UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

VYTVORENIE A NATRÉNOVANIE STROJOVÉHO PREKLADU DO SLOVENČINY POMOCOU NEURÓNOVÝCH SIETÍ

DIPLOMOVÁ PRÁCA

2024 **BC. MATÚŠ KLEŠTINEC**

UNIVERZITA KONŠTANTÍNA FILOZOFA V NITRE

FAKULTA PRÍRODNÝCH VIED A INFORMATIKY

VYTVORENIE A NATRÉNOVANIE STROJOVÉHO PREKLADU DO SLOVENČINY POMOCOU NEURÓNOVÝCH SIETÍ

DIPLOMOVÁ PRÁCA

Študijný odbor:	18. Informatika
Študijný program:	Aplikovaná informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	prof. RNDr. Daša Munková, PhD.

Nitra 2024 Bc. Matúš Kleštinec

Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Matúš Kleštinec
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: Diplomová práca
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Vytvorenie a natrénovanie strojového prekladu do slovenčiny pomocou neurónových sietí

Anotácia: Strojový preklad (Machine translation, MT) je automatické konvertovanie textu z jedného prirodzeného jazyka do druhého prirodzeného jazyka. Kvalita, ktorú MT produkuje je rôzna, pohybuje sa od veľmi vysokej až po veľmi nízku v závislosti od jazykových vlastností daného textu a jazykového páru. Cieľom práce je vytvoriť a natrénovať neurónový MT v smere prekladu z angličtiny do slovenčiny. Pripraviť trénovacie a evalvačné datasety s malým množstvom paralelných dát. Zvoliť si framework (napr. Marian) pre natrénovanie prekladových modelov a realizovať a vyhodnotiť experimenty s vybranými technikami (napr. transfer learning a semi-supervised learning) a nakoniec navrhnúť vylepšenie vybraných techník.
Charakter práce:
Výskumný – stanovenie predpokladov/hypotéz, metodika výskumu, výsledky
výskumu (štatistická interpretácia), interpretácia výsledkov výskumu (vecná
interpretácia).

Predmetové prerekvizity:
Korpusová a počítačová lingvistika (2., Mgr.)
Úvod do spracovania prirodzeného jazyka (2., Mgr.)
Úvod do strojového učenia (2., Mgr.)
Neurónové siete (1., Mgr.)
Matematické princípy spracovania jazyka a strojového učenia (1., Bc.)
Spracovanie údajov v jazyku Python (1., Bc.)

Univerzita Konštantína Filozofa v Nitre
Fakulta prírodných vied a informatiky

Najdôležitejšie kompetentnosti získané spracovaním témy:
uplatniť postupy štatistickej analýzy
vykonať analýzu údajov
analyzovať veľké dáta (big data)
realizovať procesy dátovej kvality referovať o výsledkoch analýzy
zhodnotenie a navrhnutie algoritmu pre riešenie konkrétnej zadanej úlohy

Školiteľ:	prof. RNDr. Daša Munková, PhD.
Konzultant:	Mgr. František Forgáč
Katedra:	KI - Katedra informatiky

Dátum zadania: 03.11.2022

Dátum schválenia: 07.03.2024

vedúci/a katedry

ABSTRAKT

KLEŠTINEC, Matúš: Vytvorenie a natrénovanie strojového prekladu do slovenčiny pomocou neurónových sietí. Diplomová práca. Univerzita Konštantína Filozofa v Nitre. Fakulta prírodných vied a informatiky. Školiteľ: prof. RNDr. Daša Munková, PhD. Stupeň odbornej kvalifikácie: Magister odboru Aplikovaná informatika. Nitra: FPVaI, 2024. 82 s.

V tejto práci sa zaoberáme strojovým prekladom, konkrétne jeho postupom od získania textov až po vyhodnotenie modelu strojového prekladu. Naše modely strojového prekladu boli trénované zo vstupného jazyka anglického do cieľového jazyka slovenského. Skúmali sme vplyv niektorých faktorov na model strojového prekladu ako veľkosť testovacej a validačnej množiny, veľkosť slovnej zásoby, vplyv použitých algoritmov tokenizácie, kvality textov a veľkosti dvojjazyčných textov. Po natrénovaní strojového prekladu sme aj skúmali, aký vplyv majú parametre prekladu na výsledný text. Vyhodnocovali sme pomocou automatických metrík BLEU, METEOR a COMET a na základe manuálnej kontroly viet. Zistili sme, že parametre, ktoré sme skúmali mali vplyv na model strojového prekladu a prípadne ideálne nastavenia pre naše modely.

Kľúčové slová: Strojový preklad. Korpus. Neurónová sieť.

ABSTRACT

KLEŠTINEC, Matúš: Creating and training of machine translation into Slovak using neural networks. Master Thesis. Constantine the Philosopher University in Nitra. Faculty of Natural Sciences and Informatics. Supervisor: prof. RNDr. Daša Munková, PhD. Degree of Qualification: Master of Applied Informatics. Nitra: FNSaI, 2024. 82 p.

In this paper, we focus on machine translation, specifically its process from obtaining texts to evaluating the machine translation model. Our machine translation models were trained from source language English to target language Slovak. We explored the influence of various factors on the machine translation model, such as the size of the test and validation sets, vocabulary size, the impact of tokenization algorithms used, text quality and size of bilingual texts. After training the machine translation model, we also examined the effects of translation parameters on the result text. Evaluation was done using automatic metrics such as BLEU, METEOR, and COMET, as well as manual inspection of sentences. We found that the parameters we investigated had an impact on the machine translation model and optimal settings for our models.

Keywords: Machine translation. Corpus. Neural network.

OBSAH

ÚVOD

Jazyk je systém komunikácie, ktorý sprevádza ľudstvo po značnú časť jeho existencie. Po tisícky rokov jazyky vznikali a vyvíjali sa v dôsledku potreby vyjadriť nové predmety a javy alebo v dôsledku vplyvu iných národov, ktoré používali iné dialekty alebo jazyky. Komunikácia medzi skupinami ľudí ovládajúci odlišné jazyky bola dlhú dobu jedine možná pomocou prekladateľov, ktorí museli ovládať oba jazyky, či už ústne a/alebo písomne, čo si vyžadovalo čas aj námahu.

V súčasnosti v dôsledku globalizácie ľudia potrebujú na dennej báze komunikovať alebo prijímať informácie z iných častí sveta, ale nemusia sa nutne učiť cudzí jazyk. Informácie môžu byť preložené prekladateľmi, ale na rozdiel od minulosti, dnes môžeme použiť strojový preklad. Vďaka strojovému prekladu dokážeme v krátkom čase preložiť text z jedného jazyka do druhého. Prvýkrát bol strojový preklad definovaný v roku 1949 Warrenom Weaverom (Weaver, 1949). Snaha o vytvorenie strojového prekladu existovala, ale v dôsledku komplexnosti jazykov a malej výpočtovej sile počítačov v tej dobe nenastali veľké pokroky. Na rozdiel od prekladu vykonávaného človekom, strojový preklad si vyžaduje viacero krokov, aby sme zo vstupného textu získali správne preložený text v cieľovom jazyku. Jazyky sa okrem odlišných tvarov slov môžu líšiť vetnou skladbou, sémantikou, výslovnosťou a inými prvkami jazyka. Kvôli týmto odlišnostiam nie je možné produkovať zrozumiteľný preklad len na základe slovníka, ktorý by nahradil slovo z jedného jazyka do druhého. Pre jednoduché a krátke vety by takýto prístup možno fungoval alebo na preklad textu, aby čitateľ pochopil jeho kontext. Avšak pre zrozumiteľne preložený text by to bolo nedostačujúce riešenie. Z tohto dôvodu si strojový preklad vyžaduje viacero krokov, aby dosiahol preklad požadovaných kvalít.

1 ANALÝZA SÚČASNÉHO STAVU

1.1 STROJOVÝ PREKLAD

Strojový preklad je automatický preklad z jedného prirodzeného jazyk do druhého pomocou algoritmov a modelov strojového učenia (Keary, 2023). Model strojového prekladu natrénovaný pomocou dát, v tomto prípade viet v požadovaných jazykoch, je schopný vyhodnotiť najvhodnejšie slová v cieľovom jazyku, a tým vytvoriť preklad v požadovanom jazyku (Keary, 2023). V nasledujúcich podkapitolách prejdeme jednotlivé kroky pre vytvorenie strojového prekladu.

1.2 PARALELNÝ KORPUS

Aby sme dosiahli kvalitný preklad potrebujeme natrénovať model strojového prekladu pomocou neurónových sietí. Predtým, než začneme trénovať je potrebné získať paralelný (dvojjazyčný) textový korpus, na ktorom sa bude daný model trénovať. Korpus je súbor, ktorý obsahuje text z jedného alebo z viacerých jazykov, ktorého účelom je čo najvšeobecnejšie reprezentovať použité jazyky (Dash, 2018). Jazyk, z ktorého prekladáme označujeme aj ako zdrojový/východiskový jazyk, prípadne jeho obsah ako zdrojový/východiskový text (Keary, 2023). Jazyk do ktorého prekladáme označujeme ako cieľový jazyk, prípadne jeho obsah ako cieľový text (Keary, 2023). Všeobecnosť textu je potrebná, aby náš model strojového prekladu dokázal preložiť text z akejkoľvek domény. Ak by sme trénovali model len na základe korpusu získaného zo špecializovaných resp. odborných domén, napríklad z medicínskych článkov, tak by mal problémy preložiť texty iných domén, keďže by chýbala modelu slovná zásoba. Tieto korpusy môžeme získať alebo vytvoriť viacerými spôsobmi:

- Je možné vytvoriť korpus aj ručným písaním/kopírovaním, ale aj pri relatívne malom množstve viet by to bolo časovo neefektívne.
- Vytvorené a dostupné na internetových stránkach:
  https://www.statmt.org/europarl/ - paralelné korpusy z jednaní Európskeho parlamentu.
  https://www.paracrawl.eu/ - viacjazyčné paralelné korpusy medzi anglickým jazykom a ostatnými jazykmi používanými na území Európskej únie.
  https://opus.nlpl.eu/ - zbierka paralelných korpusov z webu

- Vytvorenie korpusu pomocou WebCrawlera a Web Scrapingu.

WebCrawler (anglicky označované aj ako spider, robot alebo wanderer) je softvér, ktorý prechádza internetom a zbiera URL adresy (Yadav et al., 2012). Z týchto zdrojov sa následne pomocou Web Scrapingu extrahuje text priamo z webových stránok alebo sa stiahnu súbory napr. vo formáte (pdf, docx, txt, csv), z ktorých môžeme čerpať požadovaný text. Tento proces je automatizovaný použitím nami naprogramovaného programu alebo pomocou niektorých z dostupných nástrojov. Výhodou tohto prístupu je, že môžeme ovplyvniť, aký obsah bude tvoriť náš korpus/y. Tento prístup je samozrejme časovo náročnejší pri tvorbe veľkého korpusu. Okrem toho je aj finančne náročnejší, keďže si vyžaduje potrebný hardvér, spotreba energie za dané časové obdobie a v prípade ak chceme použiť komerčný nástroj, tak musíme počítať aj s cenou licencie. Web Crawler je používaný aj veľkými spoločnosťami ako napríklad Google. Na trénovanie ich prekladača nahradili model na základe slovníka Sentence Embedding modelom pre 14 veľkých jazykových párov, čo zvýšilo počet zozbieraných viet v priemere o 29% bez straty na presnosti (Yu et al., 2020). Sentence Embedding sa snaží vety rôznej dĺžky transformovať do rovnakej dĺžky bez straty kontextu vety (Guo et al., 2018)

1.3 PREDSPRACOVANIE TEXTU

Po získaní textu je nutné ho predspracovať, keďže by náš strojový preklad mal problém sa správne natrénovať. V prípade paralelného textu môže byť problém rozdielnosť znakov medzi jazykmi (Tabuľka 1) alebo rovnaká veta môže byť v rámci textu prirovnaná k odlišným prekladom (Tabuľka 1) (Rikters, 2018). Je potrebné si dávať pozor aj na cudzie slová, ktoré sa môžu nachádzať v našich textoch. To môže byť v dôsledku citácií alebo z obsahu ako titulky seriálov/filmov, spravodajstvo, originálne názvy diel a iné.

Tabuľka 1 Odlišné znaky medzi jazykmi[1]

Angličtina	**Estónčina**
Address: Akariah 3, 8th Floor, Olaya St.	Çáã çÕIú ÇáÝÇHÒÉ : 3

[1] Zdroj Tabuľka 1: https://arxiv.org/pdf/1810.08392.pdf

Tabuľka 2 Odlišnosť prekladu s rovnakým výsledkom

Angličtina	**Slovenčina**
I will harden Pharaoh's heart and he will pursue them.	A ja opäť zatvrdím faraónovo srdce, a bude sa hnať za vami.
And I will harden Pharaoh’s heart, and he will pursue you.	A ja opäť zatvrdím faraónovo srdce, a bude sa hnať za vami.

Aby sme dosiahli kvalitný paralelný text, musíme z neho odstrániť alebo upraviť riadky, ktoré by mohli negatívne ovplyvniť učenie nášho modelu strojového prekladu. Uvedieme si možné úpravy spomenuté v práci (Rikters, 2018), ale aj iné:

1. Odstránenie prázdnych riadkov,
2. Normalizácia na základe unicode znakov,
3. Odstránenie duplicitných riadkov,
4. Odstránenie riadkov, ktorých veta je príliš dlhá – dĺžka záleží od jazyka, niektoré jazyky majú v priemere dlhšie slová,
5. Odstránenie HTML značiek,
6. Odstránenie riadkov, ktorého obsah zo značnej časti tvoria znaky iné ako abecedné, či už v jednom alebo v oboch textoch,
7. Zmena veľkých písmen na malé (v niektorých prípadoch to nemusí byť žiadúce),
8. Odstránenie odkazov na iné zdroje alebo obrázky a tabuľky,
9. Odstránenie riadkov, kde viacero odlišných viet jedného jazyka sa rovná jednej vete druhé jazyka,
10. Filtrovanie riadkov pomocou knižnice langdetect,
11. Zamiešanie riadkov.

Množstvo riadkov, ktoré nám zostane po odstránení vyššie spomenutých problémov záleží od kvality korpusu. Na demonštráciu môžeme použiť napríklad paralelný korpus NLLB v1 (Fan et al., 2020; Schwenk et al., 2020) . Neočistený paralelný korpus má 38 096 241 riadkov. Po aplikovaní vyššie spomenutých úprav môžeme vidieť výsledok nasledovne (Tabuľka 3). Pri čistení sme použili druhý krát odstránenie prázdnych riadkov na konci, pretože niektoré predošlé kroky mohli vytvoriť prázdny riadok.

Tabuľka 3 Čistenie textov od nežiadúcich prvkov v texte

Veľkosť datasetu (riadky: 38096241)	
Odstránenie riadkov, kde je NaN	Riadky: 38 096 241
Odstránenie duplikátov	Riadky: 38 095 832
Odstránenie riadkov, kde sa riadok source rovná riadku target	Riadky: 38 095 832
Odstránenie príliš dlhých viet source/target	Riadky: 37 881 382
Odstránenie HTML znakov	Riadky: 37 881 382
Zmena veľkých písmen na malé	Riadky: 37 881 382
Odstránenie riadkov, kde je NaN	Riadky: 37 881 099

Riadky v Tabuľke 3 reprezentujú aktuálny počet riadkov v textoch. Ako môžeme vidieť niekedy nie je ani nutné použiť všetky metódy čistenia textu. To záleží na tom, akou metódou sme získali text alebo z akých zdrojov sme čerpali. Keďže sme použili už existujúci korpus, tak počet odstránených riadkov nie je veľmi veľký (215 142, čo reprezentuje 0.57% z celého korpusu). Výsledok by bol pravdepodobne vyšší, ak by sme náš korpus vytvárali sami pomocou Web Scrapingu a nepoužili by sme dostupný korpus. Samozrejme všetko záleží od obsahu z ktorého čerpáme.

Normalizáciu unicode znakov je vhodné taktiež použiť, keďže v texte sa môžu nachádzať rôzne reprezentácie jedného znaku. Ako príklad si môžeme uviesť znak A. Veľké A môžeme reprezentovať kódovým označením U+0041 alebo ako „Fullwidth Latin Capital letter A“ s kódovým označením U+FF21 (Microsoft, 2021). Okrem písmen to môžu byť aj znaky ako úvodzovky. V niektorých textoch môžu byť použité úvodzovky “ alebo ”. Toto riešenie pomáha z hľadiska kompatibility, ale v prípade strojového prekladu hlavne redukuje počet znakov do jednotnej formy. To môže pomôcť pri spracovaní textu, čo pomáha pre tvorbu slovnej zásoby a tým aj pri trénovaní modelu. Existujú 4 druhy normalizácie unicode znakov a to NFD, NFC, NFKD, NFKC (Unicode Consortium, 2023). Každý druh normalizácie môže byť vhodný na iný účel. V rámci predspracovania budeme používať NFKC (Compatibility Decomposition, Canonical Composition). NFKC transformuje znaky do vhodnej podoby a ak je možné, tak vykoná ďalšiu transformáciu do zložených znakov s tým, že zložené znaky budú kompatibilné. NFKC sa snaží vizuálne zachovať znaky aj po transformácií (Unicode Consortium, 2023).

V strojom preklade zmena veľkých písmen na malé pomáha pre zovšeobecnenie dát. Je na zváženie, kedy tento krok použiť. Ak by sme pracovali s jazykom ako je nemčina, tak tento krok by nemusel byť žiadúci, keďže v nemeckom jazyku sa píšu veľkým začiatočným písmenom napríklad aj všetky podstatné mená. V prípade slovenského jazyka by tento krok nemusel byť tak drastický, keďže veľkým písmenom okrem začiatku vety označujeme len vlastné podstatné mená.

Filtrovanie riadkov pomocou Python knižnice Langdetect je vhodný nástroj, ak sa v textoch nachádzajú slová alebo celé vety, ktoré sa nezhodujú s požadovaným jazykom alebo jazykmi. Langdetect prechádza po n-gramoch zvoleného textu a následný výstup je pravdepodobnosť, že text pochádza z určitého jazyka (Lopez, 2017). Kód by mohol byť nasledovný:

```
from langdetect import detect_langs
detect_langs("Aby sme dosiahli kvalitný paralelný text, musíme z neho odstrániť alebo upraviť riadky")
Výstup: [sk:0.9999962526806121]
```

V kóde vyššie môžeme vidieť, že text je na 99,99% slovenský. Ak by sme chceli len jazyk, ktorý je najpravdepodobnejší, tak použijeme nasledovný kód:

```
from langdetect import detect
detect("Aby sme dosiahli kvalitný paralelný text, musíme z neho odstrániť alebo upraviť riadky")
Výstup: 'sk'
```

Pomocou tohto riešenia dokážeme filtrovať vety podľa ich identifikovaného jazyka. Langdetect dokáže identifikovať 55 jazykov, vrátane slovenčiny (Pypi, 2021).

Posledný krok predspracovania pred tokenizáciou je zamiešanie respektíve zmena poradia viet. Tento krok je vhodný pre vytvorenie náhodnosti v datasete. V prípade zozbieraných textov sa v nich môžu nachádzať rôzne témy obsahujúce svoju slovnú zásobu, ktoré sa nemusia vyskytovať vo zvyšku viet. Pri rozdelení dát na trénovaciu, testovaciu a validačnú množinu tieto informácie môžu chýbať a to môže ovplyvniť výsledný model (Dupont, 2023). Tento krok je možné vykonať počas úvodných krokov predspracovania alebo počas rozdelenia na trénovaciu, testovaciu a validačnú množinu.

Porozumením našich textov môžeme vyhodnotiť, čo je nutné odstrániť alebo upraviť a tým môžeme ušetriť čas a prostriedky, ktoré boli alokované na čistenie. Pri

relatívne malom množstve textu pre nás môže byť pridanie kroku čistenia zanedbateľné, ale ak naše texty obsahujú milióny riadkov, tak sa čas predspracovania môže značne predĺžiť, keďže každý krok si vyžaduje prejdenie celým korpusom. Na druhej strane je nutné podotknúť, že pri obzvlášť veľkých korpusoch je náročné určiť, čo všetko sa môže nachádzať v texte, a preto je dobré sa presvedčiť, že sa v nich nenachádza nič, čo by negatívne ovplyvnilo náš strojový preklad. Je stále výhodnejšie stráviť viac času na vytvorenie kvalitných textov, ako stráviť dni trénovaním modelu strojového prekladu, ktorý by nakoniec nebol dobrý. Ako príklad by sme si mohli uviesť prácu Riktersa (2018), ktorý pracoval s paralelnými korpusmi v jazykochých pároch anglicko-estónsky/fínsky/litovsky. Použil pre porovnanie tri dostupné paralelné korpusy a to Paracrawl, Rapid a Europarl. Jednotlivé kroky čistenia pre dané korpusy sú zobrazené v tabuľke 4.

