SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4816

Analiza teksta pomoću dubokih neuronskih mreža

Borna Bešić

Zagreb, lipanj 2017.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Kako biste uklonili ovu stranicu, obrišite naredbu `\izvornik`.

Ovim putem želim se zahvaliti mentoru doc. dr. sc. Marku Subašiću na profesionalnom usmjerenju, stručnim savjetima te svim omogućenim resursima za izradu ovog rada. Također, zahvalan sam svojim roditeljima na bezuvjetnoj podršci tijekom svih ovih godina školovanja.

SADRŽAJ

1. Uvod

Svakog dana čovječanstvo proizvodi otprilike zapanjujućih 2.5 kvintilijuna (10^{30}) okteta podataka. Količina podataka kojima raspolažemo toliko je velika da je 90% trenutno dostupnih podataka nastalo u zadnje dvije godine [1]. S obzirom na gotovo eksponencijalan rast količine podataka, vrlo mali udio podataka je uspješno strukturiran. Zbog ogromne brzine prikupljanja nestukturiranih podataka iz velikog broja izvora, takvi podaci u tom obliku su jednostavno neiskoristivi. Složenost i veličina raznih baza podataka onemogućuju tradicionalnoj programskoj podršci mogućnost kvalitetne analize. Takva vrsta podataka popularno se naziva *Big Data*.

Razne tvrtke, organizacije te države i njihove uprave kao rezultat korištenja vlastitih informacijskih sustava u poslovanju prikupljaju takve podatke vrlo velikom brzinom. U današnje vrijeme sve češće to rade s ciljem kako bi analizom podataka došli do određenih znanja, no većina podataka u prošlosti je nastala kao nusprodukt poslovanja, prvenstveno zbog nemogućnosti obrade uslijed raznih ograničenja. Najčešće su to korisnički podaci, razni zapisnici sustava, podaci o zaposlenicima, anketni podaci, financijska izvješća, razne verzije dokumenata, sudski dokumenti, ugovori i slično. Valja primijetiti da se u ovom slučaju radi o podacima koji su gotovo svi u tekstualnom obliku [2], čiji je sadržaj prvenstveno prirodni jezik.

Obrada prirodnog jezika (*eng. NLP - natural language processing*) područje je računarske znanosti, točnije umjetne inteligencije i računarske lingvistike koje se bavi računalnim metodama u svrhu obrade i razumijevanja prirodnog jezika. Samo neki od zadataka obrade prirodnog jezika su slijedeći:

- klasifikacija (dokumenti, paragrafi, rečenice, riječi)
- prepoznavanje izraza te imenovanih entiteta (*eng. NER - named entity recognition*)
- parsiranje (gradnja sintaksnog stabla)
- prepoznavanje govora
- strojno prevođenje
- generiranje sažetka (*eng. text summarization*)
- sentimentna analiza

U današnje vrijeme vrlo popularan i vrlo uspješan pristup rješavanju problema umjetne inteligencije, pa tako i u obradi prirodnog jezika, jest strojno učenje. Strojno učenje danas postiže *state-of-the-art* rezultate na AI-teškim problemima zahvaljujući konektivističkom pristupu, točnije umjetnim neuronskim mrežama (*eng. ANN - artificial neural networks*). Umjetna neuronska mreža skup je međusobno povezanih čvorova (umjetnih neurona) kojim se nastoji simulirati rad ljudskog mozga. Za razliku od tradicionalnog pristupa programiranju računala, kod ove biološki inspirirane paradigme, računalo nema eksplicitne upute kako riješiti zadani problem. Računalo samo uči iz priloženih podataka te je s vremenom sve bliže idealnom rješenju.

Posebnu kategoriju umjetnih neuronskih mreža čine duboke neuronske mreže (*eng. DNN - deep neural networks*). Njihova glavna karakteristika su slojevi koji služe za ekstrakciju značajki (*eng. feature extraction*) čime se umanjuju pa čak i rješavaju problemi inženjerstva značajki. Također, ulančavanjem različitih vrsta i arhitektura neuronskih mreža tvori se hijerarhija različitih razina apstrakcije.

Problem koji se rješava u ovom radu jest **klasifikacija odsječaka teksta različite duljine** pomoću neuronskih mreža u razvojnoj biblioteci TensorFlow koristeći programski jezik Python. Svaki od prikupljenih skupova tekstualnih podataka sadrži označene kategorije kojima primjeri pripadaju te je podijeljen u dva podskupa: podskup za treniranje te podskup za testiranje. Treniranje modela neuronske mreže provodi se kao nadzirano strojno učenje.

2. Vektorske reprezentacije riječi

2.1. Motivacija

Sustavi za obradu slike i zvuka koriste visokodimenzionalne skupove podataka kodirane u vektore (npr. intenziteti piksela kod slika). S druge strane, sustavi za obradu prirodnog jezika tradicionalno tretiraju riječi kao diskretne simbole. Kodiranje takvih simbola je proizvoljno i ne pruža korisnu informaciju sustavu u pogledu odnosa koji postoji između određenih riječi. Na primjer, to znači da model može iskoristiti vrlo malo znanja o mačkama u trenutku kada obrađuje podatke o psima (bez obzira što su oboje životinje s četiri noge te kućni ljubimci). Predstavljajući riječi kao jedinstvene, diskretne simbole nadalje vodi u veliku raspršenost podataka (*eng. data sparsity*) što često znači da nam treba veća količina podataka da bismo uspješno istrenirali statistički model. Koristeći vektorske reprezentacije riječi možemo se riješiti navedenih prepreka [3]. Modeli u vektorskom prostoru (*eng. VSM - vector space models*) preslikavaju riječi u točke u kontinuiranom vektorskom prostor gdje su točke semantički bliskih riječi jedna blizu druge.

2.2. Word2vec

Model za vektorsku reprezentaciju riječi koji ćemo koristiti za problem klasifikacije jest Word2vec. Word2vec je računalno efikasan prediktivni model koji dolazi u dvije varijante: *continuous bag-of-words* (CBOW) i *skip-gram*. Algoritamski, obje varijante su slične jedino što CBOW predviđa ciljnu riječ iz konteksta zadanih riječi dok skip-gram radi upravo obrnuto odnosno predviđa riječi konteksta iz zadanih ciljnih riječi.

Word2vec bazira se na dvoslojnoj neuronskoj mreži koja procesuira tekstualni korpus određenog jezika. Rezultat treniranja takve neuronske mreže jest skup vektora za svaku od riječi iz korpusa. Iako Word2vec nije duboka neuronska mreža, riječi preslikane u brojčane vektore su jezik koji duboke neuronske mreže mogu razumjeti. Primjene Word2vec koncepta nadilaze granice obrade prirodnog jezika. Koncept se također može primijeniti na gene, programske kodove, popise pjesama te ostale verbalne i simboličke nizove u kojem se mogu otkriti razni uzorci. Uz dovoljno velik tekstualni korpus, Word2vec može vrlo precizno

predvidjeti značenje riječi na temelju prethodnih pojavljivanja. Takva predviđanja mogu se iskoristiti kako bi se odredila povezanost određene riječi s ostalim riječima (npr. "muškarac" naprema "dječak" je isto kao i "žena" naprema "djevojčica") ili čak grupirati dokumente te ih klasificirati prema temi.

Kao mjera sličnosti dvaju vektora $\mathbf{x_1}$ i $\mathbf{x_2}$ dobivenih Word2vec modelom, najčešće se koristi kosinusna sličnost koja daje vrijednosti iz intervala $[-1, 1]$:

$$cos(\alpha) = \frac{\mathbf{x_1}^T \mathbf{x_2}}{||\mathbf{x_1}|| \cdot ||\mathbf{x_2}||} \tag{2.1}$$

Za naš problem iskoristiti ćemo javno dostupne gotove vektorske reprezentacije riječi engleskog jezika istrenirane na 100 milijardi riječi Google News članaka [1]. Model sadrži vektore 3 milijuna riječi i izraza gdje je svaki vektor definiran je kao $\mathbf{x} \in \mathbb{R}^{300}$ [4]. Primjer kosinusnih sličnosti za dotični preuzeti skup vektora prikazan je u tablici 2.1.

Tablica 2.1: Popis nasličnijih riječi za zadanu riječ *france*

Riječ	Kosinusna sličnost
belgium	0.665923
netherlands	0.652428
italy	0.633130
switzerland	0.622323
luxembourg	0.610033
portugal	0.577154
russia	0.571507
germany	0.563291
catalonia	0.534176

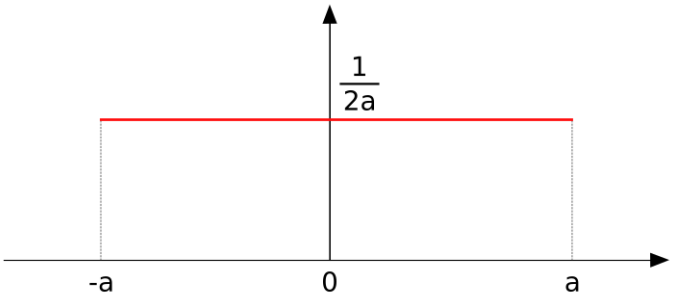
2.2.1. Vektori za nove riječi

Za svaku riječ iz podskupa za treniranje za koju ne postoji vektor u početnom Word2vec modelu, stvaramo novi vektor tako da se i-ti element novog vektora inicijalizira se iz uniformne razdiobe iz intervala $[-a_i, a_i]$. a_i određen tako da uniformna razdioba iz koje se bira ima varijancu jednaku varijanci svih izvornih elemenata na i-tom mjestu. Iz skice sa slike 2.1 proizlazi:

$$Var(x_i) = \frac{1}{3} a_i^2$$
$$a_i = \sqrt{3 \cdot Var(x_i)} = \sigma_i \sqrt{3} \tag{2.2}$$

