VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHATBOT POSTAVENÝ NA UMĚLÝCH NEURONOVÝCH SÍTÍCH

CHATBOT BASED ON ARTIFICIAL NEURAL NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUKÁŠ RICHTARIK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. IGOR SZŐKE, Ph.D.**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Richtarik Lukáš**
Obor: Informační technologie
Téma: **Chatbot postavený na umělých neuronových sítích**
Chatbot Based on Artificial Neural Networks
Kategorie: Zpracování řeči a přirozeného jazyka

Pokyny:

1. Seznamte se se základy chatbotů postavených na generativních modelech. Nastudujte základy strojového učení.
2. Najděte vhodná a veřejně dostupná data. Navrhněte jednoduchého chatbota (vstupem a výstupem bude text). Otestujte kvalitu jeho odpovědí pomocí vhodně zvolené metriky.
3. Zdokonalte chatbota (např. více dat, lepší machine learning techniky) a průběžně sledujte jeho úspěšnost.
4. Otestujte chatbota na několika uživatelích. Diskutujte dosažené cíle a navrhněte směry dalšího vývoje.
5. Vyrobte A2 plakátek a cca 30 vteřinové video prezentující výsledky Vaší práce.

Literatura:

- Dle pokynů vedoucího
- http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2 ze zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szőke Igor, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
L.S. 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Táto práca sa zaoberá problematikou chatbotov postavených na umelých neurónových sieťach a generatívnych modeloch. Popisuje postup a možnosti pri návrhu takéhoto chatbota a taktiež samotnú implementáciu a testovanie pomocou BLEU metriky. Práca obsahuje experimenty s rôznymi modelmi chatbotov, ich vyhodnotenie a porovnanie, testovanie na užívateľoch a niekoľko návrhov na budúce vylepšenia.

Abstract

This work deals with the issue of chatbots, which are based on artificial neural networks and generative models. It also describes options and process of designing the chatbot as well as an implementation and testing using BLEU metrics. The work contains multiple experiments with different models of chatbots, their performance evaluation and comparison, user experience and several suggestions for future enhancements.

Klíčová slova

chatbot, neurónová sieť, BLEU, generatívny model, sequence to sequence

Keywords

chatbot, neural network, BLEU, generative model, sequence to sequence

Citace

RICHTARIK, Lukáš. *Chatbot postavený na umělých neuronových sítích.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Chatbot postavený na umělých neuronových sítích

Prohlášení

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Igora Szőkeho, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Lukáš Richtarik
16. května 2018

Poděkování

Rád by som poďakoval pánu Ing. Igorovi Szőkemu, Ph.D. za vedenie tejto bakalárskej práce, odbornú pomoc a cenné rady.

Obsah

Kapitola 1

Úvod

V súčasnosti sa stále viac dostávajú do popredia technológie na spracovanie prirodzeného jazyka a s nimi aj snahy o vytvorenie dokonalých chatbotov s ktorými by vedeli ľudia komunikovať prirodzenou ľudskou rečou. Alan Mathison Turning v roku 1950 zadefinoval kritéria, nazvané Turningov test, ktoré mali otestovať inteligenciu programov. V tomto teste človek komunikuje s dvoma respondentmi, ktorých nemá šancu vidieť. Jedným z nich je človek a druhým počítač. Komunikácia prebieha výlučne v textovej forme, trvá päť minút a konverzačné témy nie sú obmedzené. Spolu s komunikujúcim je prítomná aj komisia, ktorá má tiež určiť či daná komunikácia prebieha s človekom alebo s počítačom. Ak program dokáže presvedčiť 30% hodnotiteľov, že komunikácia prebieha s človekom, program testom prešiel a môže byť považovaný za inteligentný. Na základe tohto testu sa každý rok koná súťaž Loebnerova cena, v ktorej súťažia programy, simulujúce ľudskú konverzáciu. Víťazom tejto súťaže sa stáva najúspešnejší zo súťažiacich programov. V roku 2017 vyhral prvé miesto chatbot Mitsuku.

V posledných rokoch je na vzostupe rozvoj umelej inteligencie pomocou neurónových sietí, ktorú sa snažia mnohé firmy a vedecké tímy použiť na spracovanie a pochopenie prirodzeného jazyka a vytvorenie generatívnych chatbotov, ktorí by boli schopní sami tvoriť vety a komunikovať s človekom. Takýto chatboti by neboli obmedzení len na špecifickú úlohu ako zákaznícka podpora, či poradenstvo v eshope. Mohli by komunikovať o hocičom a ich odpovede by boli len ťažko rozoznateľné od ľudských. Mohli by taktiež pomáhať ľuďom v každodennom živote a plniť ich rozličné príkazy.

V dnešnej dobe sú takýto chatboti len predmetom výskumu. Jedným z najpoužívanejších modelov ktorý sa používa na tvorbu generatívnych chatbotov je sequence to sequence. Tento model sa preslávil hlavne v oblasti strojového prekladu. V kombinácii s rozličnými vylepšeniami ako sú attention mechanizmus, bidirectional LSTM neurónové vrstvy či word embedding je tento model veľmi úspešný aj v problematike tvorby chatbotov.

V tejto práci sa pokúsim o návrh, vytvorenie a otestovanie jednoduchého chatbota, ktorý bude fungovať na princípe umelej inteligencie založenej na neurónových sieťach. Použijem pri tom sequence to sequence architektúru. Výsledného chatbota vytvorím modulárne, aby doňho bolo možné pridávať rôzne modely neurónových sietí a rôzne postupy spracovávania a reprezentovania vstupných viet. Pomocou experimentov s týmto modelom sa pokúsim nájsť najlepšiu kombináciu parametrov pri ktorých sa bude chatbot trénovať čo najefektívnejšie a jeho odpovede budú čo najkvalitnejšie. Komunikáciu s výsledným chatbotom potom otestujem na užívateľoch.

V druhej kapitole je popísaná základná problematika chatbotov - čo sú chatboti, ich história a využitie. Taktiež sa v nej zaoberám základným rozdelením chatbotov. Sú tam

popísané princípy ich fungovania a ich výhody a nevýhody. Ďalej nasleduje kapitola popisujúca základy neurónových sietí a frameworky, ktoré boli použité pri vytváraní chatbota v tejto práci. V tejto kapitole je tiež opísaný jeden z najpoužívanejších modelov, ktorý sa používa pri tvorbe generatívnych chatbotov - sequence to sequence. V štvrtej kapitole sú popísané princípy a možnosti pri návrhu chatbota. Zaoberám sa tam najmä tým, ako fungujú metódy tokenizácia, padding, word embedding, teacher forcing a hodnotenie odpovedí pomocou BLEU skórovania. Ďalej nasledujú dve kapitoly popisujúce moju implementáciu chatbota a experimenty s ním. Tiež sú tam popísané výsledky testovania chatbota na užívateľoch. V poslednej kapitole sú spísané vylepšenia chatbota, ktorými sa chcem v budúcnosti zaoberať.

Kapitola 2

Úvod do problematiky chatbotov

Chatboti, tiež nazývaní konverzační agenti sú v súčasnosti veľmi diskutovanou témou. Vytvorenie programu, ktorý by bol schopný porozumieť prirodzenému jazyku a dokázal by reagovať na akúkoľvek otázku, by revolučne zmenil zaužívané formy komunikácie, marketingu, či zákazníckeho servisu.

Chatbot je počítačový program, ktorý simuluje konverzáciu pomocou súboru pravidiel, prípadne umelej inteligencie. Typicky je chatbot navrhnutý na komunikáciu s človekom, alebo s iným chatbotom. V súčasnosti prevládajú chatboti, ktorí zvládajú len určitý okruh tém, o ktorých vedia konverzovať. Používajú sa na špecifickú funkciu, napríklad pomoc pri online nákupe, prípadne pomoc s jednoduchším technickým problémom. Len málo súčasných chatbotov dokáže konverzovať o akejkoľvek téme. Takýto chatboti sú v súčasnosti predmetom výskumu.

2.1 História chatbotov

Dnešný chatboti dosiahli úroveň, kedy sa už človek zamýšľa nad tým, či si momentálne píše so softvérom alebo ľudským partnerom. Tomu však predchádzal dlhý vývoj, ktorý začal už v roku 1994. Eliza, prvý chatbot na svete, bola vyvinutá profesorom Jozephom Weizenbaumom z MIT v roku 1966 a mala iba 200 riadkov kódu. Tento program simuloval Rogersovského psychoterapeuta tým, že väčšinu viet od užívateľa pretransformoval zase do otázky. Fungoval na princípe pravidiel, podľa ktorých vedel odpovedať na otázky.

V roku 1972 Kenneth Mark Colby vytvoril chatbota, ktorý simuloval paranoidného schizofrenika. Nazval ho Parry. Bol zložitejší ako Eliza, mal v sebe viacej scenárov a dokázal odpovedať na viacej otázok.

V roku 1988 bol vytvorený program Jabberwacky, ktorého autorom bol Rollo Carpenter. Tento chatbot nefungoval na princípe aplikovania súboru pravidiel. Učil sa z interakcie s užívateľom a vytváral si databázu z minulých konverzácií a z nich potom vyberal najlepšie odpovede.

Sbaitso bol naprogramovaný v roku 1992 pre systém MS-Dos. Tento chatbot simuloval psychológa, ktorý disponuje len základnými frázami, najčastejšie "Why do you feel that way?", alebo "That's not my problem". Jeho výhodou bolo, že umožňoval komunikáciu aj pomocou zvuku.

V roku 1995 Richard Wallace vytvoril chatbota s názvom ALICE. Tento chatbot pracoval s XML schémou AIML (artificial intelligence markup language), s ktorou je možné špecifikovať súbor pravidiel, podľa ktorých dokáže chatbot odpovedať na otázky. Neskôr

v roku 2001 bola zverejnená špecifikácia AIML a táto schéma bola použitá vo veľa ďalších chatbotoch a používa sa dodnes. Alice stále funguje a jej databáza pravidiel sa stále zväčšuje.

V roku 2001 bol vytvorený chatbot SmarterChild, ktorý bol predchodcom Siri. Program bol dostupný v MSN Messanger a bol určený pre vytváranie zábavných konverzácií.

Watson z firmy IBM bol naprogramovaný v roku 2006. Bol vytvorený za účelom vyhrať v americkej vedomostnej súťaži Jeopardy. Túto súťaž v roku 2011 vyhral proti 2 šampiónom. Tento chatbot mal prístup k informáciám o veľkosti cca 4 terabajtov. V súčasnosti sa vďaka veľkému množstvu informácií ku ktorým ma prístup, používa experimentálne napríklad v nemocnici Memorial Sloan Kettering Cancer Center v New Yorku.

Spoločnosť Apple vyvinula v roku 2010 program Siri. Siri je inteligentný asistent a poradca, ktorý dokáže komunikovať v prirodzenom jazyku. Dokáže porozumieť textu aj reči, rozpoznávať obrázky a videá a vykonávať príkazy zadané užívateľom.

V roku 2012 bol vytvorený Google Now ako konkurencia k Siri. Google Now dokáže vykonávať príkazy užívateľa, rozpoznávať reč, texty, obrázky, polohu užívateľa, dopravu a používa systém predikcie, kedy vie predpokladať čo užívateľ bude chcieť alebo potrebovať.

Alexa od firmy Amazon a Cortana od firmy Microsoft vznikli v toku 2015. Títo chatboti používajú algoritmy na spracovanie prirodzeného jazyka, dokážu rozpoznávať príkazy užívateľa, odpovedať v prirodzenom jazyku, posielať emaily, smsky a získavať rôzne informácie, uložené na internete. Cortana používa pri hľadaní odpovedí nástroj na vyhľadávanie od firmy Microsoft - Bing [13].

2.2 Využitie

Využitie chatbotov je veľmi široké. Dajú sa použiť v internetovom obchode, kedy užívateľ nedokáže nájsť produkt ktorý hľadá, prípadne sa chce ohľadom tohto produktu niečo opýtať. Chatboti sú v tomto smere veľmi užitoční. Z konverzácie s užívateľom vedia vyčítať o aký produkt má užívateľ záujem, vedia mu poradiť, prípadne odporučiť nejaké produkty a pomôcť s objednávkou. Taktiež s ich ľudským prístupom dokážu ľahšie nakloniť nakupujúceho ku kúpe daného produktu.

Ďalšou oblasťou využitia je zákaznícky servis. Firma tak poskytuje 24 hodinovú podporu pre užívateľoch, ktorý nemusia byť v strese z čakania na operátora. Jednoduché požiadavky zákazníka tak dokáže vyriešiť chatbot a zložitejšie môže poslať operátorovi. Takýto chatbot dokáže obslúžiť neobmedzene veľa užívateľov naraz. Výrazne to zlepší meno firmy a spokojnosť zákazníkov.

Chatboti môžu taktiež zefektívniť interné procesy vo firme a produktivitu zamestnancov. Môžu zamestnancovi pomôcť vyhľadať informácie a ušetriť čas, prípadne by mohli byť zapojení do spracovania žiadostí o dovolenku alebo uchádzaní sa o voľné pozície vo firme. To môže viesť k zníženiu počtu zamestnancov, ktorí vykonávajú túto činnosť a menším nákladom na fungovanie firmy.

Chatboti majú uplatnenie aj vo vzdelávaní. Sú používaní napríklad pri učení cudzích jazykov. Užívateľ sa tak dokáže rýchlo naučiť nové slová a frázy bez pociťovania stresu z lektora alebo vyučujúceho. Taktiež sa pri takomto vzdelávaní nestretne s netrpezlivosťou a podráždenosťou zo strany vyučujúcich.

Veľké uplatnenie majú v dnešnej dobe chatboti ako sú Alexa, Cortana alebo Siri. Ich schopnosť pochopiť prirodzený jazyk v spojení s prístupom k údajom užívateľa, internetu a možnosti vykonávať príkazy ich robí veľmi dobrými asistentmi a spoločníkmi pri práci s mobilom alebo počítačom. Užívateľ sa tak môže spýtať na predpoveď počasia na nasledujúci

deň, aktuálny čas, môže si pomocou takéhoto asistenta nastaviť budík alebo pripomienku, alebo požiadať o získanie akejkoľvek informácie z internetu.

2.3 Rozdelenie chatbotov

Existuje mnoho delení chatbotov. Najčastejšie sa delia na 3 druhy. Na tých, ktorí používajú machine learning a spracovanie prirodzeného jazyka (generative a retrieval-based) a tých, ktorí sa riadia len určitým súborom pravidiel, ktoré popisujú ako majú odpovedať na jednotlivé otázky (rule based).

