojni materijali na temu gradijetnog spusta i algoritma propagacije greške unatrag.

Specifičnost jezičnog log-bilinearnog modela je istovremeno učenje parametara linearne funkcije W i b, te matrice reprezentacija vokabulara R, kao i u jezičnom modelu temeljenom na neuronskoj mreži. U log-bilinearnom modelu dodatno je zanimljivo što se matrica R koristi na dva mjesta u funkciji. Prvo za dobivanje reprezentacija konteksta, a potom za podudarnost reduciranog konteksta sa svim riječima vokabulara. Smisleno je zapitati se bi li korištenje dviju matrica reprezentacija (jednu za prethodni kontekst, a drugu za uvjetovanu riječ) bilo prikladnije. Pogotovo jer postoje modeli koji idu tako daleko da za svaku riječ definiraju zasebnu matricu kompozicije s drugim riječima [11]. Istovremeno, korištenje dviju matrica reprezentacija osjetno povećava parametarski prostor, što može otežati treniranje i rezultirati lošijim modelom. Zbog vremenskih ograničenja pokušaj učenja modela korištenjem dviju reprezentacijskih

matrica nije isproban. U radu koji definira log-bilinearni model [9] je korištena samo jedna, a pošto su dobiveni rezultati vrlo konkurentni drugim modelima, može se zaključiti da korištenje jedinstvene matrice R funkcionira dobro.

5.3. Složenost modela

Log-bilinearni model, kao i neuronska mreža, trenira se gradijentnim smanjenjem greške. Već je objašnjeno kako broj parametara modela snažno utječe na vremensku i memorijsku složenost treniranja i korištenja mreže, iste opservacije vrijede i za log-bilinearni model. U tom smislu je zanimljivo usporediti broj parametara log-bilinearnog modela i neuronske mreže.

Matrica distribuiranih reprezentacija R sadrži $|V|d$ parametara, matrica preslikavanja W sadrži $(n-1)d^2$ parametara, a vektor pristranosti d parametara. Sveukupno, log-bilinearni model sadrži $((n-1)d + |V| + 1)\, d$ parametara. Složenost ovisi o dominantnom faktoru. Pošto je u pravilu $|V|$ puno veći broj od $(n-1)d$, konačna složenost je $O(d|V|)$, što je n puta manje od memorijske složenosti neuronske mreže. Isto tako, ta složenost ne ovisi o parametru n, što znači da se veličina konteksta log-bilinearnog modela može povećavati bez da se znatno poveća memorijska složenost modela.

Vremensku složenost treniranja i korištenja modela isto tako je teško procjeniti, kao i kod neuronske mreže. Trajanje treniranja modela svakako ovisi o broju parametara, pa je aproksimacija vremenske složenosti kao $O(d|V|)$ opravdana, i opet n puta manja od vremenske složenosti treniranja neuronske mreže.

5.4. Implementacija

Log-bilinearni model malo je manje uobičajen od unaprijednih neuronskih mreža. Nisu poznati alati koji bi omogućili definiranje takvog modela na visokoj razini apstrakcije dostupnoj za neuronske mreže, ali su alati za numeričku optimizaciju malo niže razine apstrakcije svejednako od velike koristi. Konkretno, postoji više komercijalnih i besplatnih alata koji omogućavaju jednostavnu simboličku definiciju log-bilinearnog modela, te automatizirani izračun gradijenata i optimizirani proces treniranja.

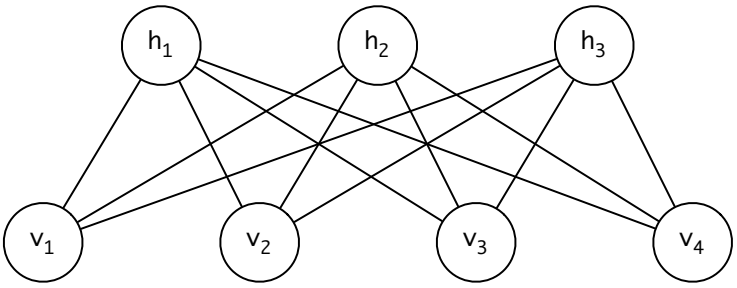
Kao i za model jezika temeljen na neuronskog mreži, za implementaciju log-bilinearnog modela korištena je knjižnica Theano. Knjižnica je korištena za automatsko izračunavanje gradijenta te optimalno treniranje modela na centralnim i grafičkim procesorima.

Kao i kod implementacije neuronske mreže, većina koda ima svrhu procesiranja teksta, ekstrakcije vokabulara i pretvaranje teksta u *n*-grame rednih brojeva riječi vokabulara. Definicija modela je u Theano knjižnici relativno jednostavna, kao i treniranje. Koristi se RMSprop inačica gradijentnog spusta, koja naspram "običnog" jako ubrzava proces učenja.

6. Ograničeni Boltzmannov stroj

U radu [9] razmatra se i jezični model temeljen na ograničenom Boltzmannovom stroju (u nastavku "RBM", od engl. *Restricted Boltzmann Machine*). RBM je generativni probabilistički model [2] zanimljive formulacije, te ga stoga vrijedi spomenuti, iako je u spomenutom radu manje uspješan od log-bilinearnog modela.