Tabuľka 4 Výsledok čistenia korpusov v jazykoch Anglicko-Estónsky/Fínsky/Litovsky[2]

	Paracrawl		Rapid			Europarl		
	Ang-Est	Ang-Fi	Ang-Est	Ang-Fi	Ang-Lt	Ang-Est	Ang-Fi	Ang-Lt
Veľkosť korpusu	1298103	624058	226978	583223	306588	652944	1926114	638789
Unikátne	26	37	23	161463	80894	23218	52686	19652
	0.00%	0.01%	0.01%	**27.68%**	**26.39%**	3.56%	2.74%	3.08%
Riadok zdrojového jazyka = riadku cieľového jazyka	242816	41611	428	3488	2929	490	528	707
	18.71%	**6.67%**	0.19%	0.60%	0.96%	0.08%	0.03%	0.11%
Viecero riadkov zdrojového jazyka = 1 cieľovému jazyku	267235	17239	1108	1513	990	1176	6631	979
	20.59%	2.76%	0.49%	0.26%	0.32%	0.18%	0.34%	0.15%
Viecero viet cieľových jazykov = 1 vete zdrojového jazyka	69225	9532	752	1016	329	462	3536	435
	5.35%	1.53%	0.33%	0.17%	0.11%	0.07%	0.18%	0.07%
>50% neabecedných znakov	200338	12919	1226	5647	1699	66	285	72
	15.43%	2.07%	0.54%	0.97%	0.55%	0.01%	0.01%	0.01%
Nekonzistentnosť abecedných znakov	23777	12737	6674	13311	6361	7211	24847	4012
	1.83%	2.04%	2.94%	2.28%	2.07%	1.10%	1.29%	0.63%
Opakujúce sa tokeny	11210	1397	175	396	171	727	2594	703
	0.86%	0.22%	0.08%	0.07%	0.06%	0.11%	0.13%	0.11%
Nerovnosť jazykov	283152	36233	14762	24854	8739	8924	10932	3301
	21.81%	**5.81%**	**6.50%**	**4.26%**	2.85%	1.37%	0.57%	0.52%
Σ odstránených	1097779	131705	25148	211688	102112	42274	102039	29861
	85%	**21%**	**11%**	**36%**	**33%**	6%	5%	5%

Rikters (2018) použil väčšinu úprav, ktoré sme si opísali vyššie a konečný počet odstránených riadkov ako môžeme vidieť v poslednom riadku v (Obrázok 3) sa značne líši ako medzi jazykmi tak aj medzi použitými korpusmi. Paracrawl môžeme považovať za najviac problematický, keďže ku koncu čistenia bolo odstránených až 85% viet. Hlavnými dôvodmi boli duplicitné vety tvoriacich 18,71%, viacero odlišných riadkov jedného jazyka sa rovnalo jednému riadku v druhom jazyku v pomere 20,59%, 15,43% riadkov obsahovalo viac ako 50% znakov, ktoré neboli abecedné a zle zarovnané vety medzi textami, teda veta jedného jazyka sa nerovná prekladu druhého jazyka, čo predstavovalo 21,81% z celého korpusu. Najlepšie dopadlo čistenie textov

[2] Zdroj Tabuľka 4: https://arxiv.org/pdf/1810.08392.pdf

korpusu Europarl s celkový počtom odstránených viet v rozmedzí 5% - 6% kde na rozdiel od ParaCrawlu jednotlivé kroky čistenia odstránili značne menšie množstvo viet. Tento praktický príklad sme si uviedli ako ukážku, ako veľmi sa od seba môžu líšiť jednotlivé korpusy.

1.3.1 TOKENIZÁCIA DÁT

Čistením sme dosiahli texty, ktoré sú vhodné na ďalší krok predspracovania dát. Aby sme dokázali natrénovať strojový preklad, potrebujeme transformovať text do podoby, ktorá bude vhodnejšia pre učenie a zároveň, aby sme nestratili kontext textu. Ide o proces, pri ktorom sa korpus rozdelí na menšie časti nazývané tokeny, ktorým následne bude pridelené unikátne identifikačné číslo. Táto transformácia sa nazýva aj tokenizácia (Awan, 2023). Text môžeme tokenizovať viacerými spôsobmi (postupnosť je od najväčších segmentov po najmenšie):

- Tokenizácia na slová,
- Tokenizácia na subwordy,
- Tokenizácia na znaky,

Pri tokenizácií na slová, ako už napovedá názov, sa bude text tokenizovať na celé slová. Táto transformácia je pomerne jednoduchá, keďže prevedenie môže nastať jednoduchým rozdelením slov pomocou medzier a interpunkčných znamienok (Munawwar, 2020). Nevýhodou tohto riešenia je potreba veľkej slovnej zásoby, keďže sa nerozlišuje podobnosť medzi slovami. Napríklad pri tokenizácií slov by sa považovali slová *krivý*, *krivá* a *krivé* za odlišné slová, a tým by si vyžadovali 3 miesta v slovnej zásobe. Veľkosť slovnej zásoby sa dá vyriešiť jej limitáciou na určitú veľkosť na základe najčastejšie vyskytujúcich slov. Slová, ktoré sa nedostanú do slovnej zásoby budú mať označenie <unk>, teda neznáme. Veľmi veľká slovná zásoba môže byť výpočtovo náročná, keďže model musí pracovať s veľkým množstvom parametrov v rôznych fázach trénovania. Vetu – „*nevedú k elektrickému prúdu*“ by sme tokenizovali nasledovne: ['nevedú', 'k', 'elektrickému', 'prúdu']

Pri tokenizácií na znaky sa text bude rozdeľovať na jednotlivé znaky, čo môžeme považovať taktiež za jednoduchý spôsob tokenizácie. Výhodou tohto riešenia je pomerne malý slovník, keďže unikátnych znakov nie je veľmi veľa. Môžeme povedať, že veľkosť slovníka bude v rádoch desiatok. Znaky môžu byť písmená, čísla, interpukčné znamienka aj medzery. Nevýhodou tohto riešenia je náročnosť modelu vyhodnotiť vzťah medzi jednotlivými znakmi, a tým by vznikol problém vytvoriť

zmysluplné slová a vety (Munawwar, 2020). Vyššie spomenutú vetu sme tokenizovali nasledovne: ['n', 'e', 'v', 'e', 'd', 'ú', ' ', 'k', ' ', 'e', 'l', 'e', 't', 'r', 'i', 'c', 'k', 'é', 'm', 'u', ' ', 'p', 'r', 'ú', 'd', 'u']

Tokenizácia na subwordy je kompromisom medzi tokenizáciou na slová a znaky. Tento typ tokenizácie rozdeľuje vety na bežné, počtom významnejšie slová v texte a menej vyskytujúce sa slová v texte. Podľa (Khanna, 2021) sa frekventované slová bez úprav vkladajú do slovnej zásoby a menej vyskytujúce sa slová sú rozdelené na časti, ktoré sú viac frekventované. Ako príklad môže byť uvedené slovo „*obyvateľstvom*“. Pri tejto tokenizácií sa slovo rozdelí na „*obyvateľstvo*“ a „*m*“. Medzere medzi slovami budú nahradené znakom „_“ napríklad *„_obyvateľstvo m*“. Pre lepšie pochopenie vetu „*nevedú k elektrickému prúdu*“ by sme tokenizovali nasledovne: [_ne vedú _k _elektrické mu _prúdu]. Týmto spôsobom tokenizácie dosiahneme rozumnú veľkosť slovnej zásoby a zároveň sa model môže naučiť kontext medzi slovami. Dôležité je aj dodať, že slovná zásoba je pri používaní časti slov (subwordov) nastaviteľná podľa potreby modelu.

Tokenizácia môže byť napríklad vykonaná metódou Byte pair encoding alebo metódou Unigram. Byte pair encoding iteratívne spája dokopy najbežnejšie páry tokenov do nového tokenu. Tento krok sa opakuje, pokým nenaplníme definovanú veľkosť slovnej zásoby alebo ak už nie je možné vykonať ďalšie spojenie tokenov (Mohan, 2022). Tokenizácia pomocou Unigram vytvorí na začiatku slovnú zásobu zo všetkých tokenov, ktoré sa nachádzajú v korpuse a následne sa odstraňujú tokeny na základe ich početnosti v korpuse. Tento proces pokračuje, kým sa nedostaneme k požadovanej veľkosti slovnej zásoby.

Na tokenizáciu a spätnú detokenizáciu je možné použiť knižnicu Sentencepiece, ktorá rozdeľuje vstupný text na subwordy (Park, 2023). Pre tokenizovanie textu je nutné najprv natrénovať Sentencepiece model samostatne pre vstupný jazyk (napr. anglický) aj výstupný jazyk (napr. slovenský) a následne pomocou novovytvorených modelov tieto texty tokenizovať. Na tokenizáciu pomocou Sentencepiece môžeme využiť už vyššie spomenutý Byte pair Encoding alebo Unigram. Sentencepiece model pre výstupný (cieľový jazyk) bude užitočný aj pri detokenizovaní, keďže preklad bude v základe vo forme subwordov. Je nutné transformovať (desubwordovať) text zo subwordov späť do čitateľnej podoby. Príklad subwordovaného súboru prekladu a po transformácií do čitateľnej podoby v tabuľke 5.

Tabuľka 5 Proces desubwordovania textu

Subwordovaná veta prekladu	▁nap lá nu jte ▁si ▁cestu ▁ako ▁sa ▁dostať ▁vla kom
Po transformácií (desubwordovaní) pomocou Sentencepiece modelu	naplánujte si cestu ako sa dostať vlakom

1.4 ROZDELENIE DÁT NA TRÉNOVANIE A TESTOVANIE MODELU

Po predspracovaní dát máme k dispozícií korpus, ktorý môže byť použitý na dosiahnutie vytvorenia modelu strojového prekladu. Pred trénovaním je nutné rozdeliť korpus na viacero častí, pretože nemôžeme použiť rovnaké dáta na trénovanie aj na testovanie dát (Baheti, 2023). Tým by sme dosiahli preučenie (ang. Overfitting) a zároveň by to bolo nepraktické, keďže cieľom nášho strojového prekladu je, aby dokázal prekladať vety, ktoré ešte nevidel. Z tohto dôvodu náš korpus delíme do troch častí:

1. Trénovacia množina - určená na trénovanie modelu. Spravidla tvorí cca. 80% z dát,
2. Validačná množina - určená na validáciu modelu, aby sme na základe získaných informácií mohli model vylepšiť. Tým je myslené napr. zmena hyperparametrov trénovania (počet vrstiev, batch size, atď.), Spravidla tvorí cca. 10% z dát, ale môže sa líšiť medzi prácami
3. Testovacia množina - určená na testovanie modelu pomocou dát, ktoré neboli použité na trénovanie ani na validáciu po natrénovaní modelu. Spravidla tvorí cca. 10% z dát, ale môže sa líšiť medzi prácami.

Pomer rozdelenia, ktorý sme napísali je len bežná prax. Môže sa líšiť od účelu modelu a veľkosti modelu.

1.5 ARCHITEKTÚRY/MODELY STROJOVÉHO PREKLADU

Prešli sme všetky potrebné kroky, aby sme mohli začať trénovať model strojového prekladu. Existuje mnoho architektúr pre vytvorenie modelov strojového učenia, ktoré sa vytvorili za niekoľko desaťročí vývoja. V rámci tejto kapitoly budeme opisovať architektúru Transformer a pozornosť v strojovom učení, ktorá sa používa aktuálne v praxi popíšeme si aj niektoré riešenia iných autorov. K jednotlivým modelom

budú k dispozícií aj výsledky ako v niektorých prácach boli modely natrénované. Vyhodnocovanie resp. evalvácia je vykonávaná pomocou automatických metrík, ako sú napríklad perplexita, BLEU, METEOR a iné, ktoré nám dokážu vyjadriť ako dobre model preloží vzorku textu. Existujú aj metriky, ktoré sú používané pri manuálnom hodnotení modelov strojového prekladu, ale tým sa nebudeme v tejto práci venovať.

Perplexita je definovaná ako exponenciálna priemerná negatívna logaritmická pravdepodobnosť postupnosti. To znamená, že ak máme postupnosť X = (x_0, x_1,....,x_t), tak perplexitu X môžeme definovať ako vzorec nižšie s tým, že logΘ (xi | x <i) je logaritmická pravdepodobnosť i-tého tokenu podmieneného predchádzajúcimi tokenmi x <i podľa nášho tokenu (Priyanka, 2022). Čím je perplexita nižšia, tým lepšie dokáže model preložiť text.

$$PPL(X) = \exp\{-\frac{1}{t}\sum_{i}^{t}\log p_\Theta(x_i|x_{<i})\}$$

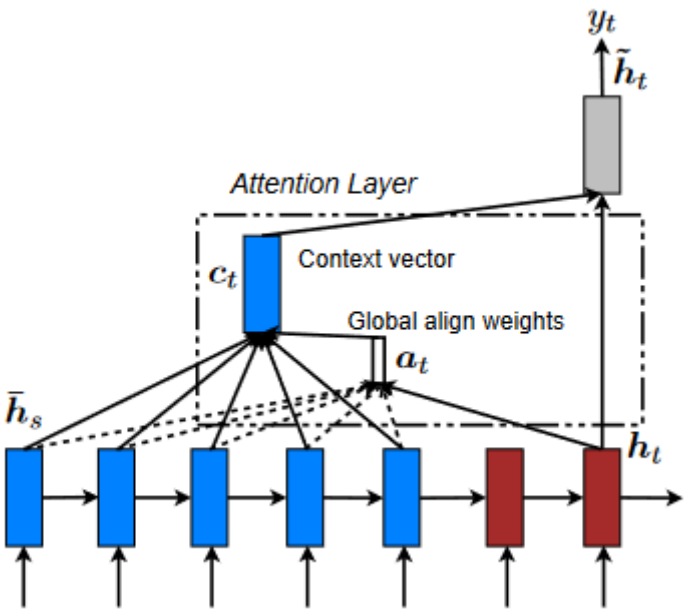
BLEU (BiLingual Evaluation Understudy) funguje na princípe počítania správne preložených n-gramov v cieľovom texte. Na poradí n-gramov nezáleží. Vyššie BLEU skóre znamená, že model strojového prekladu dokáže lepšie preložiť zo zdrojového textu na cieľový text (Papineni, 2002).

METEOR (Metric for Evaluation of Translation with Explicit Ordering) je metrika, ktorá porovnáva referenčné vety s preloženými vetami. Text rozloží na časti, a následne sa vykonáva výpočet METEOR skóre pomocou pokrytia (ang. recall), F-skóre a presnosť unigramov. Následne sa vypočíta vážený aritmetický priemer týchto výpočtov a vyjde nám výsledok v rozmedzí od 0 (zlá kvalita prekladu) po 1 (dobrá kvalita prekladu) (Banerjee et al., 2005).

COMET je framework pre trénovanie evalvačných modelov, ktorý využíva pre evalváciu vety zdrojového, ale aj cieľového jazyka (Rei et al., 2020). Výsledok sa pohybuje v rozmedzí od 0 (zlá kvalita prekladu) po 1 (dobrá kvalita prekladu). V našej práci budeme používať konkrétne COMET-22, ktorý je kombináciou COMET s viacúčelovým modelom natrénovaným na odhad skóre na úrovni viet s tagmi OK/BAD na úrovni slov odvodených od multidimenzionálnych metrík kvality s chybovými anotáciami. Tieto modely sú skombinované s využitím vyhľadávania vhodných hyperparametrov na základe odlišných vlastností získaných z evalvačných modelov a následne sú skombinované do jedného skóre (Rei et al., 2022).

1.5.1 POZORNOSŤ V STROJOVOM PREKLADE

Staršie modely používané na strojový preklad fungujú relatívne dobre na krátke vety, ale pri dlhších vetách strácajú prehľad o kontexte a tým sa znižuje kvalita samotného prekladu. Tento problém rieši pozornosť (ang. Attention mechanism). Pozornosť v strojovom preklade funguje na princípe zameriavania sa na konkrétne časti vety na strane vstupného jazyka (Luong et al., 2015). Autori (Luong et al., 2015) klasifikujú pozornosť do dvoch skupín a to na globálnu a lokálnu. Globálna pozornosť sa zameriava na všetky stavy enkóderu, keď odvodzuje obsah vektoru c_t.



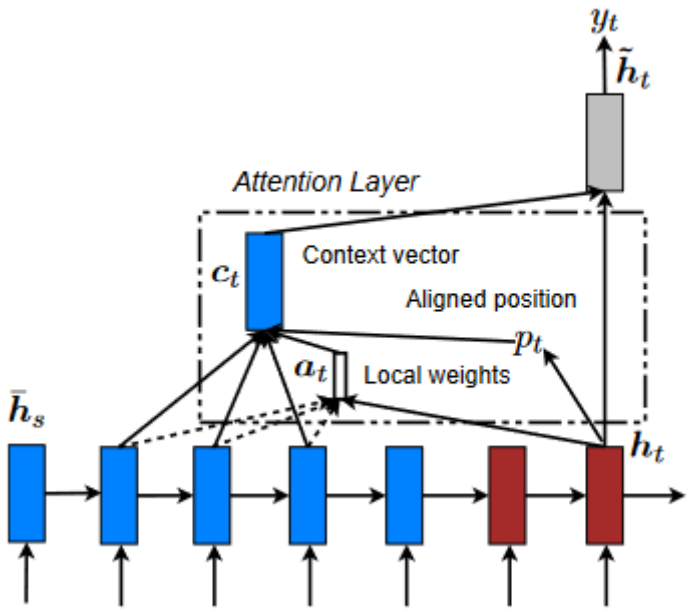
Obrázok 1 Model globálnej pozornosti[3]

Na Obrázku 1 môžeme vidieť ako funguje model globálnej pozornosti. Každým krokom t, model odvodzuje dĺžku premennej vektoru a_t, ktorej dĺžka sa rovná počtu krokov na strane vstupného textu (zdroja), na základe momentálneho stavu cieľového textu h_t a všetkých vstupných stavov $\bar{h}_s$. Vzorec pre výpočet je nasledovný:

$$a_t(s) = align(h_t, \bar{h}_t) = \frac{\exp(score(h_t, \bar{h}_t))}{\Sigma_{t'} \exp(score(h_t, \bar{h}_{t'}))}$$

[3] Zdroj Obrázok 1: https://arxiv.org/pdf/1508.04025v5.pdf

Nevýhodou globálnej pozornosti je jej pomerne vysoká výpočtová náročnosť, keďže, ako už bolo spomenuté skôr, globálna pozornosť sa zameriava na všetky stavy zdrojového textu. Tento problém rieši lokálna pozornosť. Lokálna pozornosť na rozdiel od globálnej pozornosti sa zameriava len na malú časť pozícií zdrojového textu pre jedno cieľové slovo.



Obrázok 2 Model lokálnej pozornosti[4]

Obrázok 2 vizualizuje ako funguje model lokálnej pozornosti. Model najprv odhaduje jednu pozíciu p_t pre aktuálne cieľové slovo, a následne je použité na výpočet kontextu vektoru c_t, čo je vážený priemer zdrojového textu (slova). Váhy a_t sú odvodené od aktuálneho cieľového stavu h_t a stavov zdrojového textu $\bar{h}_s$. Vzorec pre výpočet je nasledovný:

$$a_t(s) = align(h_t, \bar{h}_t) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

V rámci tejto kapitoly bude viackrát spomenutá zbierka datasetov WMT 2014. WMT 2014 je zbierka datasetov použitá v rámci workshopu, ktorého cieľom bolo porovnať strojové preklady jednotlivých vedeckých skupín za cieľom rozvoja techník strojového prekladu (Bojar et al., 2014).

Autori (Luong et al., 2015) porovnávali efektívnosť prekladu pri použití pozornosti. Popis je nasledujúci:

[4] Zdroj Obrázok 2: https://arxiv.org/pdf/1508.04025v5.pdf

- Použili zbierku datasetov WMT 2014, ktorá obsahovala 4,5 milióna párov viet v jazykoch angličtina a nemčina v pomere 116 miliónov a 110 miliónov slov,
- Nastavili veľkosť slovnej zásoby (ang. Vocabulary) na 50000 najčastejších slov,
- Slová, ktoré sa nenachádzajú v slovnej zásobe sa premenia na token <unk>, ktorý bude reprezentovať menej frekventované slová,
- Páry viet, ktorých dĺžka je dlhšia ako 50 znakov boli odstránené,
- Použil sa model LSTM (Long short-term memory), ktorý mal 4 vrstvy po 1000 neurónoch a vektorová reprezentácia slov (ang. Word embedding) v rozmere 1000,
- Trénovanie prebiehalo v 10 epochách s použitím stochastického gradientového zostupu,
- Využitie dropout s pravdepodobnosťou 0,2 (20%). To znamená, že neurón v modely má 20% šancu na vypnutie a tým sa nepoužije pri trénovaní.