2.3.1 Rule-based chatbot

Rule-based chatbot (chatbot založený na pravidlách) používa súbor pravidiel, podľa ktorých dokáže odpovedať na otázky užívateľa. Tieto pravidlá môžu byť jednoduché, alebo aj zložité s množstvom premenných a regulárnych výrazov. Je veľmi efektívny pri odpovedaní na vety, ktoré zahrňujú jeho pravidlá. Naopak, má veľký problém s odpoveďou na vety, ktoré jeho pravidlá nedefinujú. Najznámejší programovací jazyk, pre popis pravidiel chatbota je AIML (Artificial Intelligence Markup Language). Je to značkovací jazyk vychádzajúci z XML. V tomto značkovacom jazyku je možné tvoriť kategórie (category), v ktorých je zadefinovaná otázka (pattern) a odpoveď na túto otázku (template). V jazyku AIML je možné používať premenné a regulárne výrazy a chatbot naprogramovaný v tomto jazyku je schopný učiť sa nové informácie. Ďalšou obdobou AIML sú jazyky SIML (Synthetic Intelligence Markup Language) a Chatscript. Na obrázku 2.1 je ukážka kódu v jazyku AIML, zobrazujúca ako sa definujú pravidlá pre fungovanie chatbota [2].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aiml>
    <category>
        <pattern>START</pattern>
        <template>
            Hi <bot name="first_name"/>! I am a demo bot and I do awesome things.
        </template>
    </category>
    <category>
        <pattern>HOW ARE YOU</pattern>
        <template>
            I am just a chatbot and I am doing great. Test me often to improve!
        </template>
    </category>
    <category><pattern>*</pattern><template>Sorry I didn't get it =(.</template></category>
</aiml>
```

Obrázek 2.1: Ukážka kódu v jazyku AIML

Keďže rule-based chatboti dokážu reagovať len na vety a scenáre na ktoré boli naprogramovaní, nedokážu byť nikdy použití na otvorené konverzácie hocijakých tém. Bolo by nemožné vytvoriť pravidlá, ktoré by dokázali pokryť všetky scenáre. Preto je ich najčastejšie využitie v zákazníckej podpore a internetových obchodoch kde sa od nich nevyžaduje schopnosť vedieť reagovať na akúkoľvek otázku. V týchto oboroch sú však veľmi efektívni. Najznámejší rule-based chatboti sú Alice a Mitsuku [23].

2.3.2 Retrieval-based model

Chatboti ktorí používajú retrieval-based model, sú trénovaní na sade otázok a ich možných odpovediach. Pri konverzácii si potom ku každej otázke vyberú vždy niekoľko možných odpovedí z vopred danej sady. Výber možných odpovedí môže prebiehať pomocou jednoduchých algoritmov, alebo aj pomocou zložitejších algoritmov strojového učenia. Finálnu odpoveď zo súboru možných odpovedí si potom vyberú podľa skórovacieho algoritmu, ktorý hodnotí kvalitu odpovedí. Takýto chatbot nedokáže vygenerovať nové vety. Vie používať len tie, ktoré mu boli vopred dodané. Kvôli tomu sa títo chatboti využívajú prevažne len na špecifickú úlohu, napríklad v zákazníckom servise, či obchode. Nedokážu dostatočne reagovať na pre nich neznáme témy. Výhodou je, že nerobia gramatické a syntaktické chyby [4].

2.3.3 Generatívny model

Chatboti využívajúci generatívne modely dokážu generovať nové odpovede, na ktorých neboli trénovaní. Považujú sa preto za najinteligentnejší druh chatbotov. Zvyčajne sú realizovaní pomocou neurónových sietí, obsahujúcich rekurentné neuróny, schopné pamätať si kontext viet. Vďaka svojej vlastnosti tvoriť nové vety, môžu byť použití v otvorených konverzáciách bez vopred danej témy. Ich odpovede sú však často sprevádzané syntaktickými a sémantickými chybami. Tento druh chatbotov je stále vo vývoji a je predmetom výskumu. V súčasnosti sa v praxi používa väčšinou len ako forma zábavy. Budúcnosť inteligentných chatbotov však spočíva práve v tomto modely, vďaka ich schopnosti učiť sa, chápať význam otázok a generovať nové odpovede [3].

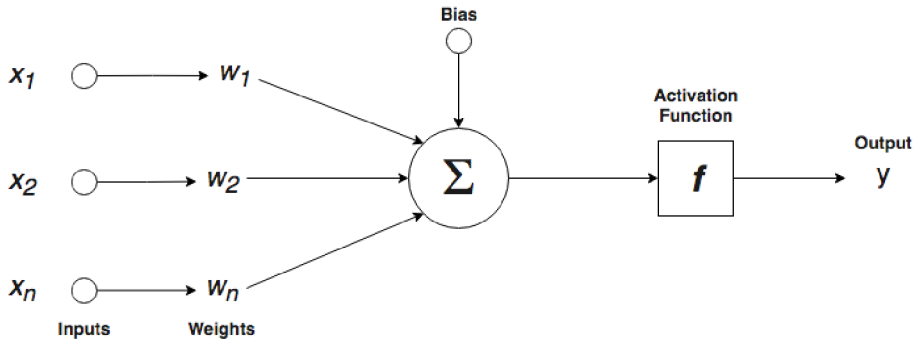
Kapitola 3

Umelé neurónové siete

Pre vytvorenie generatívneho chatbota, ktorý bude používať strojové učenie je veľmi dôležité zvoliť si vhodný model neurónovej siete. V ďalších podkapitolách nasleduje vysvetlenie toho, čo sú neurónové siete, ktoré typy neurónových sietí sa využívajú pri tvorbe chatbotov a ktoré frameworky som využil na prácu s neurónovými sieťami.

3.1 Neurónová sieť

Neurónová sieť je paralený výpočtový systém, ktorý umožňuje uchovávanie a následné spracovávanie informácií. Prvé umelé neurónové siete vznikli už na konci 50 rokov minulého storočia, no navzdory ich úspechu bol ich výskum utlmený, keďže potrebovali vyšší výpočtový výkon, než ktorý bol vtedy dostupný. S príchodom nového tisícročia sa vďaka expandujúcej technike do všetkých oblastí života a dostupnosti výpočtových zdrojov opäť vzbudil záujem o neurónové siete. V súčasnosti sú úspešným výpočtovým modelom v mnohých oblastiach ako napríklad počítačové videnie, strojový preklad, či spracovanie reči a prirodzeného jazyka.



Obrázek 3.1: Model umelého neurónu

Základnou časťou neurónovej siete je neurón, ktorého model je zobrazený na obrázku 3.1. Premenné x_1,...,x_n predstavujú vstupné hodnoty, ktoré vchádzajú do neurónu. Každý vstup má priradenú váhu. Označujeme ich w_1,...,w_n, šípkou v smere od vstupu do tela neurónu. Zmenou veľkosti váh sa do neurónu ukladá informácia. Čím vyššia je hodnota váhy, tým dôležitejší je daný vstup neurónu. Hodnota b sa nazýva bias a slúži na ovplyvňovanie

aktivačnej funkcie neurónu. Neurón má ďalej svoj vnútorný potenciál ξ, ktorý sa počíta vzťahom:

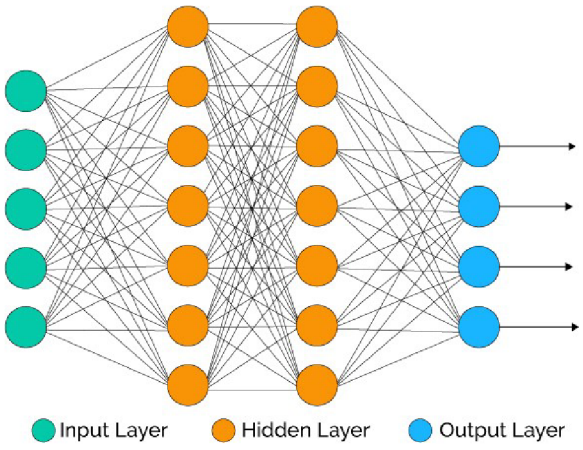
$$\xi = b + \sum_{i=1}^{N} w_i * x_i \tag{3.1}$$

Výstup z neurónu y sa počíta pomocou potenciálu neurónu ξ a aktivačnej funkcie f, ktorá slúži na prahovanie potenciálu neurónu:

$$y = f(\xi) \tag{3.2}$$

V minulosti sa používala ako aktivačná funkcia ostrá nelinearita. Neskôr ju však nahrádzali sigmoidálne funkcie, ktoré sú vhodnejšie na reprezentovanie komplexnejších funkcií. Základné funkcie ktoré sa používajú v neurónových sieťach sú logistická sigmoida, hyperbolický tangens, ReLu a ostrá nelienearita [24].

Neurónová sieť vzniká prepojením viacerých neurónov. Ako môžeme vidieť na obrázku 3.2, jednotlivé neuróny, ktoré sú vedľa seba, tvoria vrstvu neurónov. Takéto vrstvy neurónov sa často napájajú za sebou. Siete ktoré šíria vstupnú informáciu iba jedným smerom na výstup, nazývame dopredné siete. Ak v sieti existujú spätné prepojenia medzi neurónmi alebo vrstvami, hovoríme o rekurentných sieťach. V neurónových sieťach rozlišujeme 3 vrstvy - na začiatku vstupnú vrstvu (vrstvu cez ktorú sa dáta dostávajú do siete), na konci výstupnú (vrstvu odkiaľ dáta vystupujú zo siete) a medzi nimi niekoľko skrytých vrstiev (všetky vrstvy nachádzajúce sa medzi vstupnou a výstupnou vrstvou). Čím viac skrytých vrstiev sieť obsahuje, tým citlivejšie rozhodovanie môžeme zo siete dostať.



Obrázek 3.2: Jednoduchý model neurónovej siete. Zdroj: [29]

Základnou vlastnosťou neurónových sietí je schopnosť rozpoznať a abstrahovať vzťahy medzi vstupnými a výstupnými hodnotami a odvodiť pravidlá, ktoré medzi nimi platia. Následne tieto pravidlá môže neurónová sieť aplikovať na akékoľvek vstupné hodnoty. Proces abstrakcie sa nazýva učenie a môže prebiehať s učiteľom alebo bez učiteľa. Počas tohto učenia sa podľa určitých učiacich algoritmov upravujú váhy spojení. Po ukončení učenia sa tieto váhy nemenia, sieť aplikuje na vstupy svoje naučené pravidlá [12].

3.1.1 Typy neurónových sietí

Existuje mnoho typov neurónových sietí. Každá z nich má svoje špecifické vlastnosti, pomocou ktorých dokáže riešiť určitú kategóriu problémov. My si uvedieme typy, ktoré majú využitie pri spracovaní prirodzeného jazyka a ktoré teda môžu byť použité pri vytváraní chatbota [8].

Perceptrón je špeciálny prípad neurónu, v ktorom je vnútorný potenciál počítaný ako vážený súčet vstupov: $\xi = b + \sum_{i=0}^{n} w_i * x_i$, pričom x je vstup neurónu a w je jeho váha. Perceptrón môže byť diskrétny alebo spojitý. Diskrétny perceptrón na výstup posiela len dve možné hodnoty a alebo b. Používa hodnotu nazývanú prah, pomocou ktorej sa určuje hodnota na výstupe neurónu. Ak je suma váhovaných vstupov menšia ako prah, na výstup odošle hodnotu a. Ináč na výstup odošle hodnotu b. Pri spojitom perceptróne je výstupom spojitá funkcia sumy váhových vstupov. Aktivačnou funkciou takého perceptrónu je funkcia sigmoida. Sigmoida, tiež nazývaná logistická krivka je funkcia, ktorá sa často používa v neurónových sieťach. Transformuje akúkoľvek hodnotu do intervalu (0,1). Má predpis $f(t) = 1/e^{-t}$.

Perceptrón bol vynájdený v roku 1950, vedcom Frankom Rosenblattom. V dnešnej dobe používame aj novšie modely umelých neurónov, ktoré sú však inšpirované vlastnosťami a princípmi perceptrónu. Perceptrón je určený na dichotomickú klasifikáciu - rozdelenie vektorov do dvoch, lineárne separovateľných tried [12].

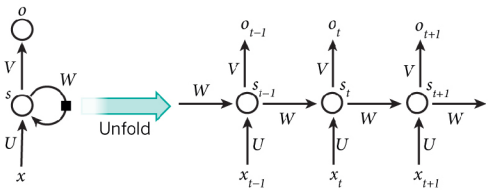
Viacvrstvová perceptrónová sieť (MLP - multilayer perceptrón) je zložená z troch a viac vrstiev perceptrónov. Každý uzol vo vrstve je prepojený s každým uzlom v nasledujúcej vrstve - to robí sieť plne prepojenú (fully connected). Táto sieť je vhodná na klasifikáciu dát, ktoré nie sú lineárne separovateľné. Používa sa pri spracovaní prirodzeného jazyka, konkrétne pri rozpoznávaní reči (speech recognition) a strojovom preklade textu (machine translation).

Konvolučná neurónová sieť je typ siete, ktorá sa používa prevažne v oblasti počítačového videnia, ale tiež aj v spracovaní reči a prirodzeného jazyka. Táto sieť je inšpirovaná vizuálnym cortexom u cicavcov. Konvolučné siete používajú rôzne variácie viacvrstvových perceptrónových sietí spojených do jednej. Ich základnou vlastnosťou je možnosť napojenia neurónu v skrytej vrstve na neurón z predchádzajúcej vrstvy. Druhou vlastnosťou je n-dimenzionálny rozmer vrstiev. Treťou hlavnou vlastnosťou je, že určité váhy môžu byť zdieľané medzi viacerými neurónmi. Napríklad všetky neuróny v jednej konvolučnej vrstve v danej hĺbke môžu mať zdieľané váhy. Konvolučné siete sú zvyčajne zložené z konvolučných vrstiev, ktoré môžu byť napojené na vstupnú vrstvu, alebo na vrstvu Poolingu, Relu, alebo opäť konvolučnú vrstvu. Výstupnej vrstve predchádza ešte jedna plne prepojená vrstva, ktorá zhusťuje dáta do jednej dimenzie. Vrstvy na začiatku konvolučnej siete rozlišujú nízkoúrovňové príznaky ako jednoduché prechody farieb a detektory hrán. Každou ďalšou vrstvou sieť rozlišuje komplexnejšie štruktúry ako napríklad viacfarebné príznaky a geometrické tvary. Posledné vrstvy už dokážu rozlíšiť jednotlivé objekty a zložité tvary [8].