RBM je dvoslojna neuronska mreža. Prvi (vidljivi) sloj sadrži neurone koji mogu biti proizvoljnog tipa, dok drugi sloj sadrži skrivene neurone koji mogu biti isključivo u stanjima 0 ili 1 (binarna aktivacija). Svaki vidljivi neuron je spojen sa svakim skrivenim, ali ne postoje veze među neuronima istog sloja. Stoga se RBM može predočiti kao potpuno spojeni bipartitni graf, što je prikazano na slici 6.1.



Slika 6.1: Vizualni prikaz ograničenog Boltzmannovog stroja. Svaki neuron vidljivog sloja je spojen sa svim skrivenim neuronima. Neuroni istog sloja nisu izravno povezani.

Moguće je izračunati vjerojatnost svakog stanja RBMa, pri čemu "stanje" znači da svi vidljivi i skriveni neuroni imaju neke konkretne vrijednosti. Vjerojatnost stanja definirana je kao eksponencirana negativna energija, kao u Boltzmannovoj distribuciji, po kojoj je model nazvan.

$$P(s) = \frac{\exp(-E(s))}{\sum_{s'} \exp(-E(s'))} \tag{6.1}$$

Pri tome je vektor s spomenuto stanje svih neurona mreže, odnosno unija vidljivih neurona v i skrivenih neurona h. Energija mreže je linearna funkcija stanja neurona. U tom smislu RBM podsjeća na log-linearni model, odnosno neuronsku mrežu sa *softmax*

slojem. Jedna od razlika je to što su u RBMu skriveni neuroni binarni. Energija RBMa je definirana na sljedeći način.

$$E(s) = E(v, h) = -vWh^T - vb_v - hb_h$$

Pri tome je W matrica težinskih veza među neuronima vidljivog i skrivenog sloja, b_v su pristranosti vidljivog sloja, a b_h skrivenog.

Sličnost s već definiranim log-bilinearnim i *softmax* modelom je očita. Što RBM čini specifičnim je to da su neuroni mreže probabilistički, a ne deterministički. Vjerojatnost binarne aktivacije skrivenog neurona v_i (preuzimanja stanja 0 ili 1) je definirana kao funkcija energije, u skladu s 6.1.

$$P(h_i = 1) = \frac{1}{1 + \exp\left(-vW_{h_i} - b_{h_i}\right)} = \sigma(vW_{h_i} + b_{h_i})$$

Pri tome W_{h_i} označava stupac matrice koji sadrži težine veza vidljivog sloja i skrivenog neurona h_i, a b_{h_i} pristranost istog neurona. $\sigma(...)$ je uobičajena oznaka za sigmoidalnu odnosno logističku funkciju. Može se primjetiti kako vjerojatnost aktivacije skrivenog neurona ovisi samo o stanjima vidljivih, a ne o ostalim skrivenim neuronima. Kao što je spomenuto, neuroni su u RBMu organizirani u bipartitan graf, dakle veze među neuronima istog sloja ne postoje.

U slučaju da su vidljivi neuroni isto binarni (što se preferira), izraz za vjerojatnost njihove aktivacije je identičan, s obzirom na stanja skrivenih neurona. U slučaju da su vidljivi neuroni drugog tipa, aktivacija isto ima probabilistički izraz, ali drukčijeg oblika.

Probabilistička aktivacija neurona specifično je svojstvo RBMova i modela temeljenih na RBMova koja ima snažno regularizacijsko djelovanje. Stoga su RBMovi često korišteni za pred-treniranje slojeva dubokih neuronskih mreža [4]. Detaljnije teoretsko i praktično razmatranje RBMa dostupno je u [12].

6.1. Jezični model temeljen na RBMu

RBM je generativni probabilistički model, što znači da modelira vjerojatnosnu distribuciju pojavljivanja primjera. Za izradu jezičnog modela RBM ovakav model se može primjeniti za modeliranje distribucije pojavljivanja *n*-grama. Takav model može dati procjenu relativnih vjerojatnosti proizvoljnih *n*-grama. Ako se pri tome prvih $(n-1)$ riječi fiksira, a posljednja varira po vokabularu, može se dobiti uvjetna vjerojatnost posljednje riječi s obzirom na prethodne.

Prvi korak u definiranju vjerojatnosne distribucije je definiranje energije modela, za ulazni primjer.

$$E(w_{i-(n-1)}, ..., w_i, h) = -\left[R_{w_{i-(n-1)}}, ..., R_{w_i}\right](Wh + b_v) - hb_h$$

Može se primjetiti kako je energija deifnirana na temelju distribuiranih reprezentacija svih riječi *n*-grama, uz poznato stanje skrivenog sloja h. Do uvjetne vjerojatnosti posljednje riječi, bez poznatog stanja skriveno sloja, doći će se u nastavku. Na temelju energije može se definirati uvjetna vjerojatnost posljednje riječi *n*-grama i stanja skrivenog sloja.

$$P(w_i, h|w_{i-(n-1)}, ..., w_{i-1}) = \frac{\exp(-E(w_{i-(n-1)}, ..., w_i, h))}{\sum_{h'}\sum_{w \in V}\exp(-E(w_{i-(n-1)}, ..., w_{i-1}, w, h'))}$$

Vidljivo je kako se normalizacija vjerojatnosti vrši po svim mogućim stanjima skrivenoh sloja, i svim riječima vokabulara. Marginalizacijom izražene po svim mogućim stanjima skrivenog sloja može se izraziti uvjetna vjerojatnost posljednje riječi *n*-grama.