[1] https://code.google.com/archive/p/word2vec/

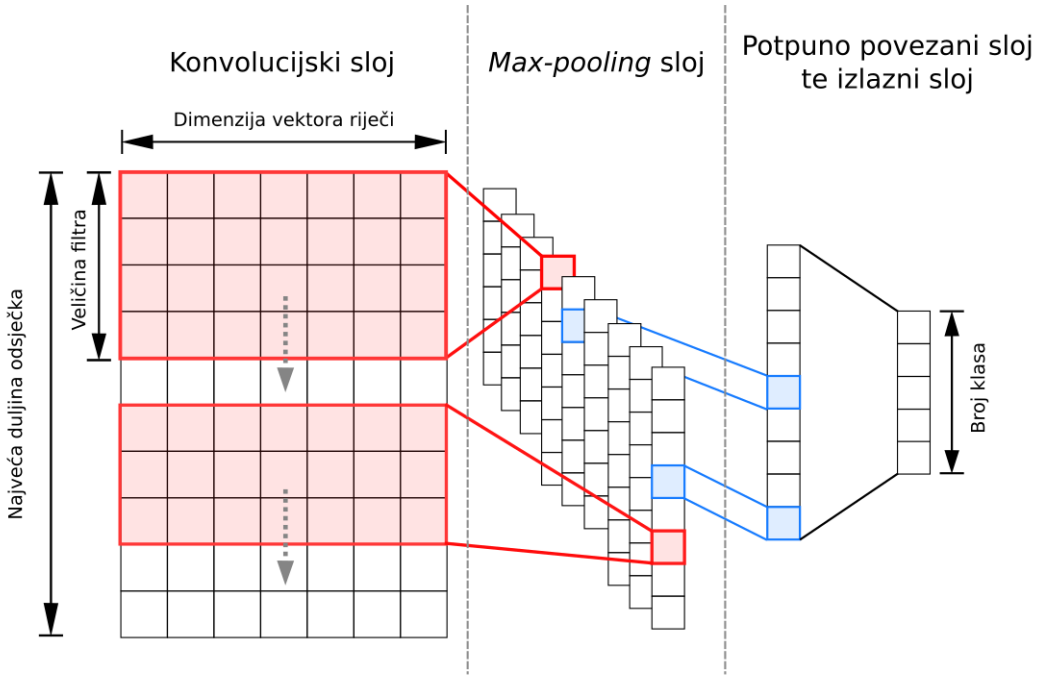
gdje je σ_i standardna devijacija i-tog elementa kroz sve vektore početnog modela.
Tijekom testiranja te samog rada istrenirane neuronske mreže, ukoliko ne postoji vektor za trenutnu riječ, tada se koristi prazan vektor inicijalno popunjen nulama $(x_1\ x_2\ ...\ x_{300}) = (0\ 0\ ...\ 0)$.



Slika 2.1: Skica gustoće uniformne razdiobe

3. Arhitektura neuronske mreže

Neuronska mreža korištena u ovom radu jest implementacija konvolucijske neuronske mreže iz rada autora Kim (2014) [5]. Navedenu arhitekturu čine ulazni sloj, konvolucijski sloj, *max-pooling* sloj, potpuno povezani sloj te izlazni sloj. Skica arhitekture prikazana je na slici 3.1. Iako izvorno primijenjena na problem klasifikacije rečenica, u ovome radu ispitati ćemo također njezinu primjenjivost i na klasifikaciju većih odsječaka teksta.



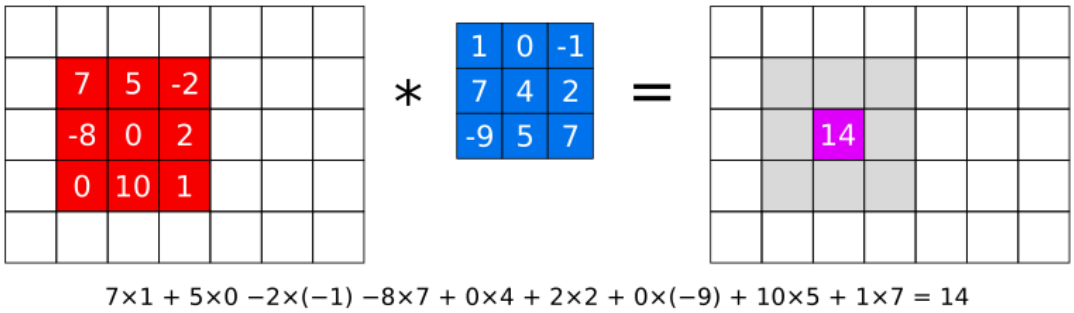
Slika 3.1: Skica korištene arhitekture neuronske mreže

3.1. Ulazni sloj

Svaka riječ iz skupa za treniranje određena je svojim Word2vec indeksom. Indeks određenoj riječi omogućava preslikavanje u jedinstveni vektor iz zadanog vektorskog prostora. Za svaku riječ na koju smo naišli, a nije u izvornom Word2vec modelu, inkrementalno se dodjeljuje prvi dostupni indeks uz stvaranje novog vektora. Poseban slučaj je indeks 0 koji je

rezerviran za riječi izvan skupa za treniranje za koje ne postoji vektorska reprezentacija u dostupnom modelu. On se preslikava u već spomenuti prazan vektor popunjem nulama.
Ulaz u neuronsku mrežu jest lista indeksa svih riječi zadane rečenice, paragrafa ili dokumenta. Kako ulaz u neuronsku mrežu mora biti fiksne veličine, ovdje nastaje problem iz razloga jer nisu svi odsječci teksta jednake duljine (nemaju jednak broj riječi). Kao rješenje, sve odječke teksta prije ulaza u neuronsku mrežu poravnati ćemo na najveću opaženu duljinu odsječka M iz skupa za treniranje koja se izračunava unaprijed, prilikom učitavanja skupa podataka. Popunjavanje se izvodi vrlo jednostavno: lista indeksa nadopuni se indeksom 0 dok njezina veličina ne dosegne najveću opaženu duljinu odjsečka. Prilikom testiranja modela, u slučaju odječka teksta koji je dulji od unaprijed određene najveće duljine, indeksi s kraja liste se odbacuju dok se ne postigne poravnanje na navedenu duljinu.
Nakon poravnanja, indeksi riječi se po principu *lookup* tablice preslikavaju u njihove vektorske reprezentacije. Tada lista indeksa riječi postaje matrica dimenzija $M \times 300$. Tako dobivena matrica sadrži sve potrebne informacije te služi kao temelj za operaciju konvolucije u slijedećem sloju.

3.2. Konvolucijski sloj



Slika 3.2: Primjer jednog koraka dvodimenzionalne konvolucije

Konvolucijski sloj jest srce konvolucijske neuronske mreže. Za razliku od klasičnih slojeva neuronskih mreža u kojima je svaki neuron povezan sa svakim u prijašnjem sloju, kod konvolucijskog sloja svaki neuron povezan je samo s ograničenim brojem neurona u prijašnjem sloju. Taj ograničeni broj neurona s kojima je povezan naziva se još i osjetilno područje neurona (*eng. receptive field*). Konvolucijski sloj prvenstveno služi za ekstrakciju značajki iz zadanog teksta, ali i smanjuje broj parametara koji je potrebno naučiti.
Osnovi slučaj konvolucije dvije funkcije je jednodimenzionalan te glasi:

$$(f * g)(k) = \sum_{n} f(k-n)g(n) \tag{3.1}$$

Jezgrena konvolucija vrši se velikim brojem jezgri (popularnije: konvolucijskih matrica, filtara) različitih veličina nad matricom iz prethodnog sloja. Svaki element filtra predstavlja jedan parametar kojeg je potrebno naučiti. Budući da se operacija jezgrene konvolucije izvodi nad dvodimenzionalnom matricom, potrebno je načiniti modifikacije. U tom slučaju izlaz za konvoluciju glasi:

$$(f * g)(x, y) = \sum_n \sum_m f(x - n, y - m)g(n, m) \tag{3.2}$$

gdje je $f(x, y)$ ulazna matrica sastavljena od vektora riječi dok je $g(x, y)$ filtar kojeg koristimo i čije elemente učimo. Na slici 3.2 prikazan je primjer jednog koraka konvolucije uz veličinu filtra 3×3. U našem slučaju, zadana veličina filtra jest zapravo visina konvolucijske matrice dok je širina implicitno postavljena na širinu matrice nad kojom se konvolucija izvodi što je u ovom slučaju dimenzija vektora riječi. Također, važan parametar konvolucije kao operacije je korak (*eng. stride*) koji za svaku dimenziju označava broj elemenata za koji se filtar pomiče tijekom prolaska preko izvorne matrice. Rezultat konvolucije je matrica koja se sastoji od onoliko dimenzija kroz koliko se filtar kretao tijekom konvolucije. Veličina rezultata D'_i za svaku dimenziju određena je sljedećom formulom:

$$D'_i = \frac{D_i - R_i}{S_i} + 1 \tag{3.3}$$

gdje je D_i dimenzionalnost početne matrice, R_i veličina filtra te S_i korak za i-tu dimenziju.

3.3. *Max-pooling* sloj

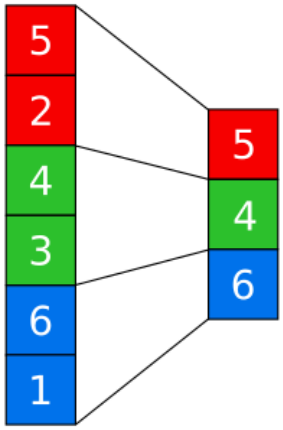
Česta praksa je dodavanje *pooling* sloja nakon jednog ili više konvolucijskih slojeva. Svrha ovog sloja jest postepeno reduciranje veličine (*eng. down-sampling*) slojeva neuronske mreže kako bi se smanjio broj parametara koje je potrebno naučiti te broj potrebnih računskih operacija. Postoji nekoliko funkcija koje se koriste za *pooling* sloj, poput aritmetičke sredine, sume, max funkcije i drugih. Iako je povijesno gledajući aritmetička sredina najčešće korištena kao funkcija u *pooling* sloju, pokazalo se da max funkcija bolje funkcionira te se ona stoga koristi i u ovom radu. Sloj koji obavlja takvu funkciju naziva se *max-pooling* sloj.