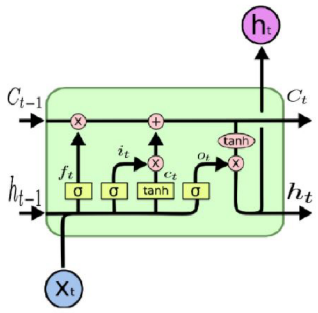
Rekurzívna neurónová sieť je typ neurónovej siete kedy je vstup spracovávaný hierarchicky pomocou vrstiev neurónov usporiadaných do stromovej štruktúry. Používa sa pri pracovaní so sekvenciami dát. Rekurzívne neurónové siete sa tiež často využívajú pri spracovávaní dát, ktoré sú štruktúrované, napríklad proteínové štruktúry, HTML webstránky, DNA regulačné siete alebo pársovacie stromy pri spracovaní prirodzeného jazyka. Sú užitočné aj pri mapovaní slov do n-rozmerného priestoru (word embedding) [5].

Rekurentná neurónová sieť je typ rekurzívnej neurónovej siete, v ktorej prepojenia medzi jednotlivými neurónmi vytvárajú orientovaný cyklus. Znamená to, že výstup neurónu závisí nielen od jeho vstupu, ale aj od jeho predchádzajúceho stavu. Vo vnútornom stave neurónu sú zakódované postupne všetky dáta, ktoré prešli cez daný neurón. Rekurentné prepojenia medzi neurónmi a vrstvami spôsobujú, že si dokáže sieť pamätať, čo bolo na vstupe v minulosti. Táto sieť dokáže prijímať dáta s časovým kontextom. Ide o sekvenčné dáta typu vývoja cien na burze, video, spracovanie sekvencií znakov alebo slov. Model jednoduchej neurónovej siete zloženej z jedného rekurentného neurónu je zobrazený na obrázku 3.3. Tento typ neurónových sietí má schopnosť pamätať si informácie. Pomáha pri spracovaní reči, ručne písaného textu alebo pri rozpoznávaní objektov v obraze (image captioning). Rekurentná sieť sa používa aj pri tvorbe chatbotov, kedy je potrebné si pamätať kontext a myšlienku vstupných viet. Ich nevýhodou je problém miznúceho gradientu (vanishing gradient). Tento problém spočíva v tom, že gradient má pri prechode cez vrstvy príliš veľkú alebo príliš malú hodnotu, nato aby mohlo byť trénovanie efektívne. Táto hodnota sa ešte stále zväčšuje alebo zmenšuje pri prechode ďalšími vrstvami. Pri spracovaní textu sa to prejavuje tým, že si sieť dokáže zapamätať kontext a potrebné informácie len pre krátke sekvencie. Pri dlhších sekvenciách kedy potrebujeme pri predikcii informáciu ktorá bola úplne na začiatku, môže byť už stratená. Sieť údaje, ktoré spracovávala dávnejšie (long-term dependencies), rýchlo zabudne [11].

LSTM (Long Short Term Memory) neurónová sieť je neurónová sieť, ktorá pozostáva z LSTM neurónov (obrázok 3.4). LSTM neuróny sú špeciálnym druhom rekurentných neurónov, ktoré sú schopné pracovať s akokoľvek minulým kontextom. Sú riešením problému miznúceho gradientu. LSTM neurón pozostáva zo vstupu, výstupu, vnútorného stavu (cell state), a troch brán - input gate, output gate a forget gate, pričom každá z nich má svoju vlastnú váhu. Input gate ovplyvňuje šírenie klasického vstupu do bloku, forget gate ovplyvňuje zapamätaný skrytý stav i vstup a output gate ovplyvňuje skrytý stav neurónu a teda aj výstup. Vďaka týmto bránam LSTM dokáže rozlíšiť, ktorá informácia je podstatná a treba si ju zapamätať a ktorú informáciu treba zabudnúť. Jej vnútorný stav reprezentuje všetky predchádzajúce relevantné informácie. LSTM neuróny sa používajú často pri spracovaní prirodzeného jazyka. Množstvo generatívnych chatbotov je založených práve na LSTM neurónových sieťach [17].



Obrázek 3.3: Model jednoduchej neurónovej siete zloženej z jedného rekurentného neurónu. Zdroj: [28]



Obrázek 3.4: LSTM neurón. Zdroj: [27]

3.1.2 Výhody a nevýhody neurónových sietí

- Výhody
  - Paralelné spracovávanie informácií

  - Nevyžadujú sa akékoľvek informácie o štruktúre dát, ktoré sa spracovávajú
  - Možnosť prispôsobenia sa na zmenu parametrov (pokiaľ sa aplikujú aj s učiacim algoritmom)
  - Rýchlosť
  - Schopnosť abstrahovať riadiace pravidlá iného regulátora (napríklad človeka) a nahradiť ich
  - Redukcia dát do menej rozmerného priestoru
  - Univerzálnosť - schopnosť aproximovať akúkoľvek spojitú funkciu s ľubovoľnou presnosťou

- Nevýhody
  - Nie sú vypracované žiadne metodiky pre návrh architektúry siete a voľbu funkcií pre jednotlivé neuróny. Pri implementácii sa používa metóda pokus-omyl. Tým sa zvyšuje časová náročnosť riešenia.
  - Nie sú vhodné pre systémy vyžadujúce presné riešenie
  - Učenie trvá zvyčajne dlho

3.2 Nástroje na prácu s neurónovými sieťami

Existuje veľa nástrojov, slúžiacich na vytváranie a trénovanie modelov založených na neurónových sieťach. Najznámejšie frameworky pre prácu s neurónovými sieťami vo výskume, ale aj v komerčnom nasadení sú Tensorflow, Keras, Caffe, Theano či Pytorch.

3.2.1 TensorFlow

TensorFlow je knižnica od spoločnosti Google. Vychádza z knižnice DistBelief V2, ktorá bola použitá v projekte Google Brain. Od roku 2015 sa nachádza na GitHub ako open-source. TensorFlow poskytuje nástroj na vizualizáciu modelovania sietí a ich výkonnosti (TensorBoard) a nástroj na uľahčenie nasadenia nových algoritmov pri zachovaní pôvodnej serverovej architektúry a rozhraní API (TensorFlow Serving). Táto knižnica podporuje jazyky Python, C++, JAVA, GO, R a Haskell. TensorFlow je podporovaný aj v prostredí Google a Amazon Cloud Enviroment. Natrénované modely môžu byť bez problémov nasadené aj na rôzne servery a mobilné zariadenia bez nutnosti inštalácie samostatného modelového dekodéru alebo interpretu Pythonu. Jeho výhodou je veľká komunita, používajúca tento nástroj, rýchlosť a možnosť trénovania na GPU. Jeho nevýhodou je zložitejšia implementácia modelov.

3.2.2 Keras

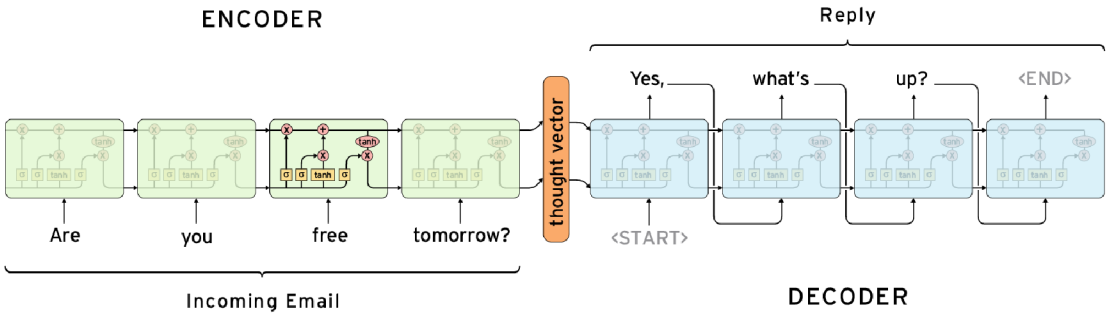
Keras je knižnica určená pre prácu s neurónovými sieťami, ktorá je napísaná v jazyku Python. Je navrhnutá modulárne, čo umožňuje jednoduché pridávanie nových prvkov, vrstiev a operácií. Keras je postavený nad knižnicou Theano, alebo TensorFlow. Dovoľuje veľmi jednoduché definovanie modelov, čo umožňuje rýchle prototypovanie modelov pre rôzne experimenty s neurónovými sieťami. Táto knižnica poskytuje 2 typy modelov: *Sequential* a *Graph*, ktoré sa tvoria postupným pridávaním vrstiev. Jeho výhodou je, že obsahuje veľké množstvo najčastejšie používaných modelov, vrstiev, objektov a aktivačných funkcií. Tvorba modelov v tomto prostredí je preto intuitívna a rýchla.

Pre implementáciu svojho chatbota som si vybral framework Keras, kvôli jeho jednoduchému používaniu, prehľadnej dokumentácii a veľkej komunite pracujúcej s týmto nástrojom. Využil som tiež mnoho vrstiev, ktoré tento framework ponúka, ako napríklad Bidirectional LSTM alebo embedding vrstvu, ktoré som nemusel nanovo implementovať. Ako backend ku Kerasu som použil knižnicu Tensorflow. Keďže GPU na mojom počítači nebolo podporované týmito frameworkmi, všetky modely som trénoval na CPU.

3.3 Model Sequence to Sequence

Sequence to sequence je model, pri ktorom sa používa enkodér-dekodér architektúra, pričom enkodér aj dekodér sú 2 rekurentné neurónové siete. Úlohou enkodéru je zakódovať vstupnú sekvenciu dát premenlivej dĺžky na vektor fixnej dĺžky. Dekóder potom na základe tohto vektora generuje výstupnú sekvenciu dát rôznej dĺžky. Tieto dve neurónové siete sú spojené do jedného modelu, aby sa maximalizoval efekt učenia. Keďže tento model používa rekurentné neurónové siete ktoré dokážu pracovať s kontextom, je tento model veľmi efektívny pri riešení problémov spracovania prirodzeného jazyka. Sequence to Sequence sa častokrát používa pri strojovom preklade, rozpoznávaní reči, alebo v dialógových systémoch [20].

Generatívny chatboti používajúci tento model vykazujú veľmi dobrú schopnosť učenia sa, pochopenia kontextu konverzácie a generovania odpovedí. Majú však problém pamätať si relevantné informácie, ktoré boli spomenuté dávnejšie v konverzácii. Je to spôsobené rekurentnou neurónovou sieťou, trpiacou problémom miznúceho gradientu, ktorý bol spomínaný v kapitole o neurónových sieťach. Tento problém je častokrát riešený použitím LSTM neurónov, ktoré sa vyznačujú veľmi dobrou schopnosťou pamätať si minulé stavy.



Obrázek 3.5: LSTM enkodér-dekodér model. Zdroj: [26]

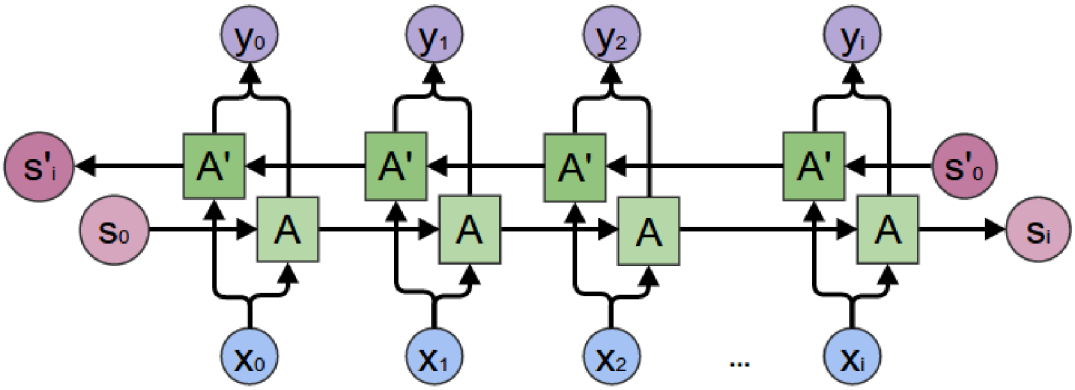
Obrázok 3.5 znázorňuje model jednoduchej enkodér-dekodér neurónovej siete. Enkóder spracováva sekvenciu slov, ktorá je pomocou metódy tokenizing a padding pretransformovaná na sekvenciu tokenov fixnej dĺžky a následne je každý token pomocou metódy word embedding namapovaný na vektor. Táto LSTM sieť spracováva vždy jedno slovo (v podobe vektoru) za jednotku času, pričom výsledný stav závisí od aktuálne spracovávaného slova a od minulého stavu. Výstup enkodéru (konečný stav) po spracovaní vety sa nazýva context vector, alebo trought vektor. Je to vektor, ktorý v sebe nesie kontext (hlavné informácie) vety. Vstupom dekodéru je trought vector, a špeciálny token *GO* alebo *START*, ktorý charakterizuje začiatok generovanej vety. Dekóder je zložený opäť z LSTM neurónovej siete, pričom za každú jednotku času vygeneruje jedno nové slovo. Vygenerované slovo závisí od predchádzajúceho stavu dekodéra a od minulého vygenerovaného slova. Sieť generuje

nové slová kým sa na výstupe dekodéra neobjaví špeciálny token *END*, alebo kým dĺžka vygenerovanej sekvencie slov neprekročí maximálnu dĺžku generovaných viet [6].

3.3.1 Bidirectional LSTMs

Bidirectional LSTM je rozšírenie LSTM enkodéra, ktoré budem testovať v jednom z mojich experimentov v šiestej kapitole. Používa sa v prípadoch, kedy je dostupná celá vstupná sekvencia tokenov predtým, ako je odoslaná do enkodéru. Má veľké uplatnenie pri problémoch strojového prekladu, rozpoznávania reči a spracovania textu.

V prípade použitia Bidirectional LSTM (obrázok 3.6) v architektúre enkodér-dekodér, enkodér obsahuje 2 LSTM neurónové vrstvy namiesto jednej. Prvá spracováva vstupnú sekvenciu tak, ako je odoslaná na vstup enkodéra a druhá spracováva jej obrátenú kópiu. Vnútorný stav obidvoch vrstiev je potom pri každom kroku spracovávania tokenu zo vstupnej sekvencie konkatenovaný do jedného výsledného stavu. To dopomáha k tomu, že v každom kroku má enkodér informáciu, aké slová sa nachádzajú aj pred aj za aktuálnym slovom. Výsledkom toho je, že sa sieť učí rýchlejšie a komplexnejšie [21].



Obrázek 3.6: Bidirectional LSTMs. Zdroj: [25]

Kapitola 4

Návrh a testovanie generatívneho chatbota