$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{\sum_h \exp(-E(w_{i-(n-1)}, ..., w_i, h))}{\sum_{h'}\sum_{w \in V}\exp(-E(w_{i-(n-1)}, ..., w_{i-1}, w, h'))}$$

Dobivena vjerojatnost iterira po svim mogućim stanjima skrivenog sloja. S obzirom da svaki neuron skrivenog sloja može biti u 2 stanja, broj mogućih stanja skrivenog sloja je $2^{|h|}$, dakle eksponencijalan je s obzirom na broj neurona skrivenog sloja. Ova izrazito nepogodna složenost može se zgodnim trikom pretvoriti u linearnu složenost s obzirom na broj neurona skrivenog sloja, kao što je opisano u [3].

6.2. Učenje

Algoritam propagacije greške unatrag teško je izravno primjeniti na RBM zbog binarnih stohastičkih neurona. Stoga se u praksi koristi algoritam "kontrastna divergencija", koji gradijentni spust aproksimira uzorkovanjem [3]. Definicija algoritma i praktične implikacije njegovog korištenja izlaze izvan okvira ovog rada, detalji se mogu pronaći u mnogim dostupnim materijalima na temu, primjerice [12].

6.3. Implementacija

Grafički modeli sa stohastičkim varijablama kompleksniji su za treniranje od determinističkih modela. Razloga je više. Prvo, egzaktni izračun vjerojatnosti tipično podrazu-

mijeva marginalizaciju po sivm mogućim stanjima skrivenog sloja. Iako se ovo može izbjeći, kao što je već spomenuto, izračun može biti numerički nestabilan. Treniranje RBMa se oslanja na generatore nasumičnih brojeva, što usporava učenje. Stohastički binarni neuroni dobro regulariziraju model, ali se treniraju sporije. Pošto se ne koristi optimizacija parametara gradijentnim spustom, parametre je potrebno podešavati "ručno" na temelju pseudo-gradijenata izračunatih kontrastnom divergencijom. Ovo otežava korištenje efikasnih numeričkih alata i čini implementaciju kompleksnijom.

Unutar ovog rada jezični model baziran na RBMu implementran je korištenjem Theano knjižnice. Model je treniran algoritmom kontrastne divergencije. Implementacija je suboptimalna zbog kompleksnosti algoritma, uz to što je po definiciji osjetno sporija od drugih isprobanih metoda. Provedeno je treniranje i vrednovanje ograničene uspješnosti, nije moguće jamčiti ispravnost ponuđene implementacije. S obzirom na kompleksnost ovog modela i poznato slabije rezultate, čini se da je praktično od najmanje koristi među modelima razmotrenim u ovom radu.

7. Vrednovanje

7.1. Metodologija

Svrha jezičnog modela je ocjena smislenosti neviđenog teksta, s obzirom na korupus teksta nad kojim je model treniran. Za ocjenjivanje uspješnosti modela vrijede isti koncepti koji se primjenjuju u strojnom učenju općenito. Ukratko, model je potrebno vrednovati korištenjem podataka koji su došli iz iste distribucije kao i podatci s kojima je model treniran, ali koji nisu sadržani u skupu za treniranje. U skladu s tim, tipičan pristup vrednovanju je da se cjelokupni skup podataka dostupan za implementaciju modela podijeli na skup za treniranje i skup za testiranje. Veličina ispitnog skupa ovisi o modelu, primjeni i količini dostupnih podataka. Pošto se jezični modeli tipično treniraju nad vrlo velikim korpusima, podataka u pravilu ima dovoljno. Stoga se za ispitni skup odvaja okvirno 5-10 posto svih podataka, dovoljno da bi ispitni skup imao sličnu distribuciju kao cijeli korpus, ali bez da se model zakine za previše podataka za učenje.

7.2. Mjera uspješnosti jezičnog modela

Za vrednovanje uspješnosti prediktivnih modela koriste se standardizirane mjere. Uobičajeno je za klasifikaciju koristiti mjeru točnosti predikcije nad ispitnim skupom. Točnost je udio primjera ispitnog skupa kojima je model dodjielio prikladnu oznaku klase. U situacijama kada klase nisu podjednako zastupljene u skupu za treniranje, prikladnija je F-mjera uspješnosti [**?**].

Navedene mjere nisu prikladne za ocjenu jezičnih modela zbog ambivalentnosti koja je jeziku inherentna. Za ilustraciju, kontekst *"Znaš da volim..."* smisleno mogu nastaviti riječi *"cirkus"*, *"mrkvu"*, *"skakati"* i brojne druge. Bilo bi stoga neispravno jezični model ocjeniti neuspješnim ako je kao nastavak predvidio riječ *"automobile"*, a u ispitnom skupu je dotični kontekst nastavljen s riječi *"motocikle"*. Istovremeno, može

se zahtjevati od modela da neku zamjetnu vjerojatnost dodijeli i riječi *"motocikle"*. Stoga se ocjenjuje upravo ta vjerojatnost.

