TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Automatické generování vtipů pomocí neuronové sítě

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **David Brož**
Vedoucí práce: Ing. Karel Paleček, Ph.D.
Ústav informačních technologií a elektroniky

Konzultant práce: Ing. Lukáš Matějů, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Automatické generování vtipů pomocí neuronové sítě

Jméno a příjmení: **David Brož**
Osobní číslo: M16000015
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se s problematikou neuronových sítí a hlubokého učení.
2. Vytvořte rešerši problematiky automatického generování textu pomocí neuronových sítí.
3. Natrénujte a otestujte vhodné modely pro generování vtipů v angličtině.
4. Prozkoumejte možnosti adaptace a využití vybraných modelů pro generování vtipů v češtině.

Rozsah grafických prací:	Dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] GOODFELLOW, I., Bengio, Y., Courville, A. Deep learning. MIT Press, 2016. ISBN: 978-0262035613
[2] BISHOP, C. Pattern Recognition and Machine Learning. 2006. ISBN 13: 978-038731073
[3] KARPATHY, A., Johnson, J., Li, F. Convolutional neural neworks for visual recognition, dostupné online: https://cs231n.stanford.edu

Vedoucí práce:	Ing. Karel Paleček, Ph.D. Ústav informačních technologií a elektroniky
Konzultant práce:	Ing. Lukáš Matějů, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

13. května 2021 David Brož

Poděkování

Velmi děkuji Ing. Karlu Palečkovi, Ph.D za odborné vedení práce, věcné připomínky, dobré rady a především za vstřícnost při konzultacích mé bakalářské práce. Dále děkuji Ing. Lucii Hambálkové za korektury práce, Bc. Romanu Staňkovi za podnětné rady a mým rodičům za pomoc s technickými potížemi a za pomoc s korekturami práce.

Abstrakt

Práce se zabývá využitím neuronových sítí pro generování vtipů v anglickém a českém jazyce. Pro vývoj byl použit programovací jazyk Python, framework PyTorch a knihovny pro matematické výpočty Pandas a NumPy. Výpočetní operace použitých neuronových sítí byly provedeny na GPU v podobě grafické karty NVIDIA GeForce GTX 1660 Ti s využitím CUDA architektury. Modely neuronových sítí byly trénovány na datech v anglickém jazyce sesbíraných platformy Reddit. Výsledkem je model neuronové sítě umožňující generovat text v podobě vtipu v anglickém jazyce. Práce prozkoumává problematiku využití natrénovaných modelů pro generování vtipů v českém jazyce.

Klíčová slova: Rekurentní neuronové sítě GRU, Hluboké učení, Generování vtipů, Neuronové sítě, PyTorch, GPU, CUDA

Abstract

The thesis deals with the proposal of neural networks for generating jokes in English and Czech. For development, the Python programming language was used along with libraries for mathematical computations Pandas and NumPy. The computations were performed using the GPU in form of an NVIDIA GeForce GTX 1660 Ti graphic card with the usage of the CUDA architecture. The models of neural networks were trained on data in the English language collected from platform Reddit. The result is a model of neural network enabling the generation of a text in form of a joke in English. The thesis explores the problematics of using the trained models for generating jokes in the Czech language.

Keywords: Recurrent neural networks, GRU, Deep learning, Joke generation, Neural networks, PyTorch, GPU, CUDA

Obsah

Seznam obrázků

Seznam zkratek

TUL	Technická univerzita v Liberci
FM	Fakulta mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci
RNN	Reccurent neural network - Rekurentní neuronová síť
LSTM	Long Short-Term Momory - Dlouhá krátkodobá paměť
NLG	Natural-language generation - Generování přirozeného jazyka
GRU	Gated recurrent unit - variance architektury LSTM
ASCII	American Standard Code for Information Interchange, standardizovaná tabulka znaků

1 Úvod

Lidé si odjakživa snažili ulehčit od práce nutné k pohodlnému životu. Začalo to kopími a luky, které umožnily lov, následovala agrokultura, která zrušila potřebu hledat potravu v okolí a umožnila ji pěstovat v pohodlné blízkosti, a další vynálezy, které ulehčily od manuální práce nutné k pohodlnému životu. V dnešní době, kdy většina manuální práce je prováděna stroji se objevuje zájem, o vynálezy, které by lidstvu ulehčily i od práce mentální.

S vynálezem počítače zmizela nutnost manuálního počítání matematických problémů, zápisy a kontroly údajů, nebo i nutnost si něco pamatovat. Jedním z dalších kroků, o který je v dnešní době zájem, je psaní textu. Psaní některých jednoduchých textů jsme již dávno dokázali zautomatizovat. Pokladny, které dokáží vytisknout účtenku, na které je napsáno, co si zákazník koupil, nebo jízdenky v autobuse, existují už mnoho let. V nedávné době se objevil vynález automatického žurnalizmu, který pomocí informačních technologií a umělé inteligence dokáže generovat články a společnosti, jako The New York Times, již tuto technologii používají [2]. Dalším krokem se zdá být generování kreativního textu; novely, básně, scénáře nebo vtipy. O to poslední se pokouší tato práce.

Automatické generování textu je problematika, kterou se lidé zabývali již před vynálezem počítače. Jeden z průkopníků generování textu byl George Philipp Harsdörffer, německý básník a jazykovědec, který žil na počátku 16. století. Experimentoval s metodami generování textu pomocí kostek s písmeny, baletu, kde tanečníci drželi cedulky s písmeny, anebo volvelly – otočné disky z tvrdého papíru, které používal pro vytvářeních slov, a které se používaly dřívěji pro astronomii [3]. Tyto metody generování textu nepřinášely příliš použitelné výsledky pro použití v praxi, proto se používaly jako inspirace pro psaní básní.

Zcela nová doba generování textu nastala s příchodem počítačů a procesu generování přirozeného jazyka NLG, který vznikl společně s program ELIZA v průběhu 70. let, a který vyvinul profesor Joseph Weizenbaum v Massachusettském technologickém institutu. ELIZA je jednoduchá umělá inteligence, která dokáže s uživatelem vést konverzaci v anglickém jazyce a imituje roli Rogerovského terapeuta. Využívá krátkých otázek založených na klíčových slovech použitých uživatelem v konverzaci [4]. S NLG se objevil zájem o využití generace textu pro komerční využití a v nedávné době se stal široce dostupný.

Velký pokrok pro generování přirozeného textu bylo použití strojového učení a modelů umělých neuronových sítí. Jedním z cílů generování přirozeného jazyka je vygenerování humoru.

2 Rešerše

Strojové učení a modely neuronových sítí se ukázaly jako užitečné nástroje pro generování přirozeného textu. Tato kapitola je rozdělena do pěti částí.

První část se věnuje obecnému úvodu do neuronových sítí. Je zde popsána funkce neuronu a neuronové sítě

Druhá část se krátce zabývá dopřednými neuronovými sítěmi.

Třetí část se zabývá rekurentními neuronovými sítěmi, které byly převážně použity pro vykonání této práce.

Čtvrtá část je věnovaná obecnému principu generování textu pomocí neuronových sítí.

Pátá část je věnována již existujícím pracím na téma využití neuronových sítí pro generování humoru.

2.1 Úvod do neuronové sítě

V této podkapitole je popsán formální neuron a jsou rozebrány nejčastější modely používané pro generování textu.

2.1.1 Formální neuron

Základním kamenem neuronové sítě je *formální neuron* (dále jen neuron). Jedná se matematické napodobení biologického neuronu, slouží ke zpracování vstupního signálu a skládá se z několika částí, které jsou vyobrazené na obrázku 2.1:

Vstupy Neuron má obecně n vstupů značené x_0 až x_n, které modelují *dendrity* biologického neuronu. Tento vstup může vyjadřovat podněty z vnějšku neuronové sítě (např. rychlost, vzdálenost, booleovskou hodnotu, teplotu atd.), někdy výstup jiných neuronů v síti.

Váhy Ke každému vstupu neuron x_i má jedenu váhu w_i. Váha je obecně reálné číslo, které ohodnocuje daný vstup a slouží k utlumení nebo zesílení signálu ze vstupu.

Bias Bias (neboli vychýlení) je reálné číslo značené b, které určuje práh, na nějž neuron reaguje. Pomáhá síti nezůstávat v lokálních minimech.