V tejto práci som sa rozhodol vytvoriť jednoduchého generatívneho chatbota, ktorý bude zvládať reagovať na jednoduché vety a otázky, prípadne by mal vygenerovať nejaké nové odpovede, ktoré sa nenachádzajú v datasete. Pri návrhu tohto chatbota si musíme zvoliť architektúru neurónovej siete, na akej sade dát túto sieť natrénujeme, ako ho budeme testovať a vyhodnocovať. Tiež je potrebné si určiť ako budú reprezentované jednotlivé vety, ktoré budú na vstupe a výstupe neurónovej siete.

4.1 Dataset

Dataset je súbor dát, na ktorých je chatbot trénovaný a testovaný. Kvalita a veľkosť datasetu výrazne vplývajú na výsledky chatbota. Ako testovacie a trénovacie dáta som použil súbor konverzácií z programu ChatterBot a Cornell Movie-Dialogs Corpus. Finálny dataset ktorý som vytvoril spracovaním dát z týchto dvoch corpusov obsahoval približne 222 000 párov otázok a odpovedí. To, ktoré vety použije chatbot z datasetu na trénovanie a testovanie závisí od vopred definovanej maximálnej dĺžky viet, s ktorými má pracovať. Chatbot si potom ako trénovacie a testovacie dáta vyberá z datasetu len tie, ktoré sú kratšie alebo rovnako dlhé ako daná maximálna dĺžka.

Dataset je potrebné rozdeliť na trénovacie a testovacie dáta. Pomocou testovacích dát je potom možné určovať kedy v priebehu trénovania došlo k pretrénovaniu siete (overfitting) a zhoršeniu generalizácie. V mojej implementácii rozdeľujem dataset na 3 časti. 80% párov otázok a odpovedí slúži na trénovanie siete. Na 10% dátach je sieť testovaná frameworkom Keras po každej epoche trénovania a 10% je použitých na ohodnotenie kvality odpovedí po každej epoche BLEU metrikou.

4.2 Tokenizácia

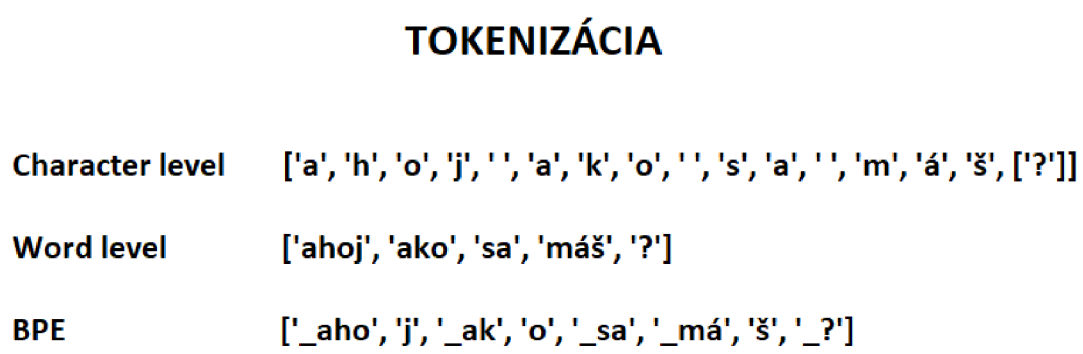
Tokenizácia je metóda, ktorá sa využíva v spracovávaní prirodzeného jazyka na lepšiu manipuláciu s vetami a ich syntaxou. Pomocou tejto metódy dokážeme previezť vetu do poľa tokenov, ktoré sa neskôr budú transformovať na vstup do neurónovej siete v podobe vektorov. Natural Language Toolkit (NLTK) je vedúca platforma pre jazyk Python, pozostávajúca z knižníc, ktoré umožňujú prácu s textom, ako je tokenizácia, klasifikácia, párovanie, tago-

vanie a sémantická analýza. Poskytuje množstvo datasetov, ktoré sú používané pri výskume spracovania prirodzeného jazyka.

Pri vytváraní chatbota si musíme zvážiť čo chceme reprezentovať jednotlivými tokenmi. Tokenizácia na úrovní písmen zaručuje nízky počet možných tokenov a tým aj menšie výpočtové zaťaženie - sieť pracuje s relatívne malými vektormi. Výhodou je taktiež to, že akékoľvek slovo dokážeme vyjadriť sériou tokenov. Nevýhodou je zas to, že tokeny predstavujúce písmená nenesú žiadnu sémantickú informáciu o danej vete a preto sa sieť učí pomaly a má problém s generalizovaním.

Pri tokenizácii na úrovni slov každý token predstavuje samostatné slovo, čo pomáha neurónovej sieti ľahšie rozlíšiť význam jednotlivých viet. Neurónová sieť sa dokáže rýchlo učiť a generalizovať. Problém nastáva v prípade, že chatbot pracuje s príliš veľkou slovnou zásobou. Sieť vtedy musí počítať s veľkými vektormi a pomaly sa trénuje. Taktiež nedokáže spracovať slová ktoré nemá v slovníku, alebo slová v ktorých nastal preklep. Tento problém je riešený zavedením špeciálneho tokenu *UNK*, ktorým sú nahradené všetky neznáme slová. Stráca sa však sémantika týchto slov.

BPE (Byte pair encoding) je metóda, pri ktorej sa text považuje za sekvenciu symbolov, pričom sa iteratívne združuje najčastejšie sa vyskytujúci pár symbolov do nového symbolu. Takto vznikajú časti slov, ktoré sú potom považované za samostatné tokeny. Výhoda tejto metódy je, že si môžeme určiť aký veľký slovník tokenov požadujeme, a iteratívne párovanie najčastejšie sa vyskytujúcich symbolov bude prebiehať, len kým sa nedosiahne požadovaná veľkosť slovníka. Pri použití malého slovníka budú tokeny predstavovať jednotlivé znaky v slove. Pri použití veľmi veľkého slovníka budú tokeny predstavovať najčastejšie sa vyskytujúce slová, či slovné spojenia. Ďalšou výhodou je, že pomocou tejto metódy je možné akékoľvek slová pretransformovať na sekvenciu tokenov - stráca sa potreba používať token *UNK*. Pri dobre zvolenej dĺžke požadovaného slovníka sú jednotlivé tokeny aj nositeľmi významu, dokonca si pomocou susedných tokenov sieť dokáže vyvodiť význam slov, ktoré v pôvodnom datasete neboli [22]. Na obrázku 4.1 je znázornený rozdiel medzi tokenizáciou na úrovni písmen, slov a tokenizáciou pomocou BPE.



Obrázek 4.1: Tokenizácia vety 'Ahoj ako sa máš?', pomocou word level tokenizing, character level tokenizing a byte pair encoding

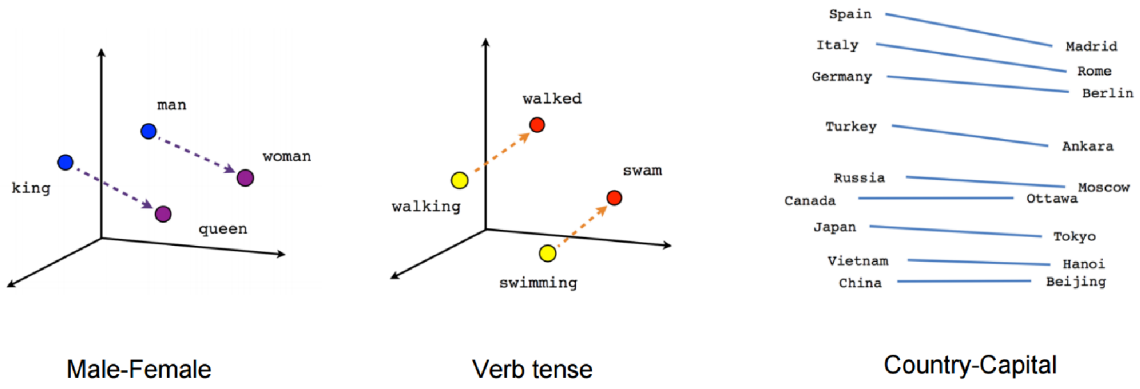
4.3 Padding

Pre neurónovú sieť je potrebné definovať veľkosť vstupného vektora, čiže počet tokenov vo vete. Problém nastáva v prípade, že vety v datasete sú rôznej dĺžky, a teda nemôžeme definovať fixný počet tokenov, ktoré budú kódované na vstupný vektor. Padding je metóda, kedy je dĺžka viet, ktorými je sieť trénovaná, prispôsobovaná na vopred daný počet tokenov. Ak je daná veta dlhšia ako požadovaná dĺžka, vetu osekáme. Ak je kratšia, doplníme na koniec vety špeciálne tokeny *PAD*, ktoré slúžia iba na zarovnanie vety na potrebnú dĺžku a nenesú žiaden význam. V praxi sa taktiež používajú ďalšie špeciálne tokeny ako sú *GO* (začiatok vety), a *EOS* (koniec vety) [20]. Zistilo sa, že neurónová sieť, určená na spracovanie viet, pochopí kontext vety lepšie, ak sú slová vo vstupnej vete v opačnom poradí. Príklad použitia paddingu na 10 tokenov môžeme vidieť nižšie:

Q : [PAD, PAD, PAD, PAD, PAD, PAD, “?”, “you”, “are”, “How”]
A : [GO, “I”, “am”, “fine”, “.”, EOS, PAD, PAD, PAD, PAD,]

4.4 Word embedding

Word embedding je technika, pomocou ktorej je možné reprezentovať slovo pomocou n-rozmerného vektoru. Každé slovo si tak vieme predstaviť ako bod v n-rozmernom priestore, pričom slová s podobným významom sú umiestnené blízko seba. Na obrázku 4.2 je znázornený word embeddingu v trojrozmernom priestore. Touto metódou je možné zachytiť sémantický význam slov, a vzťahy medzi nimi. Pri použití kódovania one-hot encoding potrebujeme vektor o veľkosti slovnej zásoby, ktorú používame. Pri veľkých datasetoch to môže byť niekoľko tisíc dimenzionálny vektor. Použitím embeddingu namiesto kódovania one-hot encoding sa znižuje veľkosť vstupného vektora s ktorým sieť ďalej pracuje. Pomocou operácii s týmito vektormi potom dokážeme získavať k danému slovu synonymá, antonymá a dokonca s nimi môžeme uskutočňovať výpočty. Napríklad: "king - man + woman = queen". Existuje niekoľko techník, ktoré sa používajú pri vytváraní modelu pre word embedding.

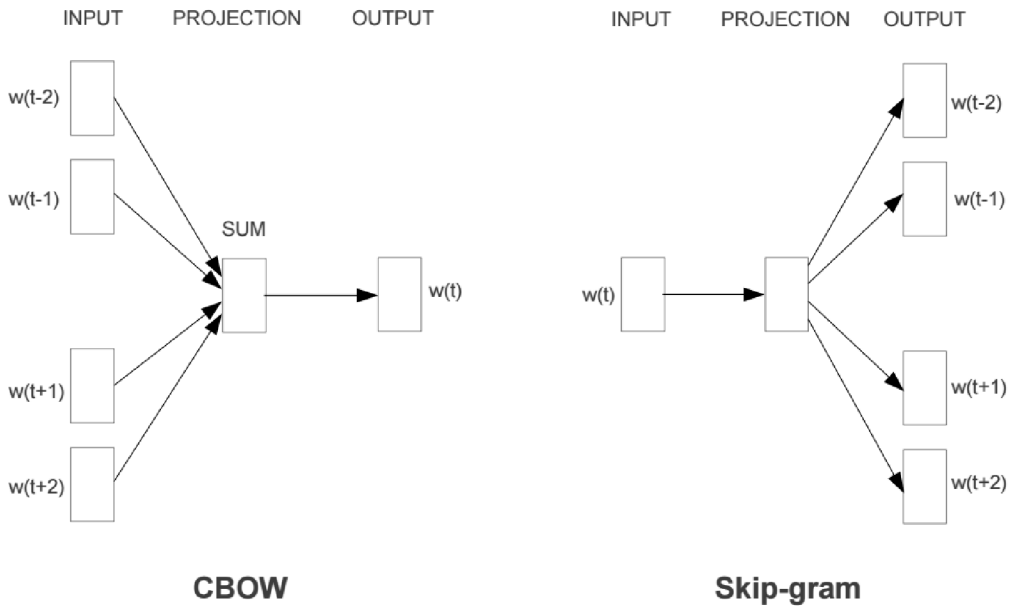


Obrázek 4.2: Word embedding. Zdroj: [30]

Na natrénovanie word embeddingu môže byť použitá **samostatná embedding vrstva**, ktorá sa nachádza na začiatku neurónovej siete a ktorá je trénovaná spolu s celým modelom na určitej úlohe. Touto úlohou môže byť napríklad spracovanie prirodzeného jazyka, klasifi-

kácia dokumentov alebo modelovanie jazyka. Pri použití tejto metódy je potrebné, aby bol vstupný text vyčistený od interpunkčných znamienok a aby jednotlivé slová boli vkladané do siete v podobe čísel, predstavujúcich indexy slov v slovnej zásobe. Veľkosť vektorového priestoru ktorý sa pred začatím trénovania určí pre túto embedding vrstvu, bude určovať veľkosť vektorov, ktorými budú reprezentované jednotlivé slová a s ktorými bude sieť ďalej pracovať. Táto veľkosť sa určuje experimentálne, zvyčajne sa pohybuje okolo hodnôt 50, 100 alebo 300. Použitie samostatnej embedding vrstvy vyžaduje veľké množstvo trénovacích dát a nevýhodou je aj pomalé trénovanie modelu. Výhodou je, že sa embedding natrénuje na rovnakej úlohe a rovnakých dátach s akými bude potom pracovať celá sieť.

Druhou z techník pre word embedding je **Word2Vec**. Bola vyvinutá Tomasom Mikolovom z firmy Google v roku 2013 [16]. Bola vytvorená so zámerom zefektívniť prácu s neurónovými sieťami, ktoré používajú word embedding. Stala sa štandardom pri práci s predtrénovaným embeddingom. Word2Vec umožňuje taktiež výpočty so sémantickým významom slov: "brought – bring + seek = sought". Pamätá si aj hlavné mestá štátov: "paris – france + poland = warsaw", či vzťah medzi mužským a ženským rodom podstatných mien. Pri Word2Vec rozlišujeme 2 rôzne modely. Continuous Bag-of-Words model (CBOW), ktorý sa učí predpovedať aktuálne slovo na základe kontextu okolitých slov a Continuous skip-gram model, ktorý zase predpovedá okolité slová na základe aktuálneho slova. Rozdiel medzi týmito modelmi je znázornený na obrázku 4.3. Obidva modely sa učia na základe kontextu okolitých slov, pričom počet okolitých slov ktoré sa berú ako kontext je konfigurovateľný. Word2Vec je veľmi efektívny pri trénovaní word embeddingu na obrovských corpusoch (niekoľko biliónov slov) a umožňuje nakonfigurovať dimenzionalitu výsledného embeddingu [10].



Obrázek 4.3: Rozdiel medzi Continuous Bag-of-Words model a Continuous skip-gram model. Zdroj: [31]

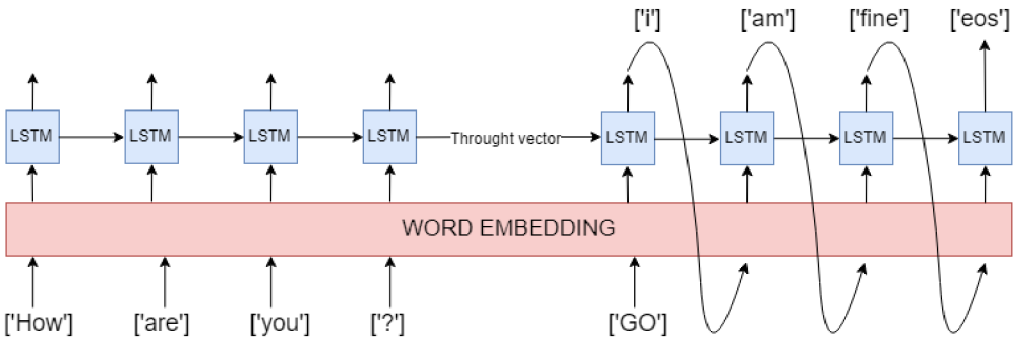
Treťou technikou je **GloVe** (The Global Vectors for Word Representation), ktorý je rozšírením Word2Vec. GloVe model sa učí transformovať slová na vektory vyhodnocovaním štatistík výskytu rovnakých skupín slov v corpuse [19]. Aj napriek tomu, že autori tejto metódy namerali v niektorých prípadoch lepšie výsledky ako pri metóde Word2Vec (až o 11%), sú tieto výsledky spochybňované. Podľa Y. Goldberga je zlepšenie oproti Word2Vec len 2-3% a podľa R.Řehůnku (tvorca knižnice gensim) dosahuje Word2Vec stále lepšie výsledky ako GloVe [32].

Gensim je voľne šíriteľná knižnica v jazyku Python, ktorá je určená na sémantické spracovanie textov. Dokáže pracovať s akýmkoľvek corpusom, z ktorého si dokáže vytvárať sémantické modely. Gensim obsahuje modely Word2Vec, LSA a LDA a implementuje funkcie na vyhľadávanie významovo podobných slov v danom modeli a operácie s nimi. Word2Vec v knižnici Gensim je súbor modelov, ktoré používajú neurónové siete na získanie lingvistického kontextu slov z textu a následnú reprezentáciu týchto slov v n-rozmernom priestore, pričom pri dostatočne veľkom corpuse môže mať tento priestor niekoľko stoviek dimenzií. Taktiež obsahuje možnosť použitia už predtrénovaného word embeddingu.

4.5 Teacher forcing

Pri trénovaní známeho modelu sequence to sequence, ktorý je spomenutý v kapitole o neurónových sieťach, je možné použiť metódu teacher forcing. Táto metóda nachádza uplatnenie hlavne pri rekurentných neurónových sieťach, ktoré používajú minulý výstup siete ako vstup pre aktuálny výpočet. Je často používaná v oblasti strojového prekladu, text summarization, a opisu obrázkov (image captioning).