Najčešće korištena mjera uspješnosti jezičnog modela je *"perplexity"* (u prijevodu *"zbrka"*, *"zbunjenost"*). Označava se simbolom $\mathcal{P}$, njena definicija slijedi.

$$\mathcal{P} = b^{-\sum_x P(w|...) log_b Q(w|...)}$$

Pri tome je $P(w|...)$ vjerojatnost da kontekst ... nastavi riječ w, $Q(w|...)$ je predikcija te vjerojatnosti od strane modela, a b baza logaritma (tipično se koriste 2 ili baza prirodnog logaritma e). U ovom radu korištena je baza prirodnog logaritma e.

$$\mathcal{P} = \exp\left(-\sum_x P(w|...) \ln Q(w|...)\right)$$

Kada se ova mjera koisti na skupu za testiranje u pravilu se pretpostavlja da je vjerojatnost primjera $P(w|...)$ jednaka udjelu tog primjera u skupu za testiranje, odnosno da je $P(w|...) = 1/N_t$, gdje je N_t broj primjera u testnom skupu.

$$\mathcal{P} = \exp\left(-\sum_x \frac{1}{N_t} \ln Q(w|...)\right)$$

Konačni izraz za mjeru vrednovanja, koji koristi uvjetnu vjerojatnost riječi s obzirom na prethodnu $(n-1)$ riječ, tada je definiran na sljedeći način.

$$\mathcal{P} = \exp\left(-\frac{1}{N_t} \sum_{w_{i-(n-1)},...,w_n} \ln Q(w_i|w_{i-(n-1)}, ..., w_{i-1})\right) \tag{7.1}$$

Za dobivanje dojma o mjeri *perplexity* najbolje je razmotriti nekoliko primjera. Može se zamisliti jezični model koji svakoj riječi dodjeljuje istu vjerojatnost (nasumično pogađa riječi), bez obzira na kontekst. Ovakav model očito je vrlo loš, ali pogoduje za ilustraciju. Vjerojatnost koju će taj model ponuditi za svaku riječ tada je $1/|V|$. Ako se ta vrijednost uvrsti u izraz 7.1, mjera *perplexity* će biti $\mathcal{P} = |V|$. Potom se može razmotriti nešto bolji model. Taj model predviđa da je za zadani kontekst ispravni nastavak jedna od 100 različitih riječi, svaka s podjednakom vjerojatnošću. Vjerojatnost koju on nudi je 0.01, a mjera *perplexity* za taj model tada je $\mathcal{P} = 100$. Dakle, za model koji primjer ispitnog skupa uvijek ocjenjuje vjerojatnošću p, mjera *perplexity* je $\mathcal{P} = 1/p$. Naravno, stvarni modeli ne daju tako uniformne i konzistentne procjene, ali navedeni primjeri trebali bi okvirno ilustrirati ponašanje mjere *perplexity*.

Potrebno je razmotriti i korištenje mjere *perplexity* za usporedbu s modelima iz drugih radova. Generalno govoreći, izravnu usporedbu treba izbjegavati. Razlog tome

je što mjera *perplexity* jako ovisi o vokabularu korištenom za izgradnju jezičnog modela, kao i veličini skupa za treniranje i evaluaciju, odnosno podudarnosti distribucija tih skupova. Primjerice, korištenjem manjeg vokabulara (koji može biti naprosto posljedica drukčijeg pretprocesiranja teksta) moguće je na istom skupu za treniranje postići niže rezultate mjere *perplexity*. Ovo može navesti na krivi zaključak kako je dotični model bolji od nekog treniranog na istom korpusu, ali korištenjem većeg vokabulara. Istovremeno, korištenje većeg vokabuara možda predikcije čini finijim, preciznijim, i time prikladnijim za neku specifičnu uporabu. U tom smislu je mjera *perplexity* prikladna isključivo za usporedbu modela treniranih na istom korpusu s istim vokabolarom, te jednakom podjelom korpusa na skup za treniranje i ispitivanje.

7.3. Korpus

Za vrednovanje jezičnih modela implementiranih u ovom radu koristit će se *Microsoft Research Sentence Completion Challenge* (u nastavku MRSCC) korpus [**?**]. Zadak u MRSCC je izgraditi jezični model koji točno kompletira 1040 rečenica ispitnog skupa. U svakoj rečenici je za jednu riječ osmišljeno četiri alternativne, koje su sintaktički ispravne, ali semantički manje smislene od originalne riječi. Rečenice su sve uzete iz Sir Arthur Conan Doyle-ovog serijala o detektivu Sherlocku Holmesu. Slijedi primjer jedne rečenice i ponuđenih alternativa.

1. The king **landed** at him in amazement.
2. The king **rejoiced** at him in amazement.
3. The king **smiled** at him in amazement.
4. The king **knocked** at him in amazement.
5. The king **stared** at him in amazement.

