Praktični vidiki uporabe podbesednih enot v strojnem prevajanju slovenščina-angleščina

Gregor DONAJ
Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru

Mirjam SEPESY MAUČEC
Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru

Večina sodobnih sistemov za strojno prevajanje temelji na arhitekturi nevronskih mrež. To velja za spletne ponudnike strojnega prevajanja, za raziskovalne sisteme in za orodja, ki so lahko v pomoč poklicnim prevajalcem v njihovi praksi. Čeprav lahko sisteme nevronskih mrež uporabljamo na običajnih centralnih procesnih enotah osebnih računalnikov in strežnikov, je za delovanje s smiselno hitrostjo potrebna uporaba grafičnih procesnih enot. Pri tem smo omejeni z velikostjo slovarja, kar zmanjšuje kakovost prevodov. Velikost slovarja besednih enot je še posebej pereč problem visoko pregibnih jezikov. Rešujemo ga z uporabo podbesednih enot, s katerimi dosežemo večjo pokritost jezika. V članku predstavljamo različne metode razcepljanja besed na podbesedne enote z različno velikimi slovarji in primerjamo njihovo uporabo v strojnem prevajalniku za jezikovni par slovenščina-angleščina. V primerjavo vključujemo še prevajalnik brez razcepljanja besed. Predstavljamo rezultate uspešnosti prevajanja z metriko BLEU, hitrosti učenja modelov in hitrosti prevajanja ter velikosti modelov. Dodajamo pregled praktičnih vidikov uporabe podbesednih enot v strojnem prevajalniku, ki ga uporabljamo skupaj z orodji za računalniško podprto prevajanje.

Ključne besede: strojno prevajanje, velikost slovarja, podbesedne enote, grafične procesne enote

Donaj, G., Sepesy Maučec, M. Praktični vidiki uporabe podbesednih enot v strojnem prevajanju slovenščina-angleščina. Slovenščina 2.0, 11(1): 275–301.

1.01 Izvirni znanstveni članek / Original Scientific Article

DOI: https://doi.org/10.4312/slo2.0.2023.1.275-301

1 Uvod

Strojno prevajanje je postopek avtomatskega prevajanja besedil iz enega naravnega jezika v drugega. Dolgo časa je bilo v praksi zavrnjeno zaradi slabe kakovosti prevodov ali pa je celo ponujalo vir humorja zaradi nesmiselnih in smešnih prevodov. V zadnjih letih se je kakovost strojnega prevajanja bistveno izboljšala, kar je privedlo do njegove uporabe v prevajalski praksi.

1.1 Pristopi strojnega prevajanja

Od svojih začetkov do danes so se pristopi strojnega prevajanja naravnega jezika večkrat spremenili (Sepesy Maučec in Donaj, 2019). Začetki segajo v 50. leta prejšnjega stoletja, ko so prvi sistemi za prevajanje temeljili na pravilih. Kasneje so se pojavili sistemi, ki so izhajali iz primerov prevodov. Algoritmi so iskali podobne pare stavkov na obeh straneh jezikovnega para. Sledilo je strojno prevajanje na podlagi statističnih modelov (Brown idr., 1993) in nekje od leta 2015 naprej strojno prevajanje, ki temelji na uporabi nevronskih mrež in trenutno daje najboljše rezultate (Bahdanau idr., 2014; Vaswani idr., 2017). Uporaba nevronskih mrež zahteva dovolj veliko računsko zmogljivost sistemov in velike množice učnega gradiva, ki morajo biti na voljo, da lahko naučimo modele za prevajanje. Prav razpoložljivost obojega v zadnjih letih je bila povod za nedavni razcvet nevronskega strojnega prevajanja (Stahlberg, 2020).

Sodobni pristopi strojnega prevajanja temeljijo na treh osnovnih arhitekturah nevronskih mrež: nevronske mreže s povratno zanko (Recurrent Neural Network – RNN), konvolucijske nevronske mreže (Convolutional Neural Network – CNN) in arhitektura transformerjev s samo-pozornostjo (self-attention), ki so danes najbolj pogoste.

1.2 Tehnični izzivi

Nevronsko strojno prevajanje prinaša tehnične izzive. V praksi je nujna uporaba grafičnih procesnih enot (Graphical Processing Unit – GPU), ki imajo omejeno velikost delovnega pomnilnika, zaradi česar v določenih primerih ne moremo uporabljati poljubno velikih nevronskih mrež.

Velikost nevronske mreže oz. modela za prevajanje v strojnem prevajalniku je odvisna od izbrane arhitekture, nastavitev hiperparametrov nevronske mreže in velikosti slovarja. Omejena velikost delovnega pomnilnika nam tako omejuje velikost slovarja, kar pomeni slabo pokritost besedišča jezika in posledično dodatne napake pri prevajanju. Uporaba sistemskega delovnega pomnilnika ni praktična, saj je komunikacija z njim nekajkrat počasnejša in v nekaterih orodjih ni podprta. Tudi jeziki, med katerimi prevajamo, predstavljajo različne izzive zaradi svojih specifičnih lastnosti. Raziskave se večinoma osredotočajo na lastnosti, ki izvirajo iz morfologije jezika. Takšni lastnosti sta na primer pregibanje besed (češčina, slovenščina) in sestavljanje besed (nemščina, kitajščina). Obe imata za posledico velik slovar besed (Tamchyna idr., 2017).

1.3 Orodja za računalniško podprto prevajanje

Orodja za računalniško podprto prevajanje (Computer-Aided Translation – CAT) omogočajo uporabo pomnilnikov prevodov – seznamov že nastalih prevodov besed ali besednih zvez, ki jih prevajalec uporablja v nadaljevanju dela, ne omogočajo pa prevajanja neznanih fraz, ki niso na seznamu. Sodobna orodja podpirajo tudi strojno prevajanje. OPUS-CAT MT Engine[1] je primer uporabniku prijaznega sistema za strojno prevajanje, ki ga lahko uporabljamo na računalniku s sistemom Windows. Za ta sistem je na voljo vtičnik za različna orodja CAT, kot so Trados Studio, memoQ in OmegaT. S tem lahko strojno prevajanje približamo poklicnim prevajalcem. Naloga prevajalca je v tem primeru popravljanje napak, ki se pojavijo v strojnih prevodih. Če so strojni prevodi v osnovi dovolj dobri, prevajalec porabi manj časa za popravljanje, kot bi ga potreboval za ročno prevajanje (Etchegoyhen idr., 2014). Vtičnik prevajalcu omogoča tudi učenje modelov za strojno prevajanje oz. njihovo prilagajanje na izbrano domeno. To pa hkrati pomeni, da mora prevajalec poznati osnovne tehnične izzive učenja modelov. Nekatere med njimi naslavljamo v tem članku.

Takšen način uporabe strojnega prevajanja je smiseln tudi v primeru zaupnih besedil, saj le-ta ne zapustijo računalnika prevajalca. To pa ne drži v primeru uporabe spletnih storitev za prevajanje.

1 https://helsinki-nlp.github.io/OPUS-CAT/

Še vedno je za tipičnega prevajalca uporaba lastnega strojnega prevajalnika morda bolj zahtevna kot uporaba preprostih pomnilnikov prevodov. Del izziva je izbira pravega modela prevajanja. Želimo, da daje model dovolj kakovostne prevode, hkrati pa moramo zagotoviti, da ga uporabnik sploh lahko zažene na svoji strojni opremi.

1.4 Struktura članka

V članku bomo predstavili različne podatkovno vodene metode za razcepljanje besed, s katerimi zmanjšamo velikost slovarja in v celoti pokrijemo jezik. Izbrali smo metode, ki so dobro znane in uveljavljene, vendar temeljijo na različnih optimizacijskih kriterijih. Te metode bomo uporabili na primeru strojnega prevajanja med slovenščino in angleščino. Predstavili bomo rezultate uspešnosti prevajanja, hitrosti učenja in prevajanja ter velikosti izdelanih modelov in njihove porabe pomnilnika GPU. Vse metode bomo primerjali z besednim modelom brez razcepljanja.

V diskusiji bomo dodali praktične vidike uporabe takšnih pristopov v strojnem prevajanju ob uporabi lastne strojne opreme.