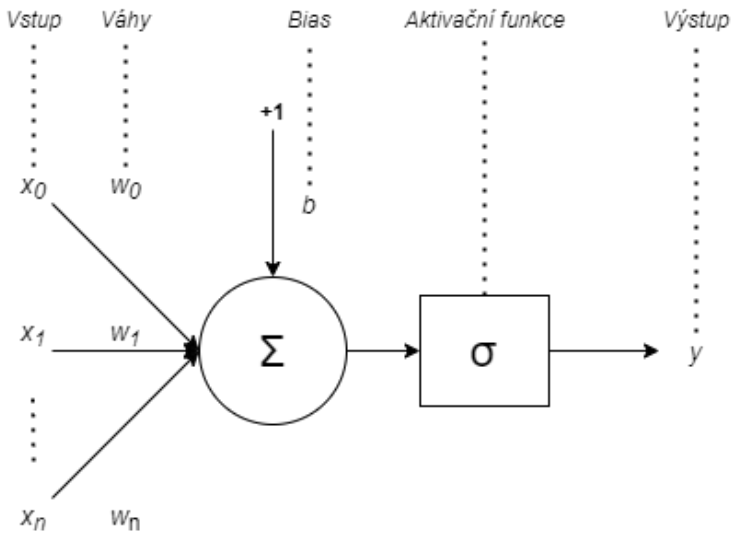
Aktivační funkce Aktivační (přenosová) funkce je nelineární funkce značená jako $\sigma(x)$, která umožňuje zpětnou propagaci zavedením nelinearity do sítě. Mezi

používané aktivační funkce patří například sigmoida, ostrá nelinearita, saturovaná lineární funkce nebo hyperbolický tangens. (Obrázky 2.2, 2.3, 2.4, 2.5)

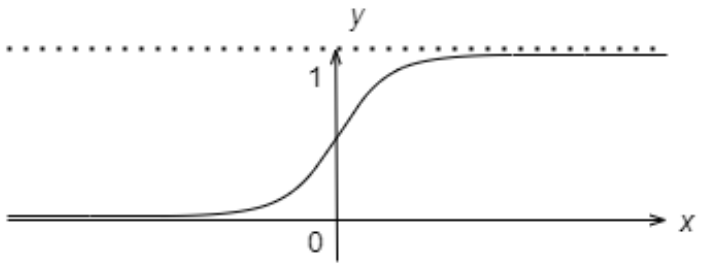
Výstup Výstup neuronu značený jako y je daný vztahem:

$$y = \sigma\left(\sum_{i=0}^{n}(x_i * w_i) - b\right)$$

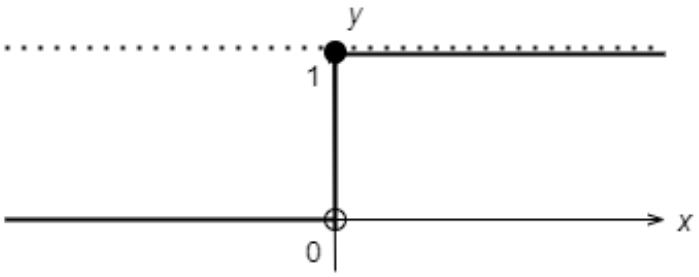
Vytvořeno na základě [5] - *1.3.1 Formální neuron*



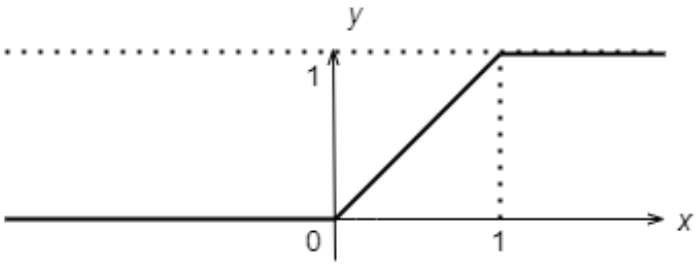
Obrázek 2.1: Model neuronu



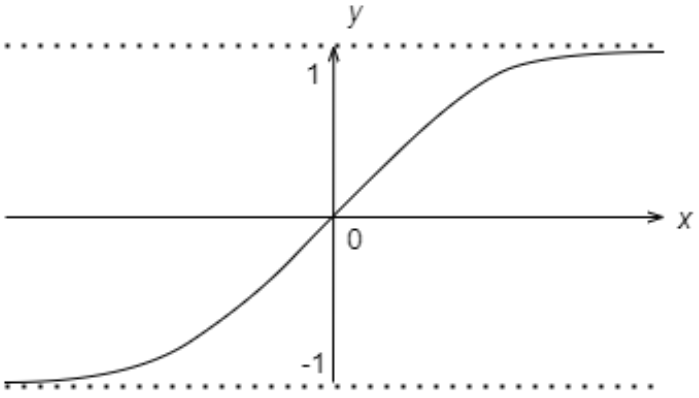
Obrázek 2.2: Standardní (logistická) sigmoida



Obrázek 2.3: Ostrá nelinearita



Obrázek 2.4: Saturovaná lineární funkce



Obrázek 2.5: Hyperbolický tangens

2.1.2 Neuronové vrstvy

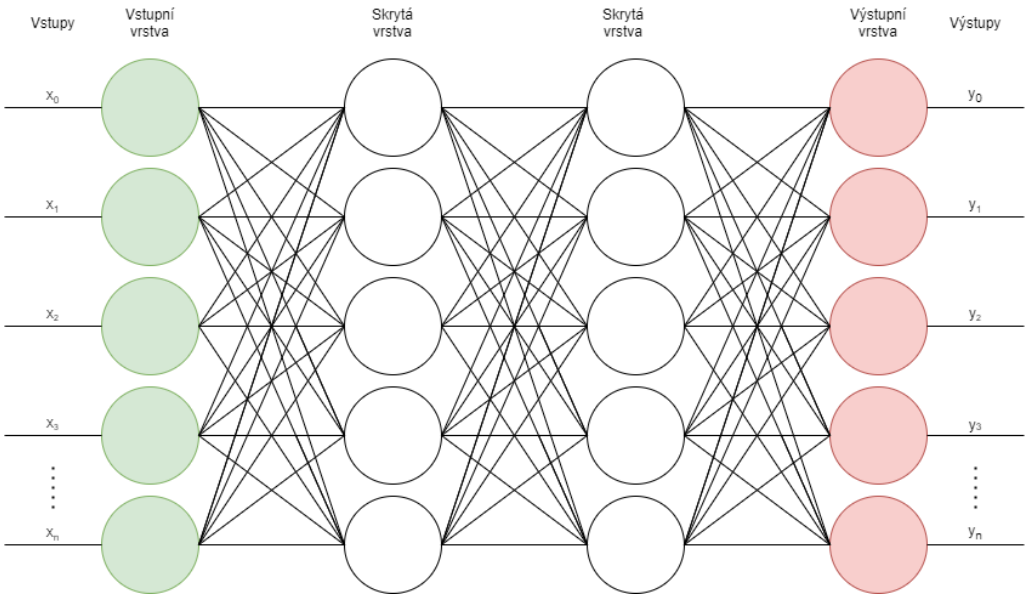
Neurony samy o sobě dokáží vyřešit pouze jednoduché úlohy. Aby neurony mohly řešit složitější úlohy, je potřeba je uspořádat do vrstev a vrstvy mezi sebou propojit. Vrstvy dělíme do tří typů:

1. **Vstupní vrstva** je první vrstva neuronové sítě a slouží ke zpracování vstupních dat z externího zdroje. Neuronům na vstupní vrstvě jsou na vstup předány vstupní signály.

2. **Skrytá vrstva** je jakákoliv vrstva mezi vstupní a výstupní vrstvou. Na vstupy neuronů skryté vrstvy se přivádí výstupy ze vstupní vrstvy nebo vyšší skryté vrstvy.

3. **Výstupní vrstva** je poslední vrstva v neuronové síti. Slouží k finálnímu zpracování dat z předchozích vrstev a přípravě k dalšímu využití.

Jednotlivé typy vrstev jsou vyobrazeny na obrázku 2.6



Obrázek 2.6: Model plně propojená dopředná neuronová síť

Neuronová síť musí mít vstupní a výstupní vrstvu. Kromě tohoto obecného rozdělení typů vrstev, každá může být zároveň jednou z následujících:

Plně propojená vrstva Každý neuron v této vrstvě je propojen s výstupem každého neuronu v předchozí vrstvě. Velké množství neuronových spojení má za následek velkou paměťovou náročnost plně propojené vrstvy. Model plně propojené vrstvy je zobrazena na obrázku 2.6.

Konvoluční vrstva Tato vrstva provádí operaci konvoluce na vstupních datech posouváním konvolučního jádra po vstupních datech a jejich výstupem jsou příznakové mapy, jejichž počet je určen počtem použitých konvolučních jader.

Vytvořeno na základě [5] - *1.3.2 Neuronová síť*

2.2 Dopředné neuronové sítě

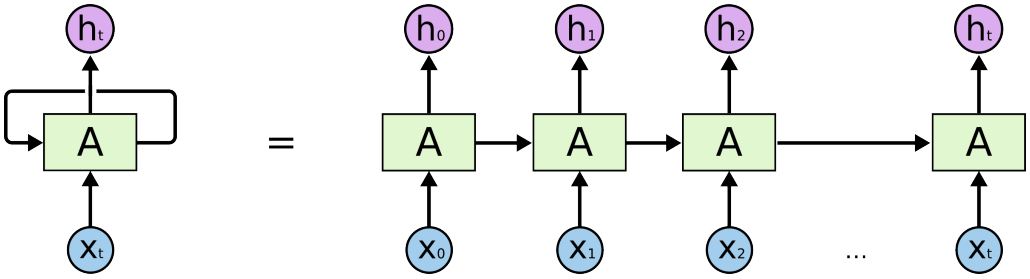
Dopředné neuronové sítě mají pouze jeden směr šíření signálu od vstupní vrstvy k výstupní vrstvě. Jsou vhodné pro použití na širokou řadu úloh a mohou se skládat pouze ze vstupní a výstupní vrstvy (tzv. jednovrstvá dopředná neuronová síť) nebo ze vstupní vrstvy, libovolného počtu skrytých vrstev a výstupní vrstvy (tzv. vícevrstvá dopředná neuronová síť). Vhodný počet vrstev je určen složitostí řešeného problému a velikostí trénovacích dat. Model dopředné neuronové sítě můžeme vidět na obrázku 2.6.