Najopćenitiji oblik ove operacije jest *k-max-pooling*, koji iz niza brojeva bira k najvećih. Operacija se izvodi pomoću klizećeg filtra određenog visinom, širinom te korakom. Za razliku od konvolucije, filtar u ovom slučaju nije matrica sa vlastitim elementima, već samo područje nad kojim se izvodi max funkcija. Stoga, popularniji naziv za filtar u ovom sloju je prozor (*eng. window*). Također, primjena *max-pooling*-a kreira efekt invarijantnosti na manje pomake u ulaznim podacima te kontrolira prenaučenost neuronske mreže (*eng. overfitting*). Zbog agresivne redukcije veličine slojeva (što je korisno za manje skupove podataka

radi izbjegavanja prenaučenosti), literatura preporuča korištenje manjih veličina filtara [6] ili čak potpuno izostavljanje ovakvih vrsta slojeva [7].

U ovom radu koristi se *1-max-pooling* odnosno biranje najvećeg elementa iz svakog rezultata konvolucije u prethodnom sloju. Biranje samo jednog najvećeg elementa empirijski je potvrđeno kao najbolja opcija [8].

Na slici 3.3 prikazan je primjer *1-max-pooling* operacije nad vektorom veličine 6×1 koristeći prozor veličine 2×1 i korak iznosa 2.



Slika 3.3: Primjer *1-max-pooling* operacije

3.4. Potpuno povezani sloj

Nakon što je obavljen *max-pooling* nad izlazima konvolucijskog sloja, rezultati se spajaju u jedan vektor. Dobiveni vektor sadrži onoliko elemenata koliko je iznosio ukupan broj filtara za konvoluciju (pošto se za svaki rezultat konvolucije uzima jedan najveći element). Tako izgrađeni vektor služi kao ulaz u potpuno povezani sloj (*eng. fully-connected layer*). Kao što samo ime govori, u ovome sloju svaki neuron povezan je sa svim neuronima prethodnog sloja. Potpuno povezani sloj tipičan je za klasične unaprijedne neuronske mreže (*eng. feed-forward neural networks*). Ovim slojem hijerarhijski završava funkcija ekstrakcije značajki te započinje funkcija klasifikacije odnosno učenje nelinearne funkcije na temelju koje na kraju donosimo odluku kojoj kategoriji pripada ulazni odsječak teksta. Svojstvo potpune povezanosti omogućava jednostavno i efikasno (pomoću sklopovske podrške) računanje izlaza koristeći matrični umnožak odnosno skalarni produkt:

$$x_{i+1} = f\left(x_i \cdot W_i + b_i\right) \tag{3.4}$$

gdje je x_{i+1} izlazni vektor, x_i ulazni vektor, W_i matrica težina, a b_i pomak (*eng. bias*) za i-ti sloj neuronske mreže. f predstavlja korištenu aktivacijsku funkciju o kojoj je više riječ

u nastavku.

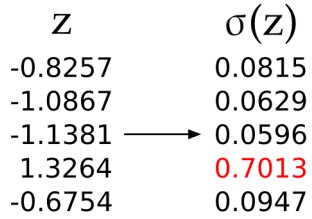
3.5. Izlazni sloj

Potpuno povezani sloj kao rezultat daje vektor C-dimenzionalni vektor gdje C predstavlja broj klasa za zadani skup podataka. Elementi navedenog vektora brojevi su iz skupa $\mathbb{R}$ dobiveni linearnom kombinacijom izlaza neurona iz prethodnog sloja. Na njih nije primijenjena aktivacijska funkcija što znači da njihovi iznosi nisu omeđeni. Svaki problem klasifikacije svodi se na određivanje vjerojatnosti pojedine kategorije. Naš zadatak je na temelju navedenih kontinuiranih elemenata odrediti diskretnu kategoriju kojoj odsječak teksta pripada. U tom slučaju, izlazni sloj se ponaša kao klasifikator.

Najednostavniji način klasifikacije jest biranje najvećeg elementa čijoj se klasi pridružuje vjerojatnost $p = 1$, a svim ostalim klasama vjerojatnost $p = 0$. U tome nas sprječava metoda optimizacije o kojoj je riječ kasnije u ovom radu. Ona od funkcije zahtjeva da bude diferencijabilna što *max* funkcija zasigurno nije. Zato ćemo upotrijebiti *softmax* funkciju koja elemente vektora preslikava na raspon $[0, 1]$ čiji je zbroj u konačnici jednak 1. Tako preslikane brojeve ćemo interpretirati kao vjerojatnosti pojedinih kategorija. *Softmax* funkcija generalizacija je logističke (sigmoidalne) funkcije i služi za klasifikaciju u slučaju više klasa pod pretpostavkom da su međusobno iskuljučive. Definirana je slijedećim izrazom:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \tag{3.5}$$

gdje je $\mathbf{z}$ spomenuti vektor, a j indeks kategorije za koju računamo vjerojatnost.
Upravo zato smo željeni izlaz neuronske mreže, koji proizlazi iz skupa podataka za treniranje, oblikovali kao *one-hot* vektor odnosno vektor u kojem je samo jedan element jednak 1 dok su svi ostali postavljeni na 0. Time dobivamo željeni scenarij: za kategoriju kojoj pripada odsječak teksta težimo dobiti siguran događaj dok za sve ostale kategorije želimo da postanu nemoguć događaj uz zadani ulaz. Pri testiranju modela neuronske mreže, kada smo prisiljeni odabrati samo jednu od kategorija, uvijek ćemo odabrati onu s najvećom vjerojatnosti.



Slika 3.4: Primjena *softmax* funkcije pri klasifikaciji

Na slici 3.4 prikazan je primjer *softmax* funkcije. Lijevi stupac $\mathbf{z}$ predstavlja vektor, izračunat u potpuno povezanom sloju, koji predstavlja 5 kategorija. Primjenom funkcije $\sigma(\mathbf{z})$ dobiva se desni stupac sa vjerojatnostima. U ovom slučaju izabrali bismo predzadnju kategoriju čija je vjerojatnost daleko viša od svih ostalih.

3.6. Aktivacijska funkcija

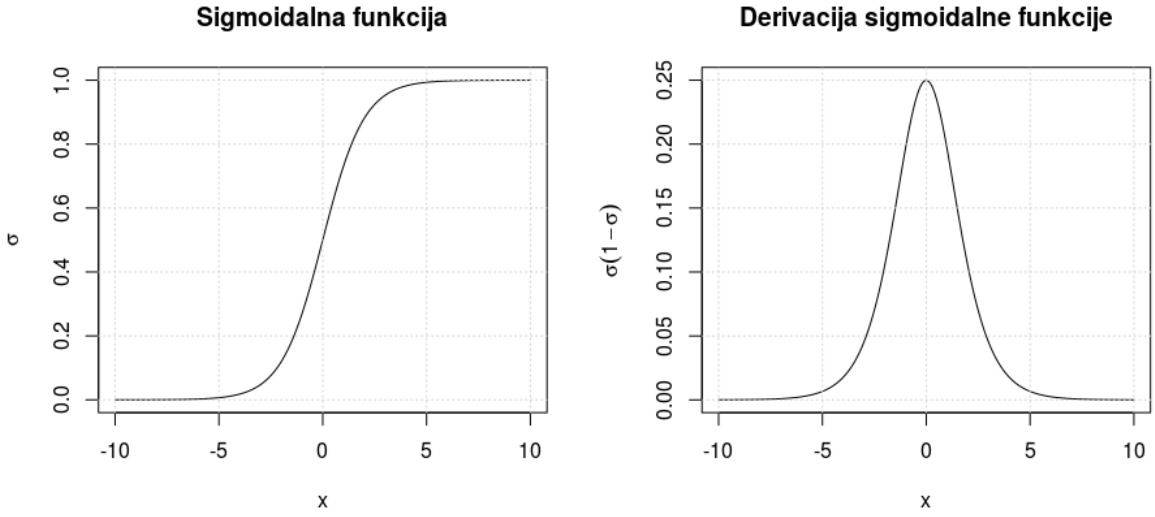
Aktivacijska funkcija čvora određene mreže, u ovom slučaju neurona u neuronskoj mreži, jest funkcija koja za jedan ili više zadanih ulaza određuje vrijednost izlaza. U radu sa neuronskim mrežama aktivacijska funkcija primjenjuje se nad linearnom kombinacijom izlaza neurona iz prethodnog sloja pomnoženih s odgovarajućim težinama. Iako bismo mogli za aktivacijsku funkciju uzeti identitetu ili bilo koju drugu linearnu funkciju, time bismo se samo ograničili na rješavanje skupa problema koji su linearno odvojivi. U praksi, velika većina problema ne može se riješiti linearnim modelom. Zato od aktivacijske funkcije želimo da bude *nelinearna*. Upravo iz tog razloga za aktivacijsku funkciju se koristi još i naziv nelinearnost. Tri najpoznatije aktivacijske funkcije su sigmoidalna funkcija, hiperbolički tangens te ReLU (*eng. Rectified Linear Unit*).

Sigmoidalna funkcija definirana je izrazom $\sigma(x) = \frac{1}{1+e^{-x}}$ i prikazana je na slici 3.5. Kao što je vidljivo, funkcija uzima realne brojeve i suzbija ih u raspon od 0 do 1. Na taj način jako veliki pozitivni brojevi teže u 1 dok jako veliki negativni brojevi teže u 0. Ova funkcija tijekom povijesti često je korištena zbog lagane interpretacije u kontekstu "paljenja" neurona: od neurona koji uopće ne pali (0) do neurona koji pali u potpunom zasićenju (1). Sa praktične strane, sigmoidalna nelinearnost danas se vrlo rijetko koristi zbog slijedećeg razloga. Vrlo neželjeno svojstvo neurona sa sigmoidalnom aktivacijom jest vrijednost derivacije u područjima zasićenja. Derivacija u repovima funkcije ima vrlo malo vrijednost blizu 0 (slika 3.5) što znači da tijekom propagacije pogreške unatrag gotovo nikakva korekcija težina neće biti provedena. Također, inicijalizaciju težina treba provesti vrlo pažljivo zbog mogućnosti zasićenja čime bi neuronska mreža bila onemogućena učiti od samog početka [9].