Ta članek je razširjena verzija prispevka na konferenci Jezikovne tehnologije in digitalna humanistika 2022 in predstavlja podrobnejšo predstavitev uporabljenih metod, dodatne rezultate in razširjeno diskusijo.

2 Slovarske enote v strojnem prevajanju

Najbolj intuitivna izbira slovarske enote prevajalnika je beseda, ki je najpogosteje izbrana osnovna enota v drugih postopkih jezikovnih tehnologij. Prinaša pa številne izzive. Za dovolj dobro pokritost besedišča jezika so potrebni veliki slovarji, kar je še posebej izrazit problem pri visoko pregibnih jezikih, kot je slovenščina. Posledica premajhnih slovarjev je visok delež neznanih besed oz. besed izven slovarja, ki močno zmanjša kakovost prevodov.

Za obvladovanje omenjenih težav so bile predlagane različne alternativne slovarske enote. Kot najmanjša slovarska enota je bila uporabljena črka oz. znak, ki se je izkazal kot zelo robustna enota, manj občutljiva na šum in razlike v domeni učnega in testnega korpusa (Heigold, Varanasi, Neumann, & van Genabith, 2018; Gupta, Besacier,

Dymetman, & Gallé, 2019). Potrebne so določene prilagoditve v arhitekturi nevronske mreže, saj je dolžina segmenta nekajkrat daljša od segmenta, ki kot slovarske enote uporablja besede. Posledica je slabše modeliranje odvisnosti na velikih razdaljah v besedilih.

Preizkušene so bile tudi slovarske enote, ki so po velikosti med črko in besedo – podbesedne enote. Podbesedne enote, dobljene s podatkovno vodenim razcepljanjem, ki kot enoto ohranja pogosta zaporedja črk, so se v splošnem izkazale kot najbolj učinkovite, saj v večji meri ohranjajo sintaktične in semantične lastnosti (Sennrich idr., 2016; Banerjee in Bhattacharyya, 2018). Ker je beseda lahko razcepljena na več različnih načinov, je bila predlagana metoda regulacije razcepljanja (Kudo, 2018). Kot slovarsko enoto bi lahko uporabili tudi jezikoslovno enoto morfem, vendar bi za pravilno delitev besed v algoritmu razcepljanja na morfeme potrebovali slovnično znanje. Podobno lahko rečemo za delitev besed na zloge.

Smiselnost razcepljanja besed na manjše slovarske enote kažejo tudi druga sorodna dela, ki se lotevajo nevronskega prevajanja za morfološko bogate jezike (Tukeyev idr., 2020; Marco idr., 2022).

Osnova za učenje modelov prevajanja so vzporedni korpusi besedil, ki vsebujejo poravnane segmente besedila. Poravnanost segmentov pomeni, da imata oba segmenta enak pomen in sta prevod drug drugega. Običajno so segmenti posamezne povedi, vendar so pogosti primeri, kjer je ena poved v nekem jeziku poravnana z dvema ali celo več povedmi v drugem jeziku. Korpus ni uporaben le za učenje prevajalnika, ampak tudi za učenje modelov za razcepljanje besed na podbesedne enote.

V nadaljevanju opisujemo različne postopke razcepljanja, ki smo jih uporabili v članku. Pri tem bo poudarek na postopku BPE, ki se najpogosteje uporablja. Vsi opisani postopki so podatkovno vodeni. To pomeni, da so modeli za razcepljanje besed naučeni izključno na korpusu, sami postopki pa so neodvisni od jezika.

2.1 Postopek Byte-Pair Encoding

Postopek BPE (Byte-Pair Encoding) je bil prvotno razvit kot postopek za stiskanje podatkov, ki deluje z iterativno zamenjavo najpogostejših

parov simbolov z novimi posameznimi simboli. Sennrich in drugi (Sennrich idr., 2016) so ta algoritem priredili za namen razcepljanje besed.

V postopku se najprej inicializira slovar, ki vsebuje vse črke in druge znake (številke, ločila, matematični simboli itd.), ki se pojavijo v korpusu, ter posebni simbol za konec besede. Vsebina korpusa se tako obravnava kot zaporedje simbolov, ki so v prvem koraku le črke in drugi znaki. Nato sledi iterativni postopek, v katerem algoritem poišče najpogostejši par zaporednih simbolov in se le-ta povsod nadomesti z novim simbolom. Te iterativne korake imenujemo združevanja, pri čemer pogostost parov simbolov izračunavamo na celotnem učnem korpusu. Pri postopku preskočimo združevanja, ki bi vključevala simbol za konec besede, kar v končnem korpusu prepreči združevanje besed, namesto njihovega razcepljanja. Tako nastane končni slovar podbesednih enot (združeni simboli) ter model, ki vsebuje seznam združevanj.

Parameter postopka je število združevanj in neposredno vpliva na velikost slovarja. Natančna velikost slovarja je enaka vsoti števila združevanj in števila simbolov v začetnem koraku. Dovolj veliko število združevanj pomeni, da se nekatere besede v celoti združijo nazaj v en simbol, enak izvorni besedi.

Najpogostejše besede v korpusu se pri uporabi modela ohranijo kot celote, redkejše pa se razcepijo na enote iz nastalega slovarja. Ker so v tem slovarju tudi posamezne črke, je skoraj zagotovljeno, da bo vsaka beseda ustrezno razcepljena in bo delež enot izven slovarja enak nič. Izjeme so zelo redke in se lahko pojavijo, ko v testnem besedilu zasledimo znak, ki se v učnem korpusu ne pojavi.

1:	d-r-ž-a-v-e	č-l-a-n-i-c-e	b-o-d-o	p-r-e-g-l-e-d-a-l-e	s-e-z-n-a-m-e	i-n	o-d	i-z-d-a-j-a-t-e-l-j-i-c-e
2:	d-r-ž-a-v-e	č-l-a-n-i-c-e	b-o-d-o	pr-e-g-l-e-d-a-l-e	s-e-z-n-a-m-e	i-n	o-d	i-z-d-a-j-a-t-e-l-j-i-c-e
5:	d-r-ž-a-v-e	č-l-an-i-c-e	b-o-d-o	pr-e-g-l-e-d-a-l-e	s-e-z-n-a-m-e	i-n	o-d	i-z-d-a-j-a-t-e-l-j-i-c-e
10:	d-r-ž-av-e	č-l-an-i-c-e	b-o-d-o	pr-e-g-l-ed-a-l-e	s-e-z-n-a-m-e	i-n	o-d	i-z-d-a-j-a-t-e-l-j-i-c-e
20:	d-r-ž-av-e	č-l-an-i-c-e	b-od-o	pr-e-g-l-ed-a-l-e	s-e-z-na-m-e	in	o-d	i-z-d-a-j-a-te-lj-i-c-e
50:	dr-ž-av-e	č-l-an-i-c-e	b-od-o	pre-g-l-ed-al-e	se-z-na-m-e	in	o-d	iz-d-a-j-a-te-lj-i-c-e
100:	držav-e	č-l-an-i-c-e	b-od-o	pre-g-l-ed-al-e	se-z-na-m-e	in	o-d	iz-da-ja-te-lj-i-c-e
200:	držav-e	član-ic-e	b-od-o	pre-g-led-al-e	se-z-na-m-e	in	o-d	iz-da-ja-te-lj-ic-e
500:	države	članice	bodo	pre-gled-al-e	se-z-na-me	in	od	iz-da-ja-te-lj-ic-e
1000:	države	članice	bodo	pregled-ale	se-z-na-me	in	od	iz-da-ja-te-lj-ic-e
2000:	države	članice	bodo	pregled-ale	se-zna-me	in	od	iz-da-ja-telj-ic-e
5000:	države	članice	bodo	pregled-ale	sezna-me	in	od	izda-ja-telj-ice
10000:	države	članice	bodo	pregled-ale	sezna-me	in	od	izda-ja-telj-ice

Slika 1: Ponazoritev delovanja postopka BPE na izbranem segmentu besedila z različnim številom združevanj od 1 do 10.000.