Vytvořeno na základě [5] - *2 Klasické modely neuronových sítí*

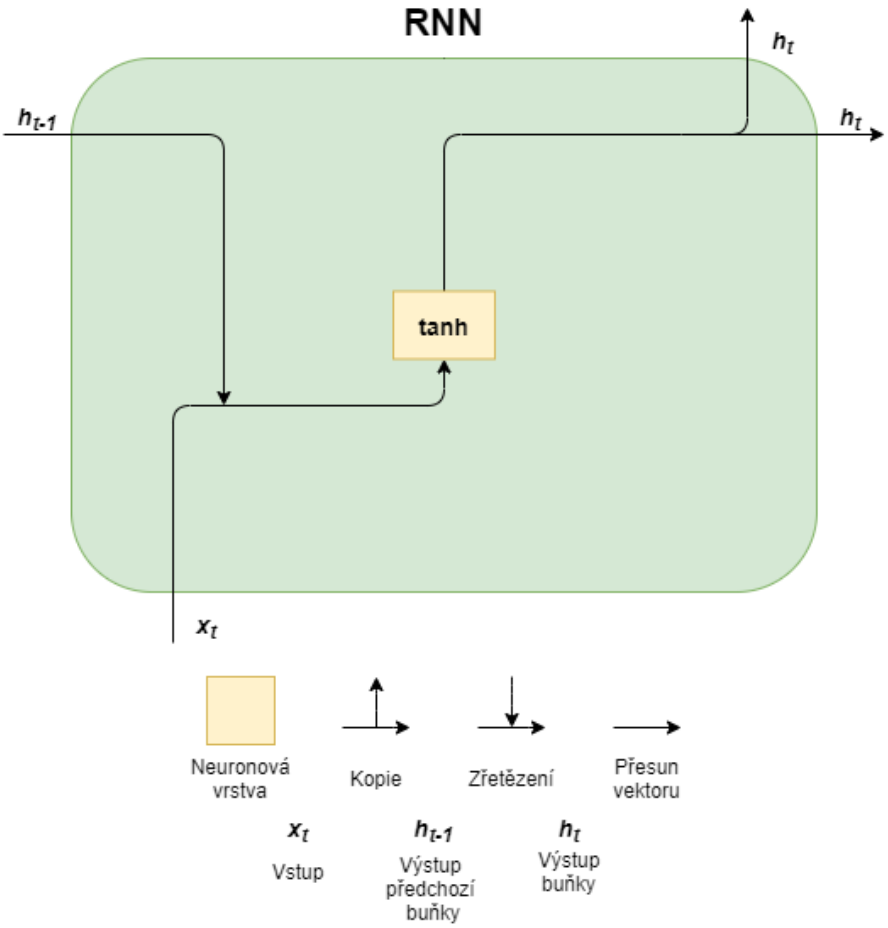
2.3 Rekurentní neuronové sítě

Rekurentní neuronová síť je odvozena od dopředné neuronové sítě. Rozdílem je existence zpětného propojení, kdy výstupy neuronů jedné vrstvy jsou vstupem do neuronů té samé vrstvy nebo jiné předchozí vrstvy. Díky tomu jsou schopny učit se závislosti mezi aktuálními a předchozími vstupy a umožňují tak dynamické zpracování dat. Jsou využívány na časově proměnná a sekvenční data, například rozpoznání řečí nebo generování textu. Tato podkapitola je vytvořena na základě [1].

Pro zjednodušení si lze rekurentní neuronové sítě představit jako zřetězené kopie té samé sítě, které si předávají informace, viz obrázek 2.7. Ve své nejjednodušší formě se jedná o jednu neuronovou vrstvu s aktivační funkcí *tanh*, do které přivedeme výstupní vektor předchozí buňky RNN, takzvaný *hidden state* (skrytý stav) zřetězený s vstupním vektorem, viz 2.8.



Obrázek 2.7: Rozbalená rekurentní neuronová síť [1]



Obrázek 2.8: Jednoduchá rekurentní neuronová síť

2.3.1 LSTM

Long Short-Term Memory, v překladu dlouhá krátkodobá paměť, dále jen LSTM, je speciální druh rekurentních neuronových sítí se schopností učit se dlouhodobé závislosti. Pro tuto schopnost byly navrženy. Tímto se liší od jednoduché RNN, u které je schopnost učit se dlouhodobé závislosti jen teoretická.

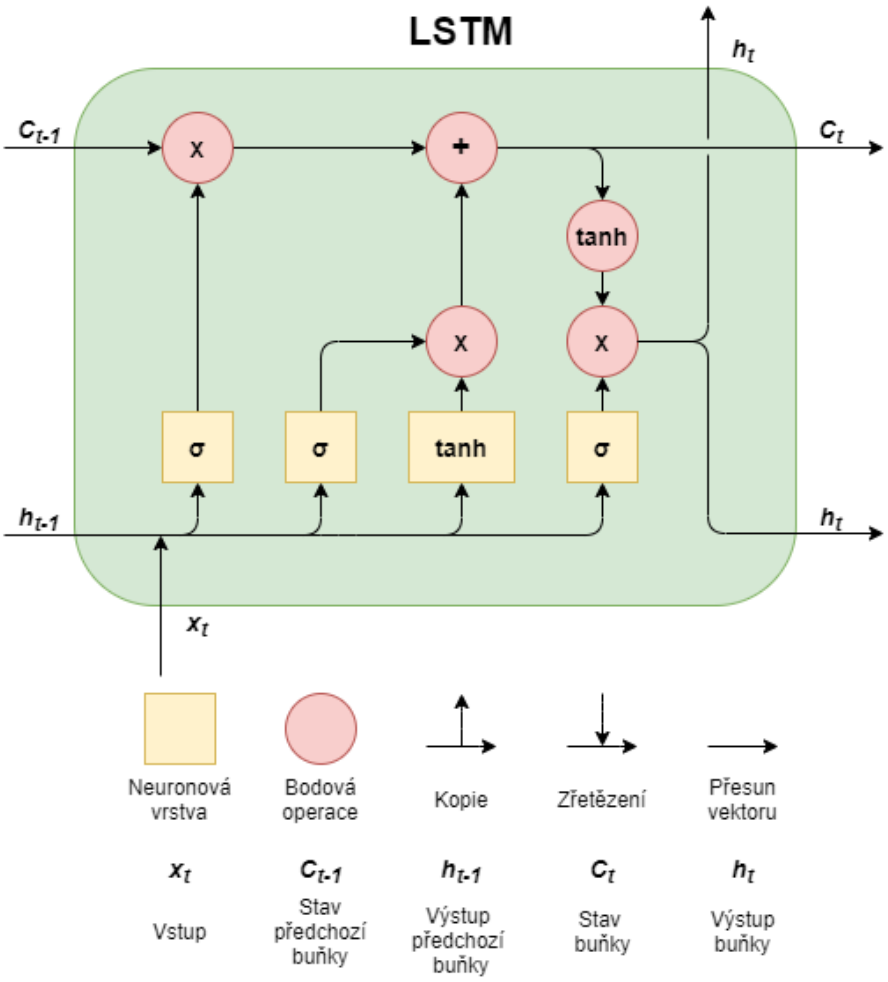
Cell state neboli stav buňky, značený na obrázku 2.9 jako C_t je v podstatě běžící pás, který posouvá informaci mezi buňkami LSTM. C_{t-1} označuje stav předchozí buňky. LSTM obsahuje takzvané brány, které mohou regulovat tok informací skrz stav buňky.

Brány toho dosahují pomocí neuronové vrstvy s aktivační funkcí sigmoida (viz obrázek 2.2) a bodové operace násobení. Tato kombinace prvků umožňuje LSTM se naučit, jakou část a jakou mírou vektor stavu předchozí buňky propustit. Výstup funkce sigmoida je mezi 0 a 1, v kontextu propouštění 0 znamená nepropustit nic a 1 propustit vše.

První brána z levé strany diagramu 2.9, určuje kolik informace stavu předchozí buňky projde dál. Tato brána se nazývá *forget gate* neboli brána zapomnění. Tato brána přijme vstupní vektor x_t a výstupní vektor předchozí buňky h_{t-1}, známý také jako *hidden state* (skrytý stav), a na jejich základě pro každé číslo vektoru stavu předchozí buňky C_{t-1} vyprodukuje číslo mezi 0 - zapomeň vše a 1 - nezapomeň nic.
Příkladem tahového zapomenutí v kontextu NLG je zapomenutí posledního rodu podmětu věty, když dostaneme podmět nový při zpracovávání nebo generování textu.

Druhá brána z levé strany se nazývá *input gate* neboli vstupní brána, která určuje, jaká informace se přidá do stavu předchozí buňky. Skládá se ze dvou neuronových vrstev. První vrstva s aktivační funkcí sigmoida určuje, jaké hodnoty se uloží. Druhá vrstva s aktivační funkcí *tanh* určuje kandidátní vektor, který by se měl přidat do stavu předchozí buňky. Obě tyto části přijímají stejný vstup jako první brána. Vynásobením hodnot výstupních vektorů těchto dvou sítí dostaneme vektor, jehož hodnoty pak můžeme přičíst k hodnotám stavu předchozí buňky profiltrované bránou zapomnění C_{t-1}, a tím dostaneme nový stav buňky C_t.