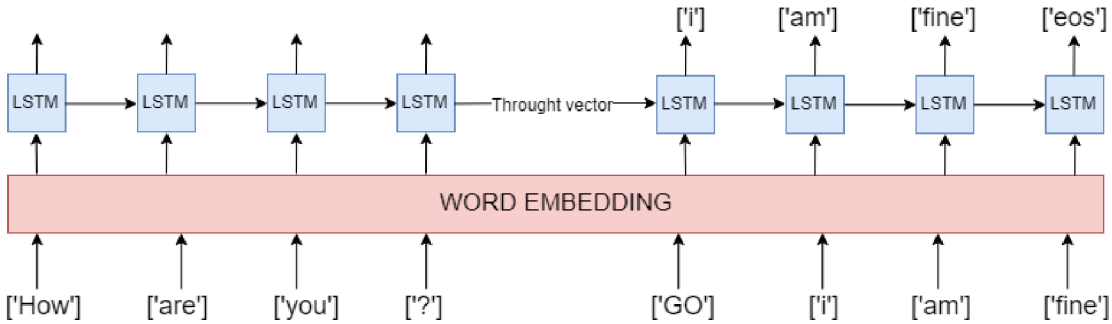
Enkóder-dekodér model ktorý je na obrázku 4.4 po natrénovaní funguje následovne. Vstupná veta v podobe tokenov namapovaných na vektor pomocou word embeddingu je odoslaná do LSTM enkodéru. Výstupom z enkodéru je výsledný vnútorný stav, trought vector. Tento vektor nesie v sebe informáciu o kontexte danej vety. Je to vlastne súbor vnútorných stavov enkodéra, počas spracovávania jednotlivých tokenov. Vstupom do dekodéra je výstupný vektor z enkodéra a špeciálny symbol *GO*. Tento symbol predstavuje začiatok procesu, kedy dekodér vygeneruje na výstupe slovo, a toto slovo bude zároveň vstupom do dekodéra v nasledujúcom timestampe. Dekóder teda na základe minulého vygenerovaného slova, a svojho minulého stavu generuje nové slovo. Tento proces sa opakuje, kým dekodér nevygeneruje špeciálny token *EOS*, alebo kým nie je prekročená maximálna dĺžka výstupnej vety.



Obrázek 4.4: Náčrt neurónovej enkodér-dekodér siete

Tento spôsob kedy sa používa minulý výstup dekodéra ako jeho aktuálny vstup, je možné použiť aj pri trénovaní siete. V mnohých prípadoch to však prináša problémy slabej konvergencie a pomalého trénovania. Metóda teacher forcing sa snaží týmto problémom predísť a urýchliť proces trénovania.

Pri použití teacher forcing v chatbotovi založenom na sequence to sequence architektúre implementovanej vo frameworku Keras musíme vytvoriť 3 neurónové siete. Prvá sieť 4.5 bude slúžiť na trénovaie chatbota. Bude zložená z LSTM enkodéra, ktorého vstup bude vektor fixnej dĺžky predstavujúci vstupnú vetu. Výstup enkodéra bude napojený na vstup LSTM dekodéra, pričom dekodér bude mať ešte jeden vstup - očakávanú odpoveď chatbota kde jednotlivé tokeny budú posunuté v čase o jeden timestamp neskôr, ako tokeny v očakávanej odpovedi. Táto sekvencia tokenov bude začínať symbolom *GO*. Dokopy bude teda tento model pri trénovaní vyžadovať 3 argumenty: vstupnú vetu, výstupnú vetu, a výstupnú vetu posunutú o jeden timestamp neskôr. Takto sa bude sieť učiť vždy predpovedať nasledujúce slovo len na základe minulého správne vygenerovaného slova. Nebude sa tak propagovať chyba, ktorá by vznikla už pri predikcii prvého zle vygenerovaného slova až na koniec generovania celej výstupnej vety.



Obrázek 4.5: Trénovací enkodér-dékóder model

Druhá neurónová sieť, takzvaný inference enkodér, bude samostatný LSTM enkodér, ktorý bude totožný s trénovacím enkodérom. Vstupom tohto enkodéra bude veta v podobe tokenov transformovaných pomocou word-embeddingu na n-dimenzionálny vektor. Výstupom bude trought vektor - vstupná veta zakódovaná do vektoru fixnej dĺžky.

Treťou neurónovou sieťou bude takzvaný inference dekodér. Bude to LSTM dekodér, ktorý bude totožný s dekodérom použitým pri tréningu. Vstupom tohto dekodéra bude trought vektor, a jeden token transformovaný na vektor pomocou word embeddingu. Výstupom bude jeden vygenerovaný token.

Inference decoder a inference encoder budú používané po natrénovaní chatbota. Veta od užívateľa v podobe série tokenov bude transformovaná pomocou word-embeddingu na vektor a bude slúžiť ako vstup do enkodéra. Enkóder vygeneruje trought vektor, ktorý bude vložený do dekodéra. Na vstup dekodéra bude taktiež vložený špeciálny symbol *GO*, ktorý predstavuje začiatok generovania ďalších slov. Dekóder následne vygeneruje token, predstavujúci prvé slovo vety vygenerovanej chatbotom. Toto slovo sa následne vloží na vstup dekodéra a ten opäť vygeneruje nový token. Tento proces sa opakuje kým dekodér nevygeneruje špeciálny symbol *EOS*, určujúci koniec vety, alebo kým sa počet slov vygenerovaných chatbotom nebude rovnať maximálnej dĺžke vety, ktorú si na začiatku zadefinujeme [7].

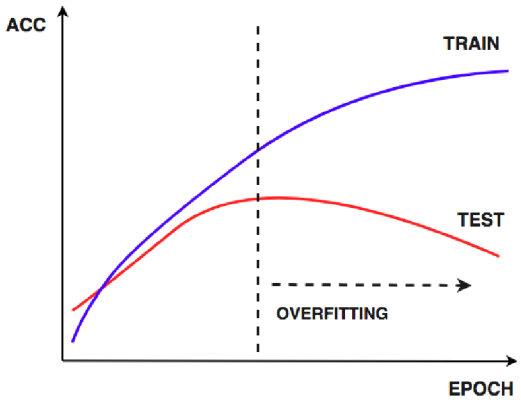
4.6 Testovanie a vyhodnocovanie

Dataset je rozdelený na trénovacie dáta (80%), testovacie dáta pre framework Keras (10%) a testovacie dáta pre BLEU skórovanie (10%). Pri trénovaní modelu je potrebné pre každú epochu trénovania otestovať jeho mieru úspešnosti na testovacích dátach. Toto robí Keras samostatne, pričom pre každú epochu vykazuje 4 hodnoty popisujúce priebeh trénovania. Sú nimi hodnota *ACC* (accuracy), *VAL_ACC* (validation accuracy), *LOSS* (error) a *VAL_LOSS* (validation error).

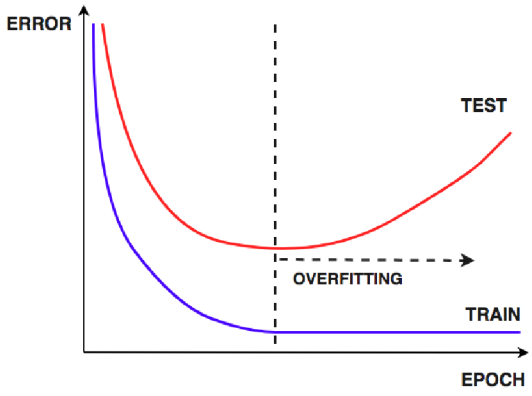
ACC a *VAL_ACC* určujú percentuálnu mieru úspešnosti siete na trénovacích a testovacích dátach. Porovnáva sa pritom každý vygenerovaný one-hot encoded token s one-hot encoded tokenom z referenčnej vety. Accuracy teda môžeme brať ako percentuálnu úspešnosť vygenerovania správneho slova na správnej pozícii vo vete.

LOSS a *VAL_ACC* určujú mieru chybovosti siete pri generovaní vety chatbotom na trénovacích a testovacích dátach. Pri trénovaní je snaha upraviť váhy siete tak, aby hodnota *LOSS* bola čo najmenšia.

Počas trénovania môže nastať prípad pretrénovania siete (overfitting), kedy si sieť začne memorovať trénovacie dáta a stráca sa pritom schopnosť generalizácie. Prejavuje sa to tak, že kým miera úspešnosti (accuracy) na trénovacích dátach počas jednotlivých epoch stúpa, miera úspešnosti na testovacích dátach začne klesať (validation accuracy). Pretrénovanie sa dá detekovať aj pomocou hodnoty *LOSS*, kedy sa miera chybovosti na trénovacích dátach znižuje, a miera chybovosti na testovacích dátach začne narastať. Príklad pretrénovania siete je zobrazený pomocou miery úspešnosti na grafe 4.6 a pomocou miery chybovosti na grafe 4.7.



Obrázek 4.6: Ilustračný obrázok pretrénovania siete. Graf zobrazuje mieru úspešnosti na trénovacích a testovacích dátach.



Obrázek 4.7: Ilustračný obrázok pretrénovania siete. Graf zobrazuje mieru chybovosti na trénovacích a testovacích dátach.

Tým, že na jendu otázku chatbota existuje množstvo potencionálne správnych a zmysluplných odpovedí, je pomerne zložité objektívne určiť, či chatbot disponuje schopnosťou generalizovať. Hodnoty *VAL_LOSS* a *VAL_ACC* zobrazujú len to, ako veľmi sa líši odpoveď vygenerovaná chatbotom od referenčnej odpovede, pričom ak *VAL_ACC* je 1, odpovede chatbota sú totožné s referenčnými odpoveďami. V prípade, že vstupná veta (otázka) bola napríklad "How are you?" a referenčná odpoveď chatbota "I am fine thanks" a chatbot vygeneroval pri testovaní odpoveď "Not very well", hodnota *VAL_ACC* bude rovná 0, aj keď chatbot odpovedal potencionálne správnou a zmysluplnou odpoveďou. Táto hodnota

bude tiež nízka v prípade poprehadzovaného slovosledu vo vygenerovanej vete. Pri trénovaní na veľmi veľkom datasete, by sa však určitá miera generalizovania mala prejaviť aj na hodnotách *VAL_LOSS* a *VAL_ACC* .

4.6.1 BLEU skórovanie

Jednou z najpoužívanejších skórovacích metrík pre vyhodnocovanie generovanej vety k referenčnej je BLEU metrika (Bilingual Evalu-ation Understudy). BLEU metrika bola predstavená Kishore Papienim v roku 2002. Táto metrika bola vytvorená kvôli potrebe automatického vyhodnocovania strojového prekladu, vzhľadom nato, že hodnotenie kvality prekladu pomocou ľudí je časovo príliš náročné. Hlavnou úlohou tejto metriky bolo zistiť, ako blízko je preklad vygenerovaný strojom k profesionálnemu prekladu. Výhodou BLEU je, že pri strojovom preklade úzko koreluje s manuálnym skórovaním pomocou ľudí.

Princíp BLEU spočíva v porovnávaní n-gramov vo vygenerovanej vete s n-gramami v referenčných vetách a počítaní zhody týchto n-gramov. 1-gram je pritom každý token, 2-gram je každý pár tokenov a n-gram je každá n-tica tokenov stojacich za sebou. Čím viac zhôd je nájdených, tým kvalitnejšia je vygenerovaná veta. Táto metrika berie do úvahy to, že jedna veta môže byť preložená viacerými možnými spôsobmi. Zameriava sa na výskyt rovnakých n-gramov v generovanej vete a v referenčných vetách bez ohľadu na ich pozíciu. Výhodou je aj to, že referenčných viet môže byť viac, a tým sa eliminuje individuálny štýl prekladateľa [18].

Pri BLEU skórovaní sa počíta takzvaná modifikovaná n-gram presnosť (modified n-gram precision). Táto hodnota sa počíta zo vzťahu:

$$P_n = \frac{\sum_{nGram \in C} Count_{clip}(nGram)}{\sum_{nGram \in C} Count(nGram)} \tag{4.1}$$

Kde P_n je modifikovaná n-gram presnosť, C je vygenerovaná veta a $Count_{clip}(nGram)$ je počet výskytov daného n-gramu v referenčnej vete. Ak sa daný n-gram vyskytuje vo viacerých referenčných vetách, berie sa počet výskytov z tej vety, v ktorej sa vyskytuje najčastejšie. $Count(nGram)$ je počet výskytov daného n-gramu vo vygenerovanej vete. Ak je modifikovaná n-gram presnosť rovná 1, znamená to, že sa vygenerovaná veta zhoduje s referenčnou.

V praxi sa najčastejšie používa kombinované BLEU skórovanie, kedy sa počíta modifikovaná n-gram presnosť pre 1-gramy až 4-gramy a výsledné ohodnotenie je ich priemer. Výpočet kombinovaného BLEU skórovania je možné zapísať vzorcom:

$$BLEU = BP.exp(\frac{1}{4}\sum_{n=1}^{4} log(P_n)) \tag{4.2}$$

Čím je dĺžka vygenerovanej vety kratšia, tým pravdepodobnejšie je, že všetky slová z vygenerovanej vety sa budú nachádzať v referenčných vetách a modifikovaná n-gram presnosť bude preto vysoká. Preto bola zavedená hodnota BP (brevity penalty), ktorá slúži

na penalizovanie systému za generovanie krátkych viet a závisí od dĺžky vygenerovanej vety c, a dĺžky referenčnej vety r. Je určená vzťahom:

$$BP = \begin{cases} 1 & \text{ak } c > r \\ e^{1-(\frac{r}{c})} & \text{ak } c \leq r \end{cases} \tag{4.3}$$

Navýhodou BLEU je to, že táto metrika nemusí vždy korelovať so subjektívnym hodnotením ľudí. Ide napríklad o prípad chatbotov, kedy na otázku užívateľa môže existovať veľa zmysluplných odpovedí vygenerovaných chatbotom, ktoré obsahujú úplne odlišné slová. Napríklad ak sa človek opýta chatbota "Are you happy?" a chatbot odpovie vetou "Yes, I am very happy", pričom referenčná odpoveď bola "No, I hate life", BLEU túto vetu ohodnotí veľmi nízkym skóre, pričom je táto veta zmysluplná a potenciálne správna. V odkazovanom článku [15] bolo preukázané, že žiadna zo súčasných, bežne používaných metrík na vyhodnocovanie chatbotov, v rátane BLEU, nekoreluje s vyhodnocovaním pomocou ľudí. V mojej práci ju však použijem pre ilustráciu, ako veľmi sa podobajú vety vygenerované chatbotom od viet v datasete. Pri veľkom množstve trénovacích a testovacích dát je pravdepodobné, že aspoň niektoré odpovede na testovacie otázky sa budú podobať odpovediam z datasetu a to by malo svedčiť o tom, že sa chatbot učí.