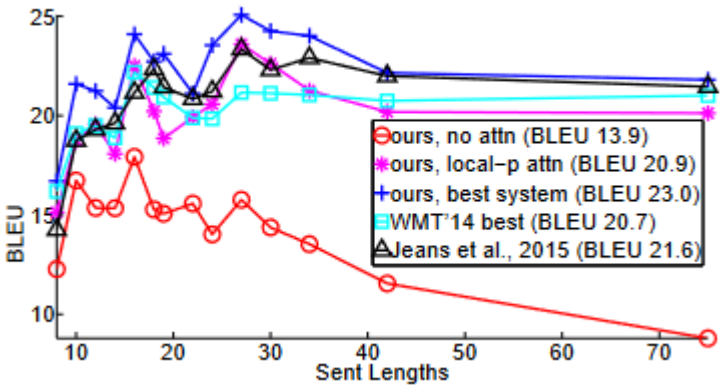
Tabuľka 6 Porovnanie hodnotení jednotlivých trénovaní na základe WMT 14 datasetov, hodnotených pomocou Perplexity a BLEU. Najlepšie skóre sú zvýraznené. [5]

System	**Perplexita**	**BLEU**
Winning WMT´14 system - phrase-based + large LM (Buck et al., 2014)		20,7
Existujúce NMT systémy		
RNNsearch (Jean et al., 2015)		16,5
RNNsearch + unk replace (Jean et al., 2015)		19,0
RNNsearch + unk replace + large vocab + ensemble 8 models (Jean et al., 2015)		**21,6**
Naše NMT systémy		
Base	10,6	11,3
Base + reverse	9,9	12,6 (+1,3)
Base + reverse + dropout	8,1	14,0 (+1,4)
Base + reverse + dropout + global attention (location)	7,3	16,8 (+2,8)
Base + reverse + dropout + global attention (location) + feed input	6,4	18,1 (+1,3)

[5] Zdroj Tabuľky 6: https://arxiv.org/pdf/1508.04025v5.pdf

Base + reverse + dropout + local-p attention (general) + feed input Base + reverse + dropout + local-p attention (general) + feed input + unk replace	5,9	19,0 (+0,9) 20,9 (+1,9)
Ensemble 8 models + unk replace		**23,0** (+2,1)

Autori (Luong et al., 2015) porovnávali riešenia bez použitia pozornosti s ich riešením, kde pozornosť použili a môžeme vidieť, že nastalo výrazné zvýšenie BLEU skóre zo 14 na 16,8 čo predstavuje zlepšenie o 2,8. Ak sa k modelu pridala globálna pozornosť a perplexita sa znížila z 8,1 na 7,3. To je zlepšenie o 0,8. Následným nahradením globálnej pozornosti lokálnou pozornosťou sa dosiahlo dodatočné zlepšenie BLEU skóre z 18,1 na 19,0 čo je dodatočné zlepšenie o 0,9 a perplexita sa znížila zo 6,4 na 5,9 čo reprezentuje zmenu o 0,6. Bola použitá aj technika „unknown replacement“, ktorej implementácia zlepšila BLEU skóre z 19 na 20,9 čo je značné zlepšenie o 1,9, perplexita zostala nemenná. Nakoniec kombináciou 8 modelov s rôznymi hyperparametrami bola dosiahnuté BLEU skóre 23 čo je dodatočný nárast o 2,1. Ako môžeme vidieť na Obrázku 6, riešenie pomocou pozornosti pomohlo k tomu, aby sa dosiahlo BLEU skóre o 1,4 väčšie ako najlepšie riešenie od iných autorov. Perplexita nie je uvedená u riešení iných autorov, čiže nie je možné použiť túto metriku na porovnanie s tými riešeniami.



Obrázok 3 Porovnanie jednotlivých prístupov ako dobre dokázali preložiť vety určitej dĺžky BLEU skóre [6]

Pozornosť má pozitívny efekt aj pri preklade dlhších viet (Obrázku 7).

[6] Zdroj Obrázok 3: https://arxiv.org/pdf/1508.04025v5.pdf

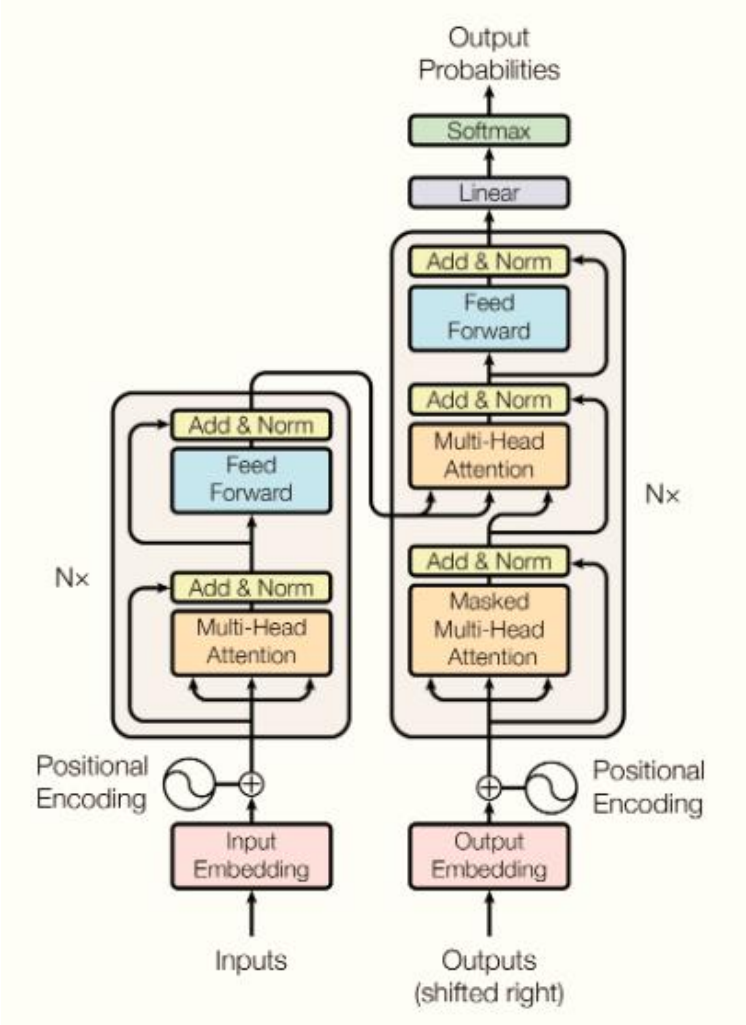
V predošlom riešení bola použitá technika Unknown replacement. Rieši problém s menej frekventovanými slovami, ktoré sú označené pomocou <unk> pri tvorbe slovnej zásoby. Pri trénovaní strojového prekladu sa použije pozícia každého slova v cieľovej vete, ktoré nie je v slovnej zásobe a priradí sa tá pozícia k zhodnému slovu vo vstupnej vete. Tieto pozície sú následne použité ako referencia pre slovník pri prekladaní textu. (Sutskever, 2015).

Ďalším variantom pozornosti je Self-attention, po slovensky by sa to dalo preložiť ako „sebapozornosť“, ale v rámci práce budeme referovať anglickým názvom. Self-attention sa zameriava na vzťahy medzi slovami v rámci rovnakej vety (Kalra, 2022). Self-attention si opíšeme bližšie v nasledujúcej kapitole.

Ako sme mohli vidieť v tejto podkapitole, využitím pozornosti môžeme výrazne zlepšiť model strojového prekladu. V ďalšej podkapitole budeme opisovať architektúru strojového učenia, ktorá používa self-attention.

1.5.2 TRANSFORMER

Jedným z populárnych modelov v dnešnej dobe je Transformer. Transformer je modelová architektúra, ktorá využíva pozornosť na vytvorenie globálnych závislostí medzi vstupom a výstupom. Dovoľuje omnoho vyššiu paralelizáciu ako staršie prístupy (napr. rekurentné neurónové siete) čo má za cieľ, že dokáže vytvoriť porovnateľne lepší výsledok v preklade za kratší čas (Vaswani et al., 2017).

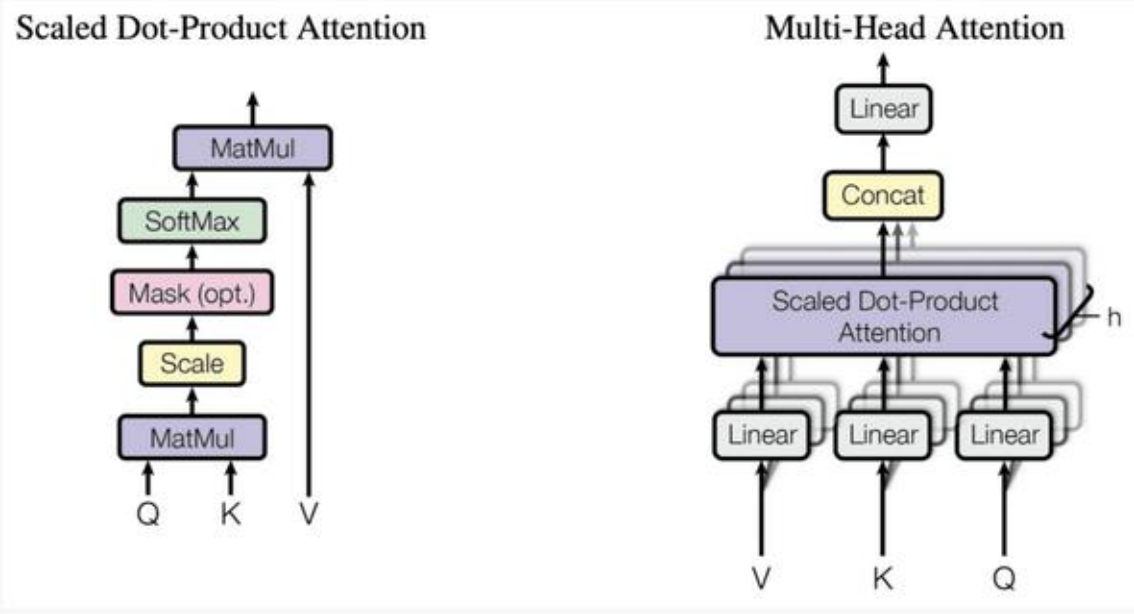


Obrázok 4 Architektúra Transformeru[7]

Enkóder (na Obrázku 4 je to žltý blok na ľavej strane) transformuje slová zo zdrojového textu (napr. z angličtiny) na číselné hodnoty a následne dekóder (na obrázku 8 je to žltý blok na pravej strane) z nich vygeneruje výstup v podobe preložených slov (napr. do slovenčiny). Nakoniec z dekódera vychádzajú preložené slová a funkcia softmax vypočíta pravdepodobnosť slov. Slovo s najvyššou pravdepodobnosťou sa zvolí ako pokračovanie vety. Proces sa opakuje, kým nie je preložená celá vstupná množina slov.

Enkóder sa skladá zo 6 identických vrstiev (N = 6) z toho každá sa skladá z dvoch podvrstiev. Prvá z týchto podvrstiev je multi-head self-attention a druhá podvrstva je dopredná neurónová sieť. (Vaswani et al, 2017). Na Obrázku 9 môžeme vidieť štruktúru scaled dot-product attention, ktorý je súčasťou multi-head attention.

[7] Zdroj Obrázok 4: https://arxiv.org/pdf/1706.03762.pdf



Obrázok 5 Pozornosť škálovaného skalárneho súčinu (vľavo) a multi-head attention (vpravo) [8]

Do pozornosti škálovaného skalárneho súčinu (ang. Scaled Dot-product) vstupujú hodnoty reprezentujúce vstupnú sekvenciu, ktoré slúžia na získanie informácií ohľadom ostatných časti vety. Na obrázku 5 reprezentované ako Q a K reprezentuje uložené informácie, ktoré sú použité vstupné hodnoty Q. Hodnoty Q a K následne budú použité v maticovom násobení (matmul), vydelené odmocninou rozmeru (dimenzionality) vektora K a na výsledky sa použije funkcia softmax, aby sa vypočítali váhy tokenov. Výsledné váhy sa následne znova násobia s hodnotou V, čo je množina skutočných hodnôt jednotlivých hodnôt vstupnej sekvencie (Vaswani et al., 2017). Vzorec je nasledovný:

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

Tento proces sa opakuje toľko krát, koľko heads (hláv) v modely máme. Hlava je hyperparameter, ktorý môžeme nastaviť podľa potreby. Každá hlava reprezentuje jednu pozornosť škálovaného skalárneho súčinu, ktorá má odlišné parametre v rámci hodnôt Q, K a V. Následne sa hodnoty všetkých hláv zreťazia a v multi-head pozornosti a máme k dispozícii konečné hodnoty váh pre každý token v slovnej zásobe.

Výstup multi-head pozornosti prechádza do doprednej neurónovej siete, ktorá pozostáva z dvoch lineárnych transformácií a jednej aktivačnej funkcie ReLU medzi

[8] Zdroj Obrázok 5: https://arxiv.org/pdf/1706.03762.pdf

nimi. V rámci každej vrstvy pracuje dopredná neurónová sieť s inými parametrami. (Vaswani et al, 2017). Vzorec je nasledovný:

$$Dopredna\_neuronova\_siet = \max(0, xW_1 + b1)\, W_2 + b_2$$

V časti dekóderu maskovanej viachlavovej pozornosti (ang. masked multi-head attention) funguje na podobnom princípe ako multi-head attention v časti enkóderu, s tým rozdielom, že tu sa aplikuje pred použitím aktivačnej funkcie softmax maska. Je nežiadúce, aby pozornosť vedela o ďalšej pozícií za aktuálnym tokenom, pretože by tým negatívne bola ovplyvnená schopnosť predpovedať nasledujúce tokeny, keďže by už dopredu vedela, čo sa tam nachádza. Preto sa nasledujúce tokeny, ktoré sú predpovedané z tokenov predošlých pozícií zamaskujú (Kierszbaum, 2020).

Z maskovanej viachlavovej pozornosti následne vychádzajú hodnoty Q a z časti enkódera vychádzajú hodnoty K a V, ktoré vstupujú do multi-head pozornosti na strane dekódera. Vykoná sa rovnaká operácia ako v multi-head pozornosti na strane enkódera, výstup sa vloží do doprednej neurónovej siete a výstup prechádza do aktivačnej funkcie softmax, ktorá vyberie tokeny s najvyššou pravdepodobnosťou výskytu (Vaswani, 2017).

Autori (Vaswani et al., 2017) nasledovne:

- Trénovali na datasete WMT 2014 v jazykoch angličtina-nemčina pozostávajúcich z 4,5 milióna párov viet. Nastavili veľkosť slovnej zásoby (ang. Vocabulary) na 37000 najčastejších slov (tokenov),
- Trénovali taktiež na datasete WMT 2014 v jazykoch angličtina-francúzština pozostávajúcich z 36 milióna párov viet. Nastavili veľkosť slovnej zásoby (ang. Vocabulary) na 32000 najčastejších slov (tokenov),
- Každá trénovacia podmnožina pozostávala približne z 25000 vstupných a cieľových tokenov,
- Použitý bol optimalizátor Adam, ktorý mal nastavený mieru učenia podľa vzorca:

$$miera\ učenia = d_{model}^{-0.5} * \min(početKrokov^{-0.5}, početKrokov * prvotnéKroky^{-1.5})$$

- Miera učenia pri prvotných krokoch (prvotnéKroky = 4000) rastie lineárne a potom klesá proporcionálne k druhej odmocnine početKrokov,
- Bol použitý dropout 0,1 pre základný model, 0,3 pre veľký model,
- Pre trénovanie bol použitý menší model v Tabuľka 7 označení ako (base model), ktorý bol trénovaný v počte 100000 krokov. Veľký model bol trénovaný v počte 300000 krokov.

Tabuľka 7 Porovnanie hodnotení jednotlivých trénovaní s aktuálnym trénovaním pomocou Transformeru na datasete WMT 2014 [9]

Model	BLEU	
	ANG - NEM	ANG - FR
ByteNet	23,75	
Deep-Att + PosUnk		39,2
GNMT + RL	24,6	39,92
ConvS2S	25,16	40,46
MoE	26,03	40,56
Deep-Att + PosUnk Ensemble		40,4
GNMT + RL Ensemble	26,3	41,16
ConvS2S Ensemble	26,36	**41,29**
Transformer (základný model)	27,3	38,1
Transformer (veľký model)	**28,4**	**41,8**

Oproti predošlým riešeniam bol Transformer úspešnejším v preklade datasetu WMT 2014 v jazykoch angličtina-nemčina v základnom i vo veľkom modely. Základný model prekonal v hodnotení BLEU predošlé najlepšie riešenie ConvS2S Ensemble o 0,94 a veľký model až o 2,04. V prípade WMT 2014 v jazykoch angličtina-francúzština základný model nedosahoval v hodnotení BLEU lepšie výsledky ako iné riešenia, ale veľký model dokázal prekonať predošlé najlepšie riešenie o 0,51.

[9] Zdroj Tabuľka 7: https://arxiv.org/pdf/1706.03762.pdf

1.6 OPENNMT

OpenNMT je toolkit hlbokého učenia, ktorý poskytuje potrebné prostriedky na trénovanie modelu strojového prekladu (Klein et al., 2018). Medzi výhody môžeme považovať napríklad:

- Možné trénovanie na procesore alebo grafickom procesore podľa dostupnosti a podpora využitia viacerých grafických procesorov na trénovanie modelu,
- modularita pre nastavenie trénovania podľa potrieb, čo je ideálne na výskumné účely,
- možné využitie viacerých architektúr ako transformer, konvolučnej alebo rekurentnej neurónovej siete

Podľa (Klein et al., 2018) existujú tri implementácie pre OpenNMT. Patria sem napríklad:

- OpenNMT-lua - prvotná verzia,
- OpenNMT-py – postavené na základe OpenNMT-lua s využitím PyTorch. Jednoduchá implementácia a je vhodná hlavne pre výskumné účely,
- OpenNMT-tf – verzia vychádzajúca z TensorFlow. Ideálne na veľké projekty s využitím funkcií TensorFlow.

2 CIELE ZÁVEREČNEJ PRÁCE

Cieľom práce je vytvoriť a natrénovať neurónový strojový preklad v smere prekladu z angličtiny do slovenčiny. Pripraviť trénovacie a evalvačné datasety s malým množstvom paralelných dát. Zvoliť si framework pre natrénovanie prekladových modelov a realizovať a vyhodnotiť experimenty s vybranými technikami ako napríklad transfer learning alebo semi-supervised learning a nakoniec navrhnúť vylepšenie vybraných techník.

Podciele alebo čiastkové ciele:

- vytvoriť a natrénovať model na základe vhodného frameworku
- otestovať ako dobre dokážu modely preložiť text,
- porovnať jednotlivé modely prekladu v rámci ich výsledkov, teda ako dobre dokážu preložiť text,
- vyhodnotiť dosiahnuté výsledky.

3 METODIKA VÝSKUMU

3.1 PREDSPRACOVANIE PARALELNÉHO TEXTU

Prvým krokom pre dosiahnutie cieľa práce je nájsť vhodný paralelný korpus, na ktorom budeme trénovať, zároveň validovať a následne testovať. Existuje viacero webových stránok, ktoré ponúkajú paralelné korpusy rôznych kvalít ako napríklad Opus. Z dôvodu experimentovania a pre zohľadnenie dostupných výpočtových zdrojov sme si zvolili relatívne malý paralelný korpus Europarl verzia 7 (Tiedemann, 2012), ktorý obsahuje 640 715 viet v jazykoch angličtina a slovenčina. Texty sú už zarovnané, čiže tento krok už nemusíme vykonávať. Je nevyhnutné, aby texty boli zarovnané, pretože model strojového prekladu by sa nedokázal správne naučiť spojitosti medzi jazykmi. Pre ukážku, ako vyzerajú konkrétne texty môžeme uviesť vzorku 5 viet pre oba jazyky:

Tabuľka 8 Vzorka paralelných textov z korpusu Europarl

Číslo riadku	**Europarl - anglicky**	**Europarl - slovensky**
2950	Here, the aim is what I have described: to review the existing legislation and determine whether it is still appropriate for today.	Cieľom je, ako som už hovoril: preskúmať existujúce právne predpisy a určiť, či sú vhodné aj v súčasnosti.
2951	It is quite clear, after all, that after around 50 years of European integration, this has to be done.	Je úplne logické, že po 50 rokoch európskej integrácie, je to očakávaný krok.
2952	On the issue of costs, it is a rather different matter.	Čo sa týka nákladov, to je niečo iné.
2953	Mr Helmer's frequent repetition of his assertions does not make them correct.	Častým opakovaním sa výroky pána Helmera nestanú pravdivými.
2954	I assume that he is not here, but I would like to make that clear once and for all.	Predpokladám, že tu nie je, ale rád by som to už raz a navždy objasnil.

Niektoré vety nie sú úplne správne preložené, ale dostatočne zachytávajú myšlienku originálnej vety a v porovnaní s inými dostupnými korpusmi je tento korpus pomerne kvalitný.

Predspracovanie korpusu a trénovanie modelu strojového prekladu budeme vykonávať v prostredí Colab pomocou grafického procerosu Tesla T4 a V100. Trénovanie neurónových sietí je výpočtovo a časovo veľmi náročné. Grafické procesory sú lepšie uspôsobené na paralelné výpočty, a preto sú lepšou voľbou pre trénovanie oproti procesorom.

Vybraný korpus je následne nutné predspracovať do vhodnej podoby, aby sme ho mohli použiť pri trénovaní modelu strojového prekladu. Načítame si anglické a slovenské texty, ktoré následne spojíme do jedného dátového súboru. Ak by sme upravovali súbory samostatne, mohla by nastať situácia, kedy by po viacerých úpravách vety už nemuseli byť zarovnané. Takto máme lepšiu kontrolu nad textami.