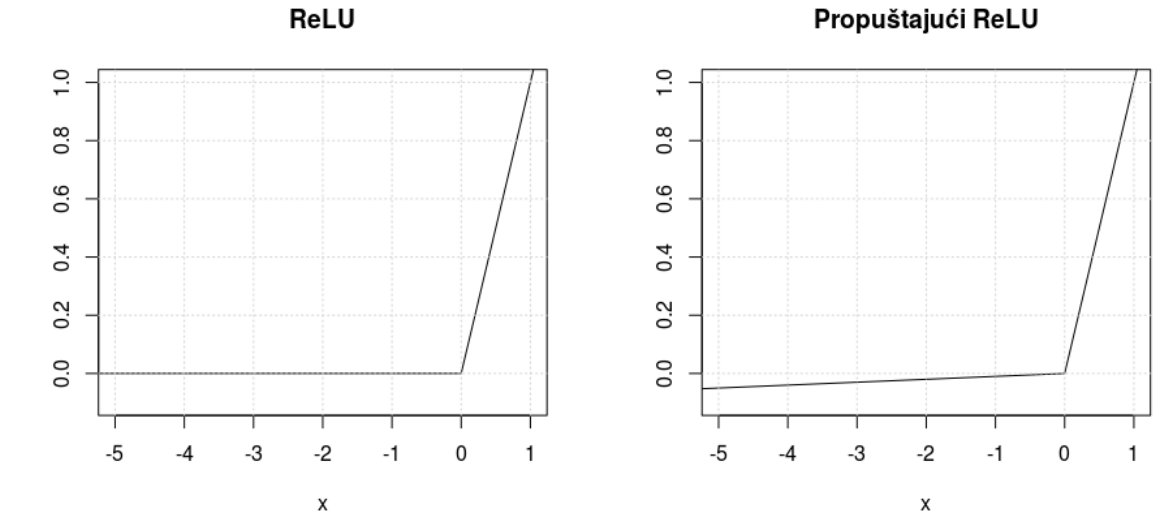
Tangens hiperbolički suzbija vrijednosti u raspon od -1 do 1 te je određen relacijom $th(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Za razliku od sigmoidalne funkcije, centriran je u odnosu na nulu (*eng. zero-centered*). Valja uočiti da je hiperbolički tangens ništa više od transformirane sigmoidalne funkcije ($th(x) = 2\sigma(2x) - 1$) što za sobom povlači slične probleme [9].

U ovome radu koristi se ReLU aktivacijska funkcija koja je postala vrlo popularna u zadnjih par godina. Određena je vrlo jednostavnim izrazom:

$$f(x) = max(0, x) \tag{3.6}$$



Slika 3.5: Prikaz sigmoidalne funkcije i njene derivacije



Slika 3.6: Usporedba ReLU funkcije i propuštajuće inačice

Korištenjem ReLU aktivacijske funkcije uočeno je ubrzanje konvergencije prilikom korištenja stohastičkog gradijentnog spusta [10] u odnosu na sigmoidalnu funkciju i hiperbolički tangens. Također, ova aktivacijska funkcija temelji se na puno efikasnijoj matematičkoj operaciji (max koja se svodi na usporedbu dvije vrijednosti) od sigmoidalne funkcije i hiperboličkog tangensa koje koriste eksponencijalnu funkciju u kombinaciji sa dijeljenjem. Nažalost, ReLU neuroni mogu biti vrlo osjetljivi te "umrijeti" tijekom treniranja neuronske mreže. U slučaju velikog iznosa gradijenta pri korekciji težina, iznosi težina mogu stvoriti stanje nakon kojeg neuron više nikada neće paliti. Postotak mrtvih neurona u ovakvoj sitaciji može doseći čak 40%. Ukoliko smanjimo stopu učenja, u velikoj mjeri možemo smanjiti učestalost ove pojave [9]. Drugi način borbe protiv mrtvih ReLU neurona je modi-

fikacija aktivacijske funkcije pod nazivom propuštajuća ReLU funkcija (*eng. leaky ReLU*), koja umjesto vrijednosti $f(x) = 0$ za $x < 0$ daje vrijednost $f(x) = ax$ gdje je a jako mala proizvoljna konstanta. Usporedba te dvije inačice prikazana je na slici 3.6. Iako neki radovi izvještavaju o poboljšanjima pri korištenju propuštajuće modifikacije, rezultati nisu uvijek konzistentni te se ova inačica ne koristi u ovome radu.

3.7. Inicijalizacija težina

Jedan od važnijih problema pri treniranju neuronske mreže jest inicijalizacija njezinih težina. Pri radu s dubokim neuronskim mrežama, način biranja početnih težina može činiti razliku između konvergencije u razumnom vremenu i oscilacije vrijednosti funkcije pogreške oko vrijednosti daleko od optimuma. Ukoliko su težine premale, varijanca ulaznog signala počinje počinje se smanjivati prolaskom kroz mrežu prema izlaznom sloju. Naposljetku, vrijednost signala pada na vrlo nisku vrijednost od koje više nemamo koristi. S druge strane, ako su težine prevelike, varijanca vrlo brzo raste prolaskom kroz svaki sloj i dolazi do "eksplozije signala". Zato ćemo iskoristiti jednu od "pametnih" metoda inicijalizacije težina.

3.7.1. Xavier

Uobičajeno je da težine inicijaliziramo potpuno nasumično, uzorkujući iz određene razdiobe. Na početku ne znamo ništa od podacima koje ćemo koristiti i nismo sigurni kako određene vrijednosti težina utječu na uspješnost treniranja neuronske mreže. Dobra praska je inicijalizacija iz Gaussove normalne razdiobe s očekivanjem 0 i nekom konačnom varijancom. Pitanje je slijedeće: kako odrediti varijancu kao parametar razdiobe iz koje uzorkujemo?

Želimo da varijanca signala ostane ista kroz sve slojeve neuronske mreže što nam pomaže u sprječavanju premalih i prevelikih vrijednosti. Drugim riječima, potrebno je inicijalizirati težine na način kako bi varijanca ostala ista za ulaz i izlaz svakog sloja neuronske mreže. Ovakav postupak inicijalizacije naziva se Xavier inicijalizacija [11].

Varijanca za inicijalizaciju težina i-tog sloja neuronske mreže određuje se na slijedeći način:

$$Var(W_i) = \frac{1}{n_{i-1} + n_i} \tag{3.7}$$

gdje je W_i matrica težina i-tog sloja, n_{i-1} broj neurona u prethodnom ($(i-1)$-tom) sloju, a n_i broj neurona u trenutnom (i-tom) sloju. Konačno, elementi matrice težina W_i uzorkuju se iz normalne razdiobe $\mathcal{N}\left(0, \frac{1}{n_{i-1}+n_i}\right)$.

3.8. Metoda optimizacije

Metoda optimizacije korištena pri treniranju neuronske mreže u ovom radu jest gradijentni spust (*eng. gradient descent*) uz algoritam propagacije pogreške unatrag (*eng. backpropagation algorithm*), a korekcija težina je provođena koristeći male grupe podataka (*eng. mini-batch*). Za razliku od korekcije težina za svaki primjer te korekcije nakon cijelog skupa za treniranje, korekcija težina koristeći male grupe primjera za treniranje uzima najbolje od obje krajnosti. Na taj način smanjuje se varijanca korekcije težina čime se postiže stabilnija konvergencija. Također, iskorištavaju se vrlo optimizirane operacije nad matricama uključene u razne razvojne biblioteke za duboko učenje. Korištena veličina grupe često varira s obzirom na primjenu. U ovom radu koristi se veličina grupe od 50 primjera za treniranje.

Ipak, klasična implementacija gradijentnog spusta uz korekciju pomoću malih grupa ne garantira konvergenciju. Izazovi s kojima se treba suočiti su slijedeći [12]:

1. Biranje valjanog faktora učenja je teško. Premali faktor učenja rezultira vrlo sporim treniranjem do konvergencije dok preveliki faktor učenja može poremetiti konvergenciju čime greška u znatnijoj mjeri titra oko minimuma ill čak divergira.

2. Klasični postupci s adaptivnim faktorom učenja tijekom faze treniranja određuje strategiju promjene faktora učenja unaprijed. Na takav način ne postoji mogućnost prilagodbe karakteristikama trenutno korištenog skupa podataka.

3. Isti faktor učenja primjenjuje se na sve korekcije težina. Ukoliko su naši podaci raspršeni, a značajke imaju vrlo različite frekvencije pojavljivanja, ne želimo ih ažurirati u istoj mjeri, već napraviti veću korekciju za značajke s rjeđom frekvencijom.

4. Težina izbjegavanja suboptimalnih lokalnih minimuma leži u pojavi sedlastih točaka. U tim točkama gradijent je praktički jednak 0 u svim dimenzijama što gradijentnom spustu jako otežava bijeg iz tog područja.