Na Sliki 1 so prikazani rezultati razcepljanja izbranega segmenta besedila s postopkom BPE, če uporabimo različna števila združevanj pri

učenju modela. Meje med podbesednimi enotami so na sliki prikazane z vezaji. Brez združevanj se vse besede najprej razdelijo na posamezne črke. V prvi vrstici na sliki je rezultat po enem koraku združevanja. Ker je najpogostejši par zaporednih simbolov »p« in »o«, ki se v tem segmentu besedila ne pojavi, so vse besede še vedno razdeljene na črke. Drugi najpogostejši par simbolov v učnem korpusu je »p« in »r«, kar lahko opazimo v drugi vrstici na sliki, kjer sta ta dva simbola združena v besedi »pregledale«. Z večanjem števila združevanj se začnejo združevati še nadaljnji simboli. Na primer, med vrsticama, ki predstavljata 500 in 1000 združevanj, lahko v besedi »pregledale« vidimo, da sta se združila para simbolov »pre« in »gled« ter »al« in »e«.

Avtorji v (Sennrich idr., 2016) so predstavili implementacijo algoritma in predlagali možnost skupnega učenja razcepljanja (Joint BPE), kjer uporabimo besedilo v obeh jezikih vzporednega korpusa kot učno gradivo za model razcepljanja. Tako tvorimo en model, ki vsebuje vsa združevanja, in po en slovar za vsak jezik v paru. Tako kot prej pri tem postopku nastavimo število združevanj in s tem skupno število združenih simbolov. Ni nujno, da se vsi združeni simboli pojavijo v obeh jezikih. Posledično sta slovarja za oba jezika manjša od števila združevanj.

2.2 Orodje Morfessor

Program Morfessor (Creutz in Lagus, 2002) je bil razvit v želji po razcepljanju besed v kompleksnih jezikih na podbesedne enote, ki približno ustrezajo morfemom – najmanjšim enotam besede, ki nosijo pomen. Želja je bila, imeti podatkovno voden postopek, ki deluje za več jezikov brez dodatnega slovničnega znanja in zgradi slovar jezikovnih enot, ki je manjši in bolj splošen kot slovar besed.

Predpostavka delovanja algoritma je, da so besede sestavljene iz zaporedja več segmentov, kot je to tipično v aglutinativnih jezikih. Razvita sta bila dva algoritma. Prvi temelji na principu najkrajše dolžine opisa, drugi pa na principu največje verjetnosti. Uporabili smo prvega.

Cilj algoritma je najti slovar podbesednih enot, ki daje optimalno vrednost funkcije cene (cost function), ki vsebuje dva dela: ceno izvornega besedila T in ceno slovarja V. Ceno opišemo z

$$C = C(T) + C(V) = -\sum_{m_i \in T} \log p(m_i) + \sum_{m_j \in V} k \cdot l(m_j),$$

kjer je *m* podbesedna enota, *T* je besedilo, *V* je slovar, $l(m_j)$ je dolžina podbesedne enote m_j (število črk) in *k* je število bitov, ki so potrebni za predstavitev ene črke in se lahko postavi na 5. Verjetnost posamezne podbesedne enote v besedilu $p(m_i)$ izračunamo z oceno največje verjetnosti kot razmerje med absolutno frekvenco te enote in številom vseh enot v besedilu.

V svojem delu smo uporabljali novejšo implementacijo programa – Morfessor 2.0 (Virpioja idr., 2013). Iskalni algoritem v tej implementaciji poišče nabor podbesednih enot, ki optimizirajo funkcijo cene, pri tem lahko ročno izbiramo uteži za obe komponenti funkcije cene ali pa izberemo želeno velikost slovarja. Mi smo v vseh primerih uporabe izbrali velikost slovarja.

2.3 Unigramski modeli

Zadnja metoda, ki smo jo pogledali, je razcepljanje besed na podlagi uporabe unigramskega modela (Kudo, 2018). V unigramskem modelu je verjetnost besedila oz. zaporedja podbesednih enot $\boldsymbol{x}=(x_1, x_2 \ldots x_M)$ modelirana kot produkt verjetnosti posameznih enot tega zaporedja:

$$P(\boldsymbol{x}) = \prod_{i=1}^{M} p(x_i),$$

kjer je *M* dolžina besedila in $p(x_i)$ verjetnost *i*-te enote v besedilu. Pri tem so vse podbesedne enote v slovarju in vsota verjetnost vseh enot mora biti enaka 1.

Najverjetnejše razcepljanje x^* besed vhodnega besedila *X* je tisto, za katerega velja

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in S(X)}{arg\,max}\, P(\boldsymbol{x}),$$

kjer je *S*(*X*) množica vseh možnih razcepljanj besed v besedilu *X*.

Verjetnosti posameznih unigramov podbesednih enot lahko določimo z algoritmom EM (Expectation Maximization), optimalno razcepljanje besed pa najdemo z Viterbijevim algoritmom (Kudo, 2018).

Primer implementacije opisanega postopka najdemo v orodju Sentence Piece (Kudo in Richardson, 2018), v katerem so implementirani še drugi postopki, vključno z BPE. Tudi v tem orodju lahko izhajamo iz želene velikosti končnega slovarja.

2.4 Izbira in primerjava postopkov

Pri izvedbi eksperimentov smo izbrali 4 kombinacije razcepljanja besed in uporabljenega orodja:

- Joint BPE – postopek BPE s skupnim učenjem združevanja na vzporednem korpusu in implementacijo Rica Sennricha, imenovano Subword NMT.
- MRF – postopek na principu najkrajše dolžine opisa, kjer se uteži v funkciji cene prilagodijo glede na želeno velikost slovarja in implementacijo Morfessor 2.0.
- SP-BPE – postopek BPE z ločenim učenjem združevanja na vsaki strani vzporednega korpusa in implementacijo v orodju SentencePiece.
- SP-UG – postopek na podlagi unigramskih jezikovnih modelov in implementacijo v orodju SentencePiece.

Dodatno smo vse eksperimente ponovili še z modelom, v katerem ne uporabljamo razcepljanja besed.

3 Eksperimentalni sistem

3.1 Korpusi

Eksperimenti so bili izvedeni na prosto dostopnem vzporednem korpusu ParaCrawl, različica 7.1 (Bañón idr., 2020). Korpus je bil zgrajen iz besedil s spletnih strani, ki imajo dovolj dvojezične vsebine. Strani in besedila so bili najdeni s pomočjo obstoječih statistik organizacije CommonCrawl. Korpus je bil samodejno poravnan. Za jezikovni par slovenščina-angleščina vsebuje približno 3,7 milijona poravnanih

segmentov, kar predstavlja 60,9 milijona besed na slovenski in 65,5 milijona besed na angleški strani. Do danes je bil korpus ParaCrawl že razširjen in za jezikovni par slovenščina-angleščina vsebuje približno 7,5 milijona segmentov.

Korpus smo razdelili na 3 dele: učni korpus, razvojni korpus in testni korpus. Razvojni korpus je namenjen sprotni validaciji tekom učenja strojnega prevajalnika, testni korpus pa končnemu testiranju in vrednotenju rezultatov. Za vsakega izmed teh dveh korpusov smo izbrali 2.000 naključnih segmentov besedila iz izvornega korpusa. Preostanek je bil uporabljen kot učni korpus.

3.2 Predprocesiranje

Nad vsemi deli korpusov smo izvedli standardno predprocesiranje za strojno prevajanje: čiščenje, normalizacijo ločil, tokenizacijo in *truecasing*. Učni korpus smo uporabili tudi za učenje modela za *truecasing*.

V koraku čiščenja smo iz datotek izločili odvečne presledke, prazne vrstice in vrstice, ki presegajo določeno dolžino (število besed) ali pa presegajo razmerja med številom besed v tem segmentu v enem jeziku in številom besed v istem segmentu v drugem jeziku. Takšni segmenti bi lahko ovirali algoritem učenja. Velikost predprocesiranega korpusa je zato rahlo manjša od velikost izvornega korpusa.

V koraku normalizacije ločil smo uredili stičnost ločil ter zamenjali nekatera ločil, ki jih ne najdemo v naboru ASCII znakov, z ustreznimi ločili iz nabora. S tem se zmanjšata raznolikost ločil in pomanjkanje podatkov za nekatere redke pojavnice v jeziku, kar lahko predstavlja težavo pri učenju velikega števila parametrov v modelih.

V koraku tokenizacije smo besedilo razdeli na pojavnice tako, da smo vrinili presledke med besede in stična ločila, nismo jih pa vrinili med besede in pike, ki skupaj predstavljajo krajšave, kot so: npr., dr., št. itd., saj krajšave predstavljajo eno besedo in jih želimo po tokenizaciji obdržati kot eno pojavnico.