Pomocou knižnice Pandas sme dokázali vytvoriť dva datasety. Source, teda zdrojový jazyk pre anglický text a target, teda cieľový jazyk pre slovenský text. Následne spojíme tieto dva datasety do jedného, ktorý sme si pomenovali jednoducho „*df*“. Rozmery datasetu sú ako sme už vyššie spomenuli 640 715 riadkov.

Pred pokračovaním je nutné podotknúť, že väčšina krokov, ktoré budeme opisovať, nie je závislá na poradí. Väčšine krokov by sme mohli zmeniť poradie a na výslednom korpuse by to nespôsobilo zmenu obsahu. Na druhej strane niektoré kroky sú výpočtovo a tým aj časovo náročnejšie, a preto je nutné zvážiť, ktoré kroky by mali byť vykonané skôr. Ideálne je začať s odstránením NaN (prázdnych) riadkov, aby sme zbytočne nimi neprechádzali v nasledujúcich krokoch. Dataset df tvoria aktuálne dva stĺpce a to vety v anglickom a vety v slovenskom jazyku zarovnane. Ak odstránime napríklad riadok, kde v jednom jazyku je NaN (prázdny riadok) a v druhom jemu priložená normálna veta, dosiahneme zachovania zarovnania textov a zároveň sa zbavíme neúplných dát. Týmto krokom sme odstránili 761 prázdnych riadkov a zostalo nám 639 954.

Unicode normalizácia

Ďalším krokom, s ktorým sme experimentovali je Unicode normalizácia. Význam tejto normalizácie je podrobnejšie opísaný v podkapitole 1.3 Predspracovanie textu. Nasledujúca operácia si vyžaduje importovať knižnicu unicodedata. Aby sme na datasete mohli vykonať túto normalizáciu je nutné transformovať obsah stĺpcov na

dátový typ string. V opačnom prípade proces neprejde, keďže sa v stĺpcoch môžu nachádzať aj iné dátové typy ako napríklad float. Následne môžeme vykonať unicode normalizáciu pomocou normalizácie NFKC.

```
df['Source'] = df['Source'].astype(str).apply(lambda x: unicodedata.normalize('NFKC', x))
df['Target'] = df['Target'].astype(str).apply(lambda x: unicodedata.normalize('NFKC', x))
```

Po vykonaní unicode normalizácie nastala zmena v 22 source (anglických) vetách a v 19 target (slovenských) vetách. V prepočte pre anglické vety zmena postihla 0.0034% a pre slovenské vety 0.0029%. To je zanedbateľné množstvo v pomere k celému korpusu, ale v niektorých prípadoch by mohla byť táto transformácia vhodná. Dôvodom, prečo sa počet normalizovaných viet medzi jazykmi líši je v dôsledku preloženia niektorých viet bez použitia špecifického znaku. Pre ukážku si uvedieme niektoré príklady zmien spomínaných viet.

Tabuľka 9 Unicode normalizácia znaku exponentu z paralelného korpusu Europarl

Pred normalizáciou: (The order of business was adopted¹)
Po normalizácií: (The order of business was adopted1)

V tabuľke 9 uvedenej vyššie normalizácia premenila exponent 1, ktorý reprezentoval pravdepodobne referenciu na zdroj, na obyčajné číslo 1. Pre účely strojového prekladu je toto lepšie riešenie, keďže v rámci vety exponent 1 nemá žiadny význam. Následne by model musel riešiť dve samostatné slová a to adopted a adopted¹. Slovo adopted1 taktiež v aktuálnom stave nemá pre nás význam, ale to bude ešte upravené oddelením čísla 1 od zvyšku v procese tokenizácie. Táto konkrétna normalizácia prebehla v anglickej aj slovenskej variante vety. Exponenty dávajú kontext v prípade, keď sa v texte vyskytuje matematická notácia ako napríklad 2^2 (v prípade korpusu Europarl taký prípad nie je) alebo v prípade plošnej miery ako km^2. V týchto prípadoch je na zváženie, či unicode normalizácia je vhodná alebo by bolo lepšie vytvoriť podmienky, ktoré by ignorovali exponent v prípade ak sa napája na číslo alebo na jednotky plochy. Tým by sme zachovali kontext vety, keďže je rozdiel medzi 2^2 a 22, čo by vzniklo následkom unicode normalizácie. Príklad takejto úpravy v kóde je nasledujúci:

```
df['Source'] = df['Source'].str.replace(r'(\b\w{4,})\s*([¹²³⁴⁵⁶⁷⁸⁹⁰]+)', r'\1', regex=True)
```

```
df['Target'] = df['Target'].str.replace(r'(\b\w{4,})\s*([¹²³⁴⁵⁶⁷⁸⁹⁰]+)', r'\1', regex=True)
```

Použijeme kód, v ktorom špecifikujeme regulárny výraz, ktorý odstráni exponent pri slovách dlhších ako 3 znaky. Tým zabezpečím, že sa neodstráni exponent pri mierach ako napríklad m, km atď.

Ďalším znakom je º (U+00BA) a ˚ (U+02DA), ktoré reprezentujú v našom prípade stupeň (Unicode, 2021). V tomto porovnaní budeme uvádzať aj unicode kód, pretože znaky º a ˚ sú si veľmi podobné.

Tabuľka 10 Unicode normalizácia znaku º (U+00BA) z paralelného korpusu Europarl

Pred normalizáciou: Our common goal is to limit global warming to the celebrated 2 ºC.
Po normalizácií: Our common goal is to limit global warming to the celebrated 2 oC.

V tabuľke 10 môžeme vidieť, že znak º (U+00BA) po unicode transformácií premenilo na malé o. Toto riešenie nemusí byť žiadúce, keďže oC na prvý pohľad neevokuje, že by sa jednalo o označenie jednotky teploty. V rámci kontextu vety na to čitateľ môže prísť, ale nie je to správne označenie.

Tabuľka 11 Unicode normalizácia znaku ˚ (U+02DA) z paralelného korpusu Europarl

Pred normalizáciou: Správa usudzuje, že strop globálneho otepľovania 2 ˚C je "strategickým cieľom" EÚ, a zároveň pripúšťa, že "takáto úroveň otepľovania by mala zásadný vplyv na našu spoločnosť a životný štýl jednotlivcov".
Po normalizácií: Správa usudzuje, že strop globálneho otepľovania 2 ℃ je "strategickým cieľom" EÚ, a zároveň pripúšťa, že "takáto úroveň otepľovania by mala zásadný vplyv na našu spoločnosť a životný štýl jednotlivcov".

Lepší výsledok môžeme vidieť v tabuľke 11. Znak ˚ (U+02DA) pomocou unicode normalizácie sa transformuje na ˚ (U+030A), čo zachová kontext, teda označenie stupňa. Možný návrh je vytvoriť pravidlo, ktoré by pred unicode normalizáciou mohlo premeniť º (U+00BA) na ˚ (U+02DA), čo by malo za následok zmeny z výsledku po normalizácií oC na ℃. Unicode normalizácia má význam, keďže ˚C a ºC by bral model ako dve odlišné slová a slúži ako dobrý nástroj na detekciu

odlišných variant znakov, ktoré by sme si priebežným čítaním nemuseli všimnúť. Pre zjednotenie znakov použijeme nasledujúci kód:

```
df['Source'] = df['Source'].str.replace('\u00BA', '\u030A')
df['Target'] = df['Target'].str.replace('\u00BA', '\u030A')
df['Source'] = df['Source'].str.replace('\u02DA', '\u030A')
df['Target'] = df['Target'].str.replace('\u02DA', '\u030A')
```

Tabuľka 12 Unicode normalizácia znaku ½ z paralelného korpusu Europarl

Pred normalizáciou: Let me point out the following: according to the formal data, every 4½ seconds an employee has an accident and every 3½ minutes an employee loses his life.
Po normalizácií: Let me point out the following: according to the formal data, every 41⁄2 seconds an employee has an accident and every 31⁄2 minutes an employee loses his life.

Posledný príklad, ktorý si uvedieme v rámci unicode normalizácie korpusu Europarl je transformácia znaku ½ (U+00BD) (Unicode, 2021). V tabuľke 12 môžeme vidieť, že slovné spojenie „3½ minutes“ zmenilo na „31⁄2 minutes“, čo úplne mení hodnotu, ktorú by malo reprezentovať. V tomto prípade unicode normalizácia nie je vhodná forma, ako predspracovať text, keďže môže úplne zmeniť kontext vety. Pre znak ½ by bolo vhodnejšie vytvoriť kód, ktorý pri prechádzaní vetami, zmenil znak ½ na ,5 , keďže ½ v prípade Europarl sa nachádza len 4 krát a vždy v spojení s číslom. Návrh kódu je nasledovný:

```
df['Source'] = df['Source'].str.replace('\u00BD', ',5')
df['Target'] = df['Target'].str.replace('\u00BD', ',5')
```

Europarl je kvalitný korpus, ktorý má vety z jedného zdroja, čiže veľmi veľa cudzích znakov sa v korpuse nevyskytuje a transformácia sa vzťahuje len na pár znakov. Podľa týchto zistení môžeme konštatovať, že unicode normalizácia pre prácu s Europarl je vhodná skôr ako nástroj na detekciu takýchto znakov, ako na úpravu. Preto pred predspracovaním korpusu vykonáme unicode normalizáciu a podľa zistených výsledkov aplikujeme kód, ktorý potrebné zmeny vykoná. V prípade použitia korpusu Europarl spomenuté návrhy kódu pre exponenty, znaky stupňa a znaku ½ (U+00BD).

Počet zmenených riadkov pre source (anglické texty) je 17 a pre target (slovenské texty) je 13. Nepomer počtu zmenených riadkov oproti použitiu unicode normalizácie je v dôsledku ponechania viet, kde exponent je vedľa miery ako napríklad km. Významnejšie zmeny, by sme videli ak by sme použili korpus vytvorený pomocou webcrawlera, kde texty sú zozbierané z desiatok až miliónov iných zdrojov. Vtedy je otázne, či by bolo možné urobiť výnimku na každý prípad.

Odstránenie rovnakých a duplicitných riadkov

Po úprave znakov nasleduje krok, v ktorom sa odstránia riadky, kde anglické a slovenské vety sú rovnaké. Model sa nemá na čom trénovať ak vety sú identické alebo nepreložené, keďže tam nie je žiadny vzorec, okrem rovnosti. Navyše šetríme výpočtovými zdrojmi, ak model nemusí prechádzať týmito vetami. Značná časť odstránených riadkov sú poradové čísla alebo nepreložené frázy v jazykoch iných ako angličtina a slovenčina, prípade nepreložené vety v angličtine, ktoré sa prepísali aj do slovenského prekladu. Vzorka takýchto prípadov je v tabuľke 13.

Tabuľka 13 Vzorka riadkov, kde sa anglický riadok rovná slovenskému

Europarl - anglicky	**Europarl - slovensky**
1.	1.
(For the results and other details on the vote: see Minutes)	(For the results and other details on the vote: see Minutes)
Pactio Olisipiensis censenda est!	Pactio Olisipiensis censenda est!
Köszönöm szépen Jóska.	Köszönöm szépen Jóska.
∘ ∘ ∘ ∘	∘ ∘ ∘ ∘

Nepriamy, ale žiadúci efekt má táto operácia aj na odstránenie riadkov, kde sa nachádzajú len poradové čísla alebo špeciálne znaky, ako sú v tabuľke 13, keďže boli prepísané do slovenského prekladu rovnako. Touto operáciou sme odstránili 3250 riadkov a zostalo nám 636 704 riadkov.

Podobným krokom je odstránenie riadkov, ktoré sa už nachádzajú v našom korpuse. Je zbytočné, aby sa model viackrát trénoval na rovnakej kombinácií viet. Zabraňujeme tým aj možnému zaujatiu modelu voči týmto príkladom, keďže by mohli reprezentovať väčšinu dostupných príkladov prekladu, nehovoriac o ušetrení

výpočtovými zdrojmi. Touto operáciou sme zabezpečili, že každá dvojica anglických a slovenských viet je unikátna. Odstránili sme 15815 nadbytočných riadkov, čo je najvyšší počet odstránených riadkov v rámci celého predspracovania. Počet zvyšných riadkov je 620 873.

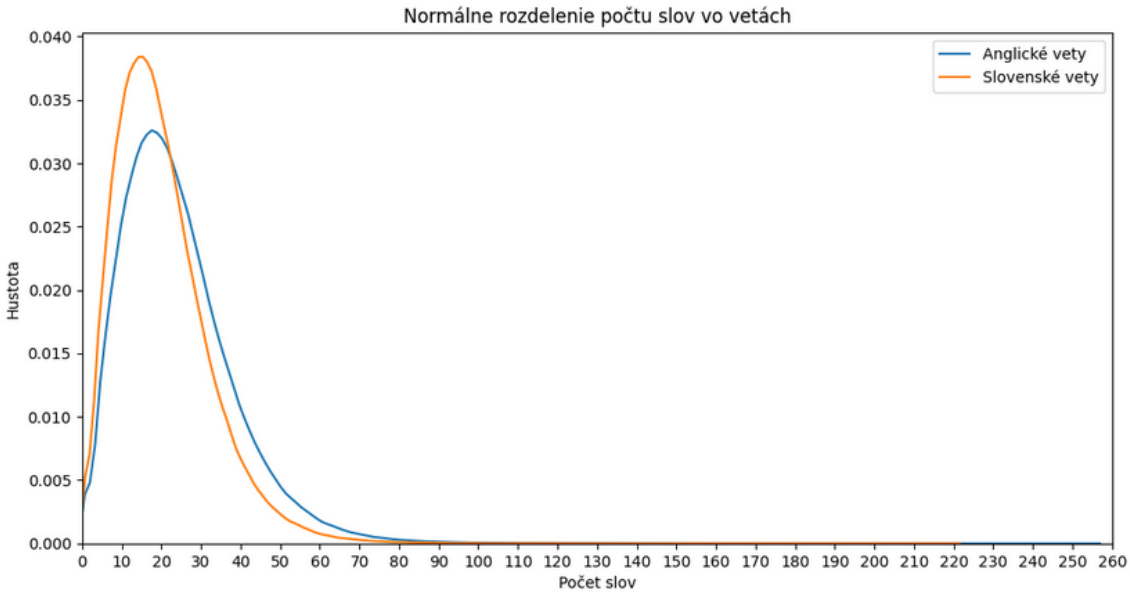
Odstránenie viet, ktorých dĺžka je príliš dlhá

Aj dĺžka viet môže ovplyvniť trénovanie strojového prekladu. Pri dlhých vetách môže byť pre model náročné zistiť kontext vety. Minimum znakov, po ktorom by sme vetu mohli považovať za príliš dlhú je diskutabilné. Priemerný počet slov alebo dĺžky slov je odlišná od jazyka. Pre zistenie možnej hranice dĺžky viet sme si vytvorili pomocný kód, ktorý vypočíta priemerný a maximálny počet slov vo vetách Europarl. Vety sme rozdelili v rámci jednotlivých jazykov na slová a následne sme aplikovali spomínané vypočítanie priemeru a maxima počtu slov.

Tabuľka 14 Priemerný a maximálny počet slov vo vetách (pred úpravami korpusu Europarl okrem odstránenie NaN hodnôt a po aplikovaní niektorých úprav)

	Anglické vety	Slovenské vety
Priemerný počet slov (pred úpravou)	≈ 24,13	≈ 20,21
Priemerný počet slov (po úpravách)	≈ 24,68	≈ 20,67
Maximálny počet slov (pred aj po úprave)	254	219

V tabuľke 14 môžeme vidieť, že priemerný počet slov sa pohybuje okolo 24 – 25 slov pre anglický text a pre slovenský 20 -21. Uvádzame dáta pre vety, ktoré prešli len odstránením NaN hodnôt z korpusu a dáta pre vety, ktoré si prešli okrem odstránenia NaN hodnôt aj úpravu niektorých znakov a odstránenie zhodných a duplicitných viet. Môžeme vidieť, že po odstránení 19098 (≈2,98%) viet nastala zmena približne pol slova. Dáta po úpravách lepšie reprezentujú korpus. Budeme pracovať s tými hodnotami. Pre vizualizáciu použijeme graf normálneho rozdelenia.

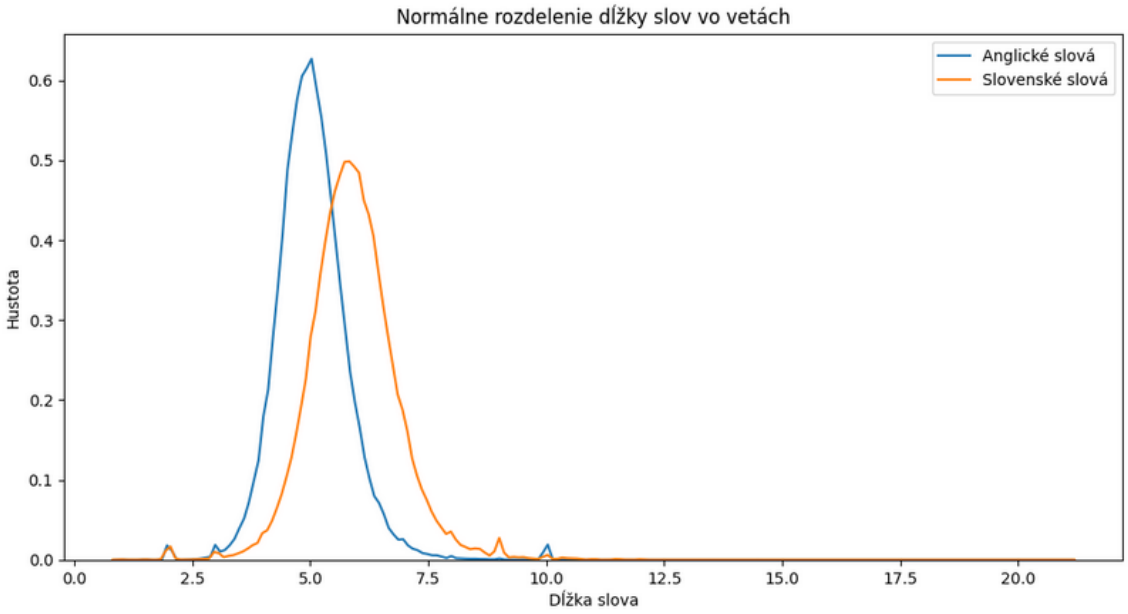


Obrázok 6 Normálne rozdelenie slov vo vetách korpusu Europarl

Na obrázku 6 vidíme, že značná časť viet má počet slov menší ako 35 – 40. Na vyhodnotenie maximálnej dĺžky viet potrebujeme zistiť aj priemernú dĺžku slov. Slová vo vetách sme rozdelili na jednotlivé znaky a vypočítali sme ich priemernú a maximálnu dĺžku.

Tabuľka 15 Priemerná a maximálna dĺžka slov vo vetách po predošlých úpravách korpusu Europarl

	Anglické vety	Slovenské vety
Priemerná dĺžka slova	≈ 5,04	≈ 5,91
Maximálna dĺžka slova	21	21



Obrázok 7 Normálne rozdelenie dĺžky slov vo vetách korpusu Europarl

V tabuľke 15 môžeme vidieť, že priemerná dĺžka slov v anglických vetách sa pohybuje okolo 5 znakov a pre slovenské vety je dĺžka približne 6 znakov. Pre zjednotenie hraničnej dĺžky by sme mohli počítať pomocou priemerného počtu slov anglického jazyka zaokrúhleného nahor, teda 24,13 -> 25. Následne túto hodnotu vynásobíme priemernou dĺžkou slovenského slova a prilepšíme o 1, keďže rozptyl na obrázku 7 má značnú časť medzi hodnotami 5 a 7. Tým sa dopracujeme k výpočtu 25 * 7 = 175. Touto operáciou sme odstránili 16 riadkov a zostalo nám 620 873. Tieto vety boli častokrát viacero viet spojených dokopy znakmi ako pomlčka alebo bodkočiarka. Možné riešenie by bolo aj ručné rozdelenie viet, ale v tomto prípade to nevykonáme.

Úprava HTML značiek

V korpusoch sa môžu nachádzať aj HTML značky, ktoré je nutné transformovať do vhodnej podoby, keďže sami o sebe nemajú význam v rámci normálnej vety. Krok pre anglické a slovenské vety je rovnaký, čiže si uvedieme len časť, ktorá upravuje source (anglický text):

```
df['Source'] = df['Source'].str.replace(r'&amp;', '&', regex=True)
df['Source'] = df['Source'].str.replace(r'&quot;', '"', regex=True)
df['Source'] = df['Source'].str.replace(r'&apos;', '\'', regex=True)
df['Source'] = df['Source'].str.replace(r'&lt;', '<', regex=True)
df['Source'] = df['Source'].str.replace(r'&gt;', '>', regex=True)
```

V prípade korpusu Europarl tento krok nie je potrebný, keďže neobsahuje HTML značky. Tento krok by bol vhodnejší pre korpus vytvorený pomocou webcrawlera, ale pre univerzálnosť kódu sme ho ponechali.