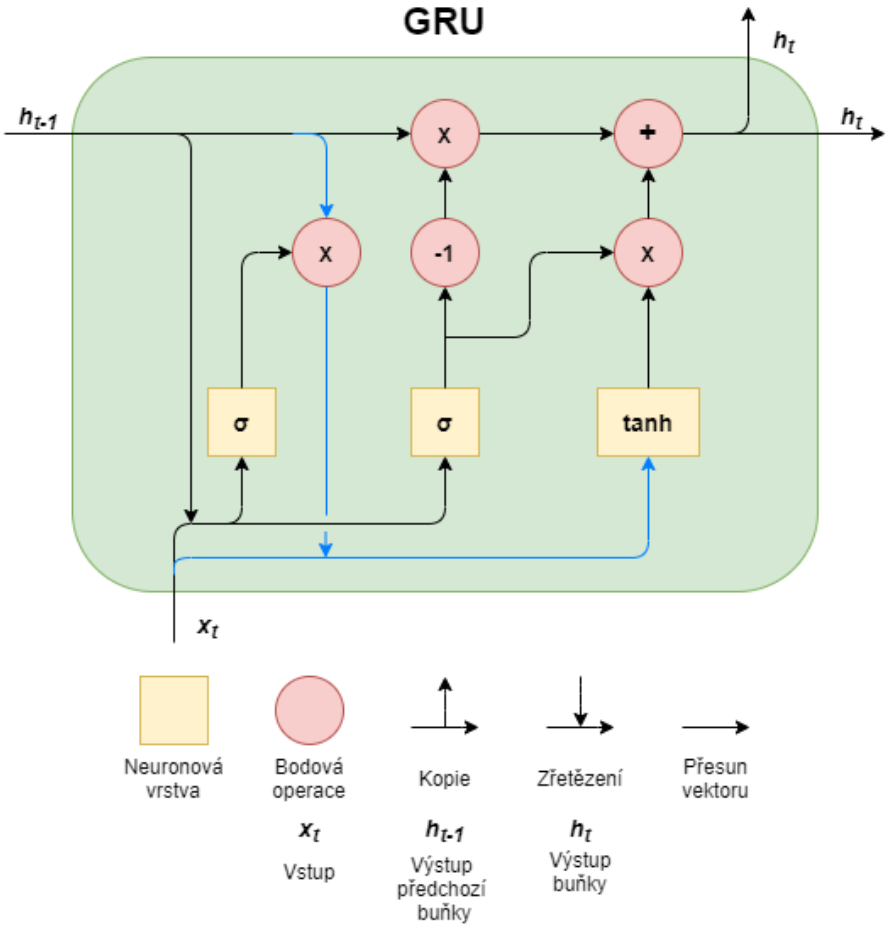
Třetí brána buňky LSTM určuje výstup buňky h_t. Tento výstup je určen ze stavu buňky C_t a výstupu předchozí buňky h_{t-1} zřetězeného se vstupem x_t. Hodnoty vektoru stavu buňky C_t projdou funkcí *tanh* za účelem stlačení hodnot mezi -1 a 1 a poté jsou profiltrovány třetí bránou, která přijímá stejné vstupy jako první a druhá brána. Výstup se předává ven ze řetězu buněk LSTM a zároveň se předá následující buňce LSTM.
Výše popsaná struktura je jen jedna z implementací LSTM. Mnoho lidí používá vlastní variace LSTM, které se od této implementace liší v detailech, ale fungují na stejném principu.



Obrázek 2.9: Model buňky LSTM

2.3.2 GRU

Gated Recurrent Unit, dále jen GRU, je jedna z variací LSTM. Kombinuje dohromady bránu zapomnění (*forget gate*) se vstupní bránou (*input gate*) do jedné aktualizační brány (*update gate*), spojuje stav buňky (*cell state*) dohromady s výstupem buňky (*hidden state* nebo také skrytý stav). Výsledkem je model jednodušší než LSTM.



Obrázek 2.10: Model buňky GRU

2.4 Princip generování textu pomocí neuronových sítí

2.4.1 Abstrakce

Neuronovou síť si lze ve velice zjednodušeném pohledu představit jako kouzelnou černou skříňku, do které se z jedné strany sypou vstupní data a z druhé vypadávají data výstupní. Aby se tato kouzelná černá skříňka mohla naučit provádět požadovanou úlohu, je nutné jí vysvětlit, jak se výstupní data liší od očekávaného výsledku pomocí dobře zvoleného hodnotícího kritéria.

Abychom mohli použít neuronovou síť pro generování textu, potřebujeme následující:

1. Sadu vstupních dat
2. Hodnotící kritérium
3. Sadu očekávaných výstupů k daným vstupním datům

Tato práce má přístup k data setu vtipů. Jelikož očekávám, že výstup této neuronové sítě bude vtip, tento data set je sada očekávaných výstupu k daným datům.

Hodnotící kritérium v tomto případě je rozdíl mezi vygenerovaným textem, který jsem dostal jako výstup neuronové sítě, a vtipem z data setu.

V případě vstupních dat se objevuje problém. Co je vstupem této neuronové sítě? Na základě čeho má síť vytvořit vtip? Toto je otázka, která nemá jednu jednoduchou odpověď. Zde je návrh potenciálních vstupů, které můžeme použít pro tuto neuronovou síť:

První slovo nebo první znak vtipu Můžeme začít vtip a nechat neuronovou síť jej dokončit.

Příklad:

Vstup: *Why*

Výstup: *Why did the chiken cross the road? To get to the other side.*

Vtip Můžeme neuronové síti předat celý text vtipu a nechat neuronovou síť odpovědět na vtip vtipem.

Část vtipu Můžeme neuronové síti dát část vtipu a nechat neuronovou síť najít pointu.

Příklad:

Vstup: *Why did the chicken cross the road?*

Výstup: *To get to the other side.*

Pro tuto možnost se nabízí využití vtipů formy otázka a odpověď, kde otázku se dá neuronové síti jako vstup, a odpověď nám poskytne výstup neuronové sítě.

Jiný text Můžeme použít jiný nevtipný text, ze kterého se neuronová síť pokusí udělat vtip.

Aby tato metoda fungovala, je potřeba mít návaznost mezi vstupním nevtipným textem a výstupním vtipem. Jedna možnost takového vstupu je popis požadovaného vtipu.

Příklad:

Vstup: *A joke about a chicken crossing a road.*

Výstup: *Why did the chiken cross the road? To get to the other side.*

Další možnost vstupu je text s kontextem, ze kterého výstupní vtip vychází. Příklad:

Vstup: <An article about chickens on roads>

Výstup: *Why did the chiken cross the road? To get to the other side.*

Data set pro využití této inicializace vstupu zatím neexistuje. Mohl by být teoreticky vytvořen přidáním popisků vtipů do již existujícího data setu vtipů, jako jsou například data sety zmíněné v sekci 4.1. Tento styl vstupních dat nebyl v této práci použit.

2.4.2 Metody

V této části je popsáno, jak neuronová síť může generovat text.

Výše popsaný neuron a neuronové sítě pracují s čísly. Aby neuronová síť mohla generovat text, musí se tento problém převést na matematický problém. Jsou dvě široce používané metody.

Po znacích Text můžeme převést na sekvenci čísel tak, že každý znak v textu nahradíme číselnou hodnotou. Vznikne tak slovník mezi čísly a znaky, který umožňuje převést problém generování další části věty na problém předpovídání pokračování sekvence čísel.

Ve výsledku se tímto přístupem vstupní text zaklíčuje slovníkem do číselné podoby a výstupní číselná hodnota se rozklíčuje pomocí stejného slovníku do textové podoby.

Po slovech Další možností je místo zakódování jednotlivých znaků nahradit celé slovo. Tímto se zkrátí délka sekvencí, se kterými neuronová síť pracuje, a zvětší se počet možných prvků v sekvenci.

Pokud neuronová síť pracuje s textem na úrovni slov, může se naučit vztahy a podobnosti mezi jednotlivými slovy pomocí vektorových reprezentací slov.

Tato vektorová reprezentace zajišťuje, že vektory podobných slov jsou si blízko. Mezi vztahy, které se ve vektorové reprezentaci slov mezi slovy projevují, patří jednoduché algebraické operace. Například *vektor("král") - vektor("muž") + vektor("žena")* dává výsledný vektor blízký vektoru *vektor("královna")*. Čerpáno z [6][7].

2.5 Existující práce

Tato podkapitola se věnuje některým již vypracovaným pracím a projektům, které se zabývaly tématikou generování humoru pomocí neuronových sítí, popisuje přístup k problému a ohodnotíme výsledek.

2.5.1 Tito joker

Na stránkách společnosti Towards Data Science Lorenzo Ampil [8] představil v prosinci 2019 svůj pokus o generování humoru. Před-trénovaný model GTP-2 od společnosti OpenAI doladil pomocí databáze vtipů získaných z portálu společnosti Kaggle, komunity datových vědců a odborníků na strojové učení. Cílem bylo natrénovat model na vtipy ve formě hádanek neboli vtipy typu otázka-odpověď. Použil Google Colab notebook s jedním Nvidia T4 grafickým procesorem specializovaným pro vývoj umělé inteligence.