U navedenom primjeru naviše smisla ima posljednja rečenica (*"stared"*). Treća rečenica (*"smiled"*) je donekle smislena, a ostale osjetno manje. Upravo zbog toga što niti jedna alternativa nije sasvim besmislena je MRSCC zadatak vrlo zanimljiv. Čovjek koji dobro govori engleski jezik lako postiže uspješnost od 90% na cijelom skupu. Istovremeno je za računalo zadatak iznimno težak, jer se bazira na semantičkom povezivanju koncepata i tumačenju konteksta rečenice. U trenutku pisanja svi računalni jezični modeli na MRSCC testu postižu točnost ispod 60%.

Za MRSCC je točno definiran korpus teksta nad kojim je dozvoljeno graditi model. Time se osigurava da sve implementacije imaju iste početne uvjete. Propisani korpus se sastoji od 522 književna djela (pretežno romana) engleskog govornog područja iz istog vremenskog perioda u kojem je pisao Conan Doyle (19. stoljeće). Cijeli korpus sadrži oko 50 milijuna riječi, a preuzet je sa stranica Guttenberg projekta[1], gdje su korištena djela dostupna u cijelosti, bez zakonskih ograničenja.

Unutar ovog rada jezični modeli su vrednovani primarno korištenjem mjere *"perplexity"* na ispitnom skupu izdvojenom iz korpusa. Vrednovanje na MRSCC zadatku je sekundarno. Razlog je to što cijeli korpus of 50 milijuna riječi zahtjeva računalne resurse i vrijeme treniranja koji za pisanje ovog diplomskog rada nisu dostupni. Moguće je implementirati modele koji se treniraju nad podskupom od 20 romana (od dostupnih 522), odnosno 1.8 milijuna riječi (od dostupnih 50). Stoga je nemoguće rezultate testiranja uspoređivati s rezultatima modela treniranih nad cijelim MRSCC korpusom, od primarnog interesa je realtivna usporedba različitih pristupa izgradnji jezičnih modela.

7.4. Pretprocesiranje i normalizacija teksta

Korpus za treniranje dostupan je u obliku u kojem je objavljen u sklopu Guttenberg projekta. Potrebno je pročistiti ga kako bi se dobio tekst prikladan za izgradnju jezičnog modela. Prvo su maknuta zaglavlja i ostali dodatci samog Guttenberg projekta, pošto oni nisu dio originalnog teksta. Brojevi pisani znamenkama se zamjenjuju posebnim tokenom. Potom se tekst dijeli na rečenice i riječi korištenjem knjižnice za procesiranje prirodnog jezika SpaCy[2]. Rezultat su riječi grupirane u rečenice, pri čemu riječ sadrži samo slova, bez interpunkcije i praznina.

U svakom većem korpusu teksta je razdioba broja pojavljivanja riječi slična. Mali broj riječi se koristi velik broj puta (to su tipično denominatori, zamjenice itd.), dio riječi je srednje učestao (često korištene imenice, glagoli i pridjevi), te je konačno velik broj riječi korišten samo nekoliko puta. Pošto računalna složenost svih jezičnih modela raste s veličinom vokabulara, riječi rijetkog pojavljivanja se često ignoriraju, odnosno zamjenjuju jednim tokenom koji ih predstavlja sve. Osim smanjenja vokabulara time se iz korpusa pročisti i dobar dio pogrešno napisanih riječi. Na korištenom podskupu MRSCC korpusa kriterij korištenja riječi je da se pojavila barem 5 puta, u barem 2 književna djela. Time je vokabular od preko 37 tisuća riječi smanjen na manje od 13

[1] https://www.gutenberg.org/

[2] https://honnibal.github.io/spaCy/

tisuća, skoro trostruko, a da pri tome odbačene riječi predstavljaju samo 4.5% korpusa.

Osim odbacivanja malo korištenih riječi, tekst se često normalizira po pitanju morfološke varijacije. Za velik broj primjena jezičnih modela nije bitno primjerice glagosko vrijeme, stoga je poželjno da se, u domeni engleskog jezika, riječi *"walk"*, *"walks"* i *"walked"* tretiraju kao ista riječ. To povećava broj primjera u kojem jezični model vidi glagol *"walk"*, što omogućava bolju generalizaciju. Naravno, postoje primjene jezičnih modela (automatsko prevođenje, provjera ispravnosti teksta) u kojima su distinkcije po glagolskom vremenu bitne. Unutar ovog rada pretpostavlja se primjena koja je prilagođena MRSCC zadatku, gdje je povezivanje semantike riječi bitnije od morfološke ispravnosti.

Za takvu primjenu potrebno je obaviti morfološku normalizaciju riječi. U tu svrhu se tipično koristi lematizacija (engl. *lemmatisation*) ili skraćivanje riječi na korijen (engl. *stemming*). Lematizacija riječ svodi na morfološki neutralan oblik iste vrste. Primjerice, lema riječi *"govorim"* je *"govoriti"*. Skraćivanje na korijen primjenom pravila infleksije odsijeca sufiks riječi. Takva normalizacija nad riječi *"govorim"* rezultira korijenom *"govori"*. Ponekad se koristi još agresivniji oblik skraćivanja gdje se od svake riječi uzima samo prvih k slova. Prednost ovog pristupa je da je vrlo jednostavan za implementaciju i dobro radi u jezicima s malim brojem složenih riječi, poput engleskog.