Dodatočné odstránenie niektorých znakov

V dôsledku relatívnej čistoty korpusu a malého množstva znakov, ktoré sme upravovali v podkapitole Unicode normalizácia, sme použili drastické riešenie pre čistenie korpusu a to odstránenie všetkých znakov, ktoré nie sú písmená, čísla alebo medzera. To sú myslené znaky ako zátvorky, pomlčky alebo bodky. Tým sme dosiahli čistý korpus, ktorý dosahoval dobré výsledky po trénovaniach. V neskorších pokusoch sme naše riešenie upravili, aby sme zachovali znaky identifikované unicode normalizáciou.

Úprava textu na malé písmená

Ďalším krokom nášho predspracovania korpusu je zmena veľkých písmen na malé. V prípade anglicko-slovenského prekladu, teda nášho, zmenenie veľkých písmen na malé ovplyvní len vlastné podstatné mená a začiatok viet. V prvých pokusoch sme zmenili všetky veľké písmená na malé.

V neskorších pokusoch sme upravili tento kód tak, aby len začiatočné slovo bolo zmenené na malé písmeno, aby sme zachovali už vyššie spomenuté vlastné podstatné mená.

Filtrovanie textov korpusu EuroParl pomocou knižnice langdetect

Ako už bolo spomenuté v kapitole 1.2 Predspracovanie dát, knižnica langdetect slúži na identifikovanie jazykov. Pre naše potreby ho použijeme na filtrovanie viet tak, aby v stĺpci source datasetu df boli len vety identifikované ako „en“, teda anglické a v stĺci target datasetu df boli len vety identifikované ako „sk“, teda slovenské. Týmto zabezpečíme, že sa v našom korpuse nebudú nachádzať cudzojazyčné vety. Pred použitím Langdetect si vytvoríme funkciu, ktorá identifikuje jazyk vety a ošetrí prípady, ak by nebolo možné identifikovať jazyk alebo ak by s konkrétnou vetou nastal problém. Langdetect je nedeterministický algoritmus, čiže pre získanie rovnakých výsledkov pri každom spustení si zvolíme hodnotu (ang. seed), ktorá zabezpečí jednotnosť. Kód je nasledovný:

```
def kontrola(text):
    try:
        return detect(text)
    except LangDetectException:
        return None

DetectorFactory.seed = 42 # hodnota pre pseudonáhodný generátor čísel
```

V rámci datasetu df si vytvoríme stĺpce „source_jazyk“ a „target_jazyk“, do ktorých sa bude uchovávať hodnota reprezentujúca jazyk identifikovaná pomocou funkcie „kontrola“. Kód je dodatočne ošetrený pre prípad ak by riadok bol náhodou prázdny alebo by obsahoval len jedno slovo. Prázdne riadky by nemali byť v rámci datasetu, keďže sa odstraňovali v predošlom kroku a pravidlo 1 slova zabezpečuje, že nebudeme porovnávať medzi sebou len jedno slovo. Dôvodom je, že 1 slovo môže mať viacero možných prekladov záležiac od kontextu. Ideálne je učiť model slovami v rámci viet, než samostatne. Kód je nasledovný:

```
df['source_jazyk'] = df['Source'].apply(lambda x: kontrola(x) if x and len(x.split()) > 1 else None)
df['target_jazyk'] = df['Target'].apply(lambda x: kontrola(x) if x and len(x.split()) > 1 else None)
```

Po vytvorení spomínaných stĺpcov porovnáme, či stĺpec „source_jazyk“ obsahuje anglický text a stĺpec „target_jazyk“ slovenský text. Ak v oboch prípadoch sa riadky zhodujú s požadovanými jazykmi, tak ich ponecháme v datasete. Porovnávanie vykonávame pomocou nasledovného kódu:

```
df = df[(df['source_jazyk'] == "en") & (df['target_jazyk'] == "sk")]
```

V tabuľke 16 máme k dispozícii vzorku viet, ktoré neprešli kontrolou jazyka. V prvom riadku anglická veta je správne identifikovaná ako anglická, ale na strane, kde by mal byť slovenský preklad máme miesto toho grécky preklad. Je zrejmé, že sa jedná o chybný prepis prekladu.

V druhom riadku bola strana pre anglický opis identifikovaná ako nórčina a na strane slovenského prekladu dánčina. V tomto prípade obe vety nezodpovedajú ich správnemu jazyku.

Tretí riadok môže byť zavádzajúci, keďže na prvý pohľad sa zdá, že anglická veta je správna. Privlastňovacie prídavné meno (ženské, dámske) v angličtine by sa malo písať ako women´s. Za bežných okolností množné číslo sa píše so „s“ na konci

slova, ale v prípade women (ženy, dámy), nie je koncovka „s“ potrebná. To je dôvod, prečo Langdetect neidentifikuje vetu ako anglickú. Slovenský preklad je správny.

Štvrtý riadok sme si uviedli ako príklad toho, že Langdetect nie je dokonalý. Anglickú vetu identifikuje správne ako anglickú vetu, ale slovenskú vetu identifikuje ako českú. Pri krátkych vetách môže mať Langdetect nedostatok informácií a na základe toho môže nesprávne identifikovať jazyk, hlavne ak sú si jazyky podobné ako slovenčina a čeština. Možné riešenie by bolo pridať dodatočnú podmienku, ktorá by vety identifikované pomocou Langdetect ako české vložila medzi slovenské. Na druhej strane vzniká riziko, že pridáme zle preložené vety, ktoré sú skutočne v češtine.

Piaty riadok si uvádzame ako príklad viet, ktoré neprešli podmienkou o počte slov, teda počet slov musí byť vyšší ako 1. Ako už bolo spomenuté, jedno slovo môže mať viacero významov podľa kontextu. Slovo simple podľa prekladača od spoločnosti Google má 13 možných jednoslovných prekladov do slovenčiny ako napríklad jednoduchý, naivný, či úprimný. Ak by sme trénovali model strojového prekladu pomocou jednoslovných fráz, tak existuje možnosť, že by v rámci prekladu do viet vkladal zlý variant slova, keďže by nebral ohľad na ostatné slová nachádzajúce sa vo vete.

Tabuľka 16 Príklady odstránených viet pomocou langdetect

	Europarl - anglicky	**Europarl - slovensky**
1	the sitting was opened at 9 am	συνεδρίαση αρχίζει στις 9 πμ
2	report elles	informe elles
3	womens immigration vote	úloha a miesto prisťahovalkýň v európskej únii hlasovanie
4	i shall start with a short announcement	začnem krátkym oznámením
5	simple	Je to jednoduché

Po vykonaní tejto operácie v datasete df pre korpus Europarl bolo odstránených 9390 riadkov (1,47% z úvodného počtu) a zostalo nám 611 479. Tento krok nám umožňuje odstrániť vety jazykov, ktoré sa nerovnajú nám zvoleným jazykom: Je nutné podotknúť, že odstráni aj mnoho viet, ktoré sú skutočne slovenské, ale knižnica Langdetect ich zle identifikovala. Z časového hľadiska trvá tento krok výrazne dlhšie ako predošlé kroky. Všetky predošlé kroky dokopy trvali približne 40 sekúnd, Langdetect trval 5516 sekúnd.

Zmena poradia viet

Posledným krokom predspracovania korpusu pred jeho tokenizáciou je zmena poradia riadkov datasetu. Tento krok nie je nevyhnutný, ale môže pomôcť prerozdeliť vety po celom korpuse. Vhodné v prípade, ak náš korpus obsahuje texty z viacero tém, ktoré majú vlastnú slovnú zásobu, ktorá by sa bežne nevyskytovala v iných témach. Kód na vykonanie tohto kroku je nasledovný:

```
df = df.sample(frac=1, random_state=42).reset_index(drop=True)
```

Funkcia sample() zvolí vzorku riadkov a zmení im poradie, frac = 1 zvolí všetky riadky datasetu a pridáme random_state, ktorým zabezpečíme pri opakovanom spustení rovnaký výsledok, aby sme mohli porovnávať. Posledná časť kódu zmení indexy riadkov (číslovanie) podľa aktuálneho usporiadania.

Zhrnutie predošlých krokov predspracovania na korpuse Europarl

Po vykonaní krokov predspracovania máme k dispozícií dataset korpusu Europarl, ktorý je v hodnej podobe na tokenizáciu. Súhrn krokov vykonaných na korpuse s hodnotami ako počet zmenených alebo odstránených riadkov sa nachádza v tabuľke 17.

Tabuľka 17 Súhrn krokov predspracovania na korpuse Europarl

Popis kroku	**Zostatok riadkov**	**Odstránené alebo upravené riadky**
Vytvorenie datasetu	640715	-
Odstránenie NaN hodnôt	639 954	761
Normalizácia na základe unicode znakov	639 954	ANG(17) / SK(13)
ANG text = SK text	636 704	3250
Odstránenie duplicitných riadkov	620 889	15815
Odstránenie príliš dlhých viet	620 873	16
Konverzia HTML znakov	620 873	0
Odstránenie niektorých znakov	620 873	nemerané
Konverzia na malé písmená	620 873	nemerané
Dodatočné odstránenie NaN hodnôt	620 869	4
Kontrola jazyka pomocou Langdetect	611 479	9390

Zmena poradia riadkov	611 479	-

Celkový čas procesu predspracovania podľa tabuľky 17 trval 5516.95 sekundy, teda približne 91 minút a zostalo nám 611 479 riadkov. Dataset df rozdelíme na dva samostatné súbory pre anglický a slovenský jazyk, na ktorých vykonáme tokenizáciu.

Predspracovanie textov korpusu ParaCrawl

V predošlých podkapitolách sme opisovali kroky predspracovania korpusu Europarl, ktorý má vety z jedného zdroja a až na niektoré prepisy viet v cudzích jazykoch ho môžeme považovať za kvalitný. Pre porovnanie predspracujeme korpus, ktorý bol vytvorený pomocou webcrawlera a tým pádom jeho obsah je tvorený vetami z mnohých zdrojov. Budeme pracovať s korpusom ParaCrawl verzia 9 pre jazyky angličtina/slovenčina (ParaCrawl, 2022). ParaCrawl verzia 9 má 22 901 690 zarovnaných viet v prípade už spomínanej kombinácie jazykov. Pre porovnanie s korpusom Europarl sme vytvorili korpus, ktorý obsahuje presne rovnaké množstvo riadkov ako Europarl, teda 640 715 riadkov. Vykonali sme rovnaké kroky ako v prípade Europarl.

Vykonali sme Unicode normalizáciu v samostatnom kroku a počet zmenených riadkov bol značne odlišný. Pri Europarl po vykonaní unicode normalizácie nastala zmena v 22 source (anglických) vetách a v 19 target (slovenských) vetách. Pre ParaCrawl po vykonaní unicode normalizácie nastala zmena v 5131 source (anglických vetách) a v 5835 target (slovenských) vetách. V tomto prípade sme neupravili zoznam znakov, ktoré by mali byť upravené a miesto toho sme vykonali len rovnakú úpravu ako v prípade Europarl, teda znaky vyjadrujúce exponent, ½ a stupne. Ako môžeme vidieť v tabuľke 18, týmito úpravami sme zmenili 109 anglických a 116 slovenských viet. ParaCrawl neobsahoval žiadne rovnaké alebo duplicitné vety, pri nastavení hraničnej dĺžky 175 znakov žiadna veta nepresiahla túto hodnotu. Zaujímavé je množstvo viet, ktoré boli odstránené v dôsledku filtrovania pomocou knižnice Langdetect. Z celkového množstva viet (640 715) bolo odstránených 128 527 viet, čo reprezentuje 20,04%. Tým nám zostalo 512 176 riadkov, ktoré možno ďalej použiť na trénovanie, testovania a validáciu modelu.

Tabuľka 18 Súhrn krokov predspracovania na korpuse ParaCrawl upravený počtom riadkov pre porovnanie s Europal

Popis kroku	**Zostatok riadkov**	**Odstránené alebo upravené riadky**
Vytvorenie datasetu	640 715	-
Riadky bez NaN hodnôt	640 703	12
Normalizácia na základe unicode znakov	640 703	ANG(109) / SK(116)
ANG text = SK text	640 703	0
Odstránenie duplicitných riadkov	640 703	0
Odstránenie príliš dlhých viet	640 703	0
Konverzia HTML znakov	620 873	0
Odstránenie niektorých znakov	620 873	nemerané
Konverzia na malé písmená	640 703	nemerané
Dodatočné odstránenie NaN hodnôt	640 703	0
Kontrola jazyka pomocou Langdetect	512 176	128 527
Zmena poradia riadkov	512 176	-

Prejdením odlišného korpusu sme si ukázali, ako sa môžu líšiť významnosťou jednotlivé kroky predspracovania. ParaCrawl obsahoval väčšie množstvo znakov, ktoré boli identifikované pomocou unicode normalizácie ako v prípade Europarl. V prípade Europarl rovnaké vety (3250) alebo duplicitné vety (15815) tvorili dokopy 19065, ale v ParaCrawl sa nenachádzala ani jedna veta tohto typu, čiže tento krok by sme mohli aj vynechať. Kontrola jazyka pomocou Langdetect priniesla omnoho rozsiahlejšie zmeny v ParaCrawl ako v Europarl. Vždy je nutné zvážiť, aké kroky sú vhodné pre predspracovanie, aby sme dosiahli kvalitný korpus.

3.2 TOKENIZÁCIA TEXTOV

Tokenizácia je taktiež krokom predspracovania textov, ale pre jej dôležitosť sme ju zahrnuli pod samostatnú kapitolu. Tento proces je rozsiahlejšie popísaní v podkapitole 1.3.1 Tokenizácia dát. V tejto podkapitole budeme tokenizovať vety korpusu Europarl na subwordy respektíve budeme pracovať s dvoma samostatnými súbormi, ktoré sme si vytvorili pomocou predošlých krokov predspracovania.

Pre tokenizáciu budeme používať knižnicu Sentencepiece. Pred tokenizáciou anglických a slovenských textov je nutné natrénovať model Sentencepiece. Definujeme si premenné source (pre anglické texty) a target (pre slovenské texty), v ktorých sa budú nachádzať špecifikácie modelu. Kód pre obe premenné je nasledovný:

```
#Parametre
vocab_size = 8000
character_coverage = 0.9995

# Model pre subwordovanie anglického textu (source)
source = f'--input={en_filt} --model_prefix=source --vocab_size={vocab_size} --hard_vocab_limit=false --character_coverage={character_coverage} --split_digits=true --model_type=bpe'
# Model pre subwordovanie slovenského textu (target)
target = f'--input={sk_filt} --model_prefix=target --vocab_size={vocab_size} --hard_vocab_limit=false --character_coverage={character_coverage} --split_digits=true --model_type=bpe'
```

Z dôvodu, že sme experimentovali s veľkosťou slovnej zásoby, vytvorili sme si samostatnú premennú vocab_size, ktorá mení veľkosť pre source aj target. Parameter character_coverage sme nemenili behom pokusov a ponechali sme ho na hodnote 0,9995, čo znamená, že model pokrýva 99,95% znakov nachádzajúcich sa v modely. Tým by sme mohli zabezpečiť o niečo menšiu slovnú zásobu, keďže sa nevytvárajú hodnoty obsahujúce znaky, ktoré sa nachádzajú v textoch len zriedkavo. Pri relatívne čistom korpuse ako je Europarl je otázne, či je vhodné použiť túto hodnotu alebo by bolo lepšie ponechať základ a to 1, teda 100%. Pre korpus ParaCrawl, ktorý obsahuje značne väčšie množstvo odlišných znakov by sme mohli očakávať lepšie využitie. V samotných premenných vkladáme súbory, s ktorými budeme pracovať v časti kódu input={sk_filt}, model_prefix=target slúži na pomenovanie, v tomto prípade výstup bude target.model, vocab_size={vocab_size} obsahuje premennú, ktorú sme si spomínali vyššie na definovanie veľkosti slovnej zásoby. Definovaním hard_vocab_limit=false dovoľujeme modelu, by sa striktne nedržal definovanej veľkosti. Tým môžeme zabrániť možnej strate časti slovnej zásoby, čo by mohlo negatívne ovplyvniť výsledný model. Model nemusí vytvoriť slovnú zásobu pre slovenský jazyk o veľkosti 8000 ako sme si definovali, ale napríklad 8210. Hodnota character_coverage obsahuje pokrytie znakov ako sme si uviedli vyššie. Rozdelenie čísel vykonávame pomocou split_digits=true. Týmto krokom transformuje napríklad číslo 1000 -> 1 0 0 0. Tým zabezpečíme, že slovná zásoba nebude obsahovať mnoho kombinácií čísel, len základné číslovky. Posledný parameter je model_type, ktorý môže

byť bpe, teda Byte Pair Encoding alebo Unigram. V rámci našich pokusov sme použili oba varianty.

Po vytvorení premenných source a target, ktoré obsahujú vyššie spomenuté parametre ich využijeme na trénovanie modelov Sentencepiece zvlášť pre anglický a slovenský jazyk. Kód je nasledovný:

```
spm.SentencePieceTrainer.train(source)
spm.SentencePieceTrainer.train(target)
```

Po vykonaní trénovania máme k dispozícií model anglického a slovenského jazyka, pomocou ktorých môžeme tokenizovať texty. Pre tokenizáciu textov na subwordy si vytvoríme funkciu „subwordovanie“. Časť kódu je nasledovná:

```
def subwordovanie(sp, vstup, vystup):
    try:
        with open(vstup, encoding="utf-8") as vstup, open(vystup, "w+", encoding="utf-8") as vystup:
            for i, rad in enumerate(vstup):
                rad = rad.strip()
                token = sp.encode_as_pieces(rad)
                vystup.write(" ".join(token) + "\n")

    except Exception as e:
        print(f"Nastala chyba v procese {vstup}: {e}")
```

Funkciu sme si vytvorili, pretože tento proces sa vykoná samostatne pre anglické a slovenské vety. Vstupná premenná sp obsahuje SentencePieceProcessory, ktorý načíta model natrénovaný pomocou SentencePieceTrainer, vstup je premenná obsahujúca cestu k súboru, ktorý bol vytvorený po vykonaní krokov predspracovania a výstup definuje cestu, kam chceme uložiť tokenizované vety. Kód je ošetrený v prípade, ak by behom tokenizácie nastala chyba. Funkcia prechádza po vetách a tie tokenizuje pomocou vytvoreného modelu. Tokenizovaná veta sa následne vkladá do nového súboru. Po prejdení slovenských a anglických viet máme k dispozícií dva tokenizované súbory na subwordy. Vzorka viet pre tokenizované vety pomocou Sentencepiece v tabuľke 19.

Tabuľka 19 Tokenizované vety pomocou Sentencepiece

Anglická veta	**Slovenská veta**
▁i ▁would ▁refer ▁in ▁particular ▁to ▁green ▁public ▁procurement s ▁which ▁will ▁allow ▁public ▁sector ▁agencies ▁to ▁play ▁a ▁leading ▁role ▁in ▁saving ▁energy ▁by ▁making ▁use ▁of ▁the ▁new ▁technological ▁applications ▁of ▁ict	▁rád ▁by ▁som ▁spomenul ▁najmä ▁ekologické ▁verejné ▁obstarávanie ▁ktoré ▁agentúra m ▁verejné ho ▁sektora ▁umožní ▁zohrávať ▁vedúcu ▁úlohu ▁pri ▁úspor e ▁energií ▁prostredníctvom ▁používania ▁nových ▁technologický ch ▁aplikáci í ▁i kt

3.3 ROZDELENIE VIET DO TRÉNOVACEJ, TESTOVACEJ A VALIDAČNEJ MNOŽINY

Rozdelíme dva súbory, teda tokenizované texty, ktoré sme si vytvorili na konci v podkapitole 3.2 Tokenizácia textov. Pre zachovanie poradia resp. zarovnania textov, si vytvoríme z anglických a slovenských viet jeden dataset. Kód je rovnaký, ako keď sme si vytvorili jeden dataset v podkapitole 3.1 Predspracovanie paralelného textu. Môžeme vykonať ešte jedno zamiešanie datasetu ako sme vykonali v podkapitole Zmena poradia viet. Tento krok nie je povinný, ale skôr ho môžeme považovať za ubezpečenie, že dataset skutočne bude zamiešaný tak, aby jednotlivé témy textov (slovná zásoby tém) boli rozložené po celom datasete. Následne rozdelíme datataset na tri datasety, ktoré budú obsahovať vety na trénovanie, testovanie a validáciu. Kód je nasledovný:

```
development = 4000
test = 4000

df_dev = df[:development]
df_test = df[development:development + test]
df_train = df[development + test:]
```

Z dôvodu experimentovania s veľkosťou validačnej a testovacej množiny sme si vytvorili samostatné premenné, v ktorých sme menili hodnoty. Dataset df_dev bude obsahovať vety pre validáciu počas trénovania a v tomto prípade bude obsahovať prvých 4000 viet z datasetu df. Dataset df_test bude obsahovať vety, ktoré použijeme na evalváciu modelu po jeho natrénovaní. Bude obsahovať v tomto prípade taktiež 4000 viet z datasetu df. Zvyšné vety použijeme na testovanie a vložíme ich do datasetu

df_train. Tieto datasety následne rozdelíme do 6 samostatných súborov, teda vety na trénovanie, validáciu a testovanie pre anglický a slovenský jazyk. Týmto krokom sme získali súbory, ktoré môžeme použiť na vytvorenie a evalváciu modelu.