3.8.1. Adam - Adaptive Moment Estimation

Adam je jedna od mnogih metoda koja računa adaptivni faktor učenja za svaki parametar odnosno težinu. Postupak pohranjuje dvije vrijednosti: eksponencijalno padajuću aritmetičku sredinu prošlih gradijenata (m_t) te njihovih kvadrata (v_t). m_t je procjena prvog momenta (očekivanja) gradijenta, dok je v_t procjena njegovog drugog momenta (varijance) od čega i dolazi ime same metode.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.8}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.9}$$

gdje je g_t iznos gradijenta, a β_1 i β_2 proizvoljno odabrani parametri.
Kako su vrijednosti oba momenta na početku inicijalizirani nulama, uočeno je kako su vrijednosti sklone oscilacijama oko 0. To se događa posebno tijekom početnih koraka te kada su faktori eksponencijalnog pada mali (β_1 i β_2 su jako blizu 1). Kako bi se poništio efekt pristranosti, računaju se korektirane vrijednosti procjene momenata:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.10}$$

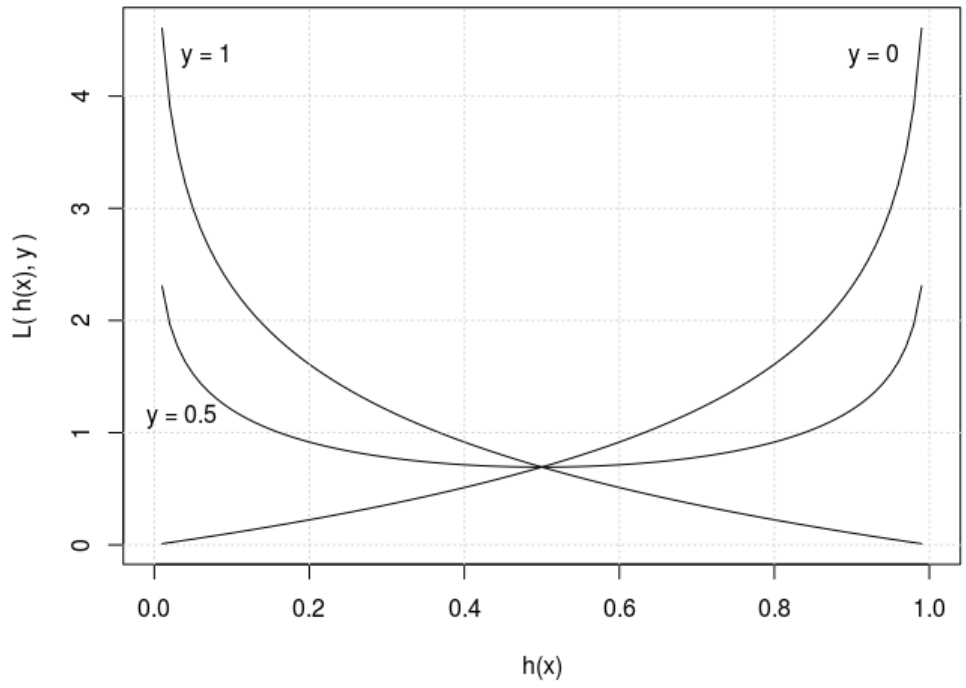
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.11}$$

Tada, korekcija težina provodi se na slijedeći način:

$$W_{i,t+1} = W_{i,t} - \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}\eta \tag{3.12}$$

η je faktor učenja, a ϵ proizvoljan jako mali broj. $W_{i,t+1}$ i $W_{i,t}$ je matrica težina u i-tom sloju u dva vremenska koraka, t i $t + 1$. Autori postupka predlažu vrijednosti od 0.9 za β_1, 0.999 za β_2 te 10^{-8} za ϵ. Iste vrijednosti koriste se i u sklopu razvojne biblioteke TensorFlow kao zadane vrijednosti.

3.9. Funkcija pogreške



Slika 3.7: Funkcija gubitka za različite vrijednosti y

Problem klasifikacije teksta zapravo je općeniti slučaj logističke regresije. U tom kontekstu za funkciju pogreške upotrijebiti ćemo unakrsnu entropiju (*eng. cross-entropy*). Prvo definiramo funkciju gubitka (za jedan par ulaz-izlaz u skupu za treniranje):

$$L\left(h(x), y\right) = -\left[y \cdot ln(h(x)) + (1-y) \cdot ln(1-h(x))\right] \tag{3.13}$$

gdje je $h(x)$ dobiveni izlaz uz trenutni ulaz x, a y željeni izlaz. Na slici 3.7 vidljivo je kako je funkcija gubitka jednaka 0 samo za savršeno točnu klasifikaciju (kada su i $h(x)$ i y jednaki). Funkcija pogreške, odnosno unakrsna entropija, u tom slučaju glasi:

$$E(w) = -\sum_{i=1}^{B} L(h(x^{(i)}|w), y^{(i)}) \tag{3.14}$$

gdje w predstavlja trenutni skup težina, a B veličinu male grupe podataka.

3.10. Regularizacija

Vjerojatno najveći problem neuronskih mreža kao modela korištenih u dubokom učenju je njihova kompleksnost. Gradeći takav jedan kompleksan model, vrlo je lako provesti prilagodbu podacima iz skupa za treniranje, i to uz zanemarivu pogrešku. No, prilikom evaluacije modela na novim podacima, rezultati su jako loši. Drugim riječima, model nema sposobnost generalizacije. Takva pojava naziva se prenaučenost (*eng. over-fitting*) i jedan je od glavnih izazova s kojim se eksperti strojnog učenja susreću. Usporedba dobre i loše generalizacije vidljiva je na slici 3.8 gdje desni graf prikazuje pojavu prenaučenosti.

Slika 3.8: Usporedba dobre i loše generalizacije

Postupak sprječavanja prenaučenosti neuronske mreže naziva se regularizacija. Neke od najčešćih metoda regularizacije su L2 regularizacija, ograničenje maksimalne norme (*eng. max-norm constraint*) te *dropout*. Sve tri metode implementirane su za korištenje po potrebi u ovome radu.

3.10.1. L2 regularizacija

Najčešća metoda regularizacije je L2 regularizacija. Implementacija ove metoda povećava vrijednost funkcije pogreške nadodajući kvadrate iznosa težina, odnosno $\frac{1}{2}\lambda \sum_i |w_i|^2$. w_i je svaka težina u neuronskoj mreži koja podliježe regularizaciji, a λ predstavlja proizvoljno odabran faktor regularizacije. Često se uočava faktor $\frac{1}{2}$ ispred sume. Konstanta je odabrana isključivo iz razloga računanja gradijenta gdje pri derivaciji članova sume bude poništena. U tom slučaju gradijent iznosi λw_i umjesto $2\lambda w_i$. Pošto se metodom optimizacije nastoji minimizirati funkcija pogreške, L2 regularizacijom se na taj način penaliziraju se ekstremne vrijednosti težina. Valja također primjetiti da zbog prisutstva člana λw_i u gradijentu svaka težina linearno pada prema nuli. U ovom radu L2 regularizacija implementirana je za korištenje se u potpuno povezanom sloju [13].

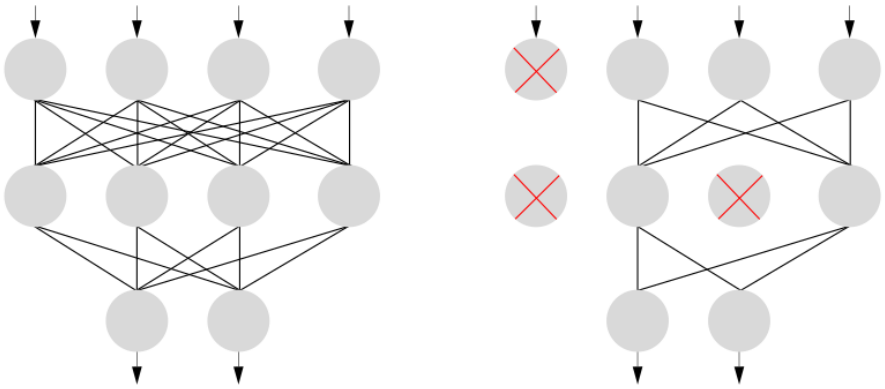
3.10.2. Ograničenje maksimalne norme

Još jedna vrsta regularizacije je ograničenje maksimalne norme. Ova metoda nastoji ograničiti vektor težina za svaki neuron omeđujući njegov iznos odozgo. U praksi se provodi na način da se izvrši korekcija težina kao inače te se nakon toga vrijednosti vektora w ograniče kako bi vrijedilo $||w||_2 \leq c$. Preciznije, svaki puta kada za vektor težina nakon korekcije vrijedi $||w||_2 > c$, vektor projiciramo nazad na kuglu radijusa c sa centrom u ishodištu. Tipične vrijednosti ograničenja c iznose oko 3 i 4. Jedno od primamljivih svojstava ove metode jest činjenica da signal u mreži ne može "eksplodirati" čak i u slučaju da je faktor učenja η prevelik pošto su iznosi korekcija težina ograničeni [13]. Što se tiče ovog rada, ograničenje maksimalne norme implementirano je nad težinama konvolucijskih filtara.

3.10.3. *Dropout*

U zadnje vrijeme vrlo popularna i uspješna metoda regularizacije dolazi od autora Srivastava et al. (2014) [14] pod nazivom *dropout*. Iako vrlo drukčija metoda od svih prethodnih, ideja je vrlo jednostavna. Tijekom treniranja neuronske mreže, svaki neuron aktivan je sa zadanom vjerojatnosti p, a u suprotnom je isključen (sa vjerojatnosti $1 - p$). Ova vrsta regularizacije može se interpretirati kao uzorkovanje nove neuronske mreže iz skupa neurona i težina početne neuronske mreže pri čemu se korekcije provode samo za uzorkovane težine. Intuitivno, provođenje regularizacije na ovaj način tjera neuronsku mrežu da bude precizna pri nedostatku određenih informacija. Također, sprječava se prenaučenost kombinirajući eksponencijalno mnogo (2^n gdje je n ukupan broj neurona) različitih arhitektura neuronske mreže [13]. Tijekom testiranja neuronske mreže *dropout* se ne primjenjuje ($p = 1$). U kontekstu ovog rada, *dropout* se koristi izravno nad *max-pooling* slojem.

Na slici 3.9 pokazan je primjer u kojem su nakon provedene *dropout* regularizacije tri neurona isključena, a korekcija se provodi za preostalih deset težina.



Slika 3.9: *Dropout* regularizacija

4. Skupovi podataka

Skupovi podataka (*eng. datasets*) služe za nadzirano treniranje te testiranje modela neuronske mreže. Svaki primjer unutar skupa podataka sastoji se od ulaza te željenog izlaza iz neuronske mreže. Radi usporedbe performansi iste arhitekture neuronske mreže pri različitim zadacima klasifikacije (različitim skupovima kategorija), koristimo više različitih skupova podataka.