V koraku *truecasing smo* odpravili velike začetnice na začetku povedi, razen če so te besede lastna imena ali druge besede, ki jih vedno pišemo z veliko začetnico (npr. svojilni pridevniki ali v angleščini zaimek »I«). Ponovno je namen zmanjšanje pomanjkanja podatkov.

```
or: Dr. Ivan Gašparovič, predsednik Slovaške republike (julij 2005) — skulptura Venetski konj.
nm: Dr. Ivan Gašparovič, predsednik Slovaške republike (julij 2005) - skulptura Venetski konj.
tk: Dr. Ivan Gašparovič , predsednik Slovaške republike ( julij 2005 ) - skulptura Venetski konj .
tc: dr. Ivan Gašparovič , predsednik Slovaške republike ( julij 2005 ) - skulptura Venetski konj .
```

Slika 2: Segment besedila iz slovenske učne množice v različnih korakih predprocesiranja: or – originalni zapis iz korpusa, nm – zapis po normalizaciji ločil, tk – zapis po tokenizaciji, tc – zapis po truecasing.

Na Sliki 2 je prikazan primer segmenta iz korpusa v posameznih korakih predprocesiranja, kjer vidimo spremembo pomišljaja v vezaj, spremembo stičnosti ločil in odpravo velike začetnice na začetku povedi. Končne velikosti vseh predprocesiranih korpusov so predstavljene v Tabeli 1.

Tabela 1: *Število segmentov besedila v učnem, razvojnem in testnem korpusu*

Korpus	Število segmentov
Učni	3.714.473
Razvojni	1.987
Testni	1.990
Skupaj	3.718.450

Večina korakov predprocesiranja je namenjena zmanjšanju pomanjkanja podatkov. Besede, ki imajo vedno isti pomen, se brez predprocesiranja lahko pojavijo v različnih oblikah glede na to, ali so na začetku povedi (velika začetnica) ali ne in ali ob njih stoji stično ločilo. Prevajalnik ločuje med temi oblikami zapisa. Težave se lahko pojavijo pri redkejših besedah, saj se morda v učni množici ne pojavijo v vseh oblikah dovolj pogosto.

Po prevajanju dobimo rezultat v *truecase* obliki, zato je potrebno postprocesiranje prevedenega besedila, kjer obnovimo velike začetnice na začetkih povedi in opravimo detokenizacijo tako, da poskrbimo za ustrezno stičnost ločil.

3.3 Razcepljanja besed

Pri razcepljanju besed smo uporabili orodja, ki so opisana v prejšnjem poglavju. Učni del korpusa smo uporabili za učenje modela razcepljanja, nato smo naučene modele uporabili za razcepljanje vseh delov korpusa. Tako smo dobili različice razcepljenih korpusov.

Ker smo izhajali iz želje po različnih velikostih končnih slovarjev, smo spreminjali ustrezne parametre pri uporabi orodij za učenje razcepljanja. Pri tem orodja uporabljajo te parametre na različne načine, kar pomeni, da velikosti končnih slovarjev niso natančno ustrezale nastavljenim vrednostim parametrov. Želene vrednosti, ki smo jih nastavili, so: 10.000, 15.000, 20.000, 25.000, 30.000, 40.000, 50.000, 60.000, 80.000, 100.000, 120.000 in 150.000. V Tabeli 2 so prikazane natančne velikosti slovarjev, ki jih dobimo na slovenski in angleški učni množici pri teh nastavitvah.

Tabela 2: *Velikost izdelanih slovenskih (sl) in angleških (en) slovarjev*

Nastavitev velikosti	Joint BPE (sl)	MRF (sl)	SP-BPE (sl)	SP-UG (sl)	Joint BPE (en)	MRF (en)	SP-BPE (en)	SP-UG (en)
10k	11.384	18.670	17.064	17.814	11.556	18.405	16.909	17.358
15k	16.273	28.251	25.525	26.716	15.739	27.822	25.455	26.177
20k	21.101	37.934	33.561	35.534	19.631	37.664	33.595	34.779
25k	25.883	46.879	41.297	44.175	23.299	46.298	41.395	43.051
30k	30.625	55.438	48.822	52.717	26.760	55.994	48.946	51.204
40k	39.890	73.766	63.478	69.530	33.593	73.960	63.132	66.839
50k	49.063	93.726	77.520	86.111	40.115	90.248	76.515	82.082
60k	58.155	109.989	91.015	102.242	46.404	105.558	89.312	96.924
80k	76.152	143.018	117.134	133.892	58.788	133.679	113.496	125.572
100k	93.938	174.026	142.294	164.877	71.043	159.419	136.198	153.190
120k	111.646	205.658	166.442	195.155	82.972	182.895	157.987	180.256
150k	138.006	238.620	201.334	239.515	101.013	210.425	188.859	218.140

Na Sliki 3 je prikazan primer segmenta, kjer smo besede razcepili z uporabo vseh štirih postopkov. Uporabili smo ciljno velikost slovarja 20.000, saj so pri tej velikosti razcepljanja besed bolj pogosta in lažje prikažemo več razlik v enem segmentu. Na sliki so mesta delitve besed nakazana z vezaji.

```
Joint BPE:     države  članice  bodo pregle-dale  sezna-me in od izdaja-te-ljice ...
Morfessor:     držav-e članic-e bodo pregled-a-le seznam-e in od izdajatelj-ice  ...
SP - BPE:      države  članice  bodo pregleda-le  sezna-me in od izdaja-telji-ce ...
SP - Unigram:  države  članice  bodo pregleda-le  seznam-e in od izdajatelj-ice  ...
```

Slika 3: Primer segmenta besedila iz testne množice z razcepljenimi besedami.

V Tabeli 3 imamo navedena števila razcepljanj na testni množici. V primeru, da je bila beseda razcepljena na dve podbesedni enoti, to štejemo kot eno razcepljanje. Če je bila razdeljena na več podbesednih enot, ustrezno štejemo več razcepljanj. Iz podatkov vidimo padanje števila razcepljanj z večanjem nastavitve velikosti slovarja. Po pričakovanjih vidimo več razcepljanj na slovenskem besedilu, kot na angleškem.

Slovenska testna množica vsebuje 36.994 pojavnic, angleška pa 39.874. Glede na podatke v tabeli lahko ugotovimo delež razcepljanj glede na število pojavnic. Pri velikostih slovarjev okoli 100.000 na slovenskem besedilu oz. 60.000 na angleškem pade ta delež pod 10 %. Izjema je pri razcepljanjih z orodjem Morfessor, kjer je ta delež še vedno večji.

Tabela 3: *Število razcepljanj besed na testni množici*

Nastavitev velikosti	Joint BPE (sl)	MRF (sl)	SP-BPE (sl)	SP-UG (sl)	Joint BPE (en)	MRF (en)	SP-BPE (en)	SP-UG (en)
10k	21.317	26.579	17.802	17.715	14.786	17.045	12.546	11.514
15k	16.893	23.086	13.816	13.688	11.198	13.608	9.215	8.518
20k	14.035	20.970	11.461	11.510	9.019	11.504	7.503	7.005
25k	12.045	19.450	9.893	10.028	7.634	10.088	6.430	6.115
30k	10.588	18.362	8.781	8.980	6.715	9.347	5.680	5.507
40k	8.559	16.266	7.227	7.533	5.388	7.914	4.789	4.732
50k	7.186	15.079	6.187	6.646	4.474	7.162	4.217	4.271
60k	6.185	13.852	5.413	5.965	3.830	6.145	3.834	3.983
80k	4.862	12.673	4.455	5.161	3.034	5.469	3.358	3.612
100k	3.963	10.874	3.803	4.655	2.482	4.699	3.082	3.436
120k	3.374	9.998	3.367	4.370	2.090	4.422	2.900	3.345
150k	2.713	9.459	2.910	4.084	1.697	4.003	2.742	3.249

V modelih brez razcepljanja besed smo uporabili natančne velikosti slovarjev: 60.000, 80.000, 100.000, 125.000, 150.000, 200.000, 250.000 in 300.000.