Unutar ovog rada isprobano je korištenje lematizatora knjižnice SpaCy i skraćivanje na prvih k slova, pri čemu je parametar k optimiziran. Inicijalno testiranje na MRSCC zadatku, korištenjem 4-grama, je pokazalo da skraćivanje na prva 4 slova daje najbolje rezultate. Iako se to može činiti kao odveć destruktivan oblik normalizacije, u skladu je s Google-ovim eksperimentima[3]. Korištenjem skraćivanja na prva 4 slova vokabular od oko 38 tisuća riječi je sveden na nešto manje od 10 tisuća prefiksa. Nakon što se primjeni odbacivanje prefiksa koji su se pojavili manje od pet puta, ostaje tek 5133 prefiksa. Iskorištenost korpusa postaje preko 99%, dakle osjetno više nego kada su se odbacivale nenormalizirane riječi s manje od 5 pojavljivanja.

7.5. Rezultati

U konačnici su vrednovana četiri modela. Prva dva se baziraju na prebrojavanju *n*-grama uz zaglađivanje aditivnom i Kneser-Ney metodom. Treći model je neuronska mreža, a posljednji je log-bilinearni model.

[3] https://www.youtube.com/watch?v=nU8DcBF-qo4

Podatci za treniranje su prvih dvadeset romana iz MRSCC korpusa, koji sveukupno sadrže oko 1.8 milijun riječi. Iz skupa za treniranje je tisuću *n*-grama izdvojeno za validaciju tijekom treniranja, a pet posto (oko 90 tisuća *n*-grama) za konačno vrednovanje.

Za modele zaglađenog prebrojavanja se korištenjem validacijskog skupa optimiraju parametri zaglađivanja *lambda* (kod aditivnog zaglađivanja) i *delta* (kod zaglađivanja Kneser-Ney metodom). Za neuronsku mrežu i log-bilinearni model su hiperparametri modela (L2 regularizacija, veličina mini-grupe) odabrani u skladu s postojećim eksperimentima [9], nisu optimirani. Isto vrijedi i za parametre gradijentnog spusta, koristi se naime RMSprop, koji veličinu gradijentnog pomaka prilagođava automatski, te rijetko zahtjeva pažljivo podešavanje. Validacijski skup je korišten za određivanje trajanja treniranja neuronske mreže i log-bilinearnog modela. Oba modela se treniraju sve dok mjera *perplexity* na validacijskom skupu pada za barem 1 u posljednje tri epohe treniranja.

7.5.1. Vrednovanje mjerom *perplexity*

Tablica 7.1 prikazuje rezultate treniranih modela na skupu za testiranje, vrednovano mjerom *perplexity*.

Tablica 7.1: Rezultati vrednovanja jezičnih modela mjerom *perplexity* za *n*-grame zaglađene aditivno i Kneser-Ney metodom, neuronsku mrežu i log-bilinearni model, po stupcima za različite vrijednosti parametra n.

	Parametar *n*		
Model	3	4	5
Additivno	305	1182	2680
Kneser-Ney	72	121	204
Neuronska mreža	117	114	113
Log-biliner	102	98	98

Najbolje rezultate daje jezični model baziran na prebrojavanju i zaglađivanju Kneser-Ney metodom. Model baziran na aditivnom zaglađivanju daje najgore rezultate, što je u skladu s očekivanjem. Vidljivo je kako rezultati modela baziranih na prebrojavanju padaju s povećanjem parametra n, odnosno veličinom konteksta. Ovo je izravno povezano s već razmatranom posljedicom povećanog konteksta, u smislu sve veće rijetkosti pojavljivanja duljih *n*-grama u ograničenom korpusu. Kada se koristi cijeli MRSCC korpus, najbolji rezultati se dobivaju za duljinu konteksta ($n \in \{4, 5, 6\}$).

Rezultati jezičnih modela baziranih na neuronskoj mreži i log-bilinearnom modelu daju dobre rezultate. Poznato je da među ta dva modela log-bilinerani radi bolje [9], što pokazuju i pokusi izvedeni u ovom radu. Zanimljiv aspekt ovih modela je kako njihove performanse rastu povećanjem konteksta. Uzrok poboljšanja je najvjerojatnije to što se distribuirane reprezentacije riječi u tim modelima računaju na temelju svih riječi u kontekstu. Stoga dulji kontekst modelu daje više informacije o međusobnoj sličnosti riječi te je učenje reprezentacija kvalitetnije. Istovremeno, za razliku od modela baziranih na prebrojavanju, neuronska mreža i log-bilinearni model su manje osjetljivi na rjetkost pojavljivanja duljeg konteksta, što izravno proizlazi iz sposobnosti modela da za semantički slične riječi pronađu slične reprezentacije.

Pitanje se postavlja zašto, s obzirom na sve njihove prednosti, neuronska mreža i log-bilinearni model daju lošije rezultate od modela zaglađenog Kneser-Ney metodom. Razlog tomu je veličina vokabulara. Korištenje vrlo malog podskupa MRSCC korpusa, uz skraćivanje na prva četiri slova, rezultira iznimno malim vokabularom. Pošto su metode bazirane na prebrojavanju osjetljivije na veličinu vokabulara (jer ne koriste distribuirane reprezentacije), korištenjem većeg vokabulara za očekivati je da log-bilinearni model postigne bolje rezultate, u skladu s postojećim eksperimentima [9]. Bitno je prisjetiti se ranije napomene kako dobiveni rezultati nisu prikladni za usporedbu s rezultatima iz drugih radova zbog korištenja različitog korpusa (podskup MRSCC korpusa naspram cijelog) i smanjenog vokabulara.