The Python Natural Language Toolkit (NLTK) obsahuje implementáciu BLEU skórovania, ktoré je možné použiť pri vyhodnocovaní generovaných viet. Vo všetkých experimentoch používam túto funkciu na počítanie kombinovaného BLEU (vzorec 4.2) na 10% viet z datasetu.

Kapitola 5

Implementácia

Moja implementácia chatbota zahŕňa súbor modulov, ktoré sú spúšťané jediným programom *chatbot.py*. Celá činnosť chatbota je ovládaná pomocou vstupných argumentov. Podrobnejšiu špecifikáciu a návod na ovládanie chatbota som uviedol v súbore *README*. V nasledujúcich podkapitolách sú popísané detailnejšie jednotlivé etapy, ktoré tvoria celkovú funkčnosť chatbota a návrhy na jeho budúce vylepšenia.

5.1 Vytvorenie datasetu

Prvým krokom k spusteniu chatbota je vygenerovanie finálneho datasetu, ktorý je tvorený konverzáciami z ChatterBot corpus a Conrell-movie dialogue corpus. ChatterBot corpus obsahuje prevažne často používané konverzácie z každodenného života. Cornell-movie dialogue corpus je tvorený niekoľkými súbormi, v ktorých sú uložené konverzácie extrahované z tituliek filmov. V súboroch sú uložené aj ďalšie metainformácie ako id osôb, žánre filmov, id konverzácií, IMDB hodnotenia atď. Parser ktorý vyexportuje konverzácie v tvare otázka-odpoveď z ChatterBot corpusu som vytvoril sám a parser pre Cornell-movie dialogue som prebral zo stránky github.com [1]. Finálny dataset obsahuje 221 952 párov otázok a odpovedí.

5.2 Príprava trénovacích dát

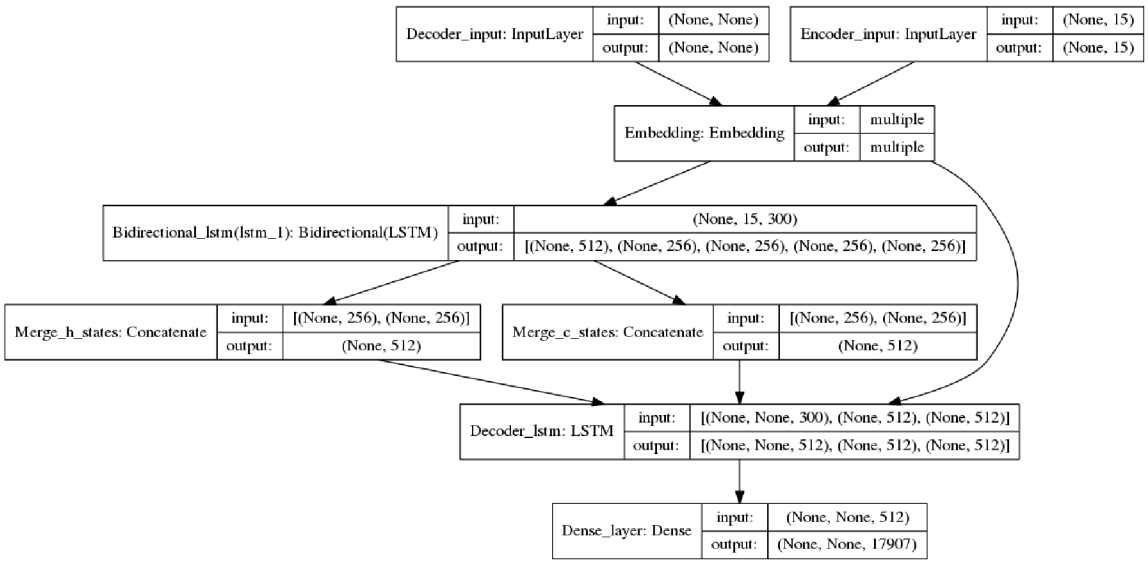
Príprava trénovacích dát zahrňuje tokenizáciu viet z datasetu, vytvorenie slovnej zásoby a následné vygenerovanie trénovacích dát v podobe vektorov, do súborov určených na trénovanie chatbota. Dataset je rozdelený na 3 časti. 80% viet je použitých na trénovanie siete, 10% na testovanie siete Kerasom počas trénovania a ďalších 10% na testovanie pomocou BLEU metriky. Trénovacie dáta sú rovno pripravené aj na trénovanie pomocou metódy teacher forcing. Tokenizácii predchádza preprocessing viet, kedy sú všetky veľké písmená zmenené na malé, sú odstránené nepovolené znaky a skrátené anglické výrazy ako *haven't* sú prepísané na dlhšiu verziu, v tomto prípade *have not*. Počas tokenizácie súčasne prebieha tvorba slovnej zásoby chatbota, kedy sa každému unikátnemu tokenu priradí jedno integer číslo, ktoré slúži ako jeho ID. Slovná zásoba sa potom uloží do samostatného JSON súboru, aby mohla byť neskôr použitá na mapovanie tokenov na čísla a opačne. Po vytvorení slovnej zásoby sa všetky tokeny vo vetách prepíšu na svoje ID čísla, pretože neurónová sieť prijíma na svojom vstupe číselné hodnoty.

[1] https://github.com/floydhub/textutil-preprocess-cornell-movie-corpus/blob/master/cornelldata.py

Pomocou vstupných argumentov je možné zvoliť si počet viet z datasetu na ktorých sa bude chatbot trénovať, maximálnu dĺžku vety na akú sa majú zarovnávať vety pomocou paddingu a maximálny počet viet, ktoré môžu byť v súbore trénovacích dát. Ak je počet viet z datasetu na ktorých sa má chatbot trénovať väčší ako maximálny počet viet v súbore trénovacích dát, vety v podobe vektorov sa uložia do viacerých trénovacích súborov. To je potrebné hlavne kvôli šetreniu pamäti na slabších počítačoch, aby sa nemuseli všetky trénovacie dáta udržovať súčasne v pamäti. Testovacie dáta sa tiež ukladajú do samostatných súborov. Pomocou osobitného argumentu je možné zvoliť si možnosť použitia metódy Byte pair encoding, kedy jednotlivé tokeny nebudú predstavovať slová, ale iba časti slov. BPE je realizované pomocou balíka sentencepiece od spoločnosti Google a predtrénovaného modelu obsahujúceho slovnú zásobu o veľkosti 10 000 možných tokenov. V prípade, že si užívateľ nezvolí BPE a tokenizácia bude prebiehať na úrovni slov, môže si užívateľ navoliť použitie predtrénovaného word-embeddingu od Googlu. Výsledný model neurónovej siete tak nebude mať osobitnú vrstvu pre embedding a tokeny sa budú transformovať na vektory pomocou embeddingu od Googlu už pri tokenizácii. Neznáme slová budú nahradené symbolom *UNK*

5.3 Trénovanie chatbota

Pri trénovaní je potrebné pomocou argumentov opäť určiť, či budeme používať predtrénovaný embedding od Googlu, BPE, alebo len embedding vrstvu zabudovanú do neurónovej siete. Taktiež je možné si zvoliť, či chceme použiť trénovaciu metódu teacher forcing z predchádzajúcej kapitoly. Voliteľný je aj počet epoch pri trénovaní a to, či chceme v enkodér-dekodér modely použiť bidirectional LSTM. Program si pri trénovaní sám dokáže zistiť koľko trénovacích súborov bolo vygenerovaných a sieť trénuje striedavo na všetkých. Kvôli lepšiemu priebehu trénovania sa pre každú epochu vyberajú trénovacie súbory v náhodnom poradí a vety v rámci jedného súboru sa tiež každou epochou náhodne poprehadzujú. Po prerušení trénovania a následnom dotrénovaní si môžeme určiť, či chceme trénovať odznovu, alebo či si program má zistiť kde skončil a pokračovať v tréningu. Počas trénovania sa v samostatnom priečinku ukladajú natrénované modely pre každú epochu, aby sa k nim dalo po tréningu vrátiť. Taktiež sa v priečinku *statistics* ukladajú štatistiky z trénovania (hodnoty *ACC*, *LOSS*, *VAL_ACC* , *VAL_LOSS*), z ktorých je neskôr možné samostatným skriptom vygenerovať grafy priebehu trénovania. V tomto priečinku sa tiež vždy pri začatí trénovania vygeneruje obrázok, v ktorom sú rozkreslené vrstvy aktuálne používaného modelu neurónovej siete a jeho vstupy a výstupy pre jednotlivé vrstvy. Na obrázku 4.5 je náčrt takto vygenerovaného obrázku trénovacieho enkodér-dekodér modelu, ktorý bol použitý aj pri testovaní na užívateľoch.



Obrázek 5.1: Trénovací enkodér-dékóder model, vygenerovaný Kerasom

5.4 Skórovanie chatbota

Osobitným argumentom sa spúšťa po natrénovaní skript, ktorý počíta BLEU ohodnotenie natrénovaných modelov pre jednotlivé epochy trénovania. Výsledné hodnoty sú potom ukladané do samostatného súboru v priečinku BLEU, kde sa nachádza aj skript na vygenerovanie grafu pre priebeh BLEU skórovania. BLEU skórovanie je počítané pomocou knižnice NLTK na 10% vetách vybraných z datasetu.

5.5 Konverzácia

Po natrénovaní má užívateľ možnosť komunikovať s chatbotom pomocou terminálu. Pre väčšiu kvalitu konverzácie disponuje chatbot sadou neutrálnych odpovedí, napríklad "What are you talking about?"alebo "What about to change the theme?". Z týchto odpovedí si náhodne vyberie jednu, ak sa vo vygenerovanej vete nachádzajú viac ako 2 tokeny *UNK*, alebo ak je súčin pravdepodobností pre jednotlivé vygenerované slová vo vete príliš nízky a použije ju namiesto aktuálne vygenerovanej odpovede. Ak užívateľ nenapíše nič dlhšie ako 25 sekúnd, chatbot vyberie jednu otázku z osobitnej sady otázok aby začal konverzáciu. Ďalším vylepšením je, že všetky čísla pri tokenizácii sú nahradené tokenom *NUMBER*. Neskôr, ak chatbot vygeneruje odpoveď, v ktorej sa bude nachádzať tento token, tak sa tento token nahradí náhodne vygenerovaným číslom. Komunikácia s chatbotom sa ukončuje príkazom *exit*.

Kapitola 6

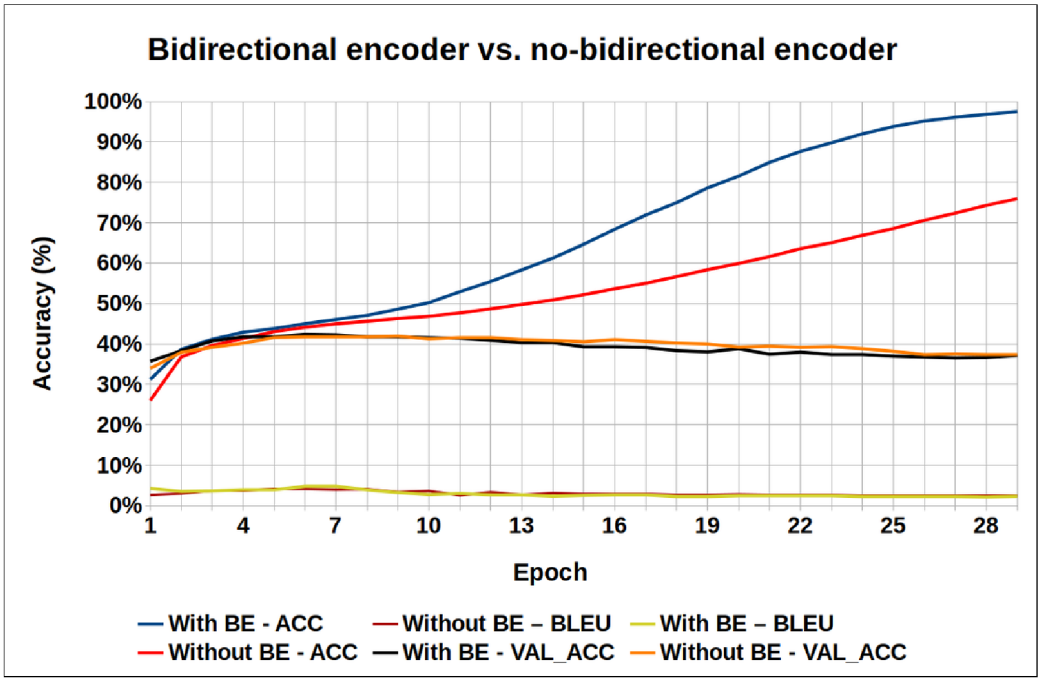
Experimenty

Chatbot je naprogramovaný tak, aby jeho funkčnosť mohla byť konfigurovateľná pomocou argumentov. Umožňuje použitie word-level tokenizácie ale aj tokenizáciu pomocou byte pair encoding. Užívateľ si môže zvoliť či chce použiť predtrénovaný embedding od Googlu, alebo osobitnú embedding vrstvu v neurónovej sieti, ktorá sa bude trénovať spolu s celou sieťou. Taktiež si môže zvoliť či chce používať obyčajný enkodér alebo bidirectional LSTM. Po nadefinovaní architektúry chatbota si môže vybrať, či chce použiť trénovanie pomocou metódy teacher forcing. V nasledujúcich kapitolách sú popísané experimenty, ktorých cieľom bolo nájdenie čo najlepšej kombinácie parametrov, pri ktorých sa chatbot bude trénovať čo najrýchlejšie a jeho odpovede budú čo najkvalitnejšie. Model ktorý sa zdal byť najlepší, bol potom použitý pri testovaní na užívateľoch.

6.1 Bidirectional enkodér

Bidirectional LSTM neurónové siete z 3. kapitoly sú rozšírením klasických LSTM neurónových sietí, ktoré môžu zlepšiť zakódovanie kontextu vstupnej vety enkodérom. Bidirectional LSTM sú dve LSTM vrstvy, pričom jedna spracováva vstupnú vetu, a druhá jej otočenú kópiu.

Cieľom prvého pokusu bolo overiť, aký bude mať vplyv na chatbota pridanie bidirectional LSTM do enkodéra. Zameral som sa na rýchlosť trénovania neurónovej siete, rýchlosť učenia sa a na výsledky BLEU metriky. Prvý model obsahoval jednoduchú LSTM vrstvu v enkodéri aj dekodéri, pričom počet neurónov bol nastavený na 256. Model bol trénovaný pomocou metódy teacher forcing. Word embedding bol realizovaný pomocou špeciálnej embedding vrstvy, ktorú ponúka framework Keras. Embedding sa trénoval spolu s celým modelom. Dimenzionalita embeddingu bola nastavená na 300 a model bol trénovaný na 10 000 vetách o veľkosti maximálne 10 tokenov. Druhý model bol taký istý ako prvý, len namiesto jednoduchej LSTM vrstvy obsahoval bidirectional lstm vrstvu, ktorú taktiež ponúka Keras. Chatbot bol v obidvoch prípadoch trénovaný 30 epoch. Graf znázorňujúci priebeh tréningu môžeme vidieť na obrázkou 6.1.



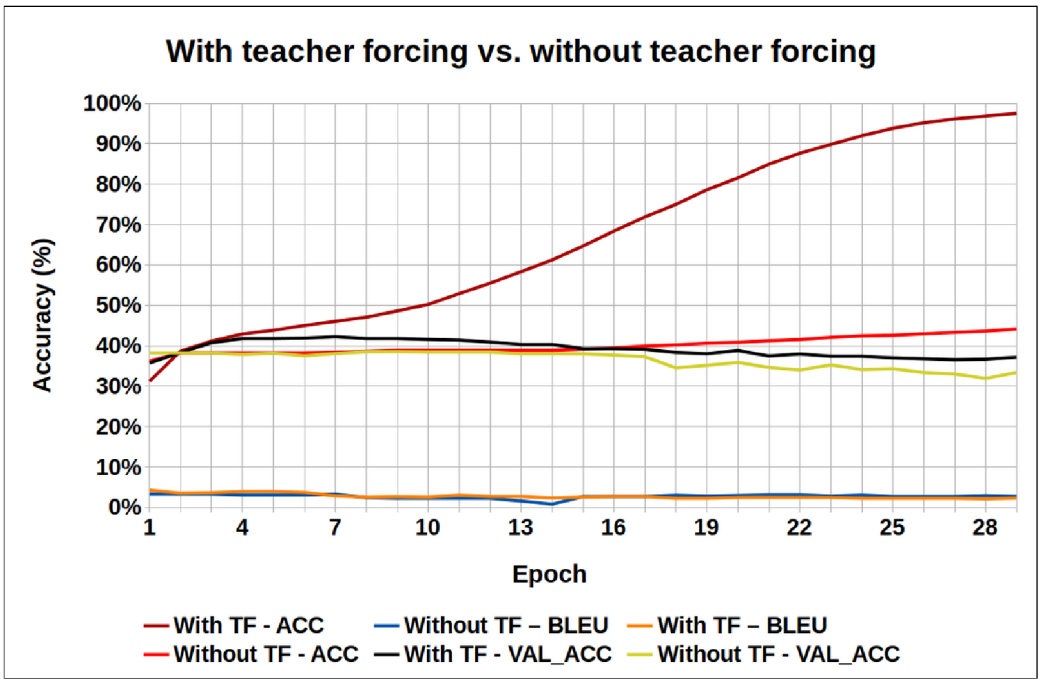
Obrázek 6.1: Bidirectional enkodér v porovnaní so simple LSTM enkodérom - miera úspešnosti generovania viet chatbotom počítaná na trénovacích a testovacích dátach a BLEU skórovanie

Zo získaného grafu môžeme vidieť, že pridaním bidirectional LSTM vrstvy sa výrazne zvýšila rýchlosť učenia. Dĺžka trvania jednej epochy pri trénovaní sa pritom zvýšila len zanedbateľne. Pretrénovanie modelu podľa grafu nastalo u obidvoch modeloch v siedmej epoche, kedy *VAL_ACC* dosiahla svoje maximum a potom začala klesať. Ohodnotenie pomocou BLEU pre jednotlivé epochy je u obidvoch modeloch takmer rovnaké, maximum dosiahli tiež v bode pretrénovania, v siedmej epoche. Tým, že sa počas trénovania počíta hodnota *VAL_ACC* ako miera úspešnosti, kedy sa kontroluje zhodnosť jednotlivých tokenov na jednotlivých pozíciách v sekvencii tokenov, zarátavajú sa do zhody aj špeciálne tokeny PAD. To spôsobuje, že hodnoty *ACC* a *VAL_ACC* sa pohybujú už od začiatku na 30% - 40% úspešnosti. Pri počítaní BLEU metriky sa počíta s vetami ktoré neobsahujú PAD symboly, preto sa BLEU ohodnotenia pohybujú na relatívne nízkej úrovni. Tvar krivky pre hodnoty *VAL_ACC* a krivky pre BLEU skórovanie sa však podobajú. V bode pretrénovania bol model, ktorý používal bidirectional enkodér úspešnejší o pár desatín percetna. Rýchlosť učenia sa použitím bidirectional enkodéru výrazne zvýšila. Hodnoty *VAL_ACC* a BLEU zobrazujú (ako aj v ďalších experimentoch) len mieru podobnosti viet vygenerovaných chatbotom k referenčným vetám z datasetu. Nie sú ukazateľom toho, ako vie chatbot generalizovať. Tento problém bol spomínaný v kapitole o BLEU skórovaní.