V naslednjem koraku smo zgradili slovarje za vse različice razcepljenih učnih korpusov in za nerazcepljen besedni učni korpus. Medtem ko v razcepljenih korpusih slovarji pokrijejo celotni korpus, se pri besednem korpusu pojavijo besede izven slovarja. V Tabeli 4 smo prikazali deleže besed izven slovarja (Out of Vocabulary – OOV) na testnem delu korpusa za oba jezika. Po pričakovanjih vidimo, da so deleži večji na slovenski strani in da padajo z večanjem slovarja.

Tabela 4: *Delež besed izven slovarja pri besednih slovarjih na slovenskem (sl) in angleškem (en) testnem korpusu*

Slovar	OOV (sl) [%]	OOV (en) [%]
60k	6,66	2,57
80k	5,38	2,07
100k	4,44	1,77
125k	3,74	1,50
150k	3,22	1,30
200k	2,53	1,08
250k	2,11	0,95
300k	1,82	0,85

3.4 Prevajalnik

Model prevajalnika je v vseh primerih nevronski strojni prevajalnik. Izbrali smo model *Amun* v orodju Marian NMT, ki sledi pristopu, opisanem v (Bahdanau, Cho, & Bengio, 2014), in temelji na arhitekturi RNN s celicami GRU (gated recurrent unit) z dimenzijo skritega stanja 1024 in dimenzijo vgrajenih vektorjev 512 (privzete nastavitve orodja). Naše dosedanje izkušnje na tej množici so kazale, da z uporabo arhitekture transformer in samo-pozornosti ne dosežemo bistvenih izboljšav. V nadaljevanju članka opisujemo tudi eksperimente in rezultate, ki pojasnjujejo te izkušnje. Dolžine segmentov besedila smo pri učenju omejili na 80 pojavnic (besed in ločil oz. podbesednih enot in ločil), kar pomeni, da upoštevamo 99,7 % vseh segmentov v učni množici brez razcepljanja. Pri modelih, kjer uporabljamo razcepljanje, tako upoštevamo med 96,3 % in 99,5 % vseh segmentov. Omejitve dolžine segmentov nismo povečevali, saj glede na omenjeno pokritost predvidevamo, da ne bi prišlo do znatnih sprememb rezultatov.

Učenje smo izvajali 10 epoh s preverjanjem rezultata na razvojni množici na vsakih 100 posodobitev parametrov modela. Najboljši model glede na razvojno množico smo nato uporabili pri vrednotenju rezultatov na testni množici.

Pri prevajanju smo uporabljali mini serije (mini-batch) velikosti 64, medtem ko je pri učenju uporabljena fleksibilna velikost, ki je prilagojena velikosti delovnega pomnilnika enote GPU, na kateri izvajamo učenje. Izbrali smo ciljno porabo pomnilnika 20.000 MiB. Pri tem so bile

povprečne velikosti mini serij za različne modele odvisne predvsem od velikost slovarja. Najmanjše mini serije so bile v povprečju dolge 124 segmentov pri besednem slovarju s 300.000 besedami in obema smerema prevajanja, najdaljše pa 1132 pri modelu s skupnim razcepljanjem BPE pri prevajanju iz angleščine v slovenščino in slovarjem z 20.000 podbesednimi enotami.

3.5 Ocenjevanje uspešnosti prevajanja

Uspešnost prevajanja oz. kakovost prevodov smo ocenjevali z metriko BLEU, ki omogoča avtomatsko vrednotenje in se v praksi tudi največ uporablja (Papineni idr., 2002). BLEU neodvisno oceni prevod vsakega segmenta, ocena BLEU celotnega testnega korpusa je uteženo geometrijsko povprečje ocen posameznih segmentov. Ocena segmenta je enaka modificirani natančnosti (precision) ujemanja n-gramov (zaporedij pojavnic dolžine n) med strojnim in referenčnim prevodom. Izračun natančnosti je modificiran tako, da se pri štetju ponovitev n-gramov upošteva največ toliko ponovitev, kot jih je prisotnih v referenčnem prevodu. Privzeto se uporabljajo n-grami do reda 4.

Ocena segmenta favorizira kratke prevode, zato je metriki dodana kazen prekratkih prevodov (brevity penalty).

3.6 Orodja

Za predprocesiranje (čiščenje, normalizacijo, tokenizacijo in *truecasing*) ter postprocesiranje (*detruecasing* in detokenizacijo) smo uporabljali skripte iz programskega paketa MOSES (Koehn idr., 2007). Za učenje prevajalnikov in prevajanje smo uporabljali orodje Marian NMT (Junczys-Dowmunt idr., 2018), ki smo ga poganjali na grafičnih procesnih enotah Nvidia Tesla V100. Za vrednotenje rezultatov z metriko BLEU smo uporabljali orodje SacreBLEU (Post, 2018), ki kot del vrednotenja izvaja ponovno tokenizacijo in vrednoti tokenizirana besedila. Orodja in algoritmi za razcepljanje besed na podbesedne enote so opisani v poglavju 2.

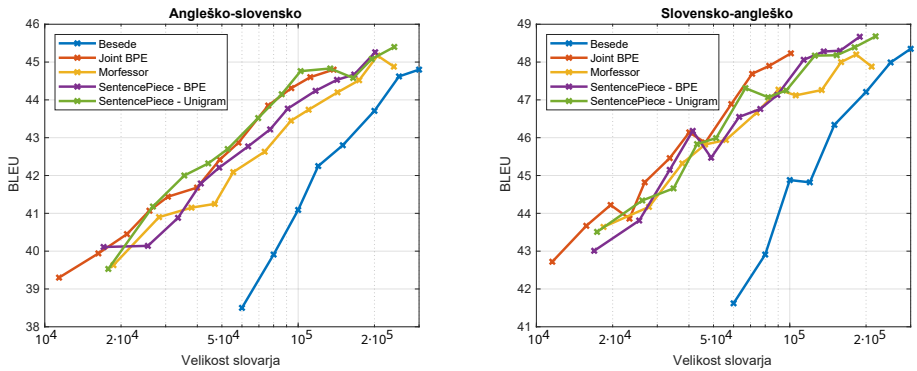
4 Rezultati

4.1 Uspešnost prevajanja

Ker je bil osnovni namen uporabe podbesednih enot zmanjšanje velikosti slovarja in s tem izvedljivost uporabe nevronskih strojnih prevajalnikov, najprej prikazujemo primer rezultatov na tipičnih velikostih slovarjev. Za besedni slovar smo izbrali velikost 60.000 besed. Velikosti med 60.000 in 65.000 so pogosto uporabljene v procesiranju naravnega jezika. V Tabeli 5 primerjamo rezultate prevajanja med besednim modelom in modelom Joint BPE z enako velikostjo slovarja. Vidimo izboljšanje uspešnosti prevajanja z uporabo podbesednih enot, kot jo tipično zasledimo v obstoječi literaturi, npr. v (Sennrich idr., 2016). Na tej točki smo še dodali rezultate vrednotenja, ki jih dobimo z metriko ChrF (β = 3) (Popović, 2015). Čeprav se ta metrika uveljavlja za vrednotenje prevajanja pri morfološko kompleksnih jezikih, smo preostale rezultate predstavili le z metriko BLEU, ki je še vedno uveljavljena in zadostuje za medsebojno primerjavo naših modelov.

Tabela 5: *Primer rezultatov uporabe besednega modela in modela z uporabo Joint BPE pri slovarju velikost 60.000*

Model	BLEU (en-sl)	BLEU (sl-en)	ChrF (en-sl)	ChrF (sl-en)
Besedni	38,50	41,62	58,43	60,68
Joint BPE	42,87	45,87	63,13	65,76

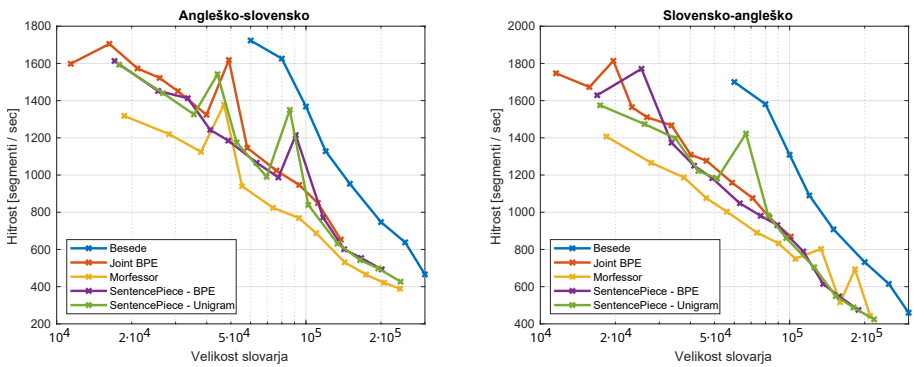


Slika 4: Rezultati uspešnosti prevajanja za vse modele.

