Případová studie: Proces tvorby strojového překladače na bázi neuronových sítí

Tomáš Svoboda, Šimon Schwarz

Obsah

Proces tvorby překladače na bázi neuronových sítí 2
Úvod 2
Obecně k využití technologií HLT v oblasti menšinových jazyků 2
Základní pojmy 3
Tvorba neuronového překladače 4
Výchozí data 5
Čištění textových dat 6
Nastavení a parametry 6
Trénink modelu 8
Evaluace modelu 8
Závěr 9
Zdroje 10

Proces tvorby překladače na bázi neuronových sítí

Úvod

V oblasti překladu se strojový překlad stává, v souvislosti s technologickým pokrokem, stále diskutovanějším tématem. Často se dokonce hovoří o možnosti nahrazení lidského překladu překladem strojovým či se odhaduje do jaké míry toto nahrazení proběhne (a jak moc již proběhlo). Hlavním důvodem je značné zlepšení ve výsledcích strojového překladu. To je způsobeno využitím umělé inteligence (AI) – a specificky pak neuronových sítí – při strojovém překladu. Ty jsou schopny efektivně pracovat se značně menším objemem výchozích dat než starší strojové překladače. Navíc mají schopnost učit se z vlastních překladů.

Tato práce je především syntézou anglofonních zdrojů, jejím účelem je poskytnout přehled tématu – tedy zmapovat a popsat proces tvorby překladače na bázi neuronových sítí. Specifické instrukce k programování neuronového překladače (řádky kódu, funkce) jsou mimo rozsah tohoto textu, ten k nim však odkazuje a jsou tedy čtenáři k dispozici v sekci „Zdroje“. Nejprve je však třeba definovat základní pojmy.

Obecně k využití technologií HLT v oblasti menšinových jazyků

Využití HLT (Human-Language Technologies – jazykových technologií zahrnujících interakci člověka a stroje) pro účely revitalizace menšinových jazyků je nejlépe doloženo zatím na příkladu baskičtiny (srov. např. články/publikace autorů: Dr. Kepa Sarasola z roku 2010), v r. 2007 byla tématu obecně věnována konference v Maďarsku („Language Revitalisation through Multimedia Technology“), včetně návrhů pro systémy strojového překladu na podporu menšinových jazyků. Problematikou se již delší dobu též mimo jiné zabývá institut SSLMIT v italském Forli.

Využití HLT v oblasti menšinových jazyků se nabízí v těchto oblastech:

a) Tématické lexikony, které výrazně usnadňují praktickou práci s jazykem, kdy uživatel v glosářích nevyhledává podle počátečních písmen hledaného slova (protože např. dané slovo nezná, nebo si na ně nemůže vzpomenout), ale tyto databáze jsou řazeny tématicky (tzv. tematická lexikografie a dialektologie umožňující tvorbu vícejazyčných tematických a terminologických zdrojů).
b) Počítačem podporované nástroje HLT pro kvalitativní kontrolu textů (např. kontrola gramatiky či pravopisu), nebo též textová predikce (podobná systému T-9), kdy po napsání prvních písmen software nabízí relevantní slovíčka. Ze zahraniční zkušenosti se ukazuje, že takovéto systémy mají nedocenitelný přínos pro standardizaci často výrazně rozkolísaného pravopisu menšinového jazyka (což platí i o psané romštině v ČR).
c) Strojový překlad, jehož přínos je v této oblasti spíše sekundární, avšak nezanedbatelný. Cílem tedy není vytvořit bezchybný systém, který by automaticky překládal, či dokonce tlumočil mezi češtinou a romštinou; to není v současné době možné. Spíše jde o vytvoření podpůrné technologie např. při realizaci vyhledávání na internetu či jako zdroj hrubé informace o určitém textu, jehož povšechná recepce by byla pro mluvčího menšinového jazyka zpočátku obtížná. Dále by strojový překladač založený na paralelních korpusech češtiny a romštiny posloužil jako sofistikovaný slovník (podobně jako někteří uživatelé využívají Překladač Google), kde lze k překladu zadat nejen osamocená slova, ale též kolokace, a překladač zohledňuje kontext použití výrazu.

