UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

ZAKLJUČNA NALOGA

POSTAVITEV SISTEMA ZA STROJNO PREVAJANJE NA OSNOVI NEVRONSKIH MREŽ

GORAN TUBIĆ

TUBIĆ 2018

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Postavitev sistema za strojno prevajanje na osnovi nevronskih mrež

(Setting up a machine translation system based on neural networks)

Ime in priimek: Goran Tubić
Študijski program: Računalništvo in informatika
Mentor: doc. dr. Jernej Vičič
Somentor: doc. dr. Branko Kavšek

Koper, september 2018

Ključna dokumentacijska informacija

Ime in PRIIMEK: Goran TUBIĆ

Naslov zaključne naloge: Postavitev sistema za strojno prevajanje na osnovi nevronskih mrež

Kraj: Koper

Leto: 2018

Število listov: 35 Število slik: 15 Število tabel: 4

Število prilog: 1 Št. strani prilog: 5 Število grafov: 1

Število referenc: 11

Mentor: doc. dr. Jernej Vičič

Somentor: doc. dr. Branko Kavšek

Ključne besede: RNN, nevronska mreža, strojno prevajanje, prevajanje, prevajalnik, NMT, strojno prevajanje na osnovi nevronskih mrež, spletni vmesnik, spletna stran

Izvleček:

V zaključni nalogi je opisan postopek razvoja sistema za strojno prevajanje na osnovi nevronskih mrež. V prvem delu se naloga osredotoči na predstavitev delovanja nevronske mreže ter opis postopkov pretvarjanja pridobljenega vhoda v izhod. Sledi opis faz pri prevajanju na osnovi nevronskih mrež. V tretjem delu naloga opiše Neural Monkey orodje, katero je uporabljeno za končni izdelek. Neural Monkey orodje je osredotočeno predvsem na prejemanje vhoda, pripravo podatkov in procesiranje le teh, za podajo izhoda. V četrtem delu sledita opis namestitve vmesnika za komuniciranje z Neural Monkey orodjem in analiza pridobljenih rezultatov, glede na opravljene teste. Zaključek zajema povzetek naloge in predstavitev idej za nadaljnje delo.

Key words documentation

Name and SURNAME: Goran TUBIĆ

Title of the final project paper: Setting up a machine translation system based on neural networks

Place: Koper

Year: 2018

Number of pages: 35 Number of figures: 15 Number of tables: 4

Number of appendix: 1 Number of appendix pages: 5 Number of graphs: 1

Number of references: 11

Mentor: Assist. Prof. Jernej Vičič, PhD

Co-Mentor: Assist. Prof. Branko Kavšek, PhD

Keywords: RNN, neural network, machine translations, translation, translator, NMT, neural machine translations, web interface, web page

Abstract:

My Bachelor dissertation is based on describing the setup of a machine translation system based on neural networks. The first part will be focused on presenting how the neural network functions, including the series of steps taken to convert an acquired input into an output. Followed by a description of phases taken in translating based on neural networks. Thirdly the dissertation will cover a description of the Neural Monkey tool, which is also used for the final project. This tool is mainly based on receiving inputs, data preparation and data processing for an output. The next chapter will consist of two subchapters; a presentation of installing an interface for communicating with the Neural Monkey tool and an analysis of acquired results, based on conducted tests. Finally, I will include a summary of the paper and ideas for further work.

ZAHVALA

Zahvaljujem se mentorju doc. dr. Jerneju Vičiču in somentorju doc. dr. Branku Kavšku, za strokovno pomoč pri izdelavi in pregledu zaključne naloge. Zahvaljujem se tudi ostalim delavcem Fakultete za matematiko, naravoslovje in informacijske tehnologije FAMNIT, za njihove nasvete in strokovno pomoč.

KAZALO VSEBINE

KAZALO PREGLEDNIC

KAZALO SLIK IN GRAFIKONOV

KAZALO PRILOG

SEZNAM KRATIC

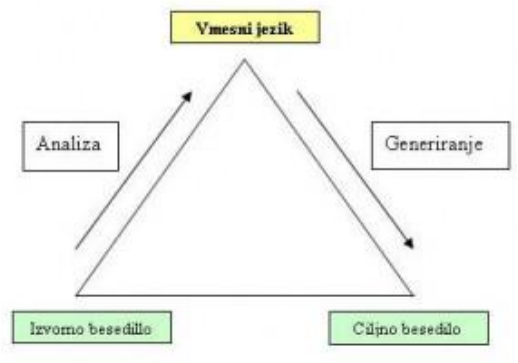
RNN	ponavljajoča se nevronska mreža
NN	nevronska mreža
S	sigmoidna funkcija
SSH	protokol za upravljanje računalnika na daljavo
IP	številka, ki natančno določa računalnik
NMT	prevajanje na osnovi nevronskih mrež
SP	strojno prevajanje

1 UVOD

V današnji dobi, ko je iskanje informacij večinoma opravljeno preko elektronskih vmesnikov, se izkaže potreba po nekem prevajalcu, ki je uporabniku prosto dostopen in obenem ponudi takojšen odziv. Za rešitev tega problema, so se pojavili strojni prevajalniki, ki olajšajo razumevanje tujega besedila. Vendar (še) ne morejo v celoti nadomestiti fizično osebo, ki se ukvarja s prevajanjem.

Slika 1: Primer prevajanja iz danščine v angleščino

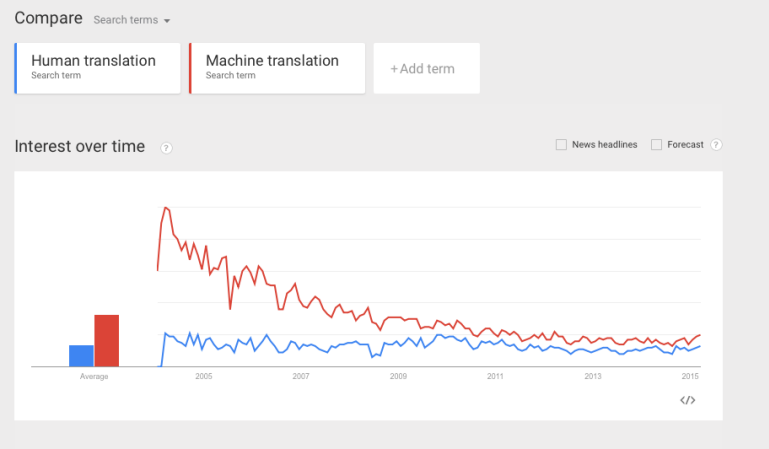
Strojno prevajanje je proces, pri katerem računalnik (velikokrat imenovan kot strežnik) pretvori podano besedilo v izbran jezik, brez posega človeške roke (vsaj med prevajanjem). Načinov oziroma pristopov prevajanja je kar nekaj. Ti so se skozi čas nadgrajevali in spreminjali. Danes imamo že zelo dobre prevajalnike, vendar ne toliko dobre, da bi lahko popolnoma zaupali v njihove prevode. Tisto, kar jih omejuje pri doseganju popolnosti, je upoštevanje različnih končnic (v primeru uporabe spola, sklanjatve, slovnice, …), podobnosti med različnimi besedami istega pomena in druge specifične lastnosti posameznega jezika. Pri strojnih prevajalnik je namreč potrebno napisati oziroma jih naučiti vsa pravila, ki jih ima določen jezik, za doseganje najboljšega cilja.



Slika 2: Postopki pri strojnem prevajanju

Danes poznamo tri glavne načine strojnega prevajanja: na osnovi pravil, statistično strojno prevajanje in prevajanje na osnovi nevronskih mrež. Prevajanje na osnovi pravil je sestavljeno iz slovničnih zakonitostih za izbran jezik ter iz slovarja najbolj pogostih besed. Pri tem se računalnik odloči za pravilno izbiro končnice ter postavitev vrstnega reda besed

glede na podana pravila in prevode. Pri statističnem prevajanju prevajalnik pravil ne pozna. Znanje, ki ga potrebuje, dobi preko statične analize besednih parov na podlagi ogromne količine podatkov. Njegova odločitev temelji na izbiri prevoda z največjim odstotkom med podanimi možnimi rešitvami (pari). Zadnji (nam znan) način je princip prevajanja na osnovi nevronskih mrež. Ta prevajalnik vsa potrebna znanja pridobi z učenjem preko "velike" nevronske mreže. V zadnjih nekaj letih je pridobil na popularnosti, saj se je, v primerjavi s fraznimi prevajalniki (več besed, en prevod), izkazal za najboljšega.