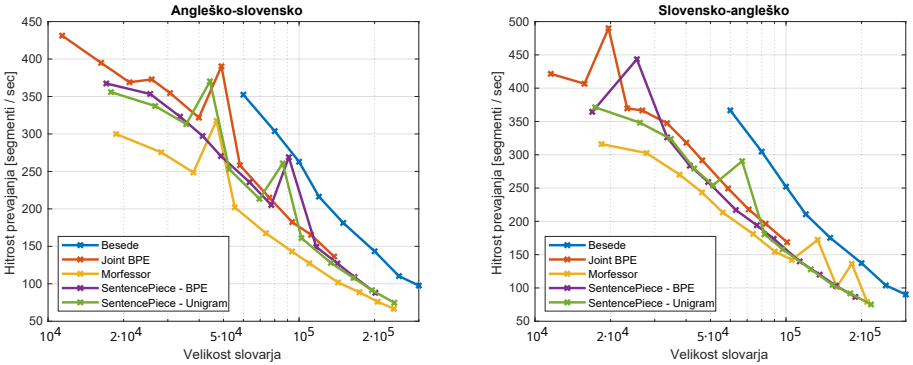
Na Sliki 4 so prikazani rezultati uspešnosti prevajanja v odvisnosti od velikosti slovarja za vse sisteme. Na slikah so velikosti slovarjev ponazorjene v logaritemskem merilu. Pri tem smo vedno upoštevali dejansko velikost slovarja, kot je razvidna iz Tabele 2.

Na splošno lahko opazimo, da uspešnost prevajanja narašča z večanjem slovarjev tako pri sistemih brez razcepljanja kot pri sistemih z razcepljanjem. Nekateri sistemi odstopajo od tega trenda, npr. prevajalnik iz slovenščine v angleščino s slovarjem 120.000 besed, ki daje slabši rezultat kot sistem slovarjem 100.000 besed. Pomemben rezultat, ki ga opazimo, je ta, da uspešnost prevajanja pri uporabi besednih modelov narašča hitreje in se pri največjih slovarjih precej približa uspešnosti prevajalnikov z razcepljanjem.

Ko primerjamo sisteme, ki uporabljajo različne načine razcepljanja besed, vidimo manjše razlike. Kljub temu lahko opazimo, da pri prevajanju iz angleščine v slovenščino večinoma daje najboljše rezultate orodje Sentence Piece z razcepljanjem na osnovi unigramov (Sentence Piece – Unigram), v nasprotni smeri pa orodje Subword NMT s skupnim učenjem (Joint BPE).



Slika 5: Hitrost učenja prevajalnika za vse modele.



Slika 6: Hitrost prevajanja pri uporabi različnih modelov.

4.2 Hitrosti učenja in prevajanja

Slika 5 prikazuje hitrost učenja modela prevajalnika, Slika 6 pa hitrost prevajanja pri njegovi uporabi. Vsi rezultati so dobljeni pri uporabi grafične procesne enote. V vseh primerih uporabljamo kot merilo za hitrost število obdelanih segmentov besedila na sekundo, saj se zaradi različnih razcepljanj število pojavnic razlikuje med sistemi. Število besed na sekundo pri učenju dobimo, če upoštevamo, da je povprečno število besednih pojavnic (besed in ločil) na segment v slovenskem besedilu 20,2, v angleškem besedilu pa 18,7.

Opazimo lahko, da se vse hitrosti zmanjšujejo z večanjem slovarja. Hitrost pri besednih modelih je večja kot hitrost pri ostalih modelih, vendar se tudi tukaj razlika pri večjih slovarjih zmanjšuje. Vidimo, da so najpočasnejši modeli tisti, ki za razcepljanje korpusa uporabljajo orodje Morfessor, vendar so počasnejši le približno 10 do 20 % glede na druge modele z razcepljanjem.

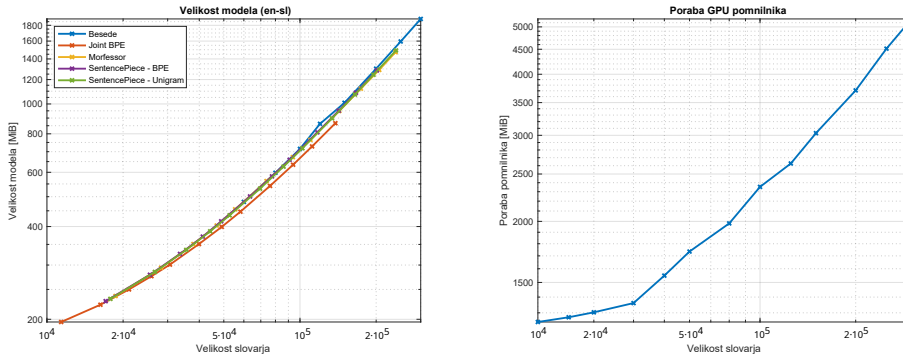
V rezultatih opazimo več točk, ki močno odstopajo od trendov. Predvidevamo, da so odstopanja nastala zaradi naključnih začetnih nastavitev nekaterih parametrov pri učenju, morebitnih odstopanj na uporabljeni strojni opremi in specifičnih lastnosti programske opreme za učenje modelov, med prilagajanjem velikosti mini serije pri različnih velikostih slovarja ali drugih vplivov.

Prikazane hitrosti prevajanja ne upoštevajo predprocesiranja in postprocesiranja besedila, ki zahtevata bistveno manj časa. Osnovno

predprocesiranje in postprocesiranje delujeta na sodobnem osebnem računalniku s hitrostjo približno 10.000 segmentov na sekundo.

Hitrost prevajanja se bistveno upočasni, če uporabljamo prevajanje na centralnih procesnih enotah računalnika CPU. Hitrost prevajanje iz angleščine v slovenščino pri besednem slovarju z 80.000 besedami je nekaj čez 300 segmentov na sekundo na enoti GPU. Če uporabimo sodobno enoto CPU lahko ta hitrost pade na približno 1 segment na sekundo. Čeprav je takšna hitrost v določenih primerih lahko še sprejemljiva za prevajanje, enote CPU niso uporabne za učenje modelov. Podobno lahko vidimo bistveno zmanjšanje hitrosti tudi pri drugih modelih.

Pri teh analizah je potrebno pripomniti, da so hitrost odvisne od optimizacije programske opreme, strojne opreme in nekaterih nastavitev orodja pri uporabi. V splošnem lahko v prihodnosti pričakujemo povečanje hitrosti prevajanja.



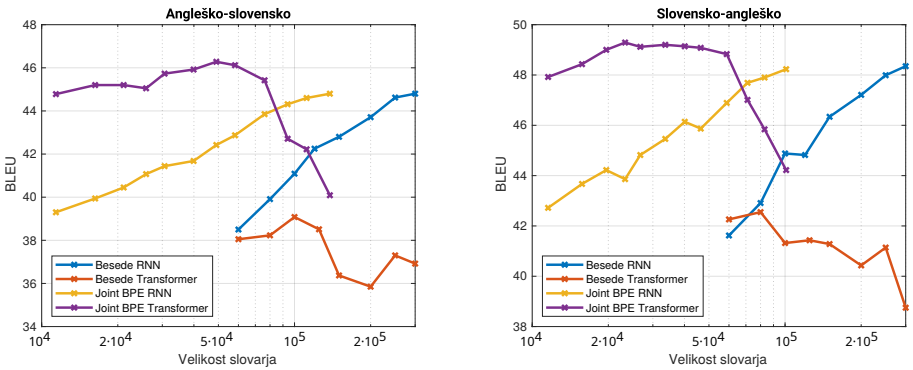
Slika 7: Velikost izdelanega modela za vse modele ter poraba pomnilnika na grafični procesni enoti med prevajanjem.