Základní pojmy

Umělá neuronová síť je počítačový systém tvořen jednotlivými *uzly* (nodes). Jeho model je založen na modelu neuronové sítě lidského mozku. Sítě mohou mít různý počet

vrstev, vždy však mají *vstupní vrstvu*, jednu či několik *skrytých vrstev* a *vrstvu výstupní*. Jednotlivé uzly tvořící vrstvy pak mají vlastní *váhu* (weight) a *práh* (threshold). Ty určují, jaké informace si uzly předávají a kdy (IBM Cloud Education, 2020).

Hloubkové učení (Deep learning) je takový algoritmus, který využívá více než tři základní vrstvy (IBM Cloud Education, 2020).

Překladače na bázi neuronových sítí jsou aplikací hloubkového učení na strojový překlad (Machine Learning Mastery, 2020). Neuronové překladače (na rozdíl od jiných metod strojového překladu) vložená data registrují na úrovni vět a nerozdělují je do jednotlivých frází – berou tedy během překladu v úvahu značně širší kontext jednotlivých jazykových prvků než klasické strojové překladače (Phrase, 2022). Nepracují podle předem naprogramovaných pravidel, ale na základě „organické“ práce s dvojjazyčnými texty, kterých potřebují značně méně než předešlé překladové modely (Phrase, 2022). Překladače jako *DeepL* pak využívají primárně tréninkovou metodu „učení pod dohledem“ (anglicky supervised learning), kde překladač během tréninku sám porovnává vlastní překlady s překlady z cvičebních dat a upravuje nastavení hodnoty *váhy*, pokud překlady neodpovídají (DeepL, 2021).

Výhodou neuronových překladačů je přesnost, rychlá schopnost učení se a nenáročnost na údržbu a úpravu (Phrase, 2022). Přes všechno tyto výhody se v české jazykové oblasti nevyskytuje významný počet zdrojů na téma vývoje překladače na bázi neuronových sítí, účelem této práce tedy je zkompilovat anglofonní zdroje z této domény a podat jejich přehled v českém jazyce.

Tvorba neuronového překladače

Potřebný software a hardware

Pro tvorbu překladače na bázi neuronové sítě je vhodné programovací prostředí *SciPy 3* využívající programovací jazyk *Python*. Dále je potřeba rozhraní pro programování aplikací (API) *Keras* se softwarovou knihovnou *TensorFlow* nebo *Theano*.

Pro proces programování je pak vhodné splňovat následující hardwarové parametry. Podle blogu firmy PugetSystems je minimálním počtem jader CPU (tedy procesoru) 16, za ideální počet však považují 32 až 64. V oblasti grafických karet pak doporučují produkty firmy NVIDIA, která trhu s GPU dominuje. Jako minimální hodnotu video paměti určují 8 gigabytů, ta však může být u složitějších programů nedostatečná. 12-24 gigabytů by však mělo stačit na většinu úkonů. Zároveň platí, že RAM (polovodičová paměť) systému by měla být alespoň dvojnásobkem grafické paměti – tedy systém s grafickou pamětí 12 gigabytů by měl pracovat s RAM alespoň 24 gigabytů (PugetSystems, n. d.). Typ disku SSD (solid state drive) je doporučován, protože nabízí vyšší rychlost zpracování a přenosu dat, pevný disk (HDD – hard disk drive), je však také přijatelný (SabrePC, 2021). Doporučovanou velikostí úložiště pak jsou 1-4 terabyty.

Výchozí data

Tvorba překladače vyžaduje data, na nichž bude umělá inteligence „trénovat“. Objem těchto dat není jasně daný, obecně však platí, že čím větší objem dat, tím větší bude přesnost překladače, pokud bude trénován dostatečně dlouho (Popel & Bojar, 2018, pp. 20). Tvůrci českého překladače CUBBIT například uvádějí, že data v řádech jednotek milionů bilingvních vět byla nedostatečná, pro vytvoření kompetentního nástroje. Využili proto zpětného překladu – vložili do překladače jednojazyčná data a nechali překladač je přeložit a využít jako výchozí data bilingvní (Popel, Tomkova, Tomek et al., 2020, pp. 2). Na tento přístup upozorňuje také Cardey (2015, pp. 288): „Překladače na bázi neuronových sítí […] velmi efektivně

kompenzují nedostatek paralelních dat.“[1] Brownleeho návod užívá data ve formátu .txt. Jedná se vždy o frázi ve výchozím jazyce oddělenou tabulátorem od fráze v jazyce cílovém:

Hi!	Hallo!
Hi!	Grüß Gott!
Run!	Lauf!