Slika 3: Primerjava trenda iskanosti v Google iskalniku med strojnim in človeškim prevajanjem

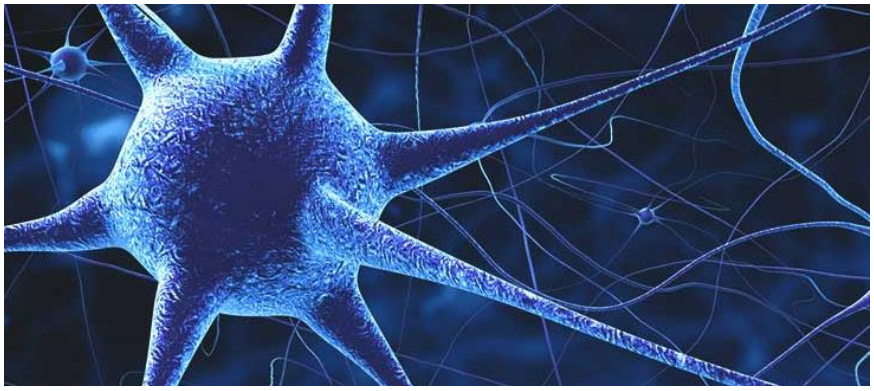
V tej nalogi se bom osredotočil na strojno prevajanje na osnovi nevronskih mrež iz slovenščine v angleščino. Uporabil bom orodje Neural Monkey[1], ki bo poskrbelo za samostojno prevajanje in učenje prevajalnika. Poleg tega bom izdelal spletni vmesnik, ki bo omogočal uporabo navadnemu uporabniku.

[1] Vir: Helcl J. in Libovický J.., Neural Monkey: An Open-source Tool for Sequence Learning, PBML, Prague, April 2017, 5-17

2 DELOVANJE NEVRONSKE MREŽE

2.1 Opis ponavljajoče in navadne nevronske mreže

Ponavljajoča se nevronska mreža (angl. Recurrent Neural Network) je nadgradnja navadne nevronske mreže. NN je zgrajena iz nevronov in uteži. Glede na vhod uporabniku poda rezultat tako, da upošteva nevrone in uteži, ki so del celotne strukture. Nevroni so elementi z nekakšno vrednostjo (v računalništvu je ta vrednost največkrat podana s številko). Ti se glede na trenutni vhod odločijo kam in kakšno količino procesiranih ali ne procesiranih podatkov posredujejo naprej naslednjemu nevronu. Pot od nevrona A do nevrona B lahko tudi nekaj stane in se prispela vrednost lahko spremeni. To ceno poti imenujemo utež. Če povzamem; navadna nevronska mreža pridobi rezultat tako, da se glede na vhod in lastno naučeno znanje (vrednosti nevronov in uteži) odloči za »pravilen« izhod. Vendar svoje znanje NN ne nadgrajuje. Zato se je uveljavila RNN s sposobnostjo neprestanega učenja. S principom ponavljanja lahko ta (ni nujno, je pa priporočljivo, drugače bi bila NN pri strojnem prevajanju nekoristna) svoje znanje izboljšuje.



Slika 4: Nevron pri človeškem nevronskem sistemu, kjer so ti povezani med sabo

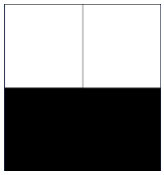
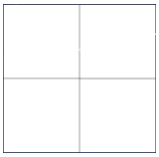
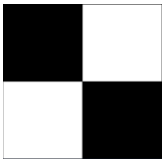
2.2 Postopki delovanja nevronske mreže

Nevronska mreža ima možnost učenja na podlagi podatkov o katerih sama še ničesar ne ve (ne pozna zgradbe, funkcije, željenih ciljev, …). Postopek kreiranja NN je sestavljen iz rekurzivne (neprestane) določitve vrednosti uteži in nevronov, ki opravljajo pomembno vlogo pri določanju izhoda, ter testiranja lastne baze znanja. Skozi celoten proces učenja NN torej potrebuje neko bazo podatkov. To bazo poimenujemo učna množica. Uporabljena je za učenje nevronske mreže. Za testiranje uporabljamo testno in validacijsko množico. Validacijska množica je uporabljena tudi za primerjanje učinkovitosti različnih algoritmov uporabljenih za učenje. Je nekakšna hibridna množica.

2.2.1 Opis problema

Delovanje nevronske mreže je najlažje opisati na primeru. Uporabimo računalniški problem, kjer je potrebno določiti katerega tipa/zapolnjenosti je kvadrat velikosti 2x2. Rezultat, glede na postavitev barvnih pik, lahko napovemo na štiri načine: vertikalna, horizontalna, polna in diagonalna. Vsak specifičen tip je narisan v spodnji tabeli.

Tabela 1: Željen tip izhoda pri določenem primeru vhoda

TIP	PRIMER	TIP	PRIMER
Vertikalna		Horizontalna	
Polna		Diagonalna	

2.2.2 Določitev vrednosti

Vsako vhodno vrednost moramo najprej opisati. V stolpcih »PRIMER« iz Tabele 1 so razvidni kvadrati velikosti 2x2. Vsakemu manjšemu kvadratu znotraj 2x2 velikega kvadrata rečemo pika. Vsak kvadrat, velikosti 2 po širini in 2 po dolžini, je torej sestavljen iz pik,. Vsako piko opišemo s številčno vrednostjo. V našem primeru, ker je barvna paleta enostavna, tj. sestavljena iz le dveh barv, je dovolj, da črni piki dodelimo vrednost -1 in beli 1. Če bi paleta vsebovala več odtenkov črne, bi imeli potem toliko več vrednosti, ki bi lahko zasedle interval od -1 do 1 z določeno vrednostjo premika (premik bi potem izračunali kot deljenje vsote absolutnih vrednosti začetka in konca intervala ter številom različnih barv).

Slika 5: Določitev vrednosti posamezne pike

2.2.3 Postavitev nevronov

Nevroni so nekakšna pravila, ki, glede na vhod, podajo nek izhod. Njihovi pogoji in postavitev se lahko spreminjajo med procesom učenja. Po opredelitvi vrednosti posamezne

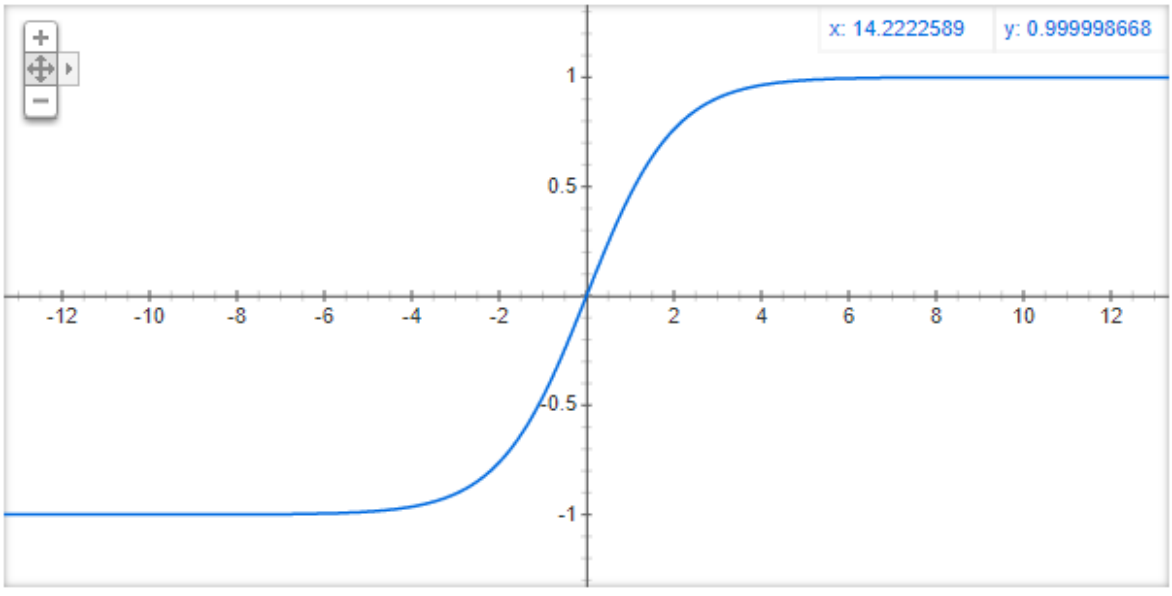
točke na vhodnem primeru dobimo iz vhoda štiri izhodne točke, ki jih opišemo s temi vrednosti. Torej vsaka vhodna vrednost (to je kvadrat v našem primeru) ima opisano vsako piko z vrednostjo 1 ali -1. Nevroni potem te vrednosti sprocesirajo in podajo rezultat v istem intervalu.