7.5.2. Uspješnost modela na MRSCC zadatku

Modeli su vrednovani i na konkretnom MRSCC zadatku odabira semantički točne riječi u rečenici, od pet ponuđenih. Mjera vrednovanja je udio točnih odabira na skupu od 1040 rečenica. Važno je napomenuti kako su rezultati koji slijede predviđeni za relativnu usporedbu korištenih modela. Usporedba s rezultatima iz drugih radova nije smislena jer je u ovom radu zbog vremenskih ograničenja korišten tek mali podskup MRSCC korpusa. Tablica 7.2 prikazuje dobivene rezultate.

Rezultati na MRSCC zadatku su u skladu s rezultatima korištenjem mjere *perplexity*. Prebrojavanje *n*-grama uz aditivno zaglađivanje daje najlošije rezultate (jednake uspješnosti nasumičnog odabira), dok prebrojavanje korištenjem zaglađivanja Kneser-Ney i log-bilinearni model daju bolje, približno jednake rezultate. I dalje je vidljivo kako modeli bazirani na prebrojavanju rade lošije kada je kontekst dulji, a log-bilinearni model bolje. Neuronska mreža konzistentno daje osrednje rezultate.

U radu unutar kojeg su se jezični modeli trenirali korištenjem cijelog MRSCC ko-

Tablica 7.2: Rezultati uspješnosti jezičnih modela na MRSCC zadatku.

	Parametar n		
Model	3	4	5
Additivno	0.19	0.19	0.19
Kneser-Ney	0.27	0.27	0.24
Neuronska mreža	0.23	0.23	0.22
Log-biliner	0.25	0.27	0.27

rupsa, najbolji rezultat postignut je korištenjem log-bilinearnog modela treniranog nad 10-gramima, prijavljena je 0.547 uspješnost (najbolja poznata u trenutku pisanja ovog rada). U istom radu model baziran na prebrojavanju 4-grama uz zaglađivanje Kneser-Ney metodom postiže 0.391 usješnost. Čini se stoga da je log-bilinearni model generalno bolji, ali valja uzeti u obzir i trošak treniranja takvog modela.

7.5.3. Trajanje treniranja

Neuronske mreže i log-bilinearni model treniraju se iterativnom metodom gradijentnog smanjenja greške. Za velike modele (modele koji imaju velik broj parametara) i velike skupove podataka za treniranje, učenje modela može biti vremenski trajno. Stoga je unutar ovog rada testirano i vrijeme potrebno za treniranje dotičnih modela. Modeli bazirani na prebrojavanju zahtjevaju samo jedan prolazak kroz n-grame skupa za treniranje i malu količinu dodatne analize. Njihovo treniranje u pravilu traje tek nekoliko minuta. Modeli bazirani na iterativnom smanjenju greške kompleksnijim algoritmima analiziraju skup za treniranje kroz veći broj iteracija (u ovom radu prosječno tridesetak iteracija). Stoga su trenirani na snažnom modernom centralnom procesoru[4] (CPU), i na modernom grafičkom procesoru srednje klase[5] (GPU). Trajanje treniranja iterativnih modela prikazano je u tablici 7.3. Bitno je napomenuti kako vremena treniranja ovise o mnogim čimbenicima kao što su optimalnost implementacije, kriterij zaustavljanja iterativnog treniranja, veličina minigrupe pri iterativnom treniranju, tip gradijentnog spusta koji se koristi itd. Stoga navedena vremena treba tumačiti kao vrlo okvirna, navedena su samo radi međusobne usporedbe.

Vidljivo je kako je vrijeme trajanja treniranja modela koji se baziraju na prebrojavanju neusporedivo kraće od treniranja neuronskih mreža i log-bilinearnog modela.

[4]Intel i7-4720, 2.6GHz, 4 fizičke jezgre, 8-dretvi, 16GB RAM

[5]nVidia GeForce GTX 960M, 4GB RAM, 640 programabilnih CUDA jezgri

Tablica 7.3: Trajanje treniranja modela, izraženo u SAT:MINUTE obliku.

	Parametar *n*		
Model	3	4	5
Prebrojavanje *n*-grama CPU	0:01	0:01	0.01
Neuronska mreža, GPU	1:47	2:15	2:17
Neuronska mreža, CPU	13:38	13:52	14.20
Log-bilinear GPU	0:49	0:48	0:46
Log-bilinear CPU	8:07	8:01	6:56

Uzme li se u obzir da model koji koristi zaglađivanje Kneser-Ney metodom postiže vrlo dobre rezultate, taj model ostaje kompetitivan u situacijama gdje su računalni resursi ili vrijeme treniranja ograničeni. Nadalje, vidljive su velike razlike između vremena treniranja korištenjem centralnog odnosno grafičkog procesora. Pri tome je bitno uzeti u obzir i to da je korišten CPU visoke klase (u vrijeme pisanja rada), a korišteni GPU više-srednje klase. Usporeba vrhunskih centralnih i grafičkih procesora rezultirala bi još većom disproporcijom.