(Brownlee, 2020)

Čištění textových dat

Vložená data je následně potřeba „vyčistit“ – tedy odstranit (případně upravit) prvky, které by mohly vytvářet problémy při zpracování dat. Může se jednat o diakritiku, interpunkci, speciální znaky (např. ostré ß), římské a arabské číslice, anglické kontrakce, vlastní jména etc. (Brownlee, 2017a). Zároveň je nutné standardizovat všechna velká písmena na malá písmena. Toho lze dosáhnout manuálně nebo pomocí programovací funkce (Brownlee, 2020). Pokud slova nejsou předem oddělena (např. pokud vkládáme plynulý text), je třeba je nejprve rozdělit na jednotlivé tokeny, toto lze opět provést programovací funkcí v rámci Pythonu (Brownlee, 2017a). Zároveň potřebujeme vstupní i výstupní data „vyplnit“ prázdnými daty tak, aby byly vstupní i výstupní texty stejně dlouhé (Li, 2018).

Nastavení a parametry

Než začneme samotný překladač trénovat, je třeba nastavit jeho parametry. Protože výsledný překladač poběží na bázi *encoder-decoder* – tedy vstupní data zakóduje do vektoru se stanovenou velikostí, a následně je v rámci procesu překladu „rozkóduje“ do výsledných dat (Brownlee, 2017b), je třeba užít způsob zakódování *one-hot* (Brownlee, 2020). *One-hot encoding* je způsob zakódování při němž platí pouze jeden *bit* (jednotka informací) v jednu

[1] Překlad vlastní

danou chvíli – tedy pouze jedna z možností výstupu (překladu) bude považována za správnou (Harris & Harris, 2013, pp. 129). V praxi tedy bude vstup, tedy zadaná data v L1, zakódován jedním modelem a následně dekódován modelem druhým do L2 (Brownlee, 2020).

Následně můžeme nastavit jednotlivé parametry překladače jako velikost vstupních a výstupních dat nebo maximální délku vstupních a výstupních frází (Brownlee, 2020). Britz, Goldieová et al. (2017) pak doporučují následující nastavení, které lze užít jako základ jenž je vhodné podle potřeby upravovat (názvy jsou zde ponechány převážně v původní angličtině vzhledem ke skutečnosti, že úzus v této oblasti informační techniky je převážně anglický bez zavedených ekvivalentů, termíny jsou vysvětleny a ad hoc přeloženy níže):

Embedding: 128 dimenzí
RNN Cells: LSTM
Encoder: Bidirectional
Encoder Depth: 2-layers (1 layer in each direction)
Decoder Depth: 4-layers
Attention: Bahdanau-style

Dimenze embeddingu značí počet dimenzí, do nichž se budou data ukládat. V „prostoru“ těchto dimenzí se pak utváří sémantické vztahy mezi jednotlivými datovými jednotkami (Google Developers). Britz & Goldieová et al. (2017) dodávají, že již 128 dimenzí podávalo kvalitní výsledky, je však možné počet dimenzí navýšit (což zvyšuje technickou náročnost procesu, ale „zjemňuje“ sémantické vztahy mezi daty). LSTM (long short-term memory) je pak typ neuronové sítě, který umožňuje překladači evaluovat data (např. slova) v jejich kontextu (Technopedia, n. d.). Počet vrstev encoderových sítí obecně koreluje s kvalitou výsledků (Brownlee, 2018), ale sítě s vyšším počtem encoderových sítí jsou méně stabilní. Britz & Goldieová et al. (2017) proto doporučují encoder s 2 až 4 vrstvami. Stejná úměra počtu a kvality se vyskytuje u decoderů, zde Britz & Goldieová et al. doporučují čtyřvrstvý decoder. Je však vhodno zmínit, že Brownlee (2018) doporučuje nejprve začít s jednovrstvými encodery a decodery, jejichž počet vrstev lze případně navýšit, pokud vyžadujeme kvalitnější výkon. Výzkum Britze & Goldieové et al. (2017) dále ukazuje, že

nejefektivnějším způsobem odeslání dat encoderu je odeslání obousměrné. Brownlee (2018) dále doporučuje užití mechanismu „přidané pozornosti“ (added attention), který představili Bahdanau et al. (2015). Tento mechanismus umožňuje programu lépe zpracovávat a překládat dlouhé věty tím, že vstupní větu rozdělí na několik vektorů místo jednoho a evaluuje každé slovo zvlášť v jeho kontextu, čímž umožňuje přesnější a sémanticky citlivější překlady (Bahdanau et al., 2015, pp. 1-2).