Za lažjo obrazložitev vzemimo primer palete z več sivimi barvami, kjer želimo določiti svetilnost vhodnega podatka. Zelo temno barvo (črno) nastavimo na -1, najsvetlejšo (belo) pa na 1. Vmesne svetilnosti zajamejo vrednosti na intervalu od -1 do 1 (ne vključujoč).

Kot primer vhoda lahko za vsako piko določimo naslednje vrednosti (ločene s podpičjem): -1; -0,5; 0; 0,5; 1. Te vrednosti nevron želi procesirati s pomočjo algoritmom. Ta je lahko tudi matematična funkcija. Denimo, da se je nevron odločil za postopek seštevanja. Za poljuben vhod je dobil skupni seštevek -1,5. Vendar ta vrednost ne sega znotraj začetnega intervala, zato uporabi nekoliko spremenjeno sigmoidno funkcijo. Ta funkcija je znana po »S« obliki. Formula za navadno S funkcijo je naslednja:

$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1},$$

ki vse podane vrednosti postavi v interval od 0 do 1. Da zadovoljimo začetne pogoje funkcijo pomnožimo z 2 in odštejemo 1. Začetni Interval navadne sigmoidne funkcije tako spremenimo od -1 do 1.



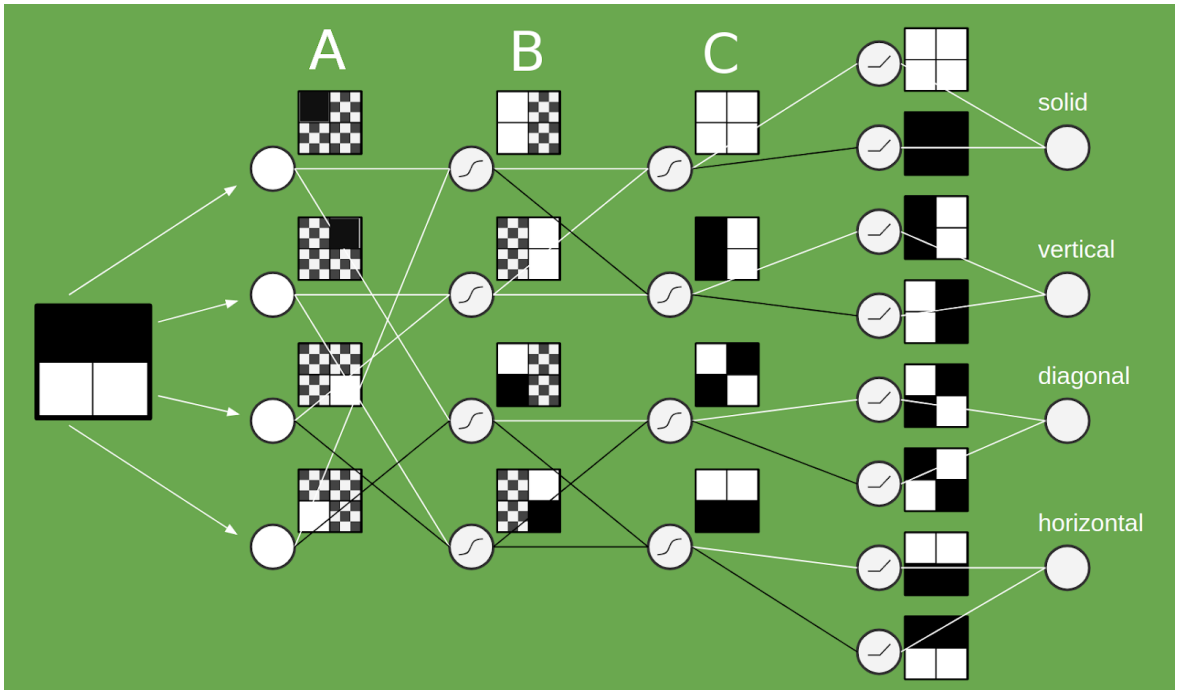
Grafikon 1: Narisana sigmoidna funkcija z rahlo spremembo vrednosti (pomnožena z 2 in odšteta z 1)

Vhodno vrednost -1,5 po izračunu nevron pretvori v približno -0.635. Rezultat tega izračuna je svetilnost vhodnega podatka, kjer velja, da je temnejša barva bolj negativna, svetlejša pa pozitivna.

V podanemu primeru smo imeli le en nevron za določitev svetilnosti. Če bi želeli določiti več podatkov, na primer ali je bolj horizontalna ali vertikalna postavitev, bi se NN med učenjem prilagajala in po možnosti dodala nevrone. Povezave med njimi bi lahko otežil oziroma jim dodal ceno. Primer večje nevronske mreže je v naslednji točki.

2.2.4 Primer nevronske mreže

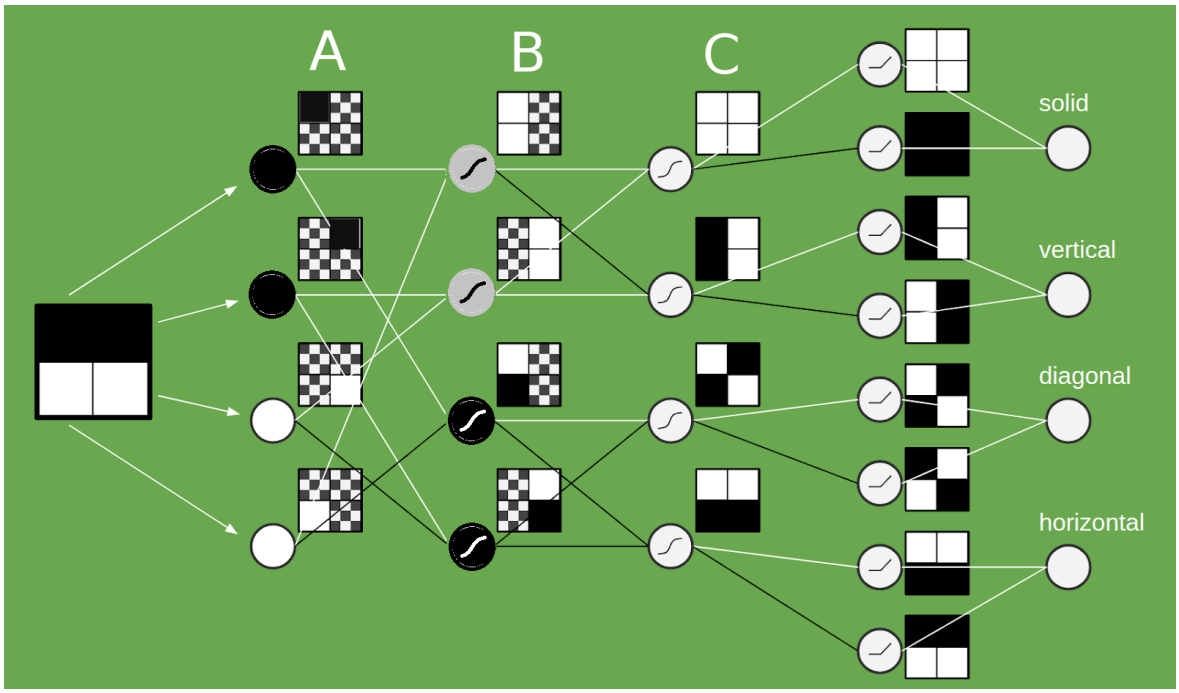
Po vseh postopkih prejšnjih točk in dodajanju nevronov nastane nevronska mreža. Ko nevrone med sabo povežemo povezave dodatno otežimo. Te uteži so označene z obarvanimi črtami. Bela barva pomeni, da vrednosti, ki je prišla do trenutnega nevrona, ne spremenimo. Črna barva povezave pomeni, da vhodno vrednost pomnožimo z -1 oziroma spremenimo predznak. Tukaj velja še omeniti, da pike, ki niso obarvane, ne vplivajo na seštevanje in nimajo določene vrednosti (so neupoštevane). Vsi premiki se dogajajo od leve proti desni. Vrednost nevrona, ki je enaka 0 je označena s sivo barvo.



Slika 6: Primer nevronske mreže pred prvim korakom sprehoda od leve proti desni, s postavljenimi nevroni in povezavami (nevron okrogle oblike, kvadrati in narisani primeri za lažje sledenje poteka)

Po prvem prehodu smo postavljeni v prvi stolpec (označen s črko A). Ta stolpec s štirimi kvadrati upošteva vrednosti pik ločeno. To pomeni, da je vsaka pika v svojem kvadratu stolpca A. Tukaj (v stolpcu A) dobimo dve "enaki" vrednosti; prva dva nevrona (kroga) sta obarvana črno (negativno naravnana), druga dva sta bele barve. Povezave do prvega stolpca so vse bele, kar pa pomeni, da vhodne vrednosti nismo spreminjali.