Data set Kolekci vtipů *Short Jokes* zveřejnil Abhinav Moudgil v roce 2017. (Dostupné z [9].) Obsahuje přes 200 000 krátkých vtipů v angličtině. Tento data set Lorenzo zmenšil na 65 394 vtipů začínající na "what", "how", "when" nebo "why", aby mohl doladit model na vtipy typu otázka-odpověď.

Model GPT-2 je velký jazykový model založený na modelu typu *transformer* s 1,5 miliardy parametrů natrénovaných na datech z 8 miliónů webový stránek [10]. Pro tento projekt byla použita nejmenší verze se 117 milióny parametrů.

Trénování Před-trénovaný model byl trénován po jednu epochu s *batch size* velikosti 2 a metodou predikce dalšího slova na základě sekvence slov vtipu z data setu.

Metoda generování Model používal jako základ pro generování vstup otázky, na kterou pak odpovídá.

Výsledek Odpovědi na otázky byly z většiny koherentní a gramaticky správné. Text byl spíše nevtipný a absurdní. Urážlivý obsah použitého data setu se občas promítl na vygenerovaném výsledku.

2.5.2 Martins Frolovs

Martins Frolovs v prosinci 2019 také představil svůj pokus na stejném základu a data setu jako v předchozí části 2.5.1, využil však grafický procesor grafické karty Nvidia GeForce 1080 Ti [11].

Data set Kolekci vtipů *Short Jokes* zveřejnil Abhinav Moudgil v roce 2017 a obsahuje přes 200 000 krátkých vtipů v angličtině (dostupné z [9]).

Trénování Před-trénovaný model byl trénován po 4 epochy metodou predikce dalšího slova na základě sekvence slov vtipu z data setu. Byl použit AdamW optimizér, verze optimizéru Adam s opravenou metodou počítání úbytku vah *weight decay* [12] používající *cross-entropy* pro výpočet hodnoty *loss*.

Model GPT-2 verze s 355 milióny parametrů.

Metoda generování Model používal jako základ pro generování vstup jednoho slova *"JOKE:"* a nechal poté model dokončit sekvenci slov. Začínajíc sekvencí o délce jednoho slova *"JOKE:"*, model vybral náhodné slovo z množiny n modelem predikovaných nejpravděpodobnějších dalších slov a přidal jej do sekvence slov. Tuto novou sekvenci opět přijal na vstup a proces opakoval, dokud model nepřidal koncový token *"</endoftext/>"* a poté generování vtipu ukončil.

Výsledek Tato metoda generování vtipů vytvářela koherentní text, které v naprosté většině dával gramatický smysl. Vtipnost textu nebyla nijak zvlášť vysoká. Vtipy byly z velké části humorné díky své absurditě. Databáze Kaggle obsahuje mnoho urážlivých vtipů, což se promítlo na vygenerovaných vtipech. Celkově byl model schopný vygenerovat spolehlivě koherentní, nespolehlivě vtipný text.

2.5.3 Humor Detection: A Transformer Gets the Last Laugh

Práce, kterou vytvořili Orion Weller a Kevin Seppi z Univerzity Brighama Younga v USA, Utah, se nezabývá generováním humorného textu, ale jeho klasifikací [13]. Není tak přímo relevantní pro generování humoru, ale naznačuje nový potenciální nástroj a přístup, který by mohl být použit pro generování humoru. Vytvořený model by potenciálně mohl být použit jako hodnotící kritérium pro trénování jiného modelu za účelem generování humoru metodou *reinforcement learning*, proto je v této práci zmíněna.

Data set Pro trénování modelu tvůrci použily 3 data sety. Data set, který pomocí API platformy Reddit ze subredditu */r/jokes* sami sesbírali obsahující vtipy s hodnocením, 13 884 nehumorných vtipů a 2 025 humorných vtipů, kolekci vtipů *Short Jokes* z Kaggle zmíněnou výše, obsahující více než 200 000 neohodnocených vtipů, a data set sesbíraný ze stránek Pun of the Day obsahující 16 001 slovních hříček a 16 002 nehumorných vět.

Trénování Trénování zařazováním textu do tříd humorné a nehumorné probíhalo po 7 epoch.

Model Byl použit model BERT (Bidirectional Encoder Representations from Transformers), více vrstvý obousměrný Transformer původně natrénovaný na korpusu obsahujícím 3 300 000 000 slov.

Výsledek Výsledkem byl model neuronové sítě schopný předpovídat, zda text bude shledán humorným komunitou na subredditu */r/jokes* lépe, než běžnou populací.

Výsledky této práce naznačují, že neuronová síť je schopna se naučit předpovídat s vysokou přesností, zda jistá komunita lidí shledá text vtipným. Na základě této práce jsem utvořil hypotézu, že by takto natrénovaný model neuronové sítě mohl být využit pro generování textu jako:

- Filtr, který vytřídí výstupy jiné neuronové sítě generující humorný text od textu, který je klasifikován jako nehumorný, a tím zvyšuje průměrnou humornost vygenerovaného textu
- Hodnotící kritérium pro neuronové sítí generující humorný text s použitím metody *reinforcement learning*.

3 Design sítě

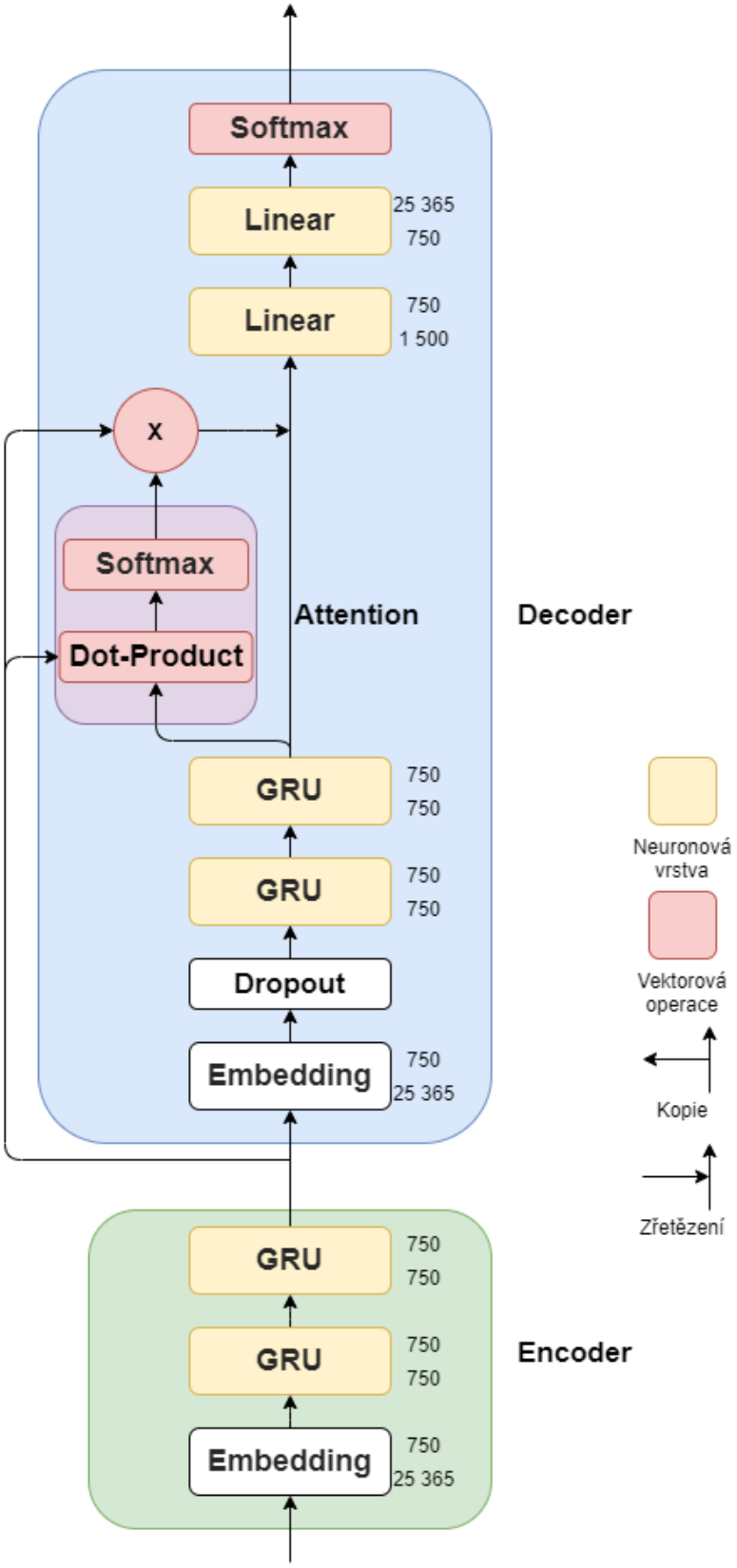
V této kapitole popíši, které postupy jsem použil pro naplnění zadání této práce. Všechny zmíněné neuronové sítě jsem implementoval pomocí jazyka Python a frameworku PyTorch.