Během procesu překladu program „prozkoumá“ svá data, aby „odhadl“, která slova budou pravděpodobně vhodná pro překlad – takzvaný *beam search* (Brownlee, 2018). Výzkum naznačuje, že vhodnými hodnotami určujícími rozsah průzkumu je 3-5 (Britz & Goldieová et al., 2017), Brownlee (2018) však doporučuje začít s hodnotou „beam=1“ a postupně ji rozšiřovat pro lepší výkon.

Trénink modelu

Po konfiguraci modelu jej můžeme začít trénovat. Nastavíme tedy velikost jedné várky (*batch*) na 64 příkladů a počet epoch (*epoch*) na 30 (Brownlee, 2020). Velikostí várky je myšlen počet příkladů, na nichž bude program trénovat – tedy u nichž vytvoří pravděpodobná řešení a následně je porovná s předpokládanými výsledky. Jedna epocha pak znamená jeden cyklus, během kterého program „projde“ veškerými daty. Oba tyto procesy mají vliv konfiguraci překladače, tedy na hodnoty vah a prahů (Brownlee, 2022). Na hardware s moderním procesorem by jedna epocha měla zabrat zhruba 30 sekund – program tedy může být zhruba natrénován za 15 minut (Brownlee, 2020).

Evaluace modelu

Potom, co byl model „vycvičen“ je třeba jeho schopnosti evaluovat. Nejprve vygenerujeme překlad sekvence a následně ji porovnáme se vstupní sekvencí. Celý proces několikrát opakujeme na značném množství příkladů (Brownlee, 2020). Manuální evaluace překladu obvykle hodnotí na základě její věrnosti originálu a její plynulosti. Hodnotící pak

mohou také zapojit „error analysis“, tedy přístup, který spočívá specificky v hledání chyb (např. ve slovosledu, chybějících či přidaných slovech atd.). Hlavní nevýhodou manuálního posuzování strojového překladu je jeho neobjektivita, která vyplývá ze skutečnosti, že neexistuje jednotná hodnotící škála kvality překladu (Defined.ai, 2021).

Z tohoto důvodu je vhodné evaluaci automatizovat. Jednou z automatických metod evaluace je BLEU (Bilingual Evaluation Understudy). Tento algoritmus porovnává výsledné fráze s výchozími bilingvními texty, a následně překlad ohodnotí na škále 0-1. Čím blíž je hodnota číslu 1, tím bližší je překlad originálu (Defined.ai, 2021). Brownlee (2020) pak ve svém návodu implementuje do svého programu právě BLEU.

Systém METEOR (Metric for Evaluation of Translation with Explicit Ordering) pak funguje na podobném principu jako BLEU, hodnotí však překlady na větné úrovni. Zároveň je schopen brát v úvahu synonyma – což mu umožňuje lépe hodnotit méně jednoznačné překlady (Defined.ai, 2021).

Po evaluaci je třeba již „jen“ dál upravovat nastavení pro zlepšení výkonu překladače a celý proces opakovat, dokud nedosáhne program výkonu optimálního.

Závěr

Text prezentoval obecnou metodiku tvorby strojového překladače na bázi neuronových sítí pomocí programovacího jazyka Python. Po definici obecných pojmů spjatých s se strojovým překladem užívajícím umělou inteligenci a přehledem doporučeného hardwaru jsme popsali výběr dat a proces jejich „čištění“. Následně poskytneme model ideálního základního nastavení strojového překladače s případnými možnostmi jeho budoucích úprav. V závěru se pak text věnuje tréninku umělé inteligence překladače a evaluaci jeho výkonu. Postup, který text popisuje není nutně jedinou možnou metodou vývoje neuronového překladače, lze zmínit například postup využívající software PyTorch (Lee, 2017). Ačkoli se tyto přístupy mohou lišit v jednotlivých krocích princip fungování

výsledných překladačů je stejný. Vzhledem k rapidnímu rozvoji informační techniky obecně a oblasti umělé inteligence a hloubkového učení specificky je zároveň pravděpodobné, že tato studie neprezentuje definitivní model strojového překladače – jejím cílem je především iluminovat oblast, která může být pro „nezasvěcené“, česky mluvící čtenáře těžko přístupná.