Tukaj velja opozoriti, da je potrebno razlikovati nevron od primera (kvadrata). Nevron je lahko pozitiven ali negativen ter upošteva ceno po pripeljani poti. Nevron naredi seštevanje vrednosti (številčne) in je okrogle oblike. Kvadrat oziroma narisan primer sešteva le prejšnje primere med sabo. Prav tako upošteva ceno poti na vsaki piki, ki ni večbarvna.



Slika 7: Primer nevronske mreže v drugem koraku, kjer so nevroni v stolpcu A in B obarvani glede na vrednosti, ki jo nosijo

Do B stolpca nas pripeljejo povezave kreirane iz prejšnjih nevronov, izmed katerih so nekatere črne druge bele. Za primer vzemimo zadnja dva kvadrata B stolpca. Naredimo seštevanje prejšnjih dveh nevronov, pri temu upoštevamo ceno poti oziroma utež. Če je povezava črne barve potem podamo konjugirano vrednost. Torej tretji nevron (od zgoraj navzdol) pri B stolpcu pridobi črno barvo oziroma negativno vrednost. To smo dobili s seštevanjem prvega kvadrata A stolpca, ki je bil črne barve, s četrtim oziroma zadnjim kvadratom A stolpca, ki je bele barve vendar konjugirane vrednosti. Skupaj torej dobimo dvakrat črne barve, kar je v našem primeru vrednost -2. Sodeč po opravljeni sigmoidni funkciji je maksimalna negativna vrednost -1 in s tem tretji nevron B stolpca pridobi črno barvo. Enako naredimo za četrti nevron tega stolpca, ki je prav tako negativne vrednosti in posledično črne barve. Prva dva nevrona tega stolpca sta po seštevanju enaka 0, kar sem označil s sivo barvo.

Ta postopek nadaljujemo do zadnjega označenega stolpca. Zadnji nevroni v neoznačenemu stolpcu uporabljajo algoritem (funkcijo), ki negativno vrednost postavi na nič, pozitivno pa enostavno pustijo. Tako dobimo rezultat, ki nam pove tip kvadrata. Prvi trije krogci zadnjega stolpca so, po vseh opravljenih postopkih, obarvani s sivo barvo. To pomeni, da

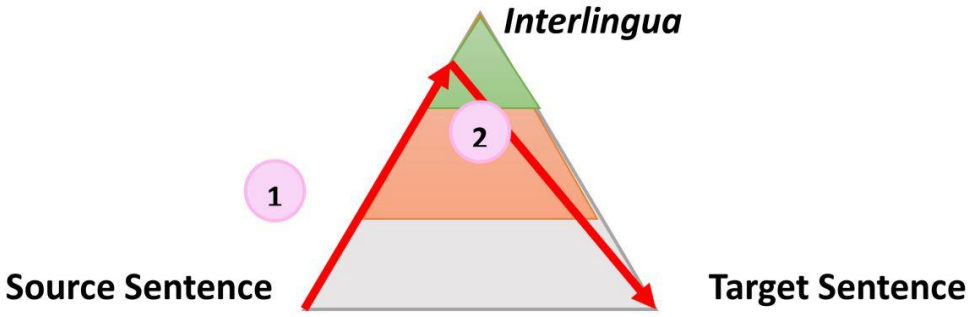
vrednosti nimajo. Zadnji krogec je obarvan z belo barvo. Ta nam pove, da je podan vnos po nekaj korakih horizontalnega tipa.

3 PREVAJANJE NA OSNOVI NEVRONSKIH MREŽ

Kot omenjeno na začetku poznamo pri strojnem prevajanju kar nekaj načinov uporabljanih v današnjem elektronskem okolju. V nadaljevanju bom opisal uporabo nevronskih mrež s strojnim prevajanjem. Ta se od drugih načinov dokaj razlikuje. Predvsem v samem postopku prevajanja od pridobljene besede do generiranega izhoda.

3.1 Osnovna zgradba strojnega prevajanja na osnovi nevronskih mrež

Na sliki spodaj (slika 8) je narisan Vauquoisov trikotnik. Ta opisuje tri različne poti skozi katere gre podan vhod. Podana beseda, stavek ali poved (na sliki "Source Sentence") je najprej analizirana in potem dostavljena naslednjemu koraku. Pri analizi se vhod kodira v zaporedje vektorjev, s katerimi računalnik lažje upravlja. Ti vektorji oziroma kodirane besede so nekakšen umetni mednarodni pomožni jezik imenovan Interlingua. Razvit je bil med letom 1937 in 1951 in se za procese prevajanja uporablja še danes[2].



Slika 8: Vauquoisov trikotnik

Po Vauquoisovem trikotniku se uporabniku kot rešitev poda preveden vhod. Proces je dokaj enostaven. Sestavljen je iz dveh korakov. Prevajalnik najprej prevaja v umetni pomožni jezik, zatem iz pomožnega jezika v končni željeni jezik. Točen postopek in imenovanji za te dve fazi sta opisani v naslednji podtočki tega poglavja.

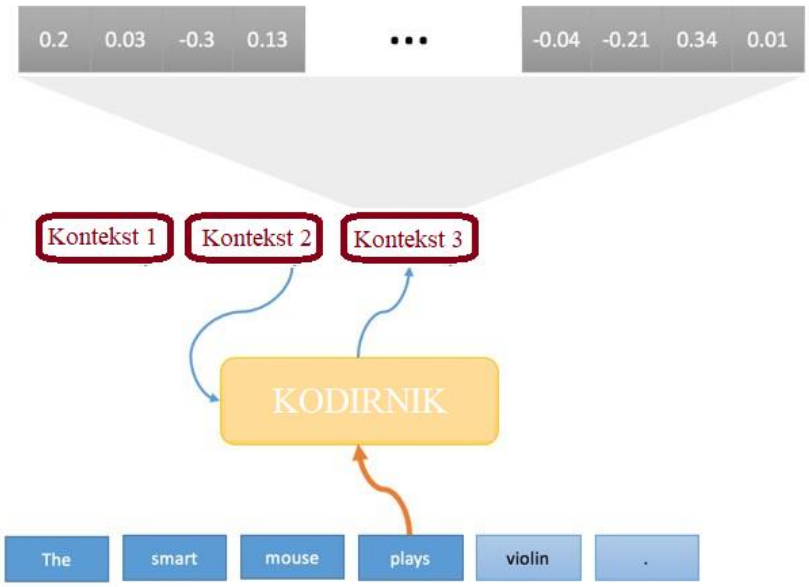
[2] Vir: Saes M., *Translation as Linear Transduction,* PBML, Uppsala University, 2011

3.2 Fazi prevoda pri NMT

Pri fazah prevajanja NMT imamo dve ločeni smeri; kodiranje in dekodiranje. V prvi fazi kodiranja prevajalnik vhodne besede najprej obdela tako, da jim postavi vektor števil. Druga faza dekodiranja nato naredi ravno obratno; vse vektorje pretvori v izhodne besede.

3.2.1 Kodiranje

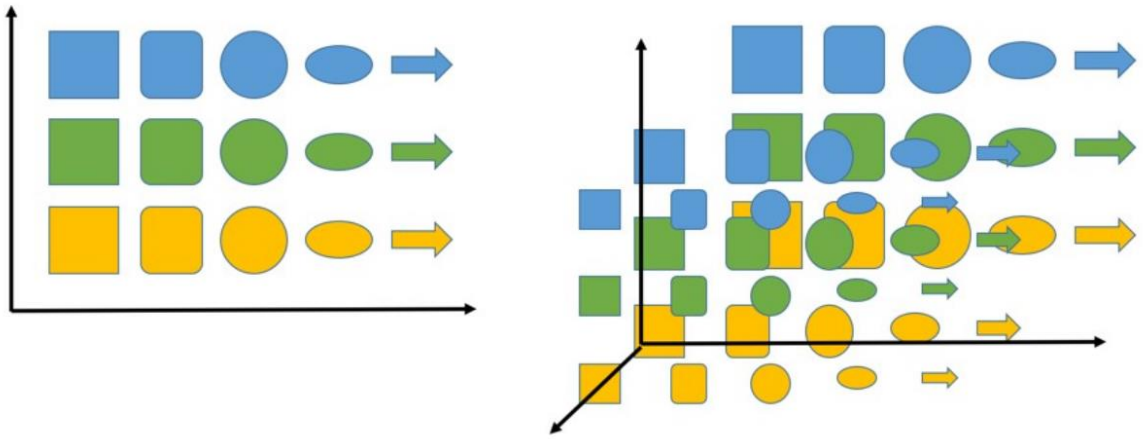
Pri kodiranju se računalnik sprehodi po podanem vhodu, kjer vsako besedo (vhod) kodira v nek kontekst (primer na sliki: »Kontekst 1«). V primeru, da se prevajalnik nahaja na vsaj drugi besedi, pri kodiranju uporabi tudi prejšnji kontekst. Vsak generiran kontekst je zaporedje nekih števil (običajno 1000 števil). Ta algoritem generira zaporedje števil imenovano vektor števil. Za lažjo razlago kodiranje temelji na določenih pravilih. Vsaka izvorna beseda je najprej pregledana v enojezičnem slovarju. Nato pregledana s kodirnikom in za vsako vhodno besedo najde vse besede, ki sodijo/bi sodile zraven ob tvorjenju stavka (skupne lastnosti).