3.1 Sequence-to-sequence vtip na vtip

První model byl inspirován chatovacími boty využívající neuronové sítě. Tyto boty dostanou na vstup otázku a na výstup dají odpověď na tuto otázku. Častý přístup je využití sequence-to-sequence neuronových sítí, trénovaných na textových záznamech rozhovorů.

Motivace Myšlenka byla model neuronové sítě naučit odpovědět na vtip vtipem. Do vstupu neuronové sítě přivést vtip z data setu vtipů a jiný náhodně vybraný vtip z data setu mít jako cílový výstup. Předpokládal jsem, že neuronová síť se naučí, že nejefektivnější metoda je vygenerovat nějaký vtip, který je minimálně závislý na vstupním vtipu. Vstupní vtip by tak fungoval jako náhodný vstup, ze kterého se model může inspirovat.

Implementace Neuronovou síť jsem vytvořil pomocí jazyka Python a frameworku PyTorch. Využil jsem architektury *encoder-decoder* s *attention*. *Encoder* i *decoder* jsou zprostředkovány pomocí dvou GRU vrstev s 750 parametry u skrytých vrstev. V modelu je *attention* s implementací metodou skalárního součinu. Použitý slovník obsahoval 25 365 slov. Model je znázorněn na obrázku 3.1



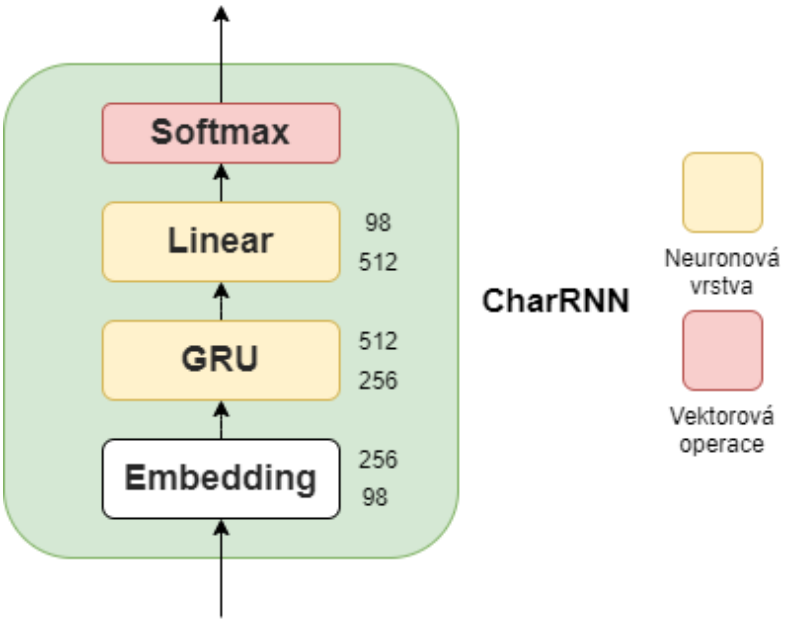
Obrázek 3.1: Použitý model sequence-to-sequence

3.2 Znaková rekurentní neuronová síť

Pro druhý design modelu jsem zvolil nejčastěji používaný přístup, a to generování textu po znacích.

Motivace Jako druhý byl zvolen bezpečný a ozkoušený přístup. Metoda generování textu po znacích je obecně uznávaná jako efektivní metoda generování textu, proslavil ji Andrej Karphaty, jehož model byl inspirací pro zvolenou architekturu. [14]

Implementace Použil jsem jednu GRU vrstvu, jednu lineární vrstvu a použitý slovník (znaky data setu Kaggle) obsahoval 98 znaků. Model je znázorněn na obrázku 3.2



Obrázek 3.2: Použitý model znakové RNN

3.3 Sequence-to-sequence odpověď a otázka

Jako třetí přístup byla znovu použita architektura sequence-to-sequence s jiným přístupem inicializace vstupu a s využitím před-trénované embedding vrstvy.

Motivace Předchozí dva přístupy nevyužívaly před-trénované modely. Embedding vrstvy byly trénovány od začátku a na malém korpusu. Proto byl zvolen přístup využití již natrénovaných modelů, které se na data setu vtipů pouze doladí.

Implementace Využil jsem před-trénovaný embedding model *GloVe* s 300 dimenzemi pro každý vektor. *GloVe* je před-trénovaný na korpusu Wikipedia 2014 + Gigaword 5 obsahující 100 miliard slov. Tento model jsem obstaral pomocí Pythonové knihovny pro modelování přirozeného jazyka *gensim*.

Kromě modelu *GloVe* jsem zvážil použití i jiných modelů jako jsou *Numberbatch* nebo *Google news - word2vec*. Pro *GloVe* jsem se rozhodl, protože obsahuje jednak *stop words*, často používaná slova s malou informační hodnotou, a jednak obsahoval interpunkci, proto jsem jej uznal za vhodný pro generování textu.

Pro architekturu sítě jsem použil variantu architektury 3.1, kterou lze vidět na obrázku 3.1. Předchozí embedding model jen nahradil před-trénovaým *GloVe* modelem a velikost a počet použitých GRU vrstev jsem několikrát změnil během trénování při průzkumu vlivu hyperparametrů na trénování a generování.

3.4 GPT-2

Jako poslední přístup jsem vyzkoušel před-trénovaný model GPT-2, který byl nejčastěji používán v předchozích pracích.

Motivace Hlavním důvodem využití tohoto modelu bylo porovnání výše popsaných přístupů s nejmodernějšími modely pro NLP.

Implementace Z důvodu nedostatku paměti jsem byl donucen použít nejmenší poskytovanou variantu s 117 milióny parametry. Pro implementaci jsem použil knihovnu *Transformers* od společnosti Hugging Face [15].

4 Trénování a analýza výsledku

V této kapitole je popsáno, jaké data sety byly použity pro trénování modelů popsaných v předchozí kapitole, jak byla data předzpracována, jak probíhalo samotné trénování sítí. Dále jsou zde vyhodnoceny výsledeky každého natrénovaného modelu.

4.1 Data sety

Pro trénování výše popsaných modelů jsem použil dva data sety anglických vtipů za účelem normalizace textu byla provedena analýza dat a odfiltrování málo používaných slov (s četností menší než 3) a vtipů obsahující tato slova. Vlastnosti každého data setu jsou popsány v tabulce 4.1.

	Kaggle - Short jokes	Reddit - joke-dataset
Autor	Abhinav Moudgil	Taivo Pungas
Datum sestavení	6. 2. 2017	30. 4. 2017
Počet vtipů	231 657	194 552
Počet unikátních znaků	98	1044
Počet unikátních slov	96 147	48 883
Redukovaný počet u. slov	36 949	25 074
Maximální délka vtipu	250 znaků	740 znaků
Dostupné z	[9]	[16]

Tabulka 4.1: Tabulka porovnání data setu

4.1.1 Kaggle - Short jokes

Tento data set obsahoval pouze znaky ASCII. Ve vtipech se často nacházely chyby a překlepy, které byly odstraněny, například slova s nízkou četností. Data set obsahoval urážlivé vtipy a artefakty ze spojování hlaviček a těla vtipu. Častým artefaktem jsou tři tečky, po kterých následují další tři tečky. V data setu se občas vyskytují data, která nejsou vtip.

4.1.2 Reddit - joke-dataset

Data tohoto data setu byla sesbírána z stránek společnosti Reddit, ze subredditu */r/jokes* (Dostupné z: `www.reddit.com/r/Jokes/`).

Tento data set na rozdíl od data setu z Kaggle obsahoval velké množství znaků, které nebyly částí anglické abecedy a diakritiky. V data setu byly obsaženy znaky arabské abecedy, japonské abecedy, čínské abecedy, korejské abecedy, hindské abecedy, azbuky, mongolské abecedy, alfabety a mnoho jiných neurčitých znaků Unicode. Tyto znaky a vtipy obsahující tyto znaky byly z data setu odstraněny. I ve vtipech tohoto data setu se také často nacházely chyby a překlepy, které byly odstraněny, například slova s nízkou četností.

Data set obsahoval vtipy rozdělené do dvou částí - titulek a tělo, které byly při zpracování spojeny v jeden řetězec. V některých případech titulek a tělo obsahovaly stejný text a v těchto případech byl titulek zahozen.