Zdroje

Bahdanau, D., Cho, K., & Bengio, Y. (2014). “Neural machine translation by jointly learning to align and translate.” https://doi.org/10.48550/arXiv.1409.0473

Britz, D., Goldie, A., Luong, M. T., & Le, Q. (2017). “Massive exploration of neural machine translation architectures”. https://doi.org/10.48550/arXiv.1703.03906

Brownlee, J. (18. 10. 2017a). „How to clean text for machine learning with Python“. *Machine learning mastery.* https://machinelearningmastery.com/clean-text-machine-learning-python/

Brownlee, J. (29. 12. 2017b). „A gentle introduction to neural machine translation“. *Machine learning* mastery. https://machinelearningmastery.com/introduction-neural-machine-translation/

Brownlee, J. (3. 1. 2018). “How to configure an encoder-decoder model for neural machine translation”. *Machine learning mastery*. https://machinelearningmastery.com/configure-encoder-decoder-model-neural-machine-translation/

Brownlee, J. (6. 10. 2020). „How to develop a neural machine translation system from scratch“. *Machine learning mastery*. https://machinelearningmastery.com/develop-neural-machine-translation-system-keras/

Brownlee, J. (10. 8. 2022). “Difference between a batch and an epoch in a neural network”. *Machine learning mastery*. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

Cardey, S. (2014). “Translation technology in France”. In S. Chan (Ed.), *Routledge encyclopedia of translation technology*. Routledge.

DeepL. (1. 11. 2021). „How does DeepL work?“. https://www.deepl.com/cs/blog/how-does-deepl-work

Defined.ai (5. 1. 2021). “Machine translation 101 – Part 3”. *Defined.ai blog.* https://www.defined.ai/blog/machine-translation-101-part-3/

Google Developers. (n. d.). “Embeddings”. *Machine learning crash course*.

Harris, D. & Harris, S. (2013). *Digital design and computer architecture* (2nd ed.). Elsevier.
IBM Cloud Education. (17. 4. 2020). „Neural networks“. https://www.ibm.com/cloud/learn/neural-networks#toc-neural-net-u3voPJVU
Lee, Ceshine. (12. 10. 2017). “Building a translation system in minutes”. *Towards data science*. https://towardsdatascience.com/building-a-translation-system-in-minutes-d82a154f603e
Li, S. (23. 6. 2018). “Neural machine translation with Python”. *Towards data science*. https://towardsdatascience.com/neural-machine-translation-with-python-c2f0a34f7dd
Luong, T. & Brevdo E. (12. 7. 2017). “Building your own neural machine translation system in TensorFlow.” *Google Research*. https://ai.googleblog.com/2017/07/building-your-own-neural-machine.html
Phrase. (23. 9. 2022). „Machines that think: the rise of neural machine translation“. https://phrase.com/blog/posts/neural-machine-translation/
Popel, Martin & Bojar, Ondřej. (2018). „Training tips for the transformer model“. *The Prague Bulletin of Mathematical Linguistics*. 110. 10.2478/pralin-2018-0002.
Popel, M., Tomkova, M., Tomek, J. et al. Transforming machine translation: a deep learning system reaches news translation quality comparable to human professionals. *Nat Commun* 11, 4381 (2020). https://doi.org/10.1038/s41467-020-18073-9
PugetSystems. (n. d.). “Hardware recommendations for machine learning & AI”. *PugetSystems*. https://www.pugetsystems.com/solutions/scientific-computing-workstations/machine-learning-ai/hardware-recommendations/
SabrePC. (1. 12. 2021). “Understanding memory requirements for deep learning and machine learning”. *SabrePC*. https://www.sabrepc.com/blog/Deep-Learning-and-AI/machine-learning-memory-requirements
Technopedia. (n. d.). “Long short-term memory”. https://www.techopedia.com/definition/33215/long-short-term-memory-lstm