Slika 9: Vhodi in izhodi kodirnika

Umestitev besed (angl. Word embedding) je proces izbire enakovrednih besed. To so besede, ki naj bi sodile skupaj. Vse besede so shranjene na način, ki tiste z enakimi značilnostmi, postavi eno poleg druge. Za primer je spodaj (slika 10) izbira besed (na sliki narisani kot objekti) s posebnimi lastnostmi. Na levi strani slike so značilnosti ločene po vrsticah in stolpcih. V vsaki vrstici je določena barva, v vsakem stolpcu določen element. Tako sta postavljeni le dve dimenziji; vertikalno in horizontalno. V primeru, da bi

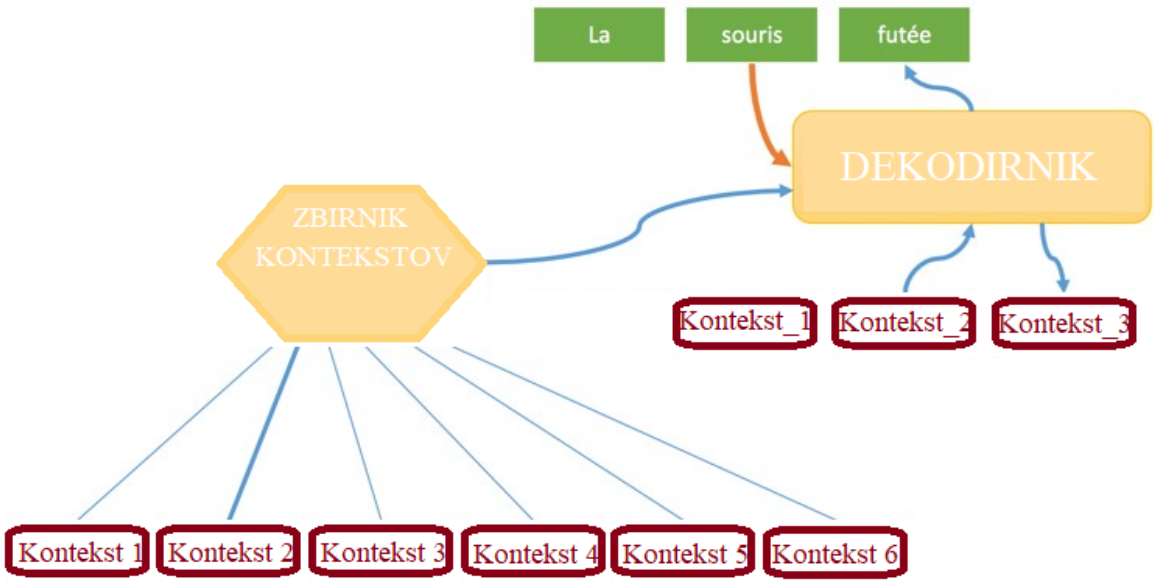
prevajalnik rabil še kakšno specifično lastnost bi kreiral dodatne dimenzije. Ker je pri jezikih ogromno lastnosti, je teh dimenzij približno 800.



Slika 10: Primer načina združevanja besed ali objektov v več dimenzijsko tabelo glede na lastnosti

3.2.2 Dekodiranje

Dekodiranje je nadaljevanje prve faze prevajanja. Prva faza je generirala kontekste skupaj s prejšnjimi. Dekodirnik v tej fazi naredi podoben postopek kot kodirnik vendar v obratnem vrstnem redu. Vse kontekste izbere, vzame prejšnjo prevedeno besedo in prevede trenutni vhodni kontekst v izbran jezik.



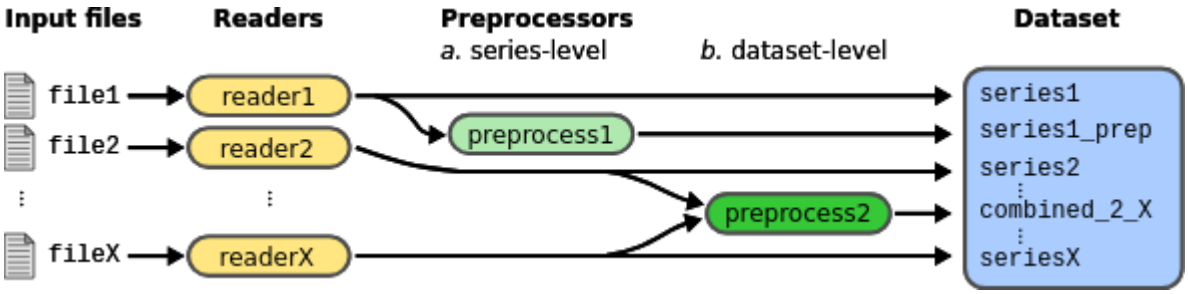
Slika 11: Vhodi in izhodi dekodirnika

4 NEURAL MONKEY

Neural Monkey je odprtokodno orodje, ki deluje na Tensorflow okolju. Namenjeno je strojnem učenju. Svoje temelje ima postavljene na nevronski mreži. Sestavljen je iz faz branja in učenja, kar je opisano v nadaljevanju.

4.1 Branje izvorne datoteke

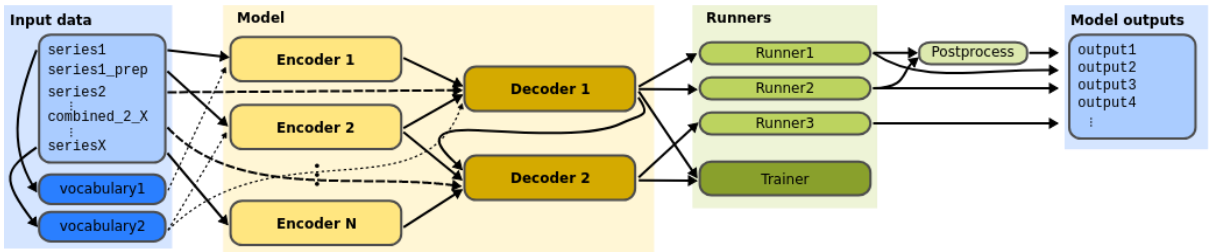
Postopek pridobitve vhodnih parametrov je sestavljen iz treh točk. Vhodna datoteka je najprej brana s pomočjo bralnika. Prebrani vhodi so potem lahko pred procesirani (na primer: byte-pair kodiranje, ki vzame vhode in segmentira izvor). Zadnja točka je pre-procesiranje na nivoju podatkovnih množic in samo kreiranje le te, kot skupek celote.



Slika 12: Potek podatkov Neural Monkey orodja pri branju izvorne datoteke

4.2 Učenje in zagon modela

Po prvi fazi branja vhodne datoteke kreiramo besednjak, ki ga uporabljajo kodirniki in dekodirniki. Sestavljen je iz indeksiranih žetonov, ki zagotavljajo funkcionalnost pretvorbe stavkov iz žetonov v matriko z indeksi in obratno.



Slika 13: Postopek procesiranja podatkov z modeli

Sam model je sestavljen iz kodirnika in dekodirnika. Kar v predhodnih točkah nisem opisal so zaganjalniki (na sliki zgoraj (slika 13) »Runners«). Njihovo delo je, da zaženejo dekoderje. To lahko izvedejo na različne načine vendar z željo pridobitve najbolj primernega prevoda. Za doseganje tega prevajalnik delo dodeli različnim zaganjalnikom.