4.3 Velikosti modelov

Na Sliki 7 so na levi prikazane velikosti datotek za vse modele prevajanja in njihova poraba pomnilnika enote GPU za besedne modele. Velikosti datotek naraščajo skoraj linearno z velikostjo slovarja (na grafu sta obe osi v logaritemskem merilu). Vsakemu modelu sta pridruženi dve datoteki z obema slovarjema, ki sta bistveno manjši. Prikazane velikosti so za modele prevajanja iz angleščine v slovenščino. Modeli v nasprotni smeri imajo primerljive velikosti.

Desno na Sliki 7 je prikazana še poraba pomnilnika pri uporabi modelov pri prevajanju. Vidimo, da ima orodje osnovno porabo pomnilnika, kar se kaže v manjših spremembah porabe pri malih slovarjih. Pri večjih slovarjih poraba pomnilnika linearno narašča. Prikazana je poraba pomnilnika za besedne modele, ki je enaka v obeh smereh prevajanja. Porabe pomnilnika pri drugih modelih so primerljive glede na velikost slovarja. Pri preverjanju porabe pomnilnika je bila uporabljena mini serija velikosti 64. Pri učenju je bila poraba pomnilnika okvirno trikrat večja.

Vsi do sedaj predstavljeni eksperimenti in rezultati so ohranjali arhitekturo samega prevajalnika. V zadnjem delu eksperimentov smo preizkusili še arhitekturo transformer in spremembe vrednosti hiperparametrov modelov.



Slika 8: Primerjava rezultatov med izbranimi modeli z arhitekturo RNN in modeli z arhitekturo transformer.

4.4 Arhitekture in hiperparametri

Na Sliki 8 je prikazana primerjava rezultatov dveh arhitektur modelov pri uporabi besednih modelov in modelov z razcepljanjem BPE s skupnim slovarjem. Arhitekturi sta RNN, za katero so rezultati prikazani že na Sliki 4, in arhitektura transformer. Vidimo, da so rezultati pri uporabi besednih enot in transformer modelov bistveno slabši. Pri uporabi deljenja besed po postopku BPE pa vidimo, da obstaja za velikost slovarja interval, znotraj katerega so modeli boljši od ostalih modelov. Vidimo tudi, da pri zelo velikih slovarjih modeli z arhitekturo RNN dohitevajo modele transformer.

Tabela 6: *Uspešnost prevajanja in velikosti modelov pri uporabi večjih vrednosti hiperparametrov*

Dimenzija vgrajenih vektorjev	Dimenzija skritega stanja	BLEU (en-sl)	BLEU (sl-en)	Velikost modela [MiB]	Poraba pomnilnika GPU pri prevajanju [MiB]	Poraba pomnilnika GPU pri učenju [MiB]
512	1024	44,80	48,35	1888	5.191	18.041
1024	1024	45,40	48,36	3673	7.427	25.171
1024	2048	**46,40**	49,34	4033	8.171	26.617
2048	1024	46,33	**49,59**	7248	11.131	32.229

Rezultati v Tabeli 6 se nanašajo na uporabo različnih nastavitev hiperparametrov modelov. Vsi rezultati v tabeli so dobljeni z modeli z arhitekturo RNN ter besednimi enotami s slovarjem velikosti 300.000. Iz tabele vidimo večanje uspešnost prevajanja pri povečevanju vrednosti hiperparametrov. Krepko označena podatka v tabeli predstavljata najboljše pridobljene rezultate za vsako stran prevajanja v tej raziskavi. Vidimo še, da so ti rezultati boljši kot najboljši rezultati pri uporabi modelov z arhitekturo transformer iz Slike 8, kjer sta najboljša rezultata 46,12 in 49,29.

Sorodne raziskave kažejo velik doprinos uporabe arhitekture transformer (Vaswani idr., 2017), vendar so te raziskave uporabljale deljenje besed in manjše velikosti slovarjev. Naši rezultati prikazujejo, da sta to pomembna faktorja pri uspešnosti teh modelov. V predhodnih eksperimentih smo uporabljali večje slovarje in smo zato dobili vtis, da transformer modeli dajejo slabše rezultate.

Z večanjem hiperparametrov se poveča velikost modela, kot je prikazano v tabeli, in sorazmerno tudi poraba pomnilnika GPU. Še večje vrednosti hiperparametrov pripeljejo do tega, da učenja modelov več ni možno izvajati na izbrani strojni opremi, čeprav bi bila uporaba še vedno možna.

Zmanjšali sta se tudi hitrost učenja in prevajanja. Ti sta pri modelu z največjimi vrednostmi hiperparametrov približno dvakrat manjši kot pa pri uporabi privzetih vrednosti (Slika 6).

5 Diskusija

Za praktično uporabo modelov prevajanja je pomembna lastnosti poraba pomnilnika GPU. Če model potrebuje večji pomnilnik, kot ga je na

voljo, ga ne moremo uporabljati na enoti GPU. Delovanje na centralni procesni enoti s sistemskim delovnim pomnilnikom (RAM) je bistveno počasnejše in predstavlja izzive za tehnično manj izkušene prevajalce.

Rezultati na Sliki 7 kažejo potrebno velikost pomnilnika GPU za uporabo modelov s privzetimi nastavitvami hiperparametrov. Tudi naši največji modeli imajo porabo pomnilnika med prevajanjem pod 6 GiB. Tako velike pomnilnike dandanes najdemo v grafičnih karticah nižjega do srednjega cenovnega razreda. S tega vidika za večino prevajalcev ne bo ovire v uporabi velikih slovarjev. To se spremeni z uporabo večjih vrednosti hiperparametrov nevronske mreže. V eksperimentih na Sliki 7 smo uporabljali RNN z dimenzijo skritega stanja 1024 in dimenzijo vgrajenih vektorjev 512. Z večanjem teh dveh parametrov lahko dosežemo izboljšane rezultate, vendar se sorazmerno poveča tudi velikost modela in s tem poraba grafičnega pomnilnika, kot to kaže Tabela 6. Naši najboljši modeli potrebujejo za delovanje pomnilnik GPU velikosti, ki je tipična za enote GPU višjega cenovnega razreda. V prihodnosti lahko pričakujemo, da bodo enote GPU za uporabo takih modelov zlahka dosegljive tudi na prenosnih računalnikih.

Potrebe po pomnilniku so bistveno večje pri učenju, kjer potrebujemo enote GPU višjega cenovnega razreda. Za naše eksperimente s povečanimi vrednostmi hiperparametrov smo potrebovali profesionalno opremo, ki tipičnemu posameznemu uporabniku ni dosegljiva za nakup.

Več parametrov pomeni, da za oceno njihovih vrednosti potrebujemo večje učne množice, sicer lahko učenje vodi v prekomerno prilegaanje. Kljub temu smo s povečanimi vrednostmi hiperparametrov dobili najboljše rezultate.

Sama hitrost uporabe modela pri prevajanju je po naših izkušnjah odvisna predvsem od pasovne širine za komunikacijo med grafičnim procesorjem in grafičnim pomnilnikom. Običajne potrošniške grafične kartice imajo lahko do petkrat manjšo pasovno širino od enote GPU, ki smo jo uporabljali v tem članku. Hitrosti so še nekoliko manjše pri enotah GPU za prenosne računalnike. Pri najpočasnejših modelih to še vedno pomeni več kot 10 segmentov besedila na sekundo. Potrebni čas za strojno prevajanje tako predstavlja neznatni delež skupnega časa, ki ga prevajalec potrebuje za uporabo prevajalnika in pregled prevodov.

Hitrost je zelo pomembna pri učenju modelov prevajanja. V eksperimentih smo izvajali učenje na korpusu s 3,7 milijona segmentov. Hitrosti učenja so se precej razlikovale. Celotni postopek učenja je trajal približno 6 ur pri najhitrejših in nekaj več kot 1 dan pri najpočasnejših modelih, ki so dajali najboljše rezultate. Na potrošniških grafičnih karticah se ponovno podaljša čas za faktor približno 10.

Strojno prevajanje je možno izvajati tudi brez standardnega predprocesiranja in postprocesiranja, vendar se pri tem lahko uspešnost prevajanja zmanjša. Razlika postaja manjša z večanjem učnih korpusov. S tem zmanjšamo potrebe po dodatnih korakih in orodjih pri uporabi strojnega prevajanja na lastni opremi.