Zanimljivo je kako kod log-bilinearnog modela kraći kontekst rezultira sporijim treniranjem. Razlog tome je povećanje broja *n*-grama kada se koristi kraći kontekst, što je posljedica toga da se *n*-grami generiraju iz rečenica, bez preklapanja. Stoga se iz svake rečenice generira $N-n+1$ *n*-gram, gdje N označava broj riječi u rečenici. Tako se iz istog korpusa rečenica generira čak 4% više trigrama nego 4-grama. Istovremeno, dodatne računske operacije potrebne za treniranje korištenjem duljeg konteksta imaju relativno mali utjecaj kada se koriste optimalne implementacije linearne algebre.

8. Zaključak

U ovom diplomskom radu vrednovani su jezični modeli izgrađeni korištenjem tri bitno različita pristupa. Prvi pristup, koji se već dugo koristi, bazira se na prebrojavanju *n*-grama. Drugi pristup bazira se na neuronskoj mreži. Treći pristup je najnoviji, bazira se na log-bilinearnom modelu koji ima znatno manji broj parametara od neuronske mreže.

Vrednovanje navedenih modela korištenjem podskupa korpusa *Microsoft Research Sentence Completion Challenge* pokazala je kako najbolje rezultate daju model baziran na prebrojavanju i zaglađivanju Kneser-Ney metodom, i log-bilinearni model. Pri tome je pokazano da na manjem korpusu prebrojavanje daje nešto bolje rezultate, dok ispitivanja u drugim radovima pokazuju da na većem korpusu log-bilinearni model daje najbolje rezultate. Ispitano je i vrijeme potrebno za treniranje svih korištenih modela, pri čemu se pokazalo da je treniranje modela baziranih na prebrojavanju neusporedivo brže od treniranja neuronske mreže i log-bilinearnog modela. U tom smislu je jezični model baziran na prebrojavanju i zaglađivanju Kneser-Ney metodom i dalje primjenjiv, bez obzira na relativno dugu povijest korištenja (prva implementacija je bila 1995. godine).

U napretku jezičnih tehnologija, poput prepoznavanja govora i automatiziranog prevođenja, jezični modeli imaju ključnu ulogu. Uz to oni nisu ograničeni samo na tu primjenu, mogu se koristiti za analizu i klasifikaciju teksta, detekciju grešaka u tekstu i za mnoge druge zadatke. Stoga je za očekivati da će se intenzivno istraživanje i razvoj ovog područja nastaviti. Trenutno je u području strojnog učenja i izadi prediktivnih modela snažan trend korištenja dubokih i konvolucijskih neuronskih mreža. Već postoje radovi koji primjenom ovih tehnika grade još sposobnije jezične modele. Njihova sposobnost prepoznavanja semantičkih uzoraka u tekstu se polako ali neosporivo približava ljudskoj. U tom smislu su jezični modeli jedna od krucijalnih komponenata izgradnje strojeva čija bi "inteligencija" bila usporediva s ljudskom.

LITERATURA

[1] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, i Richard Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

[2] Geoffrey E. Hinton. Products of experts, 1999.

[3] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800, Kolovoz 2002. ISSN 0899-7667. doi: 10.1162/089976602760128018. URL `http://dx.doi.org/10.1162/089976602760128018`.

[4] Geoffrey E. Hinton i Simon Osindero. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:2006, 2006.

[5] Reinhard Kneser i Hermann Ney. Improved backing-off for m-gram language modeling. U *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, svezak I, stranice 181–184, Detroit, Michigan, Svibanj 1995.

[6] George James Lidstone. Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities. *Transactions of the Faculty of Actuaries*, 8:182–192, 1920.

[7] Tomáš Mikolov. *Statistical language models based on neural networks*. Doktorska disertacija, Brno University of Technology, 2012.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, i Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013. URL `http://dblp.uni-trier.de/db/journals/corr/corr1310.html#MikolovSCCD13`.

[9] Andriy Mnih i Geoffrey E. Hinton. Three new graphical models for statistical language modelling. U Zoubin Ghahramani, urednik, *ICML*, svezak 227 od *ACM International Conference Proceeding Series*, stranice 641–648. ACM, 2007. ISBN 978-1-59593-793-3. URL http://dblp.uni-trier.de/db/conf/icml/icml2007.html#MnihH07.

[10] D. E. Rumelhart, G. E. Hinton, i R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. poglavlje Learning Internal Representations by Error Propagation, stranice 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL http://dl.acm.org/citation.cfm?id=104279.104293.

[11] Richard Socher, Brody Huval, Christopher D. Manning, i Andrew Y. Ng. Semantic Compositionality Through Recursive Matrix-Vector Spaces. U *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

[12] F. Stamenković. Duboke neuronske mreže. 2014.

[13] T. Tieleman i G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012.

[14] Xiang Zhang i Yann LeCun. Text understanding from scratch. *CoRR*, abs/1502.01710, 2015. URL http://arxiv.org/abs/1502.01710.

[15] M. Čupić, Dalbelo Bašić B., i Golub M. *Neizrazito, evolucijsko i neuroračunarstvo*. 2013.

9. Sažetak