5 PRIPRAVA PODATKOV IN ZAGON UČENJA

5.1 Pridobitev korpusov

Korpuse sem pridobil iz spletne strani http://opus.nlpl.eu/. Uporabil sem stolpec »alg«, ki je sestavljen iz enakih atributov »(en-sl)« in korpuse OpenSubtitles2018. Skupaj so tvorili nekaj manj kot 500 MB skrčenih podatkov.
Linki posebej:

ANG: http://opus.nlpl.eu/download/OpenSubtitles2018/en-sl/c.clean.en.gz
SLO: http://opus.nlpl.eu/download/OpenSubtitles2018/en-sl/c.clean.sl.gz

5.2 Priprava podatkov

Prevajalnik, ki temelji na nevronski mreži, je potrebno najprej naučiti prevajanja. Za pridobivanje znanja sem uporabil korpuse pridobljene na spletni strani http://opus.nlpl.eu/. Po zahtevah iz Neural Monkey dokumentacije sem kreiral testno, učno in validacijsko množico besed oziroma korpusov. To pomeni, da sem kreiral dvojice datotek za vsako množico. Ti so med seboj povezani. Vsaka datoteka ima enako število vrstic. Prav tako velja, da je poljubna vrstica, ki se nahaja na n-tem mestu ene datoteke, prevod n-te vrstice druge datoteke. Tako prevajalnik točno ve kje se nahaja prevod za vsako poved. Pri datotečnih dvojicah sem za validacijsko in testno množico uporabil različnih 5000 vrstic od vseh 19 636 519. Preostanek vrstic sem nastavil za učno množico.

Tabela 2: Uporabljeno število vrstic za testno, validacijsko in učno množico

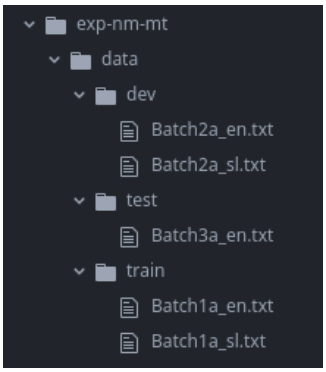
Vseh	19 636 519	**#besed za dataset**
Učna (99,94%)	[1, 19 626 519]	19 626 519
Validacijska (0,03%)	[19 626 520, 19 631 519]	5000
Testna (0,03%)	[19 631 520, 19 636 519]	5000

5.3 Postavitev okolja

Za namestitev Neural Monkey sem sledil navodilom dokumentacije[3] (uporabljeni ukazi se nahajajo v prilogi 1). Pred začetkom sem prenesel in namestil Python 3.5, pip in git programe. Sledil je prenos potrebne kode Neural Monkey okolja z git ukazom ter namestitev zahtev potrebnih za delovanje nevronske mreže z uporabo grafične kartice

[3] Vir do dokumentacije: http://neural-monkey.readthedocs.io/en/latest/

(GPU). Ob namestitvi sem zaradi pridobljene napake namestil še python numpy knjižnico. Organiziranost datotek sem postavil po strukturi na sliki spodaj (slika 14).



Slika 14: Datotečna struktura

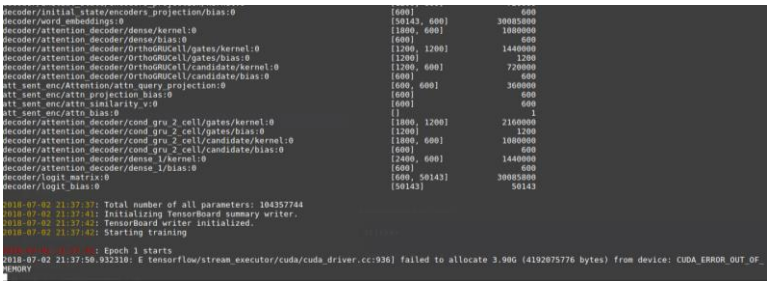
Prenešene in generirane korpuse sem postavil v treh mapah: »dev«, »test« in »train«. Pri testni množici sem kasneje za potrebe testiranja postavil še drug par dvojice. Po temu je sledilo generiranje slovarja. Proces generiranja je trajal do 10 minut. Uporabil sem algoritem BPE (Bype Pair Encoding), ki omogoča izvajanje prevodov na odprtem slovarju tako, da ta vzame pogoste in neznane besede kot sekvenco nekaj besednih enot. Torej iz ene besede je lahko kreiral več besed (na primer besedo »basketball« je razčlenil na »basket« in »ball« in podobno).

Kot zadnja malenkost je sledila izdelava konfiguracijske datoteke. S to datoteko sem prevajalniku povedal kje se nahajajo posamezni dokumenti in naredil nastavitev parametrov, ki jih Neural Monkey potrebuje za začetek delovanja[4].

5.4 Zagon učenja

Po zagonu se je izpis v konzoli nekoliko spremenil. Po nekaj minutnem branju korpusov se je prevajalnik začel učiti. Vsega skupaj je trajalo približno mesec dni. Pri tem sem uporabil grafično kartico GeForce GTX 1080 z 8GB pomnilnika. Celoten sistem je postavljen v Linux okolju.

[4] Koda dostopna na: https://github.com/orangeGoran/neuralmonkey-experiments



Slika 15: Začetni izpisi v ukaznem oknu pri zagonu učenja prevajalnika na osnovi nevronskih mrež

6 NAMESTITEV VMESNIKA ZA KOMUNICIRANJE S STREŽNIKOM

Namestitev vmesnika za komuniciranje s strežnikom[5] je bilo dokaj enostavno opravilo. Neural Monkey orodje ponuja možnost zagona prevajalnika kot spletni vmesnik. Vse to sem izvedel z enim ukazom. Da sem dostopal do strani v lastnem brskalniku sem uporabil SSH komunikacijo s posredovanjem strežniškega porta na lokalni IP naslov 127.0.0.1 na port 3333. Vendar je moje veselje ustavila obilica težav pri prevodih. Prevajalnik se je večkrat zrušil in ni uspel priti niti do začetka prevajanja ponujene besede. Te težave sem uspel odpraviti s poglabljanjem v izvorno kodo in spremembi same te, kar pa seveda ni priporočljivo. Saj se v primeru posodabljanja programske opreme lahko izgubijo narejene spremembe, vendar druge možnosti nisem imel.

7 REZULTATI

Da bi pridobil rezultate sem spisal skripto[6], ki je komunicirala s strežnikom preko namiznega računalnika tako, da je pošiljala zahteve po povpraševanju. Testno skripto sem napisal v Javascriptu. Za pogon sem uporabil ukaz “node”, kjer sem imel že prej nameščeno programsko opremo NodeJs[7], kateri si lasti ta ukaz. Po pregledu rezultatov sem zagnal še evalvacijo podatkov z Neural Monkey orodjem. Ta je veliko hitreje testiral testno množice. Po opravljeni evalvaciji sem izračunal še BLEU (Bilingual Evaluation Understudy).

7.1 Delovanje skripte za pošiljanje zahtev za povpraševanje po prevodu

[5] Dostop do spletne strani: http://www.studenti.famnit.upr.si/~89151058

[6] Koda dostopna na: https://github.com/orangeGoran/ScrapperForNMT

[7] Pridobitev preko: https://nodejs.org/en/

Skripta deluje na preprostem principu, kjer algoritem najprej prebere testni množici. Te množici sem kreiral na začetku, od katerih je ena v slovenščini druga v angleščini. Po prebiranju obeh datotek, sem vzel vsako vrstico med 5000-imi, ki je bila v slovenskem jeziku, in jo poslal s pomočjo skripte na strežnik. Ta je z uporabo Neural Monkey orodja naredil prevod za vsako prebrano vrstico. Zaradi možnosti preobremenitve prevajalnika oziroma samega strežnika sem vsako zahtevo po izračunu poslal v razmiku 100 ms. Po odgovoru strežnika sem primerjal pridobljeni prevod z izvornim prevodom in izračunal procentualno pravilnost prevajanja.

7.2 Pridobljeni rezultati

Po približno devetih minutah se je v prvem prevajanju obrnilo 5000 vrstic slovenskih povedi. Testno skripto sem zagnal trikrat zapovrstjo pri hitrosti 100ms/zahtevo in pridobil vedno znova različne rezultate.

Tabela 3: Primerjava testnih primerov glede časovne izvedbe in števila enakih ter različnih prevodov z uporabo testne skripte

	Enaki prevodi	Različni prevodi	Časovnost izvedbe
Test 1	442	4558	~ 570 sekund
Test 2	299	4701	~ 734 sekund
Test 3	190	4810	~ 1287 sekund

Pri delovanju, kot lahko opazimo, je Neural Monkey imel nekaj težav. Po pregledu ukaznega okna so se pojavljale naslednje napake: preveč odprtih datotek, premalo pomnilnika, neuspešno lociran pomnilnik, … V prvem primeru je grafična kartica delovala na 30% svoje kapacitete, dokler ni v tretjem poskusu prišla celo do 50%. Iz tega razloga sklepam, da ima Neural Monkey nastavljeno koliko pomnilnika lahko vzame za potrebe prevajanja. Teh težav nisem poskusil rešiti, ker bi zahtevalo veliko več znanja pri pregledu izvorne kode. Moje dojemanje situacije je bilo slednje; to orodje ni namenjeno velikemu številu zaporednih zahtev. Za analizo teh problemov bi rabil še veliko časa. Zato sem opravil še dodatno evalvacijo, ki jo ponuja orodje Neural Monkey.