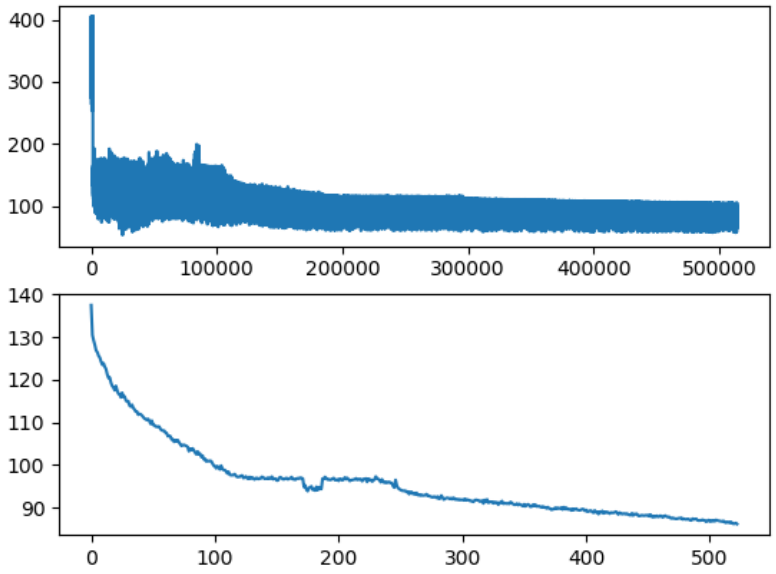
4.2 Sequence-to-sequence vtip na vtip

4.2.1 Předzpracování dat

Data byla náhodně spárována do dvojic, které se použily jako vstup a očekávaný výstup. Poté byla data převedena z *Unicode* do *ASCI*, uložena do slovníku pod index, a profiltrována od nežádoucích znaků a všech slov s nízkou četností. Do slovníku byla přidána 3 slova *SOS* - začátek věty, *EOS* - konec věty a *PAD* - vycpávka. Následně byla data převedena za pomocí vytvořeného slovníku do sekvence čísel odpovídající slovům ve vtipu a přidáváním vycpávek *PAD* sekvence byly normalizovány na stejnou délku. K sekvencím se vytvořily masky ukazující kde končí vtip a kde začíná vycpávka.

4.2.2 Proces trénování

Neuronové síti jsou předávány páry zakódovaných vtipů po dávce 40 (Batch) párů. První zakódovaný vtip v páru je síti dán jako vstup a druhý zakódovaný vtip z páru je použit pro vypočítání křížové entropie a hodnoty *loss*. Jako optimizér je použit *Adam* a úvodní rychlost učení (*learning rate*) je nastavena na $0,0001$. Mezi epochami jsou vtipy mezi páry promíchané a rychlost učení je upravována podle nutnosti, končíc na hodnotě $0,00001$. Během jedné epochy je náhodně vybráno 10% párů a u nich je použito *teacher forcing* - metoda kdy se neuronové síti předávají správná cílová slova namísto těch, co si síť sama vygenerovala. Síť je takto trénována po několik dní po 560 epoch.



Obrázek 4.1: Sequence-to-sequenct loss nad epochami

4.2.3 Analýza výsledku

Tato metoda se ukázala jako neúspěšná. Trénování probíhalo velice pomalu a i když hodnota *loss* klesala, síť nedokázala vygenerovat smysluplný text. Síť byla schopna vygenerovat zhruba základní slovosled připomínající vtip, ale výsledek byl zcela nepoužitelný. Příklady vygenerovaného textu:

- *what s the difference between a and ? ? ? ? . .*
- *what do you call a a a ? ? ? ? .*
- *how do you a a . a a a . ?*
- *what did the the say to to . . . ? ? . the the the .*
- *i was the my a . . . . . .*
- *did you hear the a a a a a a . . . . . .*
- *what do the the the a the ? a . ? ? the the the the the the the . . .*

Neúspěchu byl pravděpodobně způsobena tím, že se neuronová síť neměla čeho chytit. I kdyby síť vygenerovala opravdu vtipný text, zvolené hodnotící kritérium nedokázalo takový úspěch ohodnotit. Při pohledu na vygenerovaný text můžeme vyčíst několik věcí.

1. Model se dokázal naučit často používané začátky vtipů
2. Model našel nejúspěšnější taktiku pro uhodnutí náhodného slova, a to tipovat ty nejčastěji používaná slova v data setu a i v anglickém jazyce, neboť *the* je nejčastěji používané slovo a *a* je v prvních 10.
3. Model pochytil zvyk uživatelů píšící vtipy používat více než 3 tečky za sebou

Tento výsledek vypovídá, že pokud chceme odpovědět na vtip vtipem, musíme zvolit hodnotící kritérium, které dokáže ohodnotit kvalitu vtipu samotného bez hodnocení zvolených slov. Lepší metoda je použití *reinforcement learning* s kritériem kombinace výstupů dvou neuronových sítí, jedna hodnotící vtipnost (viz 2.5.3) a druhá gramatickou správnost textu.

4.3 Znaková rekurentní neuronová síť

4.3.1 Předzpracování dat

Z data setu se nejdříve vytvořil slovník znaků, kam byly jednotlivé znaky uloženy pod indexem, a ten pak byl profiltrován od nežádoucích znaků. Ke každému znaku se vypočítala pravděpodobnost, s jakou je daný znak prvním znakem řetězce. Tyto hodnoty se využily jako náhodný vstup pro síť při generování nového vtipu.

4.3.2 Proces trénování

Neuronové síti se jednotlivě předávaly zakódované vtipy po znacích a síť se pokoušela předpovídat další znak v sekvenci. Jako hodnotící kritérium byla použita křížová entropie. Jako optimizér byl použit *Adam* a úvodní rychlost učení (*learning rate*) byla nastavena na $0,0001$. Rychlost učení se měnila během trénování podle potřeby. Během jedné epochy se z data setu vybralo 1 000 náhodných vtipů pro trénování a proces trval 100 epoch.

4.3.3 Analýza výsledku

Tato metoda přinesla lepší výsledky než předchozí metoda. Výsledný text je více koherentní a v některých případech jsem jej shledal mírně humorným. Humornost je však spíše výjimkou než pravidlem. Vygenerovaný text však stále obsahuje velké množství zcela nesmyslných vět, obzvláště u delších textů. Vtipnost vygenerovaného textu hodnotím 5%. Příklady smysluplnějšího vygenerovaného textu vybraného z 200 pokusů:

- *What do you call a good farmer take a vegan? A same.*
- *Click and says "my friends can stop the crack" to get through the ground.*
- *A girl walks into a bar and asked me to say an embarrass*
- *How do you tell she said to the other? "This is great."*
- *I was lost a single part of my teeth to be a lot of bees and a plane store mother.*
- *A man walks into a bar... It's all goodness in the day.*
- *Did you hear about the Kamburgers last night? It was a little bell.*
- *People walk into a bar. I was going to talk about you.*

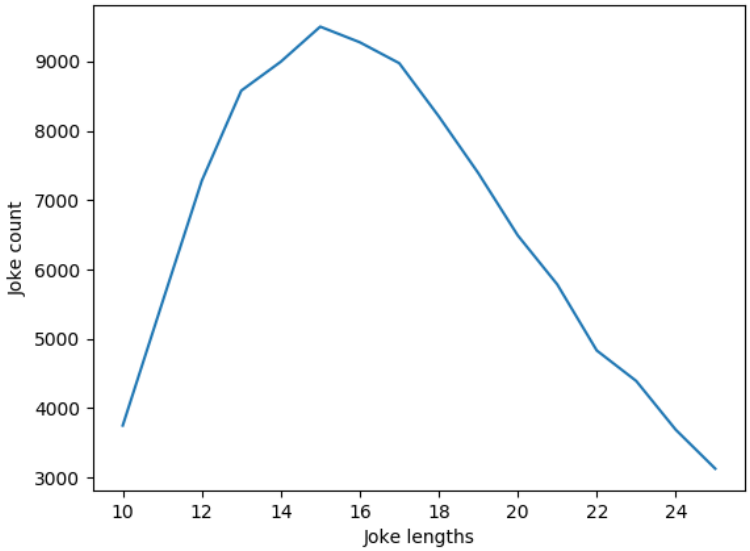
4.4 Sequence-to-sequence otázka a odpověď

4.4.1 Předzpracování dat

Data set byl nejdříve připraven pro zpracovaní přes *GloVe* embedding model. Data byla převedena na malá písmena, rozdělena na slova,a titulky spojeny s tělem vtipu. Poté byly z data setu odstraněny všechny vtipy, které obsahovaly slova neobsažena ve slovníku modelu *GloVe*.

V závislosti na obsahu vtipu byla každému vtipu přiřazena kategorie. Z data setu byly odstraněny některé z častých chyb, například opakování více vykřičníků, otazníku nebo více než 3 tečky. Nakonec byl slovník modelu *GloVe* zredukován tak, aby neobsahoval slova, která se nenacházejí v data setu.

Z data setu se následně vzala podskupina vtipů s kategorií otázka a odpověď a každý vtip byl rozdělen na otázku a odpověď. V posledním kroku se určila maximální délka vtipu a všechny delší vtipy se zahodily. Distribuci těchto délek je možno vidět na obrázku 4.2.



Obrázek 4.2: Distribuce délek vtipů v data setu

4.4.2 Proces trénování

Neuronové síti jsou předávány páry vtipů otázka a odpověď. Otázka je dávána na vstup a odpověď společně s výstupem sítě se používá pro výpočet hodnoty *loss*. Jako optimizér byl použit *AdamW* a úvodní rychlost učení (*learning rate*) byla nastavena na $0,0001$. Parametr rychlost učení se postupně měnil během trénování a skončil na hodnotě $0,00001$. *Batch size* byl nastavena na 50 a při poslední epoše na 100. *Teacher forcing* byl nastaven na 10 %.

Model je natrénován 8 krát s jinými počty vrstev GRU u enkodéru a dekodéru. Při trénování jsem nezjistil žádný zásadní rozdíl mezi počty použitých vrstev. Nejlepší model z pohledu velikosti hodnoty *loss* nad validačními daty používal enkodér s dvěma vrstvami o velkosti 300 a dekodér stejné velikosti. Tento model byl trénován po 57 epoch.