3.4 TRÉNOVANIE MODELU STROJOVÉHO PREKLADU

Na trénovanie budeme používať toolkit OpenNMT-py. Pred trénovaním si nastavíme jednotlivé parametre a určíme potrebné súbory potrebné pre natrénovanie modelu. Tieto parametre budú uložené na konfiguračnom súbore vo formáte YAML. Parametre, ktoré si definujeme nemusia byť v danom poradí. Najprv si definujeme, aké súbory budeme potrebovať na trénovanie modelu. Pre trénovanie budeme potrebovať množinu viet určenú na trénovanie, ktoré sme si vytvorili v podkapitole 3.3 Rozdelenie viet do trénovacej, testovacej a validačnej množiny, pre zdrojový (anglický) a cieľový (slovenský) jazyk. Pre validáciu si taktiež určíme súbory pre zdrojový a cieľový jazyk. Na tieto súbory sme skúsili aplikovať transformáciu filtertoolong, ktorá dodatočne odstráni príliš dlhé vety. Tento krok nie je potrebný, keďže sme odfiltrovali vety dlhšie ako 175 znakov a podľa informácií z trénovania filtertoolong filtruje vety dlhšie ako 192. Pre vytvorenie slovnej zásoby si vytvoríme zložku, kde sa uloží slovná zásoba v dvoch súboroch pre anglický a slovenský jazyk. Tieto súbory budú vytvorené na základe tokenizovaného textu vytvoreného počas procesu Sentencepiece. OpenNMT sám o sebe nevie vytvoriť slovnú zásobu, ale využíva tokenizovaný text vytvorený už spomínaním Sentencepiece. Ďalší parameter, ktorý je nutné určiť je veľkosť slovnej zásoby pre zdrojový aj cieľový jazyk. Hodnota musí byť rovnaká ako slovná zásoba vytvorená pomocou Sentencepiece. Definujeme si umiestnenie, kam sa budú ukladať modely vytvorené trénovaním strojového prekladu. Pre účely porovnania výsledkov si definujeme aj parameter seed. Definujeme parametre pre trénovanie a validáciu:

- train_steps – počet krokov trénovania,
- valid_steps – počet krokov, po ktorých sa vykoná validácia modelu pomocou množiny validačných viet,
- warmup_steps – počet prvých n krokov trénovania, kedy začína model s relatívne malou mierou učenia, ktorá postupne narastá až do momentu, kedy sa dostane na úroveň miery učenia, ktorú sme si nastavili. Vhodné pre stabilitu modelu v prvotných krokoch trénovania,
- early_stopping – počet validácií, ktoré môžu nastať bez zmeny, po ktorých sa predbežne zastaví trénovanie. Pomáha zabrániť preučeniu,

- save_checkpoint_steps – počet krokov po ktorých sa uloží priebežný model. Vhodné je ukladať model pre prípad nečakaného zastavenia trénovania a pre prehľadnosť sme si určili hodnotu tak, aby sa vždy uložil krok po validácií. V prvotných pokusoch sme mali validáciu každých 10 000 krokov a ukladal sa model každých 5000 krokov. To sme neskôr zmenili na validáciu každých 5000 krokov a uloženie každých 5000 krokov
- keep_checkpoint – počet posledných n modelov uložených pomocou save_chechpoint_steps. Je vhodné ukladať viacero modelov z dôvodu experimentovania. Niekedy model uložený v skoršom kroku môže mať niekedy lepšie výsledky ako posledný model. Hlavne ak sa začal odpočítavať early_stopping,
- report_every – slúži len na informačné účely, po každom n kroku vykoná výpis informácií o kroku trénovania. Sú tam hodnoty ako aktuálny počet kroku z celkového počtu, perplexita, presnosť, miera učenia, prejdený čas a iné.

Je potrebné definovať aj štruktúru modelu. Parametre, ktoré sme si definovali sú nasledovné:

- encoder_type: transformer – definícia typu enkódera, teda Transformer
- decoder_type: transformer – definícia type dekódera, teda taktiež Transformer,
- word_vec_size – číselná hodnota vyjadrujúca veľkosť vektorovej reprezentácie slov (ang. Word Embedding) pre zdrojový aj cieľový jazyk. Je možné nastaviť samostatnú veľkosť pre zdrojový a cieľový jazyk, ale v našich pokusoch sme tak nevykonali,
- hidden_size – číselná hodnota vyjadrujúca veľkosť skrytých stavov v enkódery a dekódery. Skryté stavy slúži na zachytávanie kontextu vstupného textu. Pri experimentoch sme túto hodnotu vždy nastavali na rovnakú hodnotu ako word_vec_size,
- layers – číselná hodnota vyjadrujúca počet vrstiev v enkódery a dekódery,
- transformer_ff – číselná hodnota vyjadrujúca veľkosť doprednej vrstvy v transformery pre enkóder aj dekóder.

- heads – počet hláv v Multi-head pozornosti

Nastavíme si optimalizačné parametre nasledovne:

- accum_count – počet vykonaných krokov trénovania, po ktorých sa aktualizujú parametre. Pomáha v prípade, ak nemáme k dispozícií silnú grafickú kartu, ktorá by zvládla vykonávať výpočty s veľkými vzorkami dát,
- optim – optimalizačný algoritmus. V našom prípade budeme pracovať s Adam.
- adam_beta1- beta1 parameter využívaný algoritmom Adam,
- adam_beta2 –beta2 parameter využívaný algoritmom Adam,
- decay_method – vyjadruje útlm miery učenia, teda znižovanie miery učenia počas trénovania. Slúži na optimalizáciu miery učenia, aby sme netrénovali príliš veľkými krokmi model,
- learning_rate – miera učenia počas trénovania modelu,
- max_grad_norm: pomáha pre zabránenie explodujúceho gradientu.

Pre inicializáciu parametrov použijeme:

- param_init – nastavíme si hodnotu parametrov resp. váh,
- param_init_glorot – metóda, ktorá udáva ako sa nastavia váhy na začiatku trénovania. Podľa dokumentácie OpenNMT-py požadované pri využití architektúry transformer.

Pre efektívne trénovanie je vhodné nastaviť veľkosť a spôsob spracovania blokov dát počas trénovania. Parametre sú nasledovné:

- batch_size – číselná hodnota vyjadrujúca veľkosť batch (bloku dát) pre trénovanie,
- valid_batch_size – číselná hodnota vyjadrujúca veľkosť bloku dát pre validáciu,
- batch_type – vyjadruje typ bloku dát. Keďže sme vety tokenizovali, tak nastavenie bude „tokens“,
- normalization – vyjadruje normalizáciu bloku dát, taktiež pre naše účely nastavené na „tokens“,
- max_generator_batches –

Nastavenia pre reguláciu modelu sú nasledovné:

- dropout – číselná hodnota vyjadrujúca mieru, koľko percent neurónov sa nevyužije počas trénovania. Pomáha zabrániť preučeniu,
- label_smoothing – číselná hodnota vyjadrujúca mieru hodnôt, ktoré budú počas klasifikácie označené zo správne klasifikovaných hodnôt na nesprávne klasifikované hodnoty (Zhang, 2021). Pomáha zabrániť preučeniu modelu.

Budeme využívať grafický procesor na trénovanie. Parametre pre využitie sú nasledovné:

- world_size – počet procesov vykonávaných počas trénovania. Budeme vykonávať výpočty len na jednej grafickej karte, čiže hodnotu ponecháme na 1.
- gpu_ranks – definovanie jednotlivých grafických procesorov. Máme k dispozícii len jeden grafický procesor, čiže nastavíme si hodnotu 0.

Popis jednotlivých krokov je z dokumentácie pre OpenNMT-py (OpenNMT, 2017). Definovali sme si všetky parametre a máme k dispozícii konfiguračný súbor pomenovaný config.yaml, ktorý môžeme použiť pre vytvorenie slovnej zásoby a samotné trénovanie.

Ako sme už spomínali na začiatku tejto podkapitoly, OpenNMT potrebuje pre vytvorenie vlastnej slovnej zásoby už existujúci tokenizovaný text. Na tento účel použijeme príkaz onmt_build_vocab. Kód je nasledovný:

```
!onmt_build_vocab -config config.yaml -n_sample -1
```

Aby onmt_build_vocab mohol niečo vykonať, je nutné si definovať parametre. Je možné jednotlivo za sebou popísať všetky parametre, ktoré sme si spomenuli vyššie alebo pomocou –config môžeme vložiť súbor obsahujúci tieto parametre. Definovaním –n_sample -1 definuje, že chceme vytvoriť slovnú zásobu zo všetkých dostupných viet. Po vytvorení slovnej zásoby pre oba jazyky máme k dispozícii všetko pre trénovanie nášho modelu strojového prekladu. Trénovanie spustíme príkazom onmt_train. Kód je nasledovný:

```
!onmt_train -config config.yaml -gpu_ranks 0
```

Pre definovanie jednotlivých parametrov použijeme –config config.yaml, v ktorom sa nachádzajú nami zvolené parametre a pomocou –gpu_ranks 0 definujeme akou grafickou kartou to má byť vykonané. Spustením začne proces trénovania. Na začiatku sa vypíšu informácie ohľadom modelu a potom sa vypisujú informácie o trénovaní modelu po určitom počte krokov, ktoré sme si nastavili v parametri report_every. Trénovanie následne prebieha, kým neprejde model požadovaným počtom krokov trénovania alebo nebude predčasne zastavený pomocou early_stopping v dôsledku stagnácie modelu.

3.5 VYTVORENIE PREKLADOVÉHO SÚBORU A DETOKENIZÁCIA

Model strojového prekladu je natrénovaný. Ďalším krokom je vytvoriť prekladový súbor, ktorý budeme potrebovať na testovanie modelu. Aj tu si pre prehľadnosť vytvoríme konfiguračný súbor, v ktorom sa budú nachádzať jednotlivé parametre. Definujeme si model, resp. cestu k modelu, ktorý sme zvolili ako najvhodnejší z procesu trénovania. Model nie je v čitateľnej podobe, čiže je potrebný súbor, kde sa nachádza testovacia množina pre zdrojový jazyk na jeho dekódovanie. Definujeme si aj miesto, kde sa uloží prekladový súbor. Pre účely porovnania výsledkov si definujeme parameter seed. Ostatné parametre si definujeme bodovo:

- beam_size - číselná hodnota vyjadrujúca počet najpravdepodobnejších možností prekladu namiesto všetkých možných. Tento parameter vytvára kompromis medzi kvalitou prekladu a výpočtovou rýchlosťou,
- min_length – číselná hodnota vyjadrujúca minimálnu dĺžku preloženého textu,
- length_penalty – v prípade avg (priemer) penalizuje vety, ktoré sú príliš dlhé alebo príliš krátke
- batch_size – číselná hodnota vyjadrujúca veľkosť bloku dát, ktorý je spracovaný naraz počas prekladu,
- gpu – číselná hodnota vyjadrujúca, či použijeme na tento proces procesor alebo grafický procesor. Pre použitie grafického procesora si zvolíme 0
- coverage_penalty – v prípade summary penalizuje preklad, aby generoval preložený text, ktorému chýba obsah zo zdrojového jazyka. Tento parameter funguje, len ak používame beam_search, ktorý sme spustili pomocou beam_size,

- beta – číselná hodnota vyjadrujúca ako silno bude penalizovať preklad coverage_penalty

Popis jednotlivých krokov je z dokumentácie pre OpenNMT-py (OpenNMT, 2017). Definovali sme si všetky parametre a máme k dispozícií konfiguračný súbor pomenovaný config_translate.yaml, ktorý použijeme v príkaze onmt_transalte. Kód je nasledovný:

```
!onmt_translate -config config_translated.yaml
```

Po vykonaní príkazu máme k dispozícií prekladový súbor v cieľovom jazyku, v našom prípade v slovenskom jazyku (tabuľka 20).

Tabuľka 20 Príklad viet v prekladovom súbore

▁po ▁prvé ▁v ▁eú ▁sú ▁k ▁dispozícii ▁nedostatočné ▁orgány ▁vo ▁svete ▁je ▁chudoba
▁s ▁pevný m ▁vnútroštátn ym ▁výmenný m ▁kurz om ▁bola ▁občiansk a ▁zodpovednosť ▁ dl ž níkov ▁v oči ▁veriteľ om ▁veľmi ▁vysoká ▁všetky ▁menové ▁riziko ▁v ▁hodnote ▁prehnan ého ▁záväzku ▁ niesli ▁ dl ž níci

Ako môžeme vidieť v tabuľke 20, vety sú stále rozdelené na subwordy. Pred evalváciou je nutné ich detokenizovať. Pre detokenizáciu použijeme Sentencepiece. Načítame si Sentencepiece procesor a doň načítame model pre cieľový jazyk, ktorý sme si vytvorili v podkapitole 3.2 Tokenizácia korpusu. Kód je nasledovný:

```
sp = spm.SentencePieceProcessor()
sp.load(target_model)
```

Následne môžeme detokenizovať prekladový text:

```
with open(translation, encoding='utf-8') as vstup, open(translation_desubword, "w", encoding='utf-8') as vystup:
    for rad in vstup:
        desub = sp.decode_pieces(rad.strip().split(" "))
        vystup.write(desub + "\n")
```

Máme k dispozícií prekladový súbor, ktorý je detokenizovaný a môžeme ho použiť pre evalváciu modelu. Okrem prekladového súboru potrebujeme aj referenčný súbor, aby sme mohli vyhodnotiť kvalitu prekladu. Ako referenčný súbor použijeme testovaciu množinu viet v slovenskom jazyku, ktorý sme si vytvorili v podkapitole 3.3 Rozdelenie viet do trénovacej, testovacej a validačnej množiny. Táto množina viet je taktiež tokenizovaná, preto je nutné vety detokenizovať. Využijeme aj v tomto prípade jazykový model cieľového jazyka.

Po vykonaní detokenizácií máme k dispozícií aj detokenizovaný text pre referenčné vety a môžeme vykonať evalváciu.

3.6 EVALVÁCIA

Pomocou evalvácie môžeme vyhodnotiť kvalitu modelu strojového prekladu. K tomu využijeme automatické metriky BLEU a METEOR. Tieto metriky sme si podrobnejšie opísali v podkapitole 1.5 Architektúry/Modely strojového prekladu. Na výpočet BLEU si importujeme knižnicu sacrebleu a dátové súbory prekladu a referenčných viet vložíme do listu pre prekladové vety a do listu pre referenčné vety. Pomocou týchto dvoch listov následne vypočítame BLEU skóre.

Pre výpočet METEOR skóre je potrebné vety tokenizovať na slová, aby METEOR dokázal porovnávať slová. Importujeme si knižnicu spaCy a vytvoríme funkciu tokenizácia, ktorá tokenizuje texty strojovo preložených textov a referenčných textov (ľudských prekladov). Následne budeme počítať skóre na základe porovnania textov pre jednotlivé vety. Skóre každej vety si budeme ukladať do listu. Po vypočítaní skóre pre všetky vety všetky hodnoty spriemerujeme a máme k dispozícií METEOR skóre pre daný model.

COMET skóre vypočítame pomocou knižnice comet. Stiahneme si model "*Unbabel/wmt22-comet-da*", ktorý je natrénovaný, aby dokázal vypočítať COMET skóre pre náš model strojového prekladu. Na výpočet budeme potrebovať vstupné dáta, teda testovaciu množinu textov zdrojového, cieľového jazyka a prekladový súbor v cieľovom jazyku. Výstup je zoznam skóre pre každú vetu a priemer pre všetky vety.

V tabuľke 21 si ukážeme porovnanie referenčnej vety s preloženou vetou. Ako môžeme vidieť, v tomto prípade medzi vetami sú odlišnosti, ale význam sa do značnej miery zachoval.

Tabuľka 21 Porovnanie referenčnej a preloženej vety

Referenčná veta	tiež by sme mali podporovať biotechnologický výskum ktorý poskytuje výskumným pracovníkom prostriedky na kultiváciu orgánov z existujúcich tkanív od samotných pacientov aj od iných darcov
Preložená veta	na záver by sme mali podporiť rozvojový výskum ktorý poskytne výskumným pracovníkom prostriedky na pestovanie orgánov z existujúcich fondov buď zo samotných pacientov alebo z iných darcov

Tento príklad je konkrétne z pokusu V4 v tabuľke 26.

4 VÝSLEDKY

Prvotné pokusy trénovania modelu strojového prekladu, ktorých trénovanie prebiehalo lokálne na počítači boli neúspešné. Počas prvotných krokov trénovania nastala veľmi rýchlo situácia, kedy model mal presnosť cez 90% a veľmi nízku perplexitu 4,5 . Po trénovaní modelu sme pomocou onmt_translate vytvorili prekladový súbor, ktorý obsahoval len neznáme tokeny. V domnení, že tento stav je v dôsledku zle nastavených parametrov sme ich upravovali alebo pridávali nové. Akékoľvek zmeny nezmenili tento stav. Následne sme skúsili trénovanie preniesť do prostredia Colab a tam trénovanie prebehlo bez problémov. Dôvod, neschopnosti natrénovať model lokálne je dodnes neznámy.

Účelom prvého úspešného pokusu trénovania modelu, bolo otestovať, či model po trénovaní bude môcť vygenerovať plynulý a adekvátny preklad. Mnoho parametrov resp. ich hodnoty boli upravené alebo pridané v dôsledku už skôr spomenutých pokusov. Parametre prvého pokusu môžeme vidieť v tabuľke 22. Názvy jednotlivých parametrov sú pomenované ich označením v rámci konfigurácie OpenNMT. Vysvetlivky k jednotlivým parametrom sa nachádzajú v podkapitole 3.4 Trénovanie modelu strojového prekladu. V nasledujúcich popísaných pokusoch sme väčšinu parametrov nemenili, čiže budeme opisovať len zmenené parametre. Až na jeden pokus sme všetky trénovania vykonávali na korpuse Europarl, čiže na túto zmenu upozorníme v príslušnom pokuse.

Tabuľka 22 Parametre trénovania pre prvý úspešný model strojového prekladu

Parameter	**Zvolená hodnota**
src_vocab_size	8000
tgt_vocab_size	8000
train_steps	200 000
valid_steps	10 000
warmup_steps	8000
early_stopping	4
decoder_type	transformer
encoder_type	transformer

word_vec_size	512
hidden_size	512
layers	6
transformer_ff	2048
heads	8
accum_count	2
optim	adam
adam_beta1	0,9
adam_beta2	0,998
decay_method	noam
learning_rate	2
max_grad_norm	0
param_init	0
param_init_glorot	true
position_encoding	true
batch_size	1024
valid_batch_size	1024
batch_type	tokens
normalization	tokens
max_generator_batches	2
dropout	0,1
label_smoothing	0,1

V tabuľke 22 uvádzame len parametre, ktoré ovplyvnia trénovanie. Cesty k potrebným súborom, informačné parametre ako report_every alebo parametre pre ukladanie modelov save_checkpoint_steps a keep_checkpoint nebudeme uvádzať, aj keď boli menené v nasledovných pokusoch. Dôležité parametre, ktoré sa menili mimo konfigurácie trénovania sú veľkosť slovnej zásoby, druh tokenizácie a podiel rozdelenia viet pre validačnú a testovaciu množinu. Veľkosť slovnej zásoby a podiel rozdelenia viet pre testovaciu a validačnú množinu budeme viac krát meniť, ale v prípade druhu tokenizácie sme vo všetkých pokusoch, až na jeden využili Byte pair encoding, čiže na túto zmenu upozorníme v príslušnom pokuse. Pokusy budeme označovať vo forme V + poradové číslo pokusu pre lepšiu prehľadnosť, o ktorý pokus ide. V prípade už vyššie

spomenutého prvého úspešného pokusu budeme sa naň odkazovať pomocou označenia V1. Pokus V1 mal pre rozdelenie testovacej a validačnej množiny nastavenú veľkosť 2000 viet , zvyšné vety sa použijú ako testovacia množina. Slovná zásoba pre zdrojový a cieľový jazyk je 8000, čo sa vždy rovná hodnotám parametrov src_vocab_size a tgt_vocab_size. Po natrénovaní modelu sme vytvorili prekladový súbor na základe parametrov v tabuľke 23.

Tabuľka 23 Parametre pre vytvorenie prekladového súboru

Parameter	**Zvolená hodnota**
beam_size	4
min_length	1
length_penalty	avg
batch_size	32
coverage	summary
beta	0,5
alpha	1

V tabuľke 23 uvádzame len parametre, ktoré ovplyvnia výsledný prekladový súbor. Cesty k potrebným súborom, teda k natrénovanému modelu a testovacej množine v zdrojovom jazyku tu nebudeme uvádzať. V nasledovných pokusoch budeme používať rovnaké parametre pre vytvorenie prekladového súboru ako sú uvedené v tabuľke 23. Zmeny budeme vykonávať až v časti, kde budeme s týmito parametrami experimentovať. V tabuľke 24 môžeme vidieť výsledky evalvácie pre model V1.