Prejšnji način za analizo, ki sem ga naredil s testno skripto, je bil napačno zastavljen, saj ne poda najboljše primerjave prevodov. V fazi učenja je Neural Monkey opravil primerjavo lastnega preverjanja s človeškimi prevodi na validacijski množici. Enako preverjanje sem nato uporabil na testni množici. Tudi njo je Neural Monkey v celoti prevedel (uspešno brez napak) ter tako na novo kreiral prevedeno datoteko s 5000-imi vrsticami. Pri prevajanju testne množice je imel program manj napak kot pri prevajanju testne skripte v prvem primeru. Enakih prevodov pri drugem načinu prevajanja z uporabo NM ukaza na testni

množici je bilo 4425. S tem sklepam, da se je zaradi pomnilniških in podobnih težav število napak pri testni skripti pojavilo vsaj 100 krat skozi vse teste prve analize.

7.3 Povzetek rezultatov

Po večkratnem testiranju s testno skripto, sem prišel do maksimalne natančnosti, ki je znašala 8,84% oziroma 11,5%, v primeru uporabe Neural Monkey elevacijskega orodja. Z rezultati nisem bil zadovoljen. Prevajalnik je pri 19 milijonih vrstic in z devetimi obhodi za učenje rabil malo manj kot en mesec. Našel sem tudi primere, kjer je napaka nastala zaradi istopomenskih besed, kljub temu, da bi program za prevod lahko uporabil katerokoli od teh besed. Prav tako sem opazil, da je precej primerov, kjer je samo slabo postavljena vejica prikazala rezultat kot napačen. Po podrobnejšem pregledu testnih množic sem naletel na veliko število znakov, katerih fizična oseba ne bi uporabila pri pisanju povedi. Za pridobitev čim bolj natančne analize sem opravil še BLEU analizo z izhodnimi podatki.

BLEU je postopek, ki ovrednoti kakovost besedil strojnega in človeškega prevoda. Čim bližji je strojni prevod človeškemu tem boljši rezultat napove. BLEU vrednosti svojega izhoda napove med 0 in 1. Vsaka vrednost, ki je bližje 1 pomeni več podobnosti med dvema prevodoma. Vsaka vrednost izhoda, ki je bližje 0 pove, da imata prevedeni povedi malo/nič podobnosti. Torej BLEU da odgovor na vprašanje: »Kako dober je prevajalnik v primerjavi s človeškim prevodom?«. Pri procesu je zelo pomembno, da za primerjavo prevodov uporabljamo testno množico, ki pri strojnem učenju ni bila uporabljena oziroma je od uporabljene množice drugačna[8].

BLEU rezultat pri procesu učenja je znašal 27,62 (pomnožen s 100; v nadaljevanju velja enako) in pri testiranju testne množice 19,33. BLEU rezultat pod 14 bi veljal za slabega, rezultat od 20 do 25 pa za dobrega. Zato lahko trdim, da sta oba dobljena rezultata dobra. Rezultat za testno množico sem pridobil z uporabo orodja dostopnega preko spletne strani https://www.letsmt.eu/Bleu.aspx.

Tabela 4: BLEU rezultati v fazah učenja in testiranja

Množica	**BLEU rezultat (pomnožen s 100)**
Validacijska množica	27,62
Testna množica	19,33

[8] Vir: Papineni K idr., *BLEU: a Method for Automatic Evaluation of Machine Translation*, ACL, Philadelphia, July 2002

8 ZAKLJUČEK IN NADALJNJE DELO

Končni rezultat zaključnega dela je bil spletni vmesnik, ki temelji na strojnem prevajanju z uporabo nevronskih mrež. Namen projekta je bil razširiti lastna znanja o nevronski mreži in jezikovnih tehnologijah, ki se uporabljajo v današnjih časih. Največ problemov sem imel z dokumentacijo, ki je dokaj skopa. V primeru napak sem težko poiskal rešitev, ker ni veliko ljudi, ki bi uporabljalo Neural Monkey orodje. Vendar sem uspel posamezne dele združiti v celoto končnega izdelka. Ker sem spreminjal tudi izvorno kodo, predvidevam, da v primeru, da bi v Neural Monkey okolju zagnal drug ukaz, bi obstajala velika možnost napake.

Nadaljnjega razvoja pri tem produktu skoraj da ni, saj je le-ta dokončan. Edina želja, ki se mi je med projektom porojila je pridobiti znanje, ki bi jih dobil s testiranjem prevajalnika z uporabo majhnih korpusov in manjšega števila iteracij. S tem bi dojel kako se prevajalnik obnaša v različnih primerih. Prav tako bi me zanimale možnosti pohitritve algoritmov učenja in opravljanje drugih možnih analiz poleg že izkoriščenih.

9 LITERATURA IN VIRI

[1] Abadi M. idr., TensorFlow: A System for Large-Scale Machine Learning, Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, Savannah, *GA*, USA, November 2016

[2] Bergstra J. idr., Theano: a CPU and GPU math expression compiler., *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010

[3] Cho idr., Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation, *EMNLP*, October 2014, 1724-1734

[4] Helcl J. in Libovický J.., Neural Monkey: An Open-source Tool for Sequence Learning, *PBML*, Prague, April 2017, 5-17

[5] Marcu D. in Wong W., A Phrase-Based, Joint Probability Model for Statistical Machine Translation, *EMNLP*, July 2002, 133-139

[6] Mesnil G., He X., Deng L in Bengio Y., Investigation of Recurrent-Neural-Network Architectures and Learning Methods for Spoken Language Understanding, *Interspeech*, Lyon, France, 2013, 3771-3775

[7] Papineni K idr., BLEU: a Method for Automatic Evaluation of Machine Translation, *ACL*, Philadelphia, July 2002

[8] Saes M., Translation as Linear Transduction, *PBML*, Uppsala University, 2011

[9] Sak H., Senior A. in Beaufays F, Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modelingm, *ISCA*, Singapore, September 2014, 338-342

[10] Shibata Y. idr., Byte pair encoding: a text compression scheme that accelerates pattern matching, *semanticscholar.org*, 1999

[11] Tiedemann J., Parallel Data, Tools and Interfaces in OPUS, In Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'2012), International, 2012

VIRI SLIK:

Slika 1: Primer prevajanja iz danščine v angleščino

Vir: narejen posnetek zaslona na: https://www.google.si/search?q=translate

Datum pridobitve: 30. 7. 2018

Slika 2: Postopki pri strojnem prevajanju

Vir: http://www.prevod-prevodi.com/strojno-prevajanje-2/

Datum pridobitve: 30. 7. 2018

Slika 3: Primerjava trenda iskanosti v Google iskalniku med strojnim in človeškim prevajanjem

Vir: na https://www.tomedes.com/translator-hub/what-can-google-trends-teach-you-about-translation.php

Datum pridobitve: 30. 7. 2018

Slika 4: Nevron pri človeškem nevronskem sistemu, kjer so ti povezani med sabo

Vir: https://www.healthline.com/health/fun-facts-about-the-nervous-system

Datum pridobitve: 2. 8. 2018

Slika 7: Vauquoisov trikotnik

Vir: http://blog.systransoft.com/how-does-neural-machine-translation-work

---

Skupaj:

Slika 12: Potek podatkov Neural Monkey orodja pri branju izvorne datoteka

Slika 13: Postopek procesiranja podatkov z modeli

Vir: http://neural-monkey.readthedocs.io/en/latest/overview.html

Datum pridobitve: 30. 7. 2018

Skupaj:

Slika 6: Primer nevronske mreže pred prvim korakom sprehoda od leve proti desni, s postavljenimi nevroni in povezavami (nevron okrogle oblike, kvadrati in narisani primeri za lažje sledenje poteka)

Slika 7: Slika 7: Primer nevronske mreže v drugem koraku, kjer so nevroni v stolpcu A in B obarvani glede na vrednosti, ki jo nosijo

Slika 9: Vhodi in izhodi kodirnika

Slika 10: Primer načina združevanja besed ali objektov v več dimenzijsko tabelo glede na lastnosti

Slika 11: Vhodi in izhodi dekodirnika

Vir: https://docs.google.com/presentation/d/1AAEFCgC0Ja7QEl3-wmuvIizbvaE-aQRksc7-W8LR2GY/edit

Datum pridobitve: 30. 7. 2018

Opomba: Slike so lahko grafično spremenjene in se razlikujejo od originalnih.