4.4.3 Analýza výsledku

Tato metoda přinesla podobné výsledky jako metoda znakové RNN, zmíněné v části 4.3. Výsledek jsem shledal na 17 % humorný a 33% koherentní. Model má problém vygenerovat koherentní text s více než 5 slovy. Zde je ukázka nejlepších vygenerovaných výsledků ze 100 generovaných vtipů:

- *what do you call a homeless pigeon? a pigeon .*
- *what is the difference between a sharply dressed man on a unicycle and a poorly dressed man on a bicycle? a tire*
- *what do you call a lawyer with a gun? a condescending suspect*
- *what is the difference between a man and a woman? wo*

4.5 GPT-2

4.5.1 Předzpracování dat

Před trénováním modelu GPT-2 došlo k minimální úpravě dat. Titulky vtipů byly spojeny s těly vtipů jako v předchozích metodách. Na začátek každého vtipu byl přidán řetězec *"JOKE:"*, sloužící jako spouštěcí slovo pro generování vtipu, a na konec každého vtipu řetězec *"<|endoftext|>"*.

4.5.2 Proces trénování

Modelu se předávala na vstup spojená sekvence vtipů z data setu, která nepřesahovala délku 300. Byl použit optimizér *AdamW* a úvodní rychlost učení (*learning rate*) byla nastavena na $0,00003$. *Batch size* je nastaven na 1 a trénování probíhalo po dvě epochy. Při druhé epoše byla rychlost učení 10 krát menší.

4.5.3 Analýza výsledku

Výsledný model GPT-2 dává velice koherentní výsledky, naprostá většina vět dává gramatický smysl. Přes to, že model GPT-2 vždy vrací koherentní text, výsledný text téměř nikdy není vtipný. Výsledný text je často smysluplný, logický a zmiňuje očividné věci. Toto jde proti tomu, co člověk shledává vtipné. Vtipné věci jsou absurdní a překvapivé, přesný opak toho co tento model nejčastěji generuje. Vtipnost vygenerovaného textu bych ohodnotil pod 1%. Zde můžete vidět ukázku vygenerovaného textu:

- *What do you call an Asian with two eyes? A nose.*
- *The best thing about a dead body is that you can't tell if the body was alive or dead*
- *Where do blind people go when they get sick? The hospital.*
- *How many lawyers does it take to change a lightbulb? One.*
- *What do you call a Mexican that is addicted to crack? A crack addict.*
- *What did the chicken say to the egg? I'm not a chicken!*
- *A blind man walks into a bar. The bartender says he's not allowed. The man says he's not allowed. The bartender asks him what he's doing. The man replies, ""I'm going to the bar."" The bartender says, ""You're not allowed.""*

5 Možnosti trénování v českém jazyce

Pro trénování modelu v českém jazyce potřebujeme dostatečně velký data set českých vtipů. To se ukázalo jako problém. V momentě psaní této práce neexistuje jednotně sesbíraný data set českých vtipů, podobný těm, které byly použity pro trénování v anglickém jazyce. Data sety anglických vtipů byly převážně sestaveny z platformy Reddit, kde je velice aktivní komunita lidí předávající si vtipy. Český ekvivalent této komunity není, české vtipy lze na internetu najít, nejsou však centralizované v takovém množství na jednom místě. Lze najít mnoho menších databází vtipů v češtině, ze kterých by se takový data set mohl sesbírat. Příklady zdrojů českých vtipů:

- panvtip.cz
- alik.cz
- ftipky.org
- vtipalek.cz
- vtipy.net
- vs-vtipy.tonikovo.cz

6 Závěr

V této práci byla popsána technologie neuronových sítí a jejich použití pro generování textu se zaměřením na rekurentní neuronové sítě. Dále byly prozkoumány některé práce zabývající se tématikou neuronových sítí v kontextu využití pro generování a klasifikování humorného textu.

Následně byly navrženy čtyři architektury sítí pro generování vtipů. Modely těchto sítí byly vytvořeny a natrénovány pomocí vtipů v anglickém jazyce. Výsledkem jsou čtyři natrénované modely generující vtipy s odlišnými výsledky.

Model sequence-to-sequence pokoušející se generovat vtipy metodou odpovědí vtipem na vtip se ukázal jako nevhodný přístup. Tento pokus pěkně ukazuje, jak je správné zvolení způsobu hodnocení naprosto nezbytné pro úspěšné natrénování neuronové sítě. Jelikož způsob hodnocení nebyl zvolen vhodně, došlo k tomu, že neuronová síť se snažila předpovídat náhodná slova obsažená v data setu vtipů z nesouvisejících vstupů. V tomto neuronová síť v rámci možností celkem dobře uspěla. Dokázala poskládat často používané fráze ve vtipech a zbytek textu doplnila nejpoužívanějšími slovy v data setu. To nebyl původní cíl, a proto se jedná o neúspěšný pokus.

Model znakové rekurentní neuronové sítě se ukázal jako lepší přístup. Jedná se o přístup, který již byl použit v předchozích pracích zabývající se generováním textu. Tento model je schopen generovat koherentní text a občas i humorný text. Průměrná humornost textu však není příliš vysoká a pokus nepovažuji za úspěšný.

Model sequence-to-sequence, pokoušející se generovat vtipy metodou doplněním vstupní otázky odpovědí, vykazuje potenciál. Vygenerovaný text dokáže být koherentní a i vtipný pokud je výsledný text dostatečně krátký. Průměrnou humornost textu jsem u tohoto model shledal nejvyšší ze všech natrénovaných modelů.

Model GTP-2 úspěšně generuje koherentní text, průměrná vtipnost vygenerovaného textu je však horší než předchozí modely. Model generuje text, který je často logický a očividný. To jsou kvality, které s humorem nekorespondují.

Navrhnutí dalších kroků Pokusem naučit model sequence-to-sequence odpovídat vtipem na vtip se ukázalo, že metoda hodnocení je velice důležitá pro dosažení chtěných výsledků. Tento neúspěšný pokus mě přivádí k hypotéze, že dosavadní pokusy o generování humorného textu trpí stejným problémem. Trénované sítě nejsou správně hodnoceny. Všechny dosavadní metody používaly hodnotící kritérium křížové entropie, která se ukázala jako efektivní kritérium pro generování koherentního textu, ale co se týče humornosti textu, toto kritérium neposkytuje žádnou zpětnou vazbu. Výsledkem je, že pokud neuronová síť někdy vygeneruje opravdu humorný

text, hodnotící kritérium není schopné toto detekovat a toto chování neuronové sítě podpořit.

Navrhuji tedy využití hodnotících kritérií, která jsou schopna dodat trénované síti zpětnou vazbu na humornost. V úvahu připadají dvě metody jak dosáhnout takové zpětné vazby. Hodnocení lidmi a hodnocení klasifikátorem humoru podobným jako byl popsán v 2.5.3.

Literatura

[1] Christopher Olah. Understanding lstm networks, 8 2015. [online] Dostupné z: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[2] Andreas Graefe. Guide to automated journalism, 1 2016. [online] Dostupné z: https://www.cjr.org/tow_center_reports/guide_to_automated_journalism.php/.

[3] Matheson Marcault. The history of text generation. MathesonMarcault.Com [online], 2020 (Přístup získán 20. 4. 2020). [online] Dostupné z: http://mathesonmarcault.com/index.php/2015/12/15/randomly-generated-title-goes-here/.

[4] C Bassett. C. the computational therapeutic: exploring weizenbaum's eliza as a history of the present. *AI & SOCIETY*, 34:803–812, 2019. [online] Dostupné z: https://doi.org/10.1007/s00146-018-0825-9.

[5] Jiří a Roman NERUDA. ŠÍMA. *Teoretické otázky neuronových sítí*. Matfyzpress, Praha, 1996. [online] Dostupné z: http://www.cs.cas.cz/~sima/kniha.html.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[7] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[8] Lorenzo Ampil. Can a robot make you laugh? — teaching an ai to tell jokes, 12 2019. [online] Dostupné z: https://towardsdatascience.com/can-a-robot-make-you-laugh-teaching-an-ai-to-tell-jokes-815f1e1e689c.

[9] Abhinav Moudgil. Short jokes, 2016. [online] Dostupné z: https://www.kaggle.com/abhinavmoudgil95/short-jokes/version/1.

[10] Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications, 2 2019. [online] Dostupné z: https://openai.com/blog/better-language-models/#sample1.

[11] Martins Frolovs. Teaching gpt-2 transformer a sense of humor, 12 2019. [online] Dostupné z: https://towardsdatascience.com/teaching-gpt-2-a-sense-of-humor-fine-tuning-large-transformer-models-on-a-single-gpu-in-pytorch-59e8cec40912.

[12] Sylvain Gugger and Jeremy Howard. Adamw and super-convergence is now the fastest way to train neural nets, 7 2018. [online] Dostupné z: https://www.fast.ai/2018/07/02/adam-weight-decay/#adamw.

[13] Orion Weller and Kevin Seppi. Humor detection: A transformer gets the last laugh, 2019.

[14] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, 5 2015. [online] Dostupné z: http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38--45, Online, October 2020. Association for Computational Linguistics.

[16] Taivo Pungas. A dataset of english plaintext jokes., 2017. [online] Dostupné z: https://github.com/taivop/joke-dataset.