Tabuľka 24 Metriky modelu V1

BLEU	**METEOR**	**COMET**
0,43715	0,6744	0,9008

Metriky BLEU, COMET a METEOR poukazujú na relatívne dobrú kvalitu modelu. Uvedieme si príklad porovnania jednej referenčnej vety s preloženou vetou v tabuľke 25.

Tabuľka 25 Porovnanie referenčnej a preloženej vety modelu V1

Referenčná veta	**Strojovo preložená veta**
celkom správne na to upozorňuje správa pre poľnohospodárstvo a rozvoj vidieka pre hodnotenie v polovici obdobia	Správa výboru pre poľnohospodárstvo a rozvoj vidieka o hodnotení v polovici obdobia celkom správne zdôrazňuje

Poradie viet je odlišné, ale kontext sa zachoval. Jediný rozdiel v slovách je slovo „*upozorňuje*“, ktoré preložilo ako „*zdôrazňuje*“ a štylistika, čo pri slovenčine nezohráva až takú veľkú úlohu, keďže slovenčina má voľný slovosled.

Po vykonaní pokusu V1 sme mali k dispozícií parametre, na ktorých sme stavali naše ďalšie modely. Naším ďalším krokom bolo porovnať, aký vplyv majú niektoré parametre na celkovú kvalitu modelu. Prvé pokusy boli vykonávané za účelom zistenia, či pomer rozdelenia trénovacích, testovacích a validačných viet má vplyv na kvalitu modelu.

4.1 VPLYV POMERU ROZDELENIA DÁT NA TRÉNOVACIU, TESTOVACIU A VALIDAČNÚ MNOŽINU NA KVALITU MODELU

Ako sme spomenuli v podkapitole 1.4 Rozdelenie dát na trénovanie a testovanie modelu, pomer rozdelenia viet do množín nie je fixný a záleží na používateľovi, aký pomer si vyberie. V rámci tejto podkapitoly budeme sledovať, či odlišný pomer tohto rozdelenia má vplyv na výsledný model. Natrénovali sme modely V2 až V5 a model V10, ktorého dôvod vytvorenia si uvedieme v nasledujúcej podkapitole. V rámci parametrov trénovania sme nemenili nič, okrem počtu krokov trénovania (parameter train_steps), ktorý sme zredukovali zo 200 000 na 100 000. Výsledky evalvácie jednotlivých modelov môžeme vidieť v tabuľke 26, kde sú jednotlivé pokusy zoradené podľa veľkosti testovacej a validačnej množiny. Hodnota Test/valid reprezentuje počet viet, ktoré boli rozdelené pre obe množiny zvlášť. To znamená, že napríklad z 2000 viet sa vytvorila testovacia množina a z ďalších 2000 validačná množina. Hodnoty zvýraznené červenou farbou reprezentujú najhorší výsledok v rámci daného stĺpca a zelená farba reprezentuje najlepší výsledok.

Tabuľka 26 Porovnanie modelov, ktoré sa líšia veľkosťou testovacej a validačnej množiny

	BLEU	METEOR	COMET	Test/valid	Test/valid v %
V10	0,39361	0,6744	0,8978	2000	≈ 0,33
V5	0,39908	0,7038	0,8993	4000	≈ 0,66
V2	0,39889	0,6765	0,9008	6000	≈ 0,98
V3	0,39959	0,6890	0,9009	30 000	≈ 4,91
V4	0,40082	0,4942	0,8992	60 000	≈ 9,81

Korpus Europarl po predspracovaní mal 611 479 riadkov. Testovaciu a validačnú množinu sme rozdelili v rozmedzí od ≈ 0,33 % po ≈ 9,81 %. Horná hranica, v našom prípade ≈ 9,81 %, by mala približne reprezentovať zaužívané rozdelenie 80/10/10 % pre trénovaciu/testovaciu/validačnú množinu. Ak by sme porovnali jednotlivé modely na základe BLEU skóre, tak výrazný rozdiel môžeme vidieť len pri modely V10 oproti ostatným modelom. Medzi modelom V10 a druhým najhoršie hodnotením modelom V2 je rozdiel 0,00528 BLEU skóre. Medzi modelom V2 a najlepšie hodnoteným modelom V4 je rozdiel 0,00193 BLEU. Rozdiel o necelých 0,002 BLEU je zanedbateľný rozdiel, ktorý vznikol pravdepodobne v dôsledku odlišnej vzorky viet, z ktorej sa počítalo BLEU skóre. Je vhodné podotknúť, že v prípade BLEU skóre vidíme mierny nárast s väčším rozdelením množín.

Metrika COMET poukazuje na ešte menšie rozdiely ako v prípade BLEU. Model V10 je znova hodnotením najhorší, ale je omnoho bližšie k ostatným modelom v porovnaní s BLEU skóre medzi modelmi. COMET skóre sa pohybuje od 0 po 1, čiže rozdiel medzi najhoršie hodnotením modelom V10 a najlepšie hodnotením modelom V3 je 0,0031, čo reprezentuje 0,31%. Tento rozdiel je taktiež zanedbateľným a môžeme predpokladať, že vznikol na základe odlišnej vzorky viet.

Metrika METEOR si udržiava konzistentné skóre medzi jednotlivými modelmi, až na prípad modelu V4. Jeho skóre je výrazne odlišné od zvyšku modelov. Ako metrika COMET aj METEOR má skóre od 0 po 1, čiže rozdiel medzi modelom V4 a druhým najhoršie hodnotením modelom V10 je 0,1802, čo reprezentuje 18%. To môžeme považovať za štatisticky významné. Tento výsledok by mohol byť v dôsledku

relatívne malej veľkosti nášho korpusu, keďže zo 611 479 viet vezmeme 120 000, čo môže znamenať, že sa model nedokáže niektoré slová dobre naučiť. Na druhej strane je zvláštne, že by taký stav neovplyvnil BLEU a COMET skóre.

Ak vezmeme do úvahy výsledky všetkých metrík medzi modelmi, tak môžeme vidieť, že malá množina testovacích a validačných viet nie je žiadúca. Model V10 v metrikách BLEU a COMET dosiahol najhoršie hodnotenie a METEOR skóre mal druhý najhorší. Väčšia množina testovacích a validačných viet už nespôsobovala významné zmeny v skóre až na prípad modelu V4, kde METEOR skóre bolo výrazne nižšie oproti ostatným modelom. Na základe týchto informácií sa domnievame, že vhodné rozmedzie pre rozdelenie spomínaných množín je od ≈ 0,66% do ≈ 4,91% čo by sme mohli zaokrúhliť na 5%. Toto rozmedzie je len orientačné a nemusí nutne prezentovať riešenie, ktoré by sa dalo aplikovať na všetky modely strojového prekladu.

4.2 VPLYV VEĽKOSTI SLOVNEJ ZÁSOBY NA KVALITU MODELU

V rámci tejto podkapitoly budeme riešiť, či má veľkosť slovnej zásoby vplyv na výsledný model. Na rozdiel od predošlých modelov sme znížili kroky validácie z 10 000 na 5000 za účelom skoršieho ukončenia trénovania, ak by sa model prestal zlepšovať. Rozdelenie testovacej a validačnej množiny sme zvolili 4000 pre tieto modely na základe zistení z predošlej podkapitoly. Väčšiu hodnotu sme si nezvolili z dôvodu dlhšieho času na výpočet metriky COMET. Okrem krokov validácie budeme samozrejme meniť parametre src_vocab_size a tgt_vocab_size, ktoré reprezentujú veľkosť slovnej zásoby pre zdrojový a cieľový jazyk. Tieto hodnoty sme vždy menili tak, aby obe mali rovnakú hodnotu. Neexperimentovali sme s odlišnými hodnotami medzi týmito dvoma parametrami. Experimentovali sme s veľkosťami slovnej zásoby od 2000 po 50 000.

V tabuľke 27 máme rozpísané modely s ich jednotlivými metrikami, veľkosť slovnej zásoby a počet krokov trénovania, kým sa model trénoval. Modely sú zoradené podľa veľkosti slovnej zásoby. Pre zdrojový a cieľový jazyk sme používali rovnaké hodnoty, preto použijeme na ich vyjadrenie len jeden stĺpec s názvom Slovná zásoba.

Tabuľka 27 Porovnanie modelov, ktoré sa líšia veľkosťou slovnej zásoby

	BLEU	METEOR	COMET	Slovná zásoba	Kroky trénovania
V12	0,39867	0,7843	0,9024	2000	100 000
V9	0,42028	0,6905	0,9058	4000	100 000
V11	0,45228	0,6905	0,9131	8000	100 000
V6	0,46227	0,6932	0,9147	16 000	80 000
V7	0,47777	0,7937	0,9147	32 000	55 000
V8	0,44686	0,7351	0,9194	50 000	35 000

S rastúcou slovnou zásobou sa zmenšoval počet krokov trénovania. U modelov, kde slovná zásoba bola vyššia ako 16 000 sa model prestal počas trénovania zlepšovať a nastalo predčasné ukončenie trénovania pomocou podmienky early_stopping.

BLEU skóre ukazuje, že modely s veľmi malou slovnou zásobou (modely V12 a V9) mali najhoršie hodnotenie v dôsledku nedostatočne veľkej slovnej zásoby. Modely V11, V6 a V7 mali rastúce BLEU skóre, kde najvyššie skóre mal model V7 so slovnou zásobou 32 000. Slovnou zásobou najväčší model V8 sa oproti modelu V7 prepadol o 0,03091 BLEU skóre. To by mohlo byť spôsobené napríklad preučením modelu. V porovnaní s experimentmi, kde sme porovnávali veľkosť testovacej a validačnej množiny tu môžeme vidieť výrazne väčší rozdiel medzi najhoršie hodnotením a najlepšie hodnotením modelom. Medzi modelom V12 a V7 je rozdiel až 0,0791 BLEU.

Metrika COMET naopak má rastúce skóre s veľkosťou slovnej zásoby. Je nutné podotknúť, že na rozdiel od BLEU, tu nevidíme výrazný nárast COMET. Rozdiel medzi Modelom V12 a V8 je len 0,0170 (0,017%).

METEOR naopak pôsobí chaoticky. Model V12, ktorý má najhoršie BLEU a COMET skóre má v prípade METEOR druhé najlepšie hodnotenie. Model V9 a V11 mali najnižšie hodnotenie. Rozdiel medzi nimi bol zanedbateľný a preto sú označené oba modely. Aj v prípade METEOR skóre najväčšia slovná zásoba nezaručuje najlepšie skóre, keďže rozdiel medzi modelom V7 a V8 je 0,0586 (0,0586%), čo je znateľný rozdiel.

Automatické metriky na vyhodnotenie modelov môžu určiť, ktoré modely sú lepšie. Ak by sme skontrolovali, ako vyzerajú preložené vety jednotlivých modelov, tak zistíme, že rozdiely nie sú až tak veľké. V tabuľke 28 môžeme vidieť ako preložilo jednu vetu z testovacej množiny viet.

Tabuľka 28 Porovnanie strojovo preložených viet modelov, u ktorých sa experimentovalo s veľkosťou slovnej zásoby

Zdrojová veta	In the current situation the european union should pay particular attention to its 20 million small and mediumsized enterprises
Referenčná veta	V súčasnej situácii by mala európska únia osobitnú pozornosť venovať svojim 20 miliónom malých a stredných podnikov
V12	V súčasnej situácii by mala európska únia venovať osobitnú pozornosť svojim 20 miliónom malých a stredných podnikov
V9	V súčasnej situácii by európska únia mala venovať osobitnú pozornosť 20 miliónom malých a stredných podnikov
V11	V súčasnej situácii by európska únia mala venovať osobitnú pozornosť 20 miliónom malých a stredných podnikov
V6	V súčasnej situácii by európska únia mala venovať osobitnú pozornosť 20 miliónom malých a stredných podnikov
V7	V súčasnej situácii by európska únia mala venovať osobitnú pozornosť 20 miliónom malých a stredných podnikov
V8	V súčasnej situácii by európska únia mala venovať osobitnú pozornosť 20 miliónom malých a stredných podnikov

Všetky vety si zachovali po preklade kontext a sú porovnateľné s referenčnou vetou s rozdielom zmeneného poradie niektorých slov. Okrem modelu V12 majú všetky modely rovnako preloženú vetu, ale to nemusí reprezentovať celú množinu preložených vie. Určili sme si len jednu vetu zo 4000 viet. Bolo by časovo náročné porovnať manuálne všetky vety.

Model V12 by sme mohli považovať za najhorší, keďže v dvoch metrikách má najhoršie skóre a V7 za najlepší, keďže v dvoch metrikách mal najlepšie skóre a v COMET mal druhé najlepšie skóre.

Hodnoty, ktoré sme použili na definovanie veľkosti slovnej zásoby nie sú použiteľné univerzálne. Ideálna veľkosť slovnej zásoby záleží od veľkosti korpusu

a obsahu daných viet. Ak budeme referovať na jednotlivé evalvačné metriky tak týmto experimentom sme poukázali na to, že malá slovná zásoba môže negatívne ovplyvniť výsledný preklad, ale zároveň príliš veľká slovná zásoba nemusí byť žiadúca. Je potrebné experimentovať pre nájdenie optimálnej hodnoty pre náš konkrétny model.

4.3 VPLYV ALGORITMU TOKENIZÁCIE NA KVALITU MODELU

V predošlých modeloch sme používali algoritmus byte pair encoding na tokenizáciu viet v tokenizátore Sentencepiece. V tomto pokuse sme tokenizovali pomocou algoritmu unigram, aby sme zistili, ktorá z týchto dvoch možností ponúka lepší model. Budeme porovnávať model, ktorý bol tokenizovaný na základe algoritmu unigram, ktorý sme si pomenovali V15 a model V5, ktorý bol tokenizovaný pomocou byte pair encoding a zároveň má nastavené rovnaké parametre ako V15. Výsledné hodnotenie môžeme vidieť v tabuľke 29.

Tabuľka 29 Porovnanie modelov na základe použitého algoritmu tokenizácie

	BLEU	**METEOR**	**COMET**	**Algorit. token.**
V5	0,39908	0,7038	0,8993	BPE
V15	0,38804	0,6220	0,8819	unigram

Model V5, tokenizovaný pomocou byte pair encoding, dosahoval vyššie skóre vo všetkých troch metrikách ako model V15, ktorý bol tokenizovaný pomocou algoritmu unigram. V tabuľke 30 môžeme vidieť, že model V15 nedokázal správne preložiť vetu.

Tabuľka 30 Porovnanie referenčnej vety so strojovo preloženou vetou modelu V15

Zdrojová veta	Firstly there are insufficient organs available in the eu
Referenčná veta	Po prvé v eú je nedostatok orgánov
V15	Po prvé v eú sú k dispozícii nedostatočné orgány

Na základe tohto pokusu sme usúdili, že pre tokenizáciu je vhodnejšie použiť algoritmus byte pair encoding narozdiel od unigram.

4.4 MODEL TRÉNOVANÝ NA KORPUSE PARACRAWL

V tejto podkapitole budeme evalvovať model V16 trénovaný na korpuse ParaCrawl. Korpus ParaCrawl a jeho predspracovanie sme preberali v podkapitole Predspracovanie korpusu ParaCrawl. V porovnaní s korpusom Europarl má ParaCrawl menšiu vzorku viet, teda 512 176 viet. Zvolili sme rovnaké parametre ako má model V5, s tým rozdielom, že V5 vykonával validáciu počas trénovania každých 10 000 krokov a model V16 vykonával validáciu každých 5000 krokov

Tabuľka 31 Výsledky evalvácie modelu V16

	BLEU	**METEOR**	**COMET**
V16	0,38998	0,3177	0,8282

Je náročné porovnávať tento model s inými modelmi z dôvodu použitia odlišného korpusu a zároveň s odlišnou veľkosťou trénovacej množiny viet, ale ak by sme porovnali len metriky modelu V16 s predošlými modelmi tak zistíme, že BLEU skóre má druhé najhoršie a METEOR s COMET dosiahli najhoršie skóre. ParaCrawl je vytvorený na základe získaných viet z internetu, a tým môže mať omnoho väčšiu slovnú zásobu, ale zároveň vety nemusia byť správne preložené. Ako príklad si môžeme uviesť tri vety z ParaCrawl v tabuľke 32.

Tabuľka 32 Porovnanie referenčných a strojovo preložených viet použitých v modely V16

Referenčná veta	**Strojovo preložená veta**
To bolo po zmene režimu v novembri 1990 pyrotechnicky odpálené do vzduchu	To bolo v novembri 1990 po zmene režimu
Robil som prácu v oblasti medicíny	Konal som v medicíne
Ndoungou hodinová predpoveď počasia	Ndoungou teplota zajtra

Týmto experimentom sme poukázali na dôležitosť kvality korpusu. Model V16 mal skoro rovnaké parametre ako model V5, ale jeho skóre metrík COMET a METEOR skóre sú výrazne nižšie.

4.5 VPLYV VEĽKOSTI KORPUSU NA MODEL STROJOVÉHO PREKLADU

V tejto podkapitole budeme sledovať, ako veľkosť korpusu môže ovplyvniť výsledný model strojového prekladu. Použili sme rovnaké parametre ako model V5 okrem krokov validácie, ktorú sme znížili na každých 5000 krokov. Na začiatku predspracovania sme korpus Europarl zmenšili o polovicu. Testovaciu a validačnú množinu sme taktiež zmenšili na polovicu, aby sme zachovali pomer rozdelenia. Trénovanie sa predčasne ukončilo v 85 000 kroku.

Tabuľka 33 Porovnanie modelov na základe veľkosti

	BLEU	**METEOR**	**COMET**	**Kroky trénovania**	**Počet viet**
V5	0,39908	0,7038	0,8993	100 000	611 479
V17	0,33371	0,1195	0,8819	85 000	305 456

Ako môžeme vidieť v tabuľke 33, zníženie objemu viet na polovicu negatívne ovplyvnilo všetky tri metriky. COMET skóre nebolo výrazne ovplyvnené, ale BLEU a METEOR sa výrazne znížili. Pre BLEU to bolo o 0,06537 BLEU skóre a pre METEOR až o 0,5843. Aj kvalita prekladu je veľmi nízka. Príklad môžeme vidieť v tabuľke 34.

Tabuľka 34 Porovnanie referenčnej a strojovo preloženej vety modelu V17

Zdrojová veta	I have caused some enquiries to be made and my first point would be that we are dealing with question time to the commission not to a particular commissioner
Referenčná veta	Dala som zistiť niekoľko informácií a moja prvá pripomienka je, že v otázky vo vyhradenom čase sú pre Komisiu a nie pre konkrétneho komisára
Strojovo preložená veta	Spôsobil mi niekoľko otázok, ktoré je

	potrebné urobiť, a mojím prvým bodom je, že sa zaoberáme hodinou otázok Komisii, nie konkrétnym komisárom.

Týmto experimentom sme ukázali, že veľkosť množiny viet pre trénovanie môže výrazne ovplyvniť výsledný model strojového prekladu. Využívame celý korpus Europarl, čiže nie je možné vyskúšať, ako výrazne by bol daný korpus ovplyvnený. Je možné, že rozdiely by boli menšie, keďže naše predošlé modely ukazujú, že dokážu celkom dobre preložiť vety.

4.6 VPLYV NASTAVENÍ EVALVÁCIE NA VÝSLEDNÝ MODEL

Okrem parametrov trénovania môže model ovplyvniť aj parametre, ktoré si nastavujeme pri vytvorení prekladového súboru. Všetky modely sme prekladali na základe nastavení v tabuľke 23. V tejto podkapitole budeme riešiť, aké nastavenia parametrov sú najvhodnejšie. Definície jednotlivých parametrov sú uvedené v podkapitole 3.4 Trénovanie modelu strojového prekladu. Prvý parameter, s ktorým sme experimentovali bol beta. Ako preklad ovplyvnil parameter beta môžeme vidieť v tabuľke 35. Číselná hodnota vedľa V17 reprezentuje poradové číslo.

Tabuľka 35 Porovnanie nastavení parametru beta pre model V17

	BLEU	METEOR	COMET	beta
V17 - 1	0,33371	0,1195	0,8819	0,5
V17 - 2	0,33401	0,1195	0,8824	0,4
V17 - 3	0,33404	0,1195	0,8825	0,3
V17 - 4	0,33490	0,1195	0,8831	0,2
V17 - 5	0,33660	0,1195	0,8837	0,1
V17 - 6	0,34263	0,1195	0,8859	0
V17 - 7	0,33374	0,1195	0,8818	0,6
V17 - 8	0,33353	0,1195	0,8818	0,7
V17 - 9	0,33336	0,1195	0,8818	0,8
V17 - 10	0,33322	0,1195	0,8817	0,9
V17 - 11	0,33312	0,1195	0,8817	1

V tabuľke 35 môžeme vidieť, že najlepšie skóre pre BLEU a COMET má model, keď beta je nastavená na hodnotu nula, čiže žiadna penalizácia coverage penalty. Skóre klesá s rastúcou hodnotou beta, z toho dôvodu v nasledovných iteráciách budeme používať beta = 0. METEOR skóre zostalo nezmenené.

Ďalším parametrom, ktorý budeme skúmať je beam_size, ktorý vyjadruje počet *n* najpravdepodobnejších možností prekladu.