PRILOGE

Priloga A: Uporabljeni ukazi pri delu

NAMESTITEV

```
python3 -m venv nm
source nm/bin/activate
git clone https://github.com/ufal/neuralmonkey
cd neuralmonkey
pip3 install numpy
pip install --upgrade -r requirements-gpu.txt
```

PRIDOBITEV KORPUSOV

Korpusi so pridobljeni iz: http://opus.nlpl.eu/
Uporabljen stolpec na spletni strani: Stolpec: alg (en-sl)
Linki posebej:

ANG: http://opus.nlpl.eu/download/OpenSubtitles2018/en-sl/c.clean.en.gz
SLO: http://opus.nlpl.eu/download/OpenSubtitles2018/en-sl/c.clean.sl.gz

ZAGON OKOLJA

BPE (Byte pair encoding) omogoča NMT izvajanje prevodov na odprtem slovarju tako, da vzame pogoste in neznane besede kot sekvenco nekaj besednih enot. Primer je naslednja beseda:
basketball => basket@@ ball

Postavimo se v »train folder« in za generiranje besednih enot uporabimo naslednji ukaz:

```
paste Batch1a_en.txt Batch1a_sl.txt |
../../../lib/subword_nmt/learn_bpe.py -s 50000 >
../../data/merge_file.bpe
```

Po narejenem slovarju sledi priprava konfiguracijske datoteke.
(http://neural-monkey.readthedocs.io/en/latest/machine_translation.html)

Za postavitev konfiguracijske datoteke kreiram v exp-nm-mt translation.ini. Najprej je potrebno pripraviti dve podatkovni množici. Ker bomo uporabljali BPE definiramo predprocesorja.

```
[train_data]
class=dataset.load_dataset_from_files
s_source="exp-nm-mt/data/train/Batch1a_en.txt"
s_target="exp-nm-mt/data/train/Batch1a_sl.txt"
preprocessors=[("source", "source_bpe", <bpe_preprocess>),
("target", "target_bpe", <bpe_preprocess>)]

[val_data]
class=dataset.load_dataset_from_files
s_source="exp-nm-mt/data/dev/Batch2a_en.txt"
s_target="exp-nm-mt/data/dev/Batch2a_sl.txt"
preprocessors=[("source", "source_bpe", <bpe_preprocess>),
("target", "target_bpe", <bpe_preprocess>)]
```

Pri podatkih za treniranje nevronske mreže povemo iz katerega jezika v kateri jezik bomo prevajali (source in target). Isto velja za validacijske podatke. Ker do sedaj nismo definirali pred in po procesorja to naredimo zdaj.

```
[bpe_preprocess]
class=processors.bpe.BPEPreprocessor
merge_file="exp-nm-mt/data/merge_file.bpe"

[bpe_postprocess]
class=processors.bpe.BPEPostprocessor
```

Za koderja in dekoderja bomo uporabili deljen slovar, ki smo ga kreirali z BPE združevanjem.

```
[shared_vocabulary]
class=vocabulary.from_bpe
path="exp-nm-mt/data/merge_file.bpe"
```

Kreiramo inicializacijo koderja in dekoderja.

```
[encoder]
class=encoders.recurrent.SentenceEncoder
name="sentence_encoder"
rnn_size=300
max_input_len=50
embedding_size=300
```

```
dropout_keep_prob=0.8
data_id="source_bpe"
vocabulary=<shared_vocabulary>

[attention]
class=attentions.Attention
name="att_sent_enc"
encoder=<encoder>
state_size=300
dropout_keep_prob=0.8

[decoder]
class=decoders.decoder.Decoder
name="decoder"
encoders=[<encoder>]
attentions=[<attention>]
rnn_size=256
embedding_size=300
dropout_keep_prob=0.8
data_id="target_bpe"
vocabulary=<shared_vocabulary>
max_output_len=50
```

Sledita trainer in runner. Trainer se uporablja za treniranje modelov in runner za zagon.

```
[trainer]
class=trainers.cross_entropy_trainer.CrossEntropyTrainer
decoders=[<decoder>]
l2_weight=1.0e-8

[runner]
class=runners.runner.GreedyRunner
decoder=<decoder>
output_series="series_named_greedy"
postprocess=<bpe_postprocess>

[bleu]
class=evaluators.bleu.BLEUEvaluator
name="BLEU-4"
```

```
[tf_manager]
class=tf_manager.TensorFlowManager
num_threads=4
num_sessions=1
minimize_metric=False
save_n_best=3
[main]
name="machine translation"
output="exp-nm-mt/out-example-translation"
runners=[<runner>]
tf_manager=<tf_manager>
trainer=<trainer>
train_dataset=<train_data>
val_dataset=<val_data>
evaluation=[("series_named_greedy", "target", <bleu>),
("series_named_greedy", "target", evaluators.ter.TER)]
batch_size=80
runners_batch_size=256
epochs=10
validation_period=5000
logging_period=80
```

Da zaženemo učenje NMT je potrebno podati naslednji ukaz:

```
bin/neuralmonkey-train exp-nm-mt/translation.ini
```

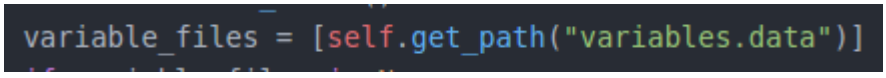
POSTAVITEV SPLETNEGA VMESNIKA

```
bin/neuralmonkey-server --configuration exp-nm-mt/translation.ini
```

Potem, ko sem zagnal ukaz za delovanje strežnika, je ta ob prevodu padel. Pojavila se je napaka, da program ni našel variables.data.0. Po pregledu kode sem v:

bin/neuralmonkey/experiment.py

spremenil 189 vrstico ter tako nastavil variable_files na:



To sem si lahko privoščil, namreč vem natanko kje se dokument nahaja ter kako je poimenovan.

Po rešitvi tega problema se je pojavil novi problem, katerega sem imel tudi v različnih testnih primerih. Problem se je pojavil pri uporabi source_bpe niza za uporaben dataset, ki

je imel shranjene serije. Po pregledu te spremenljivke sem ugotovil, da je sestavljen le iz enega parametra, zato sem se odločil, da v vsakem primeru vrnem istega, edinega možnega. Na lokaciji bin/neuralmonkey/dataset.py sem v približno 130. vrstici zamenjal return v return z atributom source:

```
return self._series['source']
```

Ugotovil se, da prevodi gredo skozi problemov. Nov problem se je pojavil pri postavitvi rešitve na uporabniški zaslon. V bin/neuralmonkey/server/server.py datoteki sem pogledal kaj se nahaja v »responsu«. Ugotovil sem, da ni atributa target, ampak je atribut target_greedy. Zato sem to spremenil v naslednji ukaz:

```
translation = " ".join(translation_response["target_greedy"][0])
```

To je bila še zadnja malenkost, katero sem moral spremeniti. Vendar pa to ni priporočljivo in vzrok dejanske napake ne vem. Rezultat testne verzije, katero sem zagnal s 100 vrsticami, enim testnim primerom ter nekaj evalvaciji, je bil grafični vmesnik na spletni strani

EVALVACIJSKI UKAZI

Za zagon skripte za test, ki je lahko deloval v lokalnem brskalniku preko ssh komunikacij, sem uporabil ukaz:
`ssh -LPORT:127.0.0.1:SERVERPORT USERNAME@SERVERIP -p SSHPORT`

Za zagon skripte za test sem uporabil ukaz:
`node IME DATOTEKE`

Evalvacijo prevajalnika brez uporabe skripte sem zagnal z naslednjim ukazom:

```
bin/neuralmonkey-run exp-nm-mt/translation.ini exp-nm-
mt/translation_run.ini
```

Po zagonu se je pojavila napaka, ki sem odpravil s spreminjanjem NM kode v neuralmonkey/learning_utils.py datoteki. Dodal sem zeleno vrstico:

```
(nm) goran@megafullkulsuperserver:~/a_project/neuralmonkey-experiments$ git diff neuralmonkey/learning_utils.py
diff --git a/neuralmonkey/learning_utils.py b/neuralmonkey/learning_utils.py
index 24027fb..69a38c5 100644
--- a/neuralmonkey/learning_utils.py
+++ b/neuralmonkey/learning_utils.py
@@ -416,6 +416,8 @@ def run_on_dataset(tf_manager: TensorFlowManager,

     if write_out:
         for series_id, data in result_data.items():
+            series_id = 'series_named_greedy'
+
             if series_id in dataset.series_outputs:
                 path = dataset.series_outputs[series_id]
                 if isinstance(data, np.ndarray):
```