6.2 Teacher forcing

Cieľom druhého experimentu bolo overiť vplyv trénovania chatbota pomocou metódy teacher forcing, ktorá je popísaná v štvrtej kapitole. Porovnával som dva enkodér-dekodér modely, pričom jeden z nich bol trénovaný metódou teacher forcing a druhý nie. Obidva modely mali bidirectional LSTM vrstvu a osobitnú vrstvu pre word embedding ako

v predchádzajúcom experimente. Opäť bola nastavená dimenzionalita embeddingu na 300 dimenzií, 256 neurónov pre LSTM vrstvy a dĺžka viet bola obmedzená na 10 tokenov. Modely boli trénované na datasete o veľkosti 10 000 viet. Na obrázku 6.2 sa nachádza priebeh trénovania modelov a ich BLEU ohodnotenie pre každú epochu.



Obrázek 6.2: Trénovanie s metódou teacher forcing a bez - miera úspešnosti generovania viet chatbotom počítaná na trénovacích a testovacích dátach a BLEU skórovanie

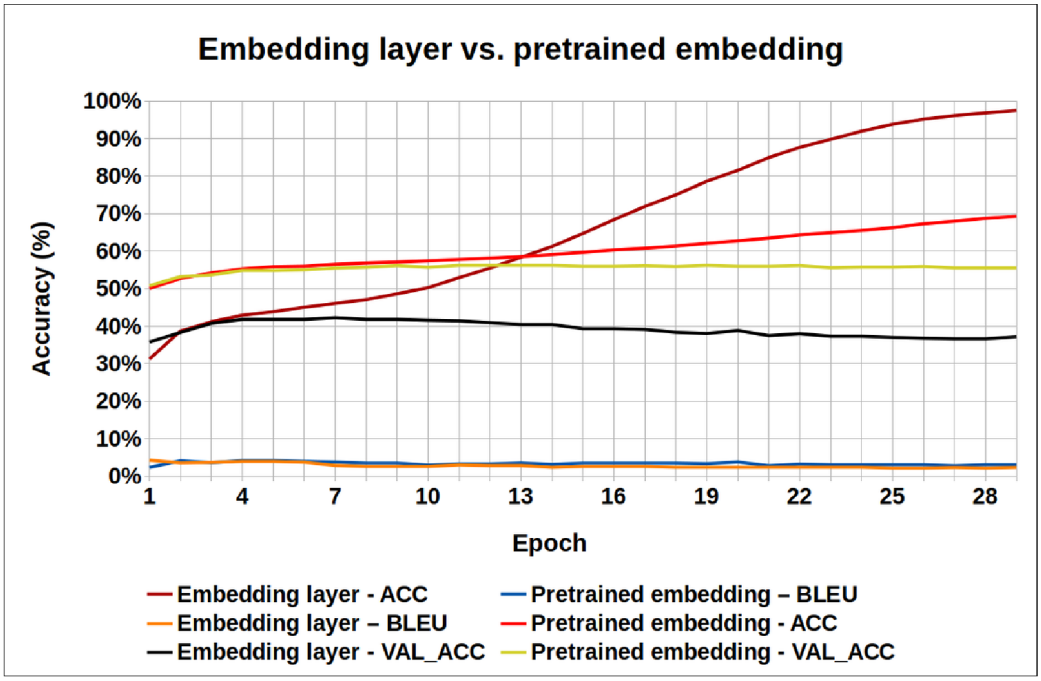
Metóda teacher forcing výrazne urýchlila trénovanie siete. Pretrénovanie siete nastalo v siedmej epoche ako v minulom experimente. Model ktorý nepoužíval metódu teacher forcing dosiahol fázu pretrénovania až v deviatej epoche, no stále na nižšej hodnote *VAL_ACC* ako model, ktorý túto metódu používal. Model ktorý bol trénovaný bez tejto metódy sa podarilo natrénovať, no potreboval až 100 epoch aby sa natrénoval na accuracy s hodnotou 0.9 (takmer 4-krát viac). BLEU skórovanie sa pohybovalo približne na rovnakej úrovni u obidvoch modeloch.

6.3 Google embedding

Ďalším experimentom bolo zistiť, aký vplyv má na trénovanie chatbota použitie predtrénovaného word embeddingu od Googlu, ktorý každé slovo mapuje na 300-dimenzionálny vektor. Transformovanie slov na vektory som robil pomocou knižnice gensim, hneď pri tokenizácii. Embedding bol takto úplne oddelený od neurónovej siete, a do enkodéra boli vkladané, už embedované slová. Neznáme slová, ktoré sa nenachádzali v slovnej zásobe použitého modelu od Googlu som nahradil špeciálnym symbolom *UNK*.

Obidva modely boli trénované pomocou metódy teacher forcing, ich enkodér obsahoval bidirectional LSTM s 256 neurónmi a dĺžka slov bola obmedzená na 10 tokenov. Druhý model obsahoval samostatnú embedding vrstvu, ktorá mapovala slová na 300-dimenzionálny

vektor. Obidva modely boli trénované na 10 000 vetách. V nasledujúcom grafe 6.3 je znázornený priebeh trénovania, a ohodnotenie jednotlivých modelov BLEU metrikou.



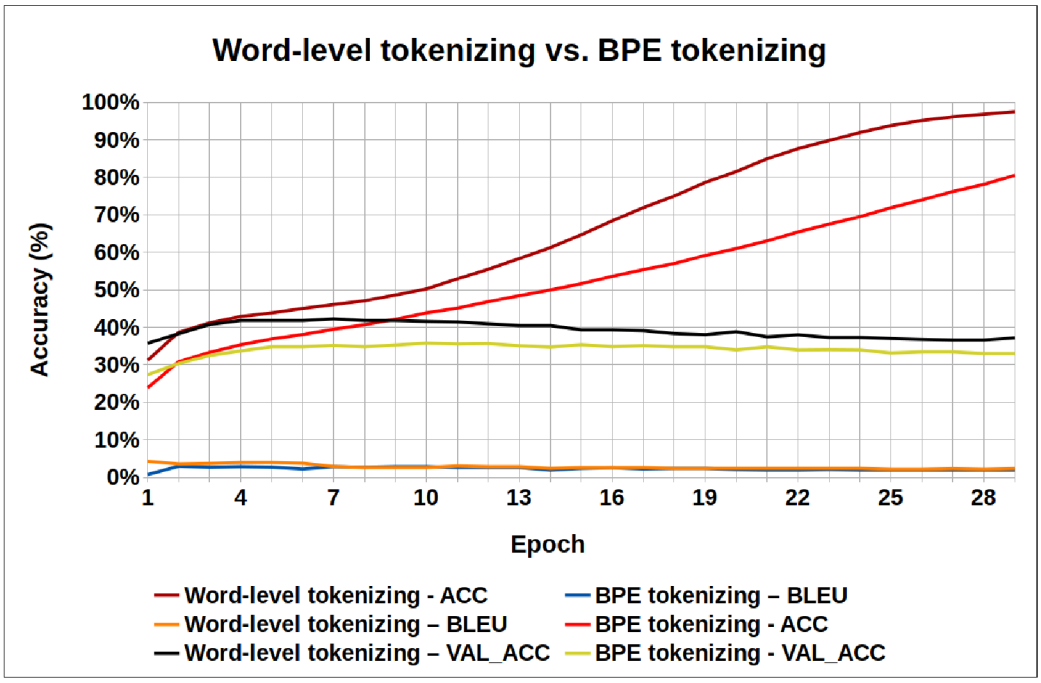
Obrázek 6.3: Osobitná embedding vrstva v porovnaní s predtrénovaným embeddingom od Googlu - miera úspešnosti generovania viet chatbotom počítaná na trénovacích a testovacích dátach a BLEU skórovanie

Použitie predtrénovaného embeddingu urýchlilo výpočet o pár sekúnd a hodnoty *ACC* aj *VAL_ACC* boli hneď od začiatku trénovania vššie ako pri modeli so samostatnou embedding vrstvou. BLEU ohodnotenie pre jednotlivé modely sa však nelíšilo. Po manuálnom otestovaní odpovedí chatbota sa ukázalo, že vysokú mieru úspešnosti zapríčiňuje špeciálny token *UNK*. Tým, že predtrénovaný model nedokázal transformovať veľa málo používaných slov (ktoré nemal vo svojej slovnej zásobe) na vektory, museli byť tieto slová nahradené symbolmi *UNK*. Pri vyhodnocovaní chatbota na testovacích dátach potom dochádzalo k častej zhode tohto symbolu v odpovediach chatbota a v referenčných odpovediach z datasetu. Slová ktoré boli najčastejšie nahrádzané symbolom *UNK* boli mená, predložky, spojky a citoslovcia. Počet *UNK* symbolov po tokenizácií datasetu s veľkosťou 10 000 viet bol približne 14 200.

Pri použití modelu s predtrénovaným embeddingom sa načítava celý model pomocou knižnice gensim do pamäte. Tento model zaberá 3.4GB. Kvôli menšiemu zaťaženiu pamäte som obmedzil načítanie tohto modelu na 1 000 000 najpoužívanejších slov. To mohlo viesť k tomu, že sa niektoré menej používané slová nepodarilo namapovať na vektor a museli byť nahradené symbolom *UNK*. Načítanie celého modelu by mohlo viesť k výraznejšiemu zlepšeniu kvality odpovedí chatbota.

6.4 Byte pair encoding

Cieľom posledného pokusu bolo zistiť, či sa zlepší kvalita odpovedí chatbota a rýchlosť trénovania, ak použijeme namiesto word-level tokenizácie byte pair encoding. Porovnával som dva totožné modely. Obidva modely obsahovali samostatnú vrstvu pre 300-dimenzionálny word embedding a bidirectional enkodér. Rozdiel bol v tom, že prvý model bol trénovaný na vetách, ktoré prešli word-level tokenizáciou (jedno slovo predstavovalo jeden token) a druhý model bol trénovaný na vetách, ktoré boli transformované na tokeny pomocou metódy byte pair encoding. Byte pair encoding bolo realizované pomocou balíka sentencepiece od spoločnosti Google a predtrénovaného modelu, ktorého slovná zásoba je 10 000 tokenov. Počet neurónov v LSTM vrstvách bol 256. Modely boli trénované opäť na dataste o veľkosti 10 000 slov. Kvôli tomu, že byte pair encoding kóduje jedno slovo na viacero tokenov, bola maximálna dĺžka vety navýšená na 20 tokenov. Model ktorý používal word-level tokenizing mal dĺžku vety obmedzenú len na 10 tokenov. Nižšie je zobrazený graf 6.4 kde je znázornený priebeh tréningu a BLEU ohodnotenie modelov pre každú epochu trénovania.



Obrázek 6.4: Word-level tokenizácia v porovnaní s tokenizáciou pomocou metódy byte pair encoding - miera úspešnosti generovania viet chatbotom počítaná na trénovacích a testovacích dátach a BLEU skórovanie

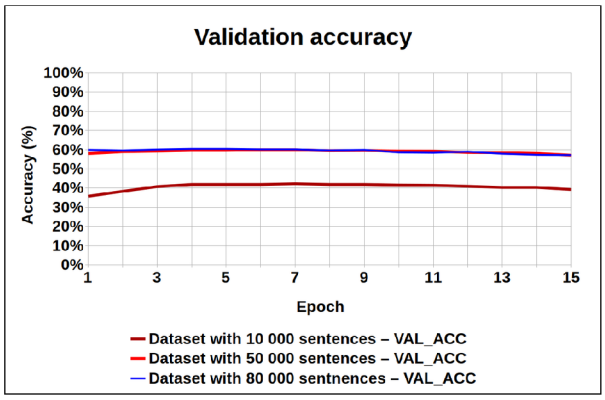
Z výsledkov tohto experimentu vyplýva, že priebeh trénovania sa pri obidvoch modeloch mierne líši. Počas celého trénovania boli hodnoty *ACC* a *VAL_ACC* pre model používajúci word-level tokenizing vyššie ako pre model používajúci BPE. Pretrénovanie pre model používajúci BPE nastalo až v 10. epoche, no hodnota *VAL_ACC* bola stále nižšia ako pri modely s word-level tokenizáciou. Pomalší priebeh tréningu pri použití BPE pripisujem zvýšenému maximálnemu počtu tokenov vo vete, a rozsekaním niektorých slov na viac tokenov. Sieti trvá pravdepodobne dlhšie pochopiť vzťahy medzi jednotlivými tokenmi. Výsledky z BLEU skórovania sa pri obidvoch modeloch výrazne nelíšia, BLEU ohodnotenie pre word-level tokenizing je počas celého trénovania len o pár desatín percenta lepšie ako

pri použití BPE. BPE však pomohlo ušetriť pamäť pri trénovaní, keďže je možné dopredu určiť, akú veľkosť slovnej zásoby potrebujeme. Rýchlosť trénovania sa o niekoľko sekúnd zhoršila, kvôli zvýšenej maximálnej dĺžke viet, s ktorou model pracoval. Pri trénovaní na veľmi veľkom datasete by však BPE bolo výhodnejšou voľbou. Vďaka relatívne malej slovnej zásobe by sieť pracovala s malými vektormi a výpočty by boli rýchlejšie a pamäťovo menej náročné.

6.5 Zhrnutie

Experimentmi v predchádzajúcich kapitolách som zistil, že ako najefektívnejší model pre chatbota sa javí enkodér-dekodér model, ktorého enkodér pozostáva z bidirectional LSTM vrstvy a ktorý je trénovaný metódou teacher forcing. Pre word embedding som sa rozhodol použiť samostatnú embedding vrstvu, pretože predtrénovaný word embedding model od Googlu má problém transformovať niektoré predložky, častice, citoslovcia a menej používané slová na vektor a tieto slová musia byť nahrádzané špeciálnym symbolom *UNK*. Pri použití modelu od Googlu s väčšou slovnou zásobou by sa tento problém mohol minimalizovať, no malo by to dopad na väčšie zaťaženie pamäte, ktoré by stroj na ktorom som modely trénoval nezvládol. Priebeh trénovania pri použití modelu s tokenizáciou pomocou BPE sa oproti modelu s word-level tokenizáciou nezlepšil. Bolo to pravdepodobne spôsobené väčšou maximálnou dĺžkou viet a rozdelením slov na viacero tokenov. Sieť sa pomalšie učila vzťahy medzi jednotlivými tokenmi. Pri pracovaní s veľkými datasetmi by použitie BPE pravdepodobne pomohlo urýchliť výpočty, ušetriť pamäť a na veľkom datasete by sa dokázala sieť lepšie naučiť vzťahy medzi jednotlivými tokenmi, z ktorých by boli zložené slová.

Pre natrénovanie chatbota ktorý by dokázal odpovedať zmysluplnými vetami je potrebné sieť natrénovať na veľkom množstve dát. So zväčšovaním datasetu ktorý používa chatbot narastá tiež čas potrebný na jednu epochu trénovania a nároky na výpočtové prostriedky. Výhodou veľkého datasetu je, že sieť dokáže lepšie generalizovať a tým pádom kvalitnejšie odpovedať na otázky užívateľa, ktoré neboli obsiahnuté v datasete. Na obrázku 6.5 je znázornený priebeh trénovania enkodér-dekodér modelu v závislosti od veľkosti použitého datasetu. Graf zobrazuje percentuálnu mieru úspešnosti na trénovacích dátach. Na obrázku 6.6 je tabuľka, ktorá zobrazuje dobu trvania jednej epochy a veľkosť slovnej zásoby s ktorou chatbot pracuje v závislosti od veľkosti použitého datasetu (počet viet).



Obrázek 6.5: Priebeh trénovania enkodér-dekodér modelu v závislosti od veľkosti použitého datasetu

Veľkosť datasetu	Priemerný čas trvania jednej epochy	Veľkosť gramatiky
10 000	9 minút	5 849 slov
50 000	42 minút	15 001 slov
80 000	1 hodina, 1 minúta	19 652 slov

Obrázek 6.6: Zobrazenie závislosti trvania trénovania jednej epochy a veľkosti slovnej zásoby od veľkosti použitého datasetu

Hodnoty *VAL_ACC* a BLEU skórovanie vyjadrujú mieru úspešnosti generovania viet vzhľadom na referenčné vety z datasetu. Určujú to, ako veľmi sa podobá odpoveď chatbota na referenčnú odpoveď, pričom sa neberie do úvahy, že vygenerovaná odpoveď sa nemusí vôbec podobať na referenčnú odpoveď a pritom môže byť správna a zmysluplná. Tento problém je bližšie popísaný v kapitole o testovaní a vyhodnocovaní chatbota, kde je spomenuté aj fungovanie BLEU metriky. Nižšie (6.1) sú niektoré príklady viet, kedy chatbot vygeneroval na otázku relatívne dobrú odpoveď, ale kvôli referenčnej odpovedi bola táto veta vyhodnotená ako nesprávna. Symbol *Q* označuje otázku pre chatbota, *CR* odpoveď chatbota, *RR* referenčnú odpoveď a BLEU označuje ohodnotenie odpovede chatbota BLEU metrikou.