4.1. TREC

TREC [1] je skup podataka koji sadrži pitanja te kategorije kojima odgovori na dotična pitanja pripadaju [15]. Sastoji se od sljedećih kategorija:

```
abbreviation (ABBR): skraćenica
entity (ENTY): entitet
description (DESC): opis
human (HUM): čovjek
location (LOC): mjesto
numeric (NUM): broj
```

Podskup podataka za treniranje sadrži 5452 primjera dok podskup podataka za testiranje se sastoji od 500 primjera. Najveća duljina odsječka teksta iznosi 33 riječi.

4.2. Reuters-21578 R8

Reuters-21578 [2] [3] (distribucija 1.0) trenutno je najkorištenija kolekcija u istraživanju klasifikacije teksta. Skup podataka izvorno su prikupili i labelirali Carnegie Group, Inc. i Reuters, Ltd. tijekom razvoj CONSTRUE sustava za klasifikaciju teksta. Zbog činjenice da je distribucija dokumenata po kategorijama vrlo nakrivljena, često se razmatraju slijedeća dva podskupa:

[1] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[2] http://www.daviddlewis.com/resources/testcollections/reuters21578

[3] http://ana.cachopo.org/datasets-for-single-label-text-categorization

– R10: skup od 10 kategorija s najvećim brojem pozitivnih primjera za treniranje

– R90: skup od 90 kategorija s najmanje jednim pozitivnim primjerom za treniranje i testiranje

Pošto je naš zadatak klasifikacija dokumenata sa samo jednom označenom kategorijom, svi dokumenti sa manje ili više od jedne kategorije su izbačeni. Prema tome, neke kategorije iz R10 i R90 skupova ostale su bez ijednog dokumenta za treniranje ili testiranje. Time smo dobili 8 od 10 kategorija (R10) te 52 od početnih 90 kategorija (R90). Dva novodobivena podskupa nazivaju se R8 i R52 [16]. U ovom radu se radi jednostavnosti koristi podskup R8 koji se sastoji od slijedećih kategorija:

```
acq
crude
earn
grain
interest
money-fx
ship
trade
```

Ovaj podskup sadrži 5485 primjera za treniranje te 2189 primjera za testiranje. Najveća duljina odsječka teksta iznosi 523 riječi, uz izbačene zaustavne (*eng. stop*) riječi engleskog jezika. Zaustavne riječi predstavljaju najčešće i redundantne riječi određenog jezika, no treba uzeti u obzir da ne postoji univerzalan popis pošto svaki alat za obradu prirodnog jezika koristi vlastiti. Skup podataka preuzet je već s izbačenim zaustavnim riječima pa nemamo uvid koje su točno izbačene.

4.3. Hrvatski portali

Hrvatski portali osobno je prikupljeni skup podataka koristeći vlastito napisani *web crawler* alat u programskom jeziku Python. Skup podataka sadrži članke s raznih hrvatskih internet portala: 24sata, Index.hr, Dnevno.hr, Večernji list te Jutarnji list. Članci su prikupljani po kategorijama posebno za svaki internet portal s vijestima kroz vremenski period od mjesec dana. Skup podataka izgrađen je presjekom kategorija portala tako da su srodne kategorije spojene u jednu zajedničku. Ostale kategorije koje se ne poklapaju su izostavljene. Detaljnija građa skupa podataka dana je u tablici 4.1 gdje oznake + predstavljaju incidentne kategorije.

Iz tablice je također vidljiva ogromna nakrivljenost distribucije članaka po kategorijama. Kao pokušaj rješenja problema nebalansiranih klasa, iskoristili smo postupak nasumičnog poduzorkovanja (*eng. under-sampling*) onih klasa koje sadrže veliki broj dokumenata. Pri

nasumičnom poduzorkovanju svaka kategorija ograničena je na najviše 3000 odsječaka teksta. Za podjelu odsječaka na podskup za treniranje te podskup za testiranje, korišten je omjer 80:20 također nasumično odabranih primjera. U procesu obrade tekstualnih odsječaka prije ulaza u neuronsku mrežu, izbačene su najčešće zaustavne riječi (preuzete sa stranice Ranks NL [4]) te su za sve riječi iskorištena mala slova.

Tijekom treniranja modela nad ovim skupom podataka, stvoren je novi skup vektorskih reprezentacija riječi (čija je veličina ograničena hiperparametrima) koji je nasumično inicijaliziran ranije opisanom Xavier metodom. U tom slučaju, za razliku od engleskog jezika i preuzetih javno dostupnih vektora, nastojimo naučiti vektorske reprezentacije riječi hrvatskog jezika "od nule".

Tablica 4.1: Građa skupa podataka *Hrvatski portali*

	Hrvatski portali				
	vijesti	sport	show	novac	kultura
24sata					
news	+				
sport		+			
show			+		
Index.hr					
vijesti	+				
sport		+			
novac				+	
Večernji list					
vijesti	+				
sport		+			
showbiz			+		
biznis				+	
kultura					+
Jutarnji list					
vijesti	+				
spektakli			+		
biznis				+	
kultura					+
Dnevno.hr					
vijesti	+				
sport		+			
novac				+	
Broj članaka	18300	7852	4649	2817	1374

[4] http://www.ranks.nl/stopwords/croatian

5. Eksperimentalni rezultati

5.1. TensorFlow uz podršku grafičkog sklopovlja

TensorFlow je programska biblioteka za dizajn i razvoj numeričkog računa s glavnim fokusom na primjenu u strojnom učenju. Biblioteka omogućava opis algoritama pomoću grafova u kojem svaki čvor predstavlja određenu operaciju. Navedene operacije mogu se izvršavati na različitim platformama, od prijenosnih uređaja i kućnih računala do visokobudžetnih poslužitelja, uz podršku grafičkog sklopovlja.

U ovom radu korištena je TensorFlow biblioteka sa CUDA podrškom za NVIDIA grafičke kartice čime se znatno ubrzalo treniranje modela neuronskih mreža u odnosu na korištenje isključivo samo glavnog procesora računala. Prilikom dobivanja eksperimentalnih rezultata korištene su tri grafičke kartice na tri različita računala, prikazane u tablici 5.1.

Tablica 5.1: Korištene grafičke kartice s pripadajućim brojem CUDA jezgri

Računalo	Grafička kartica	Broj CUDA jezgri
1	GeForce GTX TITAN X	3072
2	GeForce GTX 970	1664
3	GeForce GTX 980M	1536

5.2. Opis hiperparametara

Slijedeći parametri korišteni su pri implementaciji modela:

- **BATCH_SIZE** - veličina grupe podataka nakon koje se provodi korekcija težina
- **DROPOUT_PROB** - vjerojatnost izbacivanja neurona u *dropout* regularizaciji ($1 - p$)
- **LEARNING_RATE** - stopa učenja (η)
- **MAX_L2_NORM** - maksimalna L2 norma za elemente konvolucijskih filtara (0 znači isključeno)

- **NUM_EPOCHS** - broj epoha za treniranje
- **NUM_FILTERS** - broj filtara po svakoj zadanoj veličini
- **REGION_SIZES** - zadane veličine filtara, odvojene zarezom
- **REG_LAMBDA** - faktor L2 regularizacije u potpuno povezanom sloju (λ)
- **STATIC_EMBEDDINGS** - zastavica koja određuje primjenjuju li se ili ne korekcije elemenata preuzetih Word2vec vektora engleskog jezika

Za hrvatski jezik i skup podataka *Hrvatski portali* korišteni su dodatni parametri:

- **MAX_DOCUMENT_SIZE** - najveća duljina za poravnavanje odsječaka teksta
- **VECTOR_DIM** - dimenzija vektorskih reprezentacija riječi
- **VOCABULARY_SIZE** - broj riječi za koje će se izgraditi vektor u rječniku

5.3. TREC

Tijekom treniranja modela neuronske mreže za TREC skup podataka, radi provjere rezultata korišteni su identični parametri kao u radu Kim (2014) [5]:

```
BATCH_SIZE: 50
DROPOUT_PROB: 0.5
LEARNING_RATE: 0.0003
MAX_L2_NORM: 0.0
NUM_EPOCHS: 250
NUM_FILTERS: 100
REGION_SIZES: 3,4,5
REG_LAMBDA: 0.0
STATIC_EMBEDDINGS: True
```

Rezultati su prikazani u tablici 5.2 zajedno s utjecajem različitog broja epoha za treniranje na točnost modela neuronske mreže. U nastavku su navedeni primjeri klasifikacije pitanja s prikazanim kategorijama te pripadajućim vjerojatnostima. Zvjezdica predstavlja kategoriju s najvećom vjerojatnosti.

```
> What does FER stand for?
    ABBR          100.00 % *
    ENTY            0.00 %
    DESC            0.00 %
    HUM             0.00 %
    LOC             0.00 %
    NUM             0.00 %
```

```
> What is the name of the satellite that the Soviet Union
sent into space in 1957?
    ABBR            0.00 %
    ENTY           99.40 % *
    DESC            0.00 %
    HUM             0.02 %
    LOC             0.58 %
    NUM             0.00 %


> What is machine learning?
    ABBR            0.00 %
    ENTY            0.00 %
    DESC          100.00 % *
    HUM             0.00 %
    LOC             0.00 %
    NUM             0.00 %


> In the late 1700's British convicts were used to
populate which colony?
    ABBR            0.00 %
    ENTY            1.00 %
    DESC            0.00 %
    HUM            18.76 %
    LOC            75.54 % *
    NUM             4.71 %
```

5.4. Reuters 21578 R8

Za Reuters 21578 R8 skup podataka najprije je provedeno linearno traženje optimalne veličine filtra (tablica 5.3). Pri tome su korišteni slijedeći hiperparametri:

```
BATCH_SIZE: 50
DROPOUT_PROB: 0.5
LEARNING_RATE: 0.0003
MAX_L2_NORM: 0.0
NUM_EPOCHS: 500
NUM_FILTERS: 100
REGION_SIZES: 3,4,5
REG_LAMBDA: 0.0
```

```
STATIC_EMBEDDINGS: True
```