Tabuľka 36 Porovnanie nastavení parametru beam_size pre model V17

	BLEU	**METEOR**	**COMET**	**beam_size**
V17 - 14	0,33459	0,1195	0,8748	1
V17 - 13	0,34159	0,1195	0,8812	2
V17 - 12	0,34286	0,1195	0,8841	3
V17 - 15	0,34117	0,1195	0,8871	5
V17 - 16	0,34227	0,1195	0,8875	6
V17 - 17	0,34203	0,2528	0,8880	7
V17 - 18	0,34191	0,2528	0,8882	8
V17 - 19	0,34172	0,2528	0,8886	9
V17 - 20	0,34134	0,2528	0,8886	10
V17 - 21	0,34193	0,2590	0,8889	15
V17 - 24	0,34197	0,2590	0,8890	16
V17 - 25	0,34215	0,2590	0,8890	17
V17 - 26	0,34188	0,2590	0,8889	18
V17 - 22	0,34170	0,2590	0,8889	20
V17 - 23	0,34122	0,2590	0,8889	25

V tabuľke 36 sme zoradili verzie podľa veľkosti beam_size. V prípade beam_size už nie je úplne zrejmé, ktorá hodnota je najvhodnejšia. Prvotná hodnota parametru beam_size bola pre všetky modely 4. V tabuľke 3 môžeme vidieť, že najvyššie BLEU skóre dosiahla verzia V17 – 12, ktorá mala beam_size nastavený na 3. Ostatné verzie majú s rastúcou hodnotou beam_size podobné BLEU skóre, ktoré náhodne rastie a klesá. METEOR skóre začalo meniť svoju hodnotu pri beam_size = 7 a vyššie, kedy v rozmedzí od 7 do 10 malo hodnotu 0,2528 a od 15 sa zvýšilo na 0,2590.

Skóre COMET taktiež rástlo s veľkosťou beam_size, ale s tým rozdielom, že pri hodnote vyššej ako 17 môžeme vidieť mierny pokles. Na základe týchto zistení sme usúdili, že najlepšie nastavenie ak berieme do úvahy BLEU je verzia V17 – 12 s beam_size 3. Ak by sme verzie hodnotili na základe METEOR a COMET, tak najlepšou verziou by sa stala verzia V17 – 25, keďže mala najvyššie hodnoty v týchto dvoch metrikách a jej BLEU je vyššie ako vo verzií V17 – 24. V nasledujúcich verziách použijeme beam_size = 3 na základe najvyššieho BLEU skóre.

Nasledujúci parameter je batch_size, ktorý vyjadruje veľkosť bloku dát, ktorý sa naraz spracuje počas prekladu. Výsledky jednotlivých veľkostí batch_size môžeme vidieť v tabuľke 37. Hodnoty sú zoradené podľa veľkosti batch_size. Stĺpec s názvom Čas predstavuje čas, za koľko nastalo preloženie textu v sekundách.

Tabuľka 37 Porovnanie nastavení parametru batch_size pre model V17

	BLEU	METEOR	COMET	batch_size	Čas
V17 - 28	0,34286	0,1195	0,8841	8	71,87
V17 - 27	0,34286	0,1195	0,8841	16	40,82
V17 - 12	0,34286	0,1195	0,8841	32	24,09
V17 - 29	0,34286	0,1195	0,8842	64	17,02
V17 - 30	0,34286	0,1195	0,8842	128	14,93
V17 - 31	0,34286	0,1195	0,8843	256	15,13
V17 - 32	0,34286	0,1195	0,8843	512	16,62

Môžeme vidieť, že BLEU a METEOR skóre nie je ovplyvnené hodnotou parametre batch_size, ale u COMET môžeme vidieť malý nárast. Z časového hľadiska vidíme prudký pokles medzi V17 – 28 a V17 – 12 a minimum nastane pri batch_size 128. Pri vyššej hodnote ako 128 nastáva nárast prejdeného času na vytvorenie prekladového súboru. Ak by sme sa rozhodovali na rýchlosť prekladu, tak optimálna hodnota pre batch_size je 128. Ak by sme preferovali najlepšie COMET skóre, aj keď len s minimálnym nárastom, tak batch_size 512 je najvhodnejší.

Tabuľka 38 Porovnanie prvotnej verzie s verziami, ktoré dosahovali najlepšie skóre v metrikách

	BLEU	METEOR	COMET	beam_size	coverage_penalty	beta
V17 - 1	0,33371	0,1195	0,8819	4	summary	0,5
V17 - 25	0,34215	0,2590	0,8890	17	summary	0
V17 - 30	0,34286	0,1195	0,8842	3	summary	0

V tabuľke 38 môžeme vidieť, aký rozdiel môže byť, ak použijeme vhodné parametre na preklad. Ak by sme preferovali najvyššie BLEU skóre, tak by sme zvolili verziu modelu V17 -30. Pre najlepšie METEOR a COMET skóre zvolíme verziu V17 – 25. Tieto parametre vykazujú najlepšie skóre pre náš aktuálny model. Môže existovať viacero faktorov, ktoré môžu ovplyvniť, aké parametre môžu byť užitočné a ktoré naopak nie.

ZÁVER

V tejto práci sme si predstavili strojový preklad a jeho význam, kroky potrebné na dosiahnutie funkčného modelu strojového prekladu a riešili sme nastavenia parametrov trénovania a prekladu.

Na základe pokusov s veľkosťou rozdelenia testovacej a validačnej množiny sme dosiahli výsledok, že najvhodnejšie rozdelenie sa pohybuje v rozmedzí ≈ 0,66% do ≈ 4,91%. Hodnoty sú orientačné, ale predstavujú približné rozdelenie, ktoré nám prinieslo najlepšie výsledky. Je nutné podotknúť, že v prípade väčšieho modelu by toto rozmedzie mohlo byť odlišné. Pracovali sme s relatívne malým korpusom, čiže nebolo možné s rovnakou kvalitou viet experimentovať s väčším množstvom viet.

Pokusy s veľkosťou slovnej zásoby ukázali, že veľkosť slovnej zásoby môže ovplyvniť model. Príliš malá slovná zásoby môže negatívne ovplyvniť výsledný model, ale to platí aj pre príliš veľkú slovnú zásobu. Na základe našich poznatkov sme usúdili, že je vždy nutné zvážiť ideálnu veľkosť slovnej zásoby a pre každé riešenie môže byť hodnota odlišná. Experimentovali sme len v jazykoch angličtina a slovenčina a s jednou veľkosťou korpusu, čiže naše hodnoty platia v prípade len tejto dvojice jazykov a pre približnú veľkosť korpusu. Jazyky majú odlišne bohatú slovnú zásobu, čo by tiež mohlo ovplyvniť vhodnú veľkosť slovnej zásoby modelu strojového prekladu. Väčší korpus by mohol obsahovať bohatšiu slovnú zásobu, čo by tiež mohlo ovplyvniť vhodnú veľkosť slovnej zásoby.

Na základe porovnania modelov s odlišnými algoritmami tokenizácie sme sa dopracovali k výsledku, že byte pair encoding vykazuje lepšie výsledky ako unigram. Veta preložená modelom tokenizovaným na základe algoritmu unigram nedokázala správne preložiť správne vetu, respektíve vytvorila nezmyselnú vetu. Pokus s algoritmami tokenizácie bol vykonaný na malej vzorke, respektíve sme porovnávali len dva modely medzi sebou. Vzorka je malá, ale všetky metriky, hlavne METEOR poukazujú na prepad kvality prekladu.

Vytvorením modelu strojového prekladu na základe odlišného korpusu sme chceli poukázať, ako kvalita viet v korpuse môže ovplyvniť model. ParaCrawl je vytvorený na základe zozbieraných viet z internetu. To sa líši od Europarl, ktorý je prepisom zasadnutí európskeho parlamentu, čiže sme mohli už pred experimentom, očakávať odlišné výsledky. Metriky aj vzorka viet prekladu poukazuje, že kvalita

modelu je nižšia ako modely, ktoré sme trénovali na korpuse Europarl. Samozrejme je nutné podotknúť, že Europarl mal skoro o 100 000 viet viac ako Paracrawl po predspracovaní. Aj počet viet, ktoré chýbali na trénovanie mohli ovplyvniť výsledný model.

Myšlienka, ako výrazne môže ovplyvniť veľkosť korpusu výsledný model nadväzuje na ďalší pokus, kde sme Europarl korpus rozdelili na polovicu. Mohli sme vidieť prepad u všetkých troch metrík, najvýraznejšie u metriky METEOR. Model mal problém preložiť aj vetu z testovacej množiny a tým zmenil úplne celý kontext vety. Tým sme chceli poukázať na dôležitosť počtu viet, na ktorých trénujeme. Je nutné podotknúť, že u výrazne väčšieho korpusu by sme tak výrazný mohol byť menej výrazný, ale to by si vyžadovalo väčší korpus na ktorom by sme to mohli otestovať.

Posledným pokusom bolo zistiť, ktoré nastavenia prekladu by boli pre naše riešenie najvhodnejšie. Dokázali sme úvodné hodnoty parametrov prekladu zmeniť, aby sme dosiahli lepší výsledok, ale je dôležité dodať, že naše výsledky nie sú univerzálnym riešením. Model môže mať viacero parametrov, ktoré by mohli ovplyvniť, aké hodnoty parametrov sú najvhodnejšie.

Tieto výsledky by mohli pomôcť pochopiť fungovanie strojového prekladu a ako je dôležité rozhodnúť, ako nastavíme jednotlivé parametre trénovania a prekladu. Je náročné určiť, ktoré parametre a aké sú ideálne ich nastavenia, aby sme dosiahli najlepší model. Výsledný model môže byť ovplyvnený veľkým množstvom faktorov ako kvalita a veľkosť korpusu, ako predspracujeme korpus, ako tokenizujeme korpus, ako rozdelíme vety na množiny, mnoho parametrov, ktoré môžu ovplyvniť samotné trénovania a dokonca aj ako vytvoriť prekladový súbor. Možností je tak veľa, že by bolo časovo a výpočtovo náročné všetko vyskúšať, čo môžeme považovať zároveň za najväčšiu limitáciu tejto práce. Bolo by časovo a zároveň aj výpočtovo veľmi náročné vyskúšať všetky možné kombinácie nastavení, čiže naše výsledky pokrývajú len malú časť možných riešení pre zlepšenie strojového prekladu. Ďalšou limitáciou je relatívne malé množstvo kvalitných dvojjazyčných textov v jazykoch angličtina a slovenčina. Ako sme už uviedli skôr, Europarl je kvalitný, ale veľmi malý korpus. Vety by sa dali doplniť z iných korpusov, ale ich kvalita je sporná ako v prípade ParaCrawl. S väčším objemom viet, by sme dokázali lepšie porovnávať preklady bez rizika, že náš preklad je ovplyvnený malým množstvom viet, na ktorých by sa mohol model strojového prekladu učiť.

ZOZNAM BIBLIOGRAFICKÝCH ODKAZOV

AWAN, A. 2023. What is Tokenization? [online]. 2023. [cit. 19 januára 2024]. Dostupné na: https://www.datacamp.com/blog/what-is-tokenization

BAHETI, P. 2023. Train Test Validation Split: How To and Best Practices. [online] [cit. 5 november 2023]. Dostupné na internete: https://www.v7labs.com/blog/train-validation-test-set

BANERJEE, S. LAVIE, A. 2005. METEOR: An Automatic Metric for MT Evaluation with Impoved Correlation with Human Judgments. V: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization, s. 65-72. 2005. Dostupné na internete: https://aclanthology.org/W05-0909.pdf

BOJAR, O. BUCK, C. FEDERMANN, C. HADDOW, B. KOEHN, P. LEVELING, J. MONZ, C. PECINA, P. POST, M. SAINT-AMAND, H. SORICUT, R. SPECIA, L. TAMCHYNA, A. 2014. Findings of the 2014 Workshop on Statistical Machine Translation. V: Preceedings of the Ninth Workshop on Statistical Machine Translation s. 12-58. Dostupné na: 10.3115/v1/W14-3302

DASH, N. S, ARULMOZI, S, 2018. Web Text Corpus. V: History, Features, and Typology of Language Corpora. 2018. Dostupné na: https://doi.org/10.1007/978-981-10-7458-5_8

DUPONT, L. 2023. Lesson 1.4 Data Shuffling. [online] [cit. 10. marca 2024]. Dostupné na internete: https://deci.ai/course/data-shuffling/

FAN, A. BHOSALE, S. SCHWENK, H. MA, Z. EL-KISHKY A. GOYAL, S. BAINES, M. CELEBI, O. WENZEK, G. CHAUDHARY, V. GOYAL, N. BIRCH, T. LIPTCHINSKY, V. EDUNOV, S. GRAVE, E. AULI, M. JOULIN, A. 2020. Beyond English-Centric Multilingual Machine Translation. 2020. Dostupné na: https://doi.org/10.48550/arXiv.2010.11125

GUO, M. SHEN, Q. YANG, Y. GE, H. CER, D. ABREGO, G. STEVENS, K. CONSTANT, N. SUNG, Y. STROPE, B. KURZWEIL, R. 2018. Effective Parallel Corpus Mining using Bilingual Sentence Embeddings. Dostupné na: https://doi.org/10.48550/arXiv.1807.11906

KALRA, G. 2022. Attention Networks: A simple way to understand Self Attention. [online]. [cit. 2. decembra 2023]. Dostupné na:

https://medium.com/@geetkal67/attention-networks-a-simple-way-to-understand-self-attention-f5fb363c736d

KEARY, T. 2023. Machine Translation (MT) [online]. 2023. [cit. 25. marca 2024]. Dostupné na internete: https://www.techopedia.com/definition/machine-translation-mt

KHANNA, C. 2021. Word, Subword, and Character-Based Tokenization: Know the Difference. [online] [cit. 22. januára 2024]. Dostupné na internete: https://towardsdatascience.com/word-subword-and-character-based-tokenization-know-the-difference-ea0976b64e17

KIERSZBAUM, S. 2020. Masking in Trasformers´ self – attention mechanism. [online] [cit. 23. januára 2024]. Dostupné na internete: https://medium.com/analytics-vidhya/masking-in-transformers-self-attention-mechanism-bad3c9ec235c

KLEIN, G. KIM, Y. DENG, Y. NGUYEN, V. SENELLART, J. RUSH, A. 2018. OpenNMT: Neural Machine Translation Toolkit. 2018. Dostupné na: doi.org/10.48550/arXiv.1701.02810

LOPEZ, F. 2017. langdetect. [online] [cit. 29. februára 2024]. Dostupné na internete: https://github.com/fedelopez77/langdetect?tab=readme-ov-file

LUONG, M. PHAM, H. MANNING, C. 2015. Effective Approaches to Attention-based Neural Machine Translation. 2015. Dostupné na: https://doi.org/10.48550/arXiv.1508.04025

MICROSOFT, 2021. Using Unicode Normalization to Represent Strings. [online] [cit. 17. februára 2024]. Dostupné na internete: https://learn.microsoft.com/en-us/windows/win32/intl/using-unicode-normalization-to-represent-strings

MOHAN, A. 2022. Understanding Byte-Pair Encoding (BPE) Word Piece And Unigram. [online]. [cit. 11. marca 2024]. Dostupné na internete: https://medium.datadriveninvestor.com/understanding-byte-pair-encoding-bpe-word-piece-and-unigram-d97d788de34c

MUNAWWAR, E. 2020. A comprehensive guide to subword tokenizers. [online]. [cit. 21. januára 2024]. Dostupné na internete: https://towardsdatascience.com/a-comprehensive-guide-to-subword-tokenisers-4bbd3bad9a7c

OpenNMT. 2017. Train. [online]. [cit. 16. marca 2024]. Dostupné na internete: https://opennmt.net/OpenNMT-py/options/train.html

OpenNMT. 2017. Translate. [online]. [cit. 16. marca 2024]. Dostupné na internete: https://opennmt.net/OpenNMT-py/options/translate.html

PAPINENI, K. ROUKOS, S. WARD, T. ZHU W. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. V: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), s. 311-318. 2015. Dostupné na: https://doi.org/10.3115/1073083.1073135

PARACRAWL. 2022. ParaCrawl Corpus release v9. [online] [cit. 11. marca 2024]. Dostupné na internete: https://paracrawl.eu/

PARK, S. 2023. Sentencepiece: A simple and language-independent subword tokenizer and detokenizer for neural text preocessing. [online]. [cit. 11. marca 2024]. Dostupné na internete: https://medium.com/codex/sentencepiece-a-simple-and-language-independent-subword-tokenizer-and-detokenizer-for-neural-text-ffda431e704e

PYPI. 2021. Langdetect 1.0.9. [online] [cit. 29. februára 2024]. Dostupné na internete: https://pypi.org/project/langdetect/

PRIYANKA, M. 2022. Perplexity of Language Models. [online]. [cit. 20 novembra 2023]. Dostupné na:
https://medium.com/@priyankads/perplexity-of-language-models-41160427ed72

REI, R. SOUZA, J. ALVES, D. ZERVA, C. FARINHA, A. GLUSHKOVA, T. LAVIE, A. COHEUR, L. MARTINS, A. 2022. COMET-22: Unbabel-IST 2022 Submission for the Metrics Shared Task. V: Proceedings of the Seventh Conference on Machine Translation (WMT). s. 578 – 585. Dostupné na: https://aclanthology.org/2022.wmt-1.52.pdf

REI, R. STEWARD, C. FARINHA, A. LAVIE, A. 2020. COMET: A Neural Framework for MT Evaluation. V: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). s. 2685-2702. Dostupné na: 10.18653/v1/2020.emnlp-main.213

RIKTERS, M. 2018. Impact of Corpora Quality on Neural Machine Translation V: Frontiers in Artificial Intelligence and Applications. 2018. s. 126-133. DOI: 10.3233/978-1-61499-912-6-126

SCHWENK, H. WENZEK, G. EDUNOV, S. GRAVE, E. JOULIN, A. 2020. CCMatrix: Mining Billions of High-Quality Parallel Sentences on the WEB. 2020. Dostupné na: https://doi.org/10.48550/arXiv.1911.04944

SUTSKEVER, I. LE, Q. VINYALS, O. ZAREMBA, W. 2015. Addressing the Rare Word Problem in Neural Machine Translation. V: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, s. 11-19. 2015. Dostupné na: doi:10.3115/v1/P15-1002

TIEDEMANN, J. 2012. Parallel Data, Tools and Interfaces in OPUS. V: Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'12), s. 2214-2218. [online] 2012. [cit. 17. februára 2024]. Dostupné na internete: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf

UNICODE CONSORTIUM. 2023. Unicode Normalization Forms. [online] [cit. 20. februára 2024]. Dostupné na internete: https://unicode.org/reports/tr15/#Description_Norm

UNICODE CONSORTIUM. 2021. The Unicode Standard, Version 15.1 [online] [cit. 29. februára 2024]. Dostupné na internete: http://www.unicode.org/charts/PDF/U0080.pdf

VASWANI, A. SHAZEER, N. PARMAR, N. USZKOREIT, J. JONES, L. GOMEZ, A. KAISER, L. POLOSUKHIN, I. 2017. Attention Is All You Need. Dostupné na: https://doi.org/10.48550/arXiv.1706.03762

WANG, H. WU, H. HE, Z. HUANG, L. CHURCH, K. 2021. Progress in Machine Translation. V: Engineering 18. 2022. s. 143-153. ISSN: 2096-0026. Dostupné na: https://doi.org/10.1016/j.eng.2021.03.023

WEAVER, W. 1949. The Mathematics of Communication. V: Scientific American [online]. 1949, roč. 181, č. 1. [cit. 16. októbra 2023]. Dostupné na: https://monoskop.org/images/4/48/Weaver_Warren_1949_The_Mathematics_of_Communication.pdf

YADAV, D. SHARMA, A. SANCHEZ-CUADRADO, S. MORATO, J. 2012. An Approach to Design Incremental Parallel WebCrawler. V: Journal of Theoretical and Applied Information Technology. 2012, roč. 43, č. 1. ISSN: 1992-8645

YU, T. DABIRMOGHADDAM, A. MACHEREY, K. WANG, P. TIAN, Y. KLINGNER, J. TAKEUCHI, J. SAWAI, Y. KATAWA, H. SHAH, A. JAIN, M. STEVENS, K. FENG, F. TIAN, C. RICHARDSON, J. TIBREWAL, R. FIRAT, O. CHEN, M. BAPNA, A. ARIVAZHAGAN, N. LEPIKHIN, D. WANG, W. MACHEREY, W. TOMANEK, K. GAO, Q. NIU, M. HUGHES, M. 2020.

Recent Advances in Google Translate [online] [cit. 20. októbra 2023]. Dostupné na: https://blog.research.google/2020/06/recent-advances-in-google-translate.html

ZHANG, C. JIANG, P. HOU, Q. WEI, Y. HAN, Q. LI, Z. CHENG, M. 2021 Delving Deep into Label Smoothing. V: IEEE Transactions on Image Processing, č. 30, s. 5984-5996. Dostupné na: DOI 10.1109/TIP.2021.3089942

ZOZNAM PRÍLOH

Príloha A – odkaz na Github obsahujúci kód modelu strojového prekladu

PRÍLOHA A

Príloha A

- A1 Odkaz na GIT: https://github.com/ukf-matusklestinec/Strojovy-preklad-DP