6 Sklep

V članku smo obravnavali problem pokritosti besedišča z različnimi slovarji osnovnih slovarskih enot. To je še posebej pereč problem jezikov z velikim številom pregibnih besednih vrst. Prikazali in primerjali smo nekatere najpogostejše podatkovno vodene metode za razcepljanje besed in njihovo uporabo na primeru nevronskega strojnega prevajalnika. Analiza je pokazala, da vsi štirje načini razcepljanja dajejo primerljive rezultate z le majhnimi razlikami. Pri prevajanju iz angleščine v slovenščino je najboljši Sentence Piece – Unigram, medtem ko pri prevajanju v obratni smeri Subword NMT s skupnim učenjem. Naši rezultati kažejo, da z razcepljanjem besed še vedno dosegamo boljše rezultate kot s prevajalniki brez razcepljanja, tudi v primerih, ko jim večamo slovarje. Trend kaže, da lahko z besednimi prevajalniki z nadaljnjim večanjem slovarja dohitimo modele z razcepljanjem besed. Ob trenutnem trendu razvoja in večanja pomnilniških zmogljivosti enot GPU bo takšne modele v prihodnje možno naučiti in uporabljati na potrošniških enotah GPU.

Prikazani rezultati lahko služijo raziskovalcem in uporabnikom kot orientacija pri izbiri velikosti slovarja, arhitekture modela in vrednosti hiperparametrov za strojne prevajalnike, če želijo upoštevati uspešnost prevajanja, hitrost prevajanja in velikost modela. Slednja je pomembna zaradi omejitev strojne opreme.

Natančne izbire parametrov in velikosti slovarja bodo odvisne od specifičnega primera uporabe, kjer upoštevamo: domeno prevajanja,

morebitno potrebo po adaptaciji na domeno, razpoložljiva strojna oprema prevajalca ali prevajalske agencije, razpoložljivi korpusi. Kot splošno vodilo lahko predlagamo modele s čim večjimi slovarji in vrednostmi hiperparametrov, ki jih razpoložljiva oprema še dopušča.

Za boljše razumevanje uporabnosti razcepljanja besed v strojnem prevajanju bi bilo treba izvesti še nadaljnje raziskave. V tem članku smo le v omejenem obsegu preizkušali modele transformer in spremembe vrednosti hiperparametrov. Izvedli smo le postopke podatkovno vodenega razcepljanja, ki jih v nespremenjeni obliki lahko uporabimo tudi pri drugih pregibnih jezikih. V nadaljevanju lahko proučujemo še metode razcepljanja na podlagi slovničnega znanja ali kombiniranje komplementarnih metod. Pomemben prispevek h kakovosti prevajanja pri besednih modelih imata lahko večanje učne množice in večanje hiperparametrov modela. Slednje pomeni povečanje velikosti modela in njegovo počasnejše delovanje. Nadaljnje raziskave bodo tudi vključevale podrobnejšo analizo napak, ki se pojavljajo pri različnih metodah razcepljanja.

Zahvala

Raziskovalni program št. P2-0069, v okvirju katerega je nastala ta raziskava, je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna.

Avtorji se zahvaljujejo konzorciju HPC RIVR (www.hpc-rivr.si) za sofinanciranje raziskave z uporabo zmogljivosti sistema HPC MAISTER na Univerzi v Mariboru (www.um.si).

Zahvaljujejo se tudi avtorjem vzporednega korpusa ParaCrawl za njegovo prosto dostopnost.

Literatura

Bahdanau D., Cho K., & Bengio Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations*.

Banerjee, T., & Bhattacharyya, P. (2018). Meaningless yet meaningful: Morphology grounded subword-level nmt. In *Proceedings of the second workshop on subword/character level models* (pp. 55–60). Retrieved from https://aclanthology.org/W18-1207.pdf

Bañón, M., Chen, P., Haddow, B., Heafield, K., Hoang, H., Esplà-Gomis, M., Forcada, M. L., ..., & Zaragoza, J. (2020). ParaCrawl: Web-scale acquisition of parallel corpora. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 4555–4567). doi: 10.18653/v1/2020.acl-main.417

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., & Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, *19*(2), 263–311.

Creutz, M., & Lagus, K. (2002). Unsupervised discovery of morphemes. In *Proceedings of the workshop on morphological and phonological learning of ACL-02* (pp. 21–30). doi: 10.3115/1118647.1118650

Etchegoyhen, T., Bywood, L., Fishel, M., Georgakopoulou, P., Jiang, J., Loenhout, G. V., Pozo, A. D., ..., & Volk, M. (2014). Machine translation for subtitling: A large-scale evaluation. In N. C. C. Chair et al. (Eds.), *Proceedings of the ninth international conference on language resources and evaluation* (LREC'14). Retrieved from http://www.lrec-conf.org/proceedings/lrec2014/pdf/463_Paper.pdf

Gupta, R., Besacier, L., Dymetman, M., & Gallé, M. (2019). Character-based NMT with transformer. *arXiv preprint arXiv:1911.04997*.

Heigold, G., Varanasi, S., Neumann, G., & van Genabith, J. (2018). How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th conference of the association for machine translation in the Americas* (Vol 1, pp. 68–80). Retrieved from https://aclanthology.org/W18-1807.pdf

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., ..., & Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL2018, system demonstrations* (pp. 116–121).

Koehn, P., Hoang, H.T., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., ..., & Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions* (pp. 177–180).

Kudo, T. (2018). Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 66–75). doi: 10.18653/v1/P18-1007

Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 conference on empirical methods in natural language processing: System demonstrations* (pp. 66–71). doi: 10.18653/v1/D18-2012

Marco, M. W. D., Huck, M., & Fraser, A. (2022). Modeling Target-Side Morphology in Neural Machine Translation: A Comparison of Strategies. In *Proceedings of the Conference on Machine Translation (WMT)* (Vol 1, pp. 56–67).

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311–318). Retrieved from https://aclanthology.org/P02-1040.pdf

Popović, M. (2015). chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation* (pp. 392–395). doi: 10.18653/v1/ W15-3049

Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the third conference on machine translation: Research papers* (pp. 186–191). Retrieved from https://aclanthology.org/W18-6319.pdf

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics* (Vol. 1*,* pp. 1715–1725). doi: 10.18653/v1/P16-1162

Sepesy Maučec, M., & Donaj, G. (2019). Machine Translation and the Evaluation of Its Quality. In A. Sadollah & T. S. Sinha (Eds.), *Recent Trends in Computational Intelligence*. IntechOpen.

Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69, 343–418.

Tamchyna, A., Marco, M. W. D., & Fraser, A. (2017). Modeling target-side inflection in neural machine translation. In *Proceedings of the Conference on Machine Translation (WMT)* (Vol. 1, pp. 32–42).

Tukeyev, U., Karibayeva, A., & Zhumanov, Z. H. (2020). Morphological segmentation method for Turkic language neural machine translation. *Cogent Engineering*, *7*(1), 1856500. doi: 10.1080/23311916.2020.1856500

Vaswani A., Shazeer,N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems. (pp. 5998–6008).

Virpioja, S., Smit, P., Grönroos, S.-A., & Kurimo, M. (2013). *Morfessor 2.0: Python implementation and extensions for morfessor baseline*. Aalto University.

Practical Aspects of Using Subword Units in Slovene-English Machine Translation

Most modern machine translation systems are based on neural networks. This includes web-based applications, research tools and translation tools for translators. Although neural net systems can be used on common central processing units in computers and servers, reasonable speech can only be achieved by using graphics processing units (GPUs). Given today's technology for GPUs, neural machine translation systems can only use a limited vocabulary, with negative effects on translation quality. The use of subword units can alleviate the problems of vocabulary size and language coverage. However, with further technological developments the limited vocabulary and the use of subword units are losing significance. This paper presents different word splitting methods with different final vocabulary sizes. We apply these methods to the machine translation task for the Slovene-English language pair and compare them in terms of translation quality, training and translation speed, and model size. We also include a comparison with word-based translation models, and present some practical aspects on the use of subword units when machine translation is used with computer-aided translation tools.

Keywords: machine translation, vocabulary size, subword units, graphical processing units