Nakon utvrđivanja optimalne veličine filtra, koja u ovom slučaju iznosi 16, isproban je model koji koristi veličine filtara iz okoline (14, 15 i 16), ali puno manji broj filtara po veličini zbog računalnih ograničenja. Tablica 5.4 pokazuje prosječno lošiji rezultat u odnosu na jedinstvenu veličinu filtra. To ukazuje na potrebu ispitivanja većih veličina filtara te većeg broja filtara po veličini u budućnosti, u slučaju posjedovanja dovoljno resursa za takvu pretragu što pri izradi ovog rada nije bio slučaj. Pritom su korišteni slijedeći hiperparametri:

```
BATCH_SIZE: 50
DROPOUT_PROB: 0.5
LEARNING_RATE: 0.0003
MAX_L2_NORM: 0.0
NUM_EPOCHS: 500
NUM_FILTERS: 16
REGION_SIZES: 14,15,16
REG_LAMBDA: 0.0
STATIC_EMBEDDINGS: True
```

5.5. Hrvatski portali

Treniranje modela neuronske mreže nad hrvatskim jezikom provedeno korištenjem slijedećih hiperparametara:

```
BATCH_SIZE: 50
DROPOUT_PROB: 0.5
LEARNING_RATE: 0.0003
MAX_DOCUMENT_SIZE: 0
MAX_L2_NORM: 0
NUM_EPOCHS: 30
NUM_FILTERS: 32
REGION_SIZES: 3,4,5
REG_LAMBDA: 0.001
VECTOR_DIM: 100
VOCABULARY_SIZE: 100000
```

Navedeni parametri izabrani su na temelju principa pokušaja i pogreške. Rezultati su dostupni u tablici 5.5. Također, na slici 5.1 prikazano je kretanje prosječnog gubitka po epohi.

Nije utvrđeno koliko su navedeni parametri daleko od optimalne konfiguracije što je sigurno dobro polazište za slične radove ubuduće. Slijedi nekoliko primjera kategorizacije članaka.

```
> Hoće li Adrian Šemper novu sezonu dočekati u Engleskoj? Naime,
mladi je vratar jučer otputovao u London, neslužbeno, pregovore s
nekim od tamošnjih klubova. Tajnovito je zasad o kome se radi,
zna se da je prije godinu i pol dana zbog njega u Zagreb došao
Chelsea, ponudio u startu 2,5 milijuna eura, ali Dinamo je tražio
osam. Šemper je odlaskom Eduarda preuzeo Dinamovu 'jedinicu'
početkom sezone, no, zaredali su loši rezultati, a mladi vratar
postao je jedna od žrtva krize, te je poslan na posudbu u Lokomotivu,
gdje je na proljeće branio jako dobro. Još uvijek je mlad, tek mu
je 19 godina i svi ga svjetski mediji svrstavaju među najbolje
vratare na kontinentu, a veliki ga klubovi i dalje prate. Dinamo
trenutačno ne računa sa Šemperom, Livaković i Zagorac su ispred
njega, pa ne bi trebalo čuditi ako dođe do transfera. Šemper je i
dalje tražena roba, a hoće li boravak u Londonu biti samo izlet za
mladog vratara ili će se završiti transferom, ostaje pričekati...
(izvor: sportske.jutarnji.hr)
    kultura        15.88 %
    vijesti        17.72 %
    novac          12.05 %
    show           18.64 %
    sport          35.71 % *
```

```
> 38-godišnja glumica Katie Holmes upisala je studij te pohađa program
"Biznis u medijima, sportu i zabavnoj industriji" na prestižnom
Harvardu. S Katie su u klupama i Shakirin muž nogometaš Gerard Pique,
ragbijaš Jamie Heaslip, košarkaš CJ McCollum i bivši nogometaš Rashean
Mathis. Poznata profesorica Anita Elberse predavat će slavnoj glumici,
ali i poznatim sportašima pa je tako objavila i njihovu zajedničku
fotografiju.Katie Holmes, koja ima vlastitu produkcijsku kompaniju
Noelle Productions, objavila je na Instagramu fotografiju iz "školskih"
klupa, a i Pique se pohvalio sudjelovanjem na nastavi. "Zahvalna sam i
uzbuđena što sam na Harvardu s tolikim briljantnim ljudima", napisala je
Katie. Program je očito jako popularan kod celebrityja. Već su ga završili
manekenka Tyra Banks, reper LL Cool J i glumac Channing Tatum, koji će
ovogodišnjoj generaciji doći na predavanje i pričati o tome što su novo
naučili i kako su unaprijedili svoj posao otkad su diplomirali.
(izvor: dnevnik.hr)
    kultura         0.31 %
    vijesti         0.01 %
    novac           0.01 %
    show           99.67 % *
    sport           0.00 %
```

> Vrijednost najvećih stotinu svjetskih brendova ove je godine veća za osam posto na 3,6 bilijuna dolara, a Google je uspio obraniti lanjsku titulu. Kako stoji u BrandZ Global Top 100 redovitom godišnjem istraživanju marketinških agencija WPP i Kantar Millward Brown, Google ove godine vrijedi 245,6 milijardi dolara ili sedam posto više nego lani. Među prvih pet još su Apple, Microsoft, Amazon i Facebook. Inače, među prvih 10 čak je devet američkih kompanija, a jedini "uljez" je kineski internetski div Tencent. Dominaciju američkih brendova potvrđuje i raspodjela po regijama. Prvih 10 sjevernoameričkih brendova vrijedi 12 posto više na 1,4 bilijuna dolara dok 10 najvećih europskih brendova vrijedi 274,4 milijarde dolara ili četiri posto više. Prvih 10 iz Azije vrijedi 391,1 milijardu dolara ili devet posto više. Dominaciju tehnoloških tvrtki potvrđuju i novopridošli. Prvi puta među najvećih 100 ušli su Xfinity, YouTube, Hewlett Packard Enterprise, Salesforce, Netflix i Snapchat te telekom operator Sprint. Ipak, najveći porast bilježi kompanija iz "staromodne" branše – Adidasov brend sada vrijedi 58 posto više na 8,3 milijarde dolara. Koliko se svijet ubrzano mijenja pokazuje i podatak da se ove godine među najvećih 10 nalaze samo tri kompanije – Google, Microsoft i IBM – koje su se i 2006. nalazile u tom elitnom društvu. U posljednjih 12 godina vrijednost stotinu brendova skočila je 152 posto, piše Poslovni dnevnik.
(izvor: hr.n1info.com / poslovni.hr)

kultura	0.40 %	
vijesti	9.13 %	
novac	88.39 %	*
show	0.33 %	
sport	1.74 %	

Tablica 5.2: TREC - Utjecaj broja epoha na točnost

	Broj epoha za treniranje		
	200	250	500
	92.60%	92.40%	92.80%
	93.00%	93.40%	92.80%
	93.20%	93.20%	92.60%
	92.40%	92.60%	93.80%
	93.20%	93.20%	93.20%
	92.60%	93.40%	92.60%
	92.80%	93.20%	92.40%
	91.80%	93.00%	94.00%
	92.80%	93.00%	93.00%
	91.80%	92.60%	93.00%
	92.20%	92.80%	93.80%
	93.00%	93.20%	93.00%
	93.40%	92.80%	92.00%
	93.60%	93.00%	93.60%
	93.00%	93.20%	93.20%
	92.80%	93.80%	94.00%
	92.60%	93.20%	92.60%
	93.00%	93.20%	92.40%
	92.60%	93.20%	92.40%
	93.60%	92.80%	91.40%
$\bar{x}$	92.80%	**93.06**%	92.93%
σ	4.90E-03	3.17E-03	6.64E-03
max	93.60%	93.80%	94.00%
min	91.80%	92.40%	91.40%

Tablica 5.3: Reuters 21578 R8 - ispitivanje optimalne veličine filtra

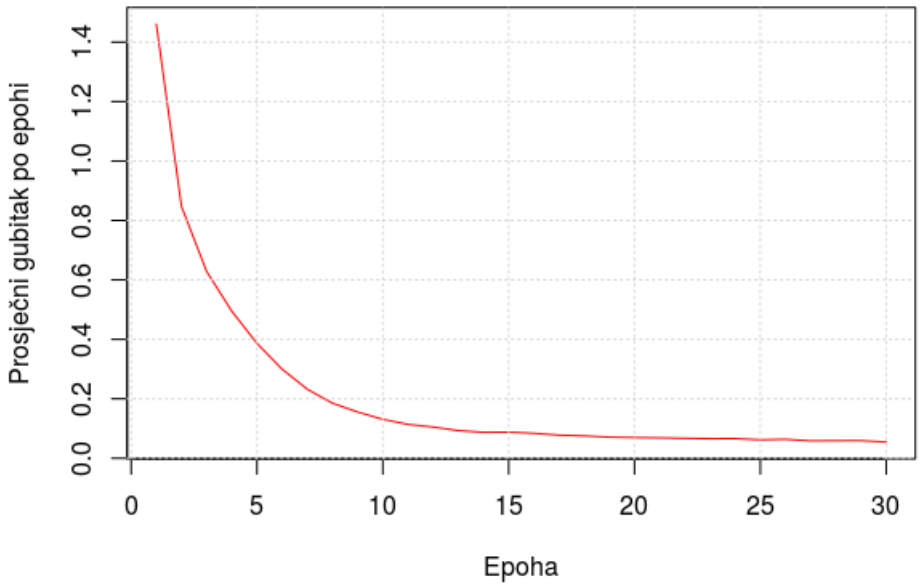
	Veličina filtra						
	3	4	5	6	7	8	9
	94.06%	94.97%	94.97%	95.34%	95.52%	95.39%	95.66%
	94.47%	95.20%	95.29%	95.57%	95.34%	95.75%	96.03%
	94.52%	95.07%	95.80%	95.80%	96.12%	95.98%	95.84%
	94.02%	95.48%	95.07%	95.07%	96.07%	95.93%	96.12%
	94.20%	95.48%	95.20%	95.48%	96.03%	95.71%	95.52%
	94.66%	95.43%	95.02%	95.43%	95.48%	95.66%	95.66%
	94.20%	95.29%	95.57%	95.48%	95.61%	95.52%	95.61%
	94.61%	94.97%	95.52%	95.20%	95.84%	95.34%	96.03%
	94.06%	95.02%	95.16%	95.75%	95.75%	95.80%	95.75%
	94.61%	95.20%	95.20%	96.07%	95.80%	95.98%	96.12%
$\bar{x}$	94.34%	95.21%	95.28%	95.52%	95.76%	95.71%	95.83%
σ	2.57E-03	2.02E-03	2.67E-03	2.96E-03	2.66E-03	2.32E-03	2.23E-03
max	94.66%	95.48%	95.80%	96.07%	96.12%	95.98%	96.12%
min	94.02%	94.97%	94.97%	95.07%	95.34%	95.34%	95.52%
	10	11	12	13	14	15	16
	96.21%	96.07%	95.80%	96.30%	96.30%	95.84%	96.53%
	96.44%	95.98%	96.25%	96.53%	96.21%	96.57%	96.62%
	95.89%	96.12%	96.48%	96.21%	96.80%	96.16%	96.35%
	95.80%	96.07%	96.07%	95.84%	96.30%	95.98%	96.30%
	96.30%	96.25%	96.16%	96.35%	96.30%	96.35%	96.03%
	95.80%	95.80%	96.30%	96.44%	95.93%	96.03%	96.21%
	96.12%	96.12%	95.66%	96.44%	96.57%	96.44%	96.57%
	95.52%	96.07%	96.44%	96.30%	96.57%	96.57%	96.80%
	96.16%	96.21%	96.30%	96.39%	96.25%	96.44%	96.53%
	95.93%	95.71%	95.89%	96.62%	95.93%	96.07%	96.57%
$\bar{x}$	96.02%	96.04%	96.14%	96.34%	96.32%	96.24%	**96.45**%
σ	2.76E-03	1.71E-03	2.76E-03	2.11E-03	2.74E-03	2.62E-03	2.28E-03
max	96.44%	96.25%	96.48%	96.62%	96.80%	96.57%	96.80%
min	95.52%	95.71%	95.66%	95.84%	95.93%	95.84%	96.03%

Tablica 5.4: Reuters 21578 R8 - ispitivanje okoline optimalne veličine filtra paralelno na dva računala

	Točnost	Vrijeme treniranja	
	95.75%	76 m 34 s	Računalo 1
	96.07%	76 m 33 s	
	96.30%	76 m 35 s	
	96.44%	76 m 0 s	
	96.16%	76 m 32 s	
	96.16%	76 m 35 s	
	96.39%	76 m 35 s	
	96.48%	76 m 6 s	
	95.89%	76 m 35 s	
	96.12%	76 m 7 s	
	96.03%	60 m 23 s	Računalo 2
	96.16%	60 m 25 s	
	95.89%	63 m 16 s	
	95.89%	61 m 43 s	
	95.93%	61 m 39 s	
	96.07%	60 m 18 s	
	95.89%	60 m 42 s	
	96.12%	61 m 22 s	
	96.53%	60 m 32 s	
	96.44%	60 m 32 s	
$\bar{x}$	96.14%		
σ	2.30E-03		
max	96.53%		
min	95.75%		

Tablica 5.5: Hrvatski portali - rezultat treniranja modela nad hrvatskim jezikom

	Točnost		
	81.43%	79.39%	82.87%
	81.24%	81.32%	79.69%
	80.60%	80.67%	80.37%
	80.03%	81.47%	81.81%
	79.84%	81.62%	81.36%
	81.70%	80.67%	81.13%
	80.26%	80.45%	81.09%
	81.05%	81.51%	82.76%
	80.30%	80.22%	79.99%
	80.56%	79.84%	80.98%
$\bar{x}$	80.87%		
σ	8.41E-03		
max	82.87%		
min	79.39%		



Slika 5.1: Hrvatski portali - kretanje prosječnog gubitka po epohi

6. Zaključak

Cilj ovog rada bio je prikazati primjenjivost dubokih neuronskih mreža na analizu teksta, točnije na klasifikaciju odsječaka teksta raznih duljina. Odabrana je jedna arhitektura konvolucijske neuronske mreže koja je onda trenirana te testirana na tri različita skupa podataka, od kojih su dva na engleskom, a jedan na hrvatskom jeziku. Za engleski jezik korištene su javno dostupne vektorske reprezentacije riječi dok ih se za hrvatski jezik nastojalo naučiti iz korištenog skupa podataka. Iako je dotična arhitektura izvorno dizajnirana za problem klasifikacije rečenica, u ovom radu ispitana je njezina primjenjivost i na dulje odsječke teksta. Za TREC skup podataka potvrđena je točnost od ~93% kao i u radu Kim (2014) [5]. Kod puno duljih odsječaka teksta u sklopu Reuters 21578 R8 skupa podataka rezultati su još i bolji (točnost od ~96%). Takav rezultat slijedi iz toga da se klasifikacija pitanja temelji prvenstveno na zaustavnim riječima (*what, how, where, who, what...*) koje kod TREC skupa podataka nisu izbačene. Pitanja su općenito kratki odsječci teksta što povećava udio redundantnih riječi. S druge strane, iz odsječaka teksta Reuters 21578 R8 skupa podataka izbačene su zaustavne riječi nakon čega su preostale riječi bitnije za samu kategoriju. Dodatno, duljina pojedinog odsječka teksta u prosjeku je puno veća od prosječnog pitanja čime se neuronskoj mreži predočava veća količina informacija temeljem koje je moguće napraviti jasniju klasifikaciju. Što se tiče hrvatskog jezika te klasifikacije nad skupom podataka Hrvatski portali, točnost modela očekivano je nešto lošija (~81%). No, uzevši u obzir relativno mali skup podataka s velikom nakrivljenosti distribucije, performanse modela su prilično zadovoljavajuće. Za razliku od engleskog jezika čije su vektorske reprezentacije učene na milijardama riječi, za hrvatski jezik pokušali smo naučiti vektorske reprezentacije, a ujedno i pravilno klasificirati članke po kategorijama iz jednog jedinog specifičnog skupa podataka. Pritom smo čak morali koristiti i poduzorkovanje radi zadržavanja ravnoteže u kategorijama. Valja također uzeti u obzir činjenicu da hrvatski jezik koristi složeniju sintaksu i semantiku od engleskog jezika. Prema tome, zaključujemo kako kvalitetno naučene vektorske reprezentacije riječi u sebi nose ključne informacije o određenom jeziku te u velikoj mjeri mogu povećati točnost modela u obradi prirodnog jezika. Učenjem vektorskih reprezentacija riječi hrvatskog jezika na većem tekstualnom korpusu (npr. na Hrvatskom nacionalnom korpusu), prije rješavanja samog problema klasifikacije, zasigurno bi se riješio veliki dio navedenih problema.

LITERATURA

[1] Ben Walker. The Impact of Big Data on our Everyday Lives - Infographic, 2015. URL https://www.vouchercloud.net/resources/big-data-infographic.

[2] Seth Grimes. Unstructured data and the 80 percent rule, 2008. URL https://breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule.

[3] Google. Vector Representations of Words, 2016. URL https://www.tensorflow.org/tutorials/word2vec.

[4] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL http://arxiv.org/abs/1301.3781.

[5] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

[6] Benjamin Graham. Fractional max-pooling. *CoRR*, abs/1412.6071, 2014. URL http://arxiv.org/abs/1412.6071.

[7] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014. URL http://arxiv.org/abs/1412.6806.

[8] Ye Zhang and Byron C. Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *CoRR*, abs/1510.03820, 2015. URL http://arxiv.org/abs/1510.03820.

[9] Andrej Karpathy. Neural networks part 1: Setting up the architecture, 2015. URL https://cs231n.github.io/neural-networks-1/.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges,

L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[11] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. 9:249–256, 13–15 May 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

[12] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2016. URL http://sebastianruder.com/optimizing-gradient-descent/index.html.

[13] Andrej Karpathy. Neural networks part 2: Setting up the data and the loss, 2015. URL https://cs231n.github.io/neural-networks-2/.

[14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[15] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072378. URL http://dx.doi.org/10.3115/1072228.1072378.

[16] Ana Cardoso-Cachopo. Improving Methods for Single-label Text Categorization. PdD Thesis, Instituto Superior Tecnico, Universidade Tecnica de Lisboa, 2007.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Analiza teksta pomoću dubokih neuronskih mreža

Sažetak

Cilj ovog rada je opisati primjenu dubokih neuronskih mreža u klasifikaciji odsječaka teksta različite duljine s obzirom na jednu pripadajuću kategoriju. U prvom dijelu objašnjavaju se motivacija za korištenje modela u vektorskom prostoru te osnovni koncepti u pozadini Word2vec modela. Predstavljena je i detaljno analizirana korištena arhitektura konvolucijske neuronske mreže s naglaskom na inicijalizaciju težina, aktivacijsku funkciju, metodu optimizacije, funkciju pogreške te regularizaciju. U drugom dijelu opisani su korišteni skupovi podataka za treniranje modela neuronske mreže te su navedeni eksperimentalni rezultati dobiveni koristeći TensorFlow biblioteku u okviru programskog jezika Python.

Ključne riječi: strojno učenje, duboko učenje, neuronske mreže, duboke neuronske mreže, umjetna inteligencija, analiza teksta, konvolucija, obrada prirodnog jezika, Word2vec, klasifikacija, treniranje, model, TensorFlow, Python

Text Analysis Using Deep Neural Networks

Abstract

The aim of this paper is to describe the use of deep neural networks in the single-label classification of various length texts. In the first part, the motivation for vector space models and the basic concepts behind the Word2vec model are explained. Then the used convolutional neural network architecture is analyzed in detail, with emphasis on weights initialization, activation function, optimization method, error function and regularization. In the second part, the datasets used to train the neural network model are described and experimental results are reported. The results were obtained using the TensorFlow library and the Python programming language.

Keywords: machine learning, deep learning, neural networks, deep neural networks, artificial intelligence, text analysis, convolution, natural language processing, Word2vec, classification, training, model, TensorFlow, Python