```
Q:     You are an artificial
       linguistic entity
CR:    yeah.
RR:    that is my name.
BLEU:  0.033

Q:     You are not immortal
CR:    i am not!
RR:    all software can be
       perpetuated indefinitely.
BLEU:  0

Q:     What is a ratchet jaw
CR:    oh, nothing.
RR:    a chatterbot.
BLEU:  0.113

Q:     You are immortal
CR:    i am not asking you
RR:    not quite, but i can be perpetuated
       indefinitely.
BLEU:  0.027

Q:     Can you mate?
CR:    of course, i can not.
RR:    no, my method of reproduction is
          cloning.
BLEU:  0.032

Q:     Hi
CR:    hi.
RR:    how are you doing?
BLEU:  0
```

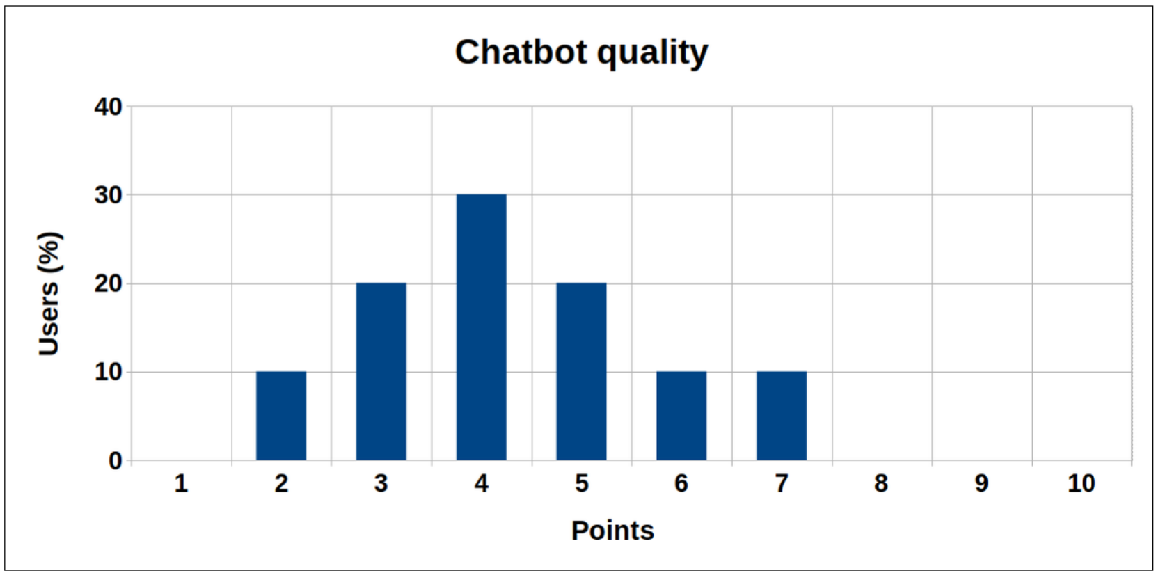
Listing 6.1: Ukážka konverzácie s chatbotom a BLEU ohodnotenie jednotlivých odpovedí chatbota

Pri ručnom testovaní chatbota po epoche, kedy nastalo pretrénovanie, chatbot nevykazoval skoro žiadnu mieru generalizovania. Vety generoval náhodne a častokrát mal tendenciu ich dookola opakovať. Preto som sa rozhodol vygenerovať automaticky odpovede chatbota po každej tretej epoche na každú z testovacích otázok. Potom som vybral model, ktorého odpovede sa javili ako najzmysluplnejšie. Takéhoto chatbota som potom testoval na užívateľoch.

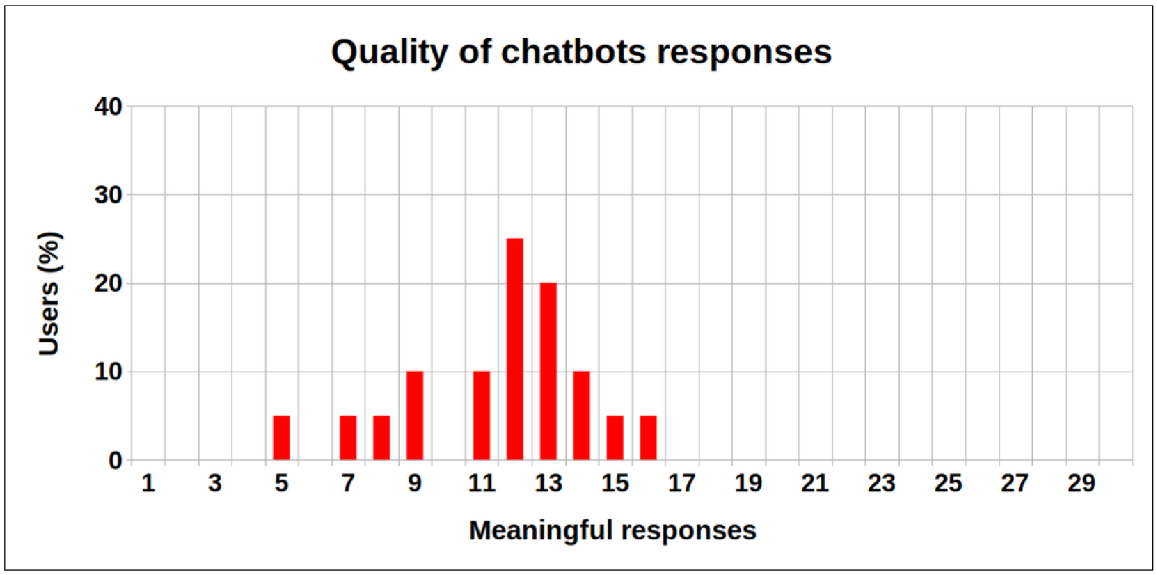
6.6 Testovanie na užívateľoch

Na základe výsledkov získaných experimentami, som sa rozhodol natrénovať chatbota, ktorého výsledky sa javili ako najlepšie. Model obsahoval bidirectional LSTM vrstvu v enkodéri a osobitnú vrstvu pre 300-dimenzionálny word embedding, ktorá sa trénovala spolu s celým modelom. Chatbot sa trénoval pomocou metódy teacher forcing na dátach o veľkosti 50 000 viet. Maximálna dĺžka viet bola nastavená na 10 tokenov. Keďže BLEU hodnotenie nebolo príliš objektívne a určovalo len to, ako sa vygenerovaná veta chatbotom podobá na referenčnú vetu, bolo potrebné kvalitu modelu otestovať aj na užívateľoch. BLEU som použil len ako jeden z parametrov, ktoré určovali v predchádzajúcich experimentoch, ktorý z modelov je lepší. Cieľom tohto testovania na užívateľoch bolo získanie objektívnej spätnej väzby o kvalite chatbota.

Chatbota som testoval na 20 ľuďoch. Väčšina z nich pochádzala z prostredia mimo IT. Testovanie prebiehalo na mojom notebooku. Každému testujúcemu boli povedané základné inštrukcie - čo je to vlastne chatbot, ako s ním komunikovať, ako pracovať s konzolou. Jednotlivé konverzácie trvali 5-10 minút, užívatelia si mali s chatbotom vymeniť 30 viet. Testujúci sa mali zamerať na správnosť syntaxe viet, či sú vety zmysluplné, či chatbot odpovedá na to, načo sa ho pýtajú. Taktiež mali zhodnotiť celkovú kvalitu konverzácie. Na konci testovania dostali testujúci 2 otázky. V prvej mali ohodnotiť kvalitu konverzácie číslom od 0 po 10, pričom 0 znamená, že chatbot nedokázal odpovedať na nič, 5 že chatbot dokázal odpovedať na polovicu otázok a 10 že chatbota by nebolo možné rozlíšiť od človeka. Druhou otázkou bolo, na koľko z tridsiatich otázok dokázal odpovedať chatbot podľa užívateľa zmysluplne. Po týchto dvoch otázkach mali testujúci povedať, čo sa im na chatbotovi páčilo a čo im prekážalo. Na nasledujúcich dvoch grafoch 6.7 a 6.8 je znázornené ohodnotenie chatbota testujúcimi.



Obrázek 6.7: Ohodnotenie celkovej kvality chatbota bodmi od 1 po 10



Obrázek 6.8: Ohodnotenie kvality a zmysluplnosti odpovedí chatbota

Užívatelia kvalitu chatbota hodnotili najčastejšie 3-5 bodmi z plného počtu 10 bodov. Z 30 odpovedí chatbota označili väčšinou 12 za zmysluplné. Chatbot mal najčastejšie problém s otázkami užívateľov, ktoré sa nevyskytovali v datasete. Často vygeneroval syntakticky správnu vetu ktorá mala zmysel, ale nenadväzovala na odpoveď užívateľa. Podarilo sa mu však v niektorých prípadoch vygenerovať zmysluplné vety a vykazoval aj určitú mieru generalizácie, kedy sa užívateľ spýtal na tú istú vec viacerými podobnými vetami a chatbot na ne odpovedal rovnako. Krátky príklad (6.2) konverzácie s užívateľom je zobrazený nižšie. Symbol *CH* označuje odpoveď chatbota a znak *U* otázku užívateľa.

```
CH: Welcome!
U:  hello
CH: hi
U:  who are you?
CH: it is martha,
U:  really? how is that
    possible?
CH: and i have a
    heard of back here.
U:  How old are you?
CH: twenty
U:  you don't say!
CH: no, i are not.
U:  what is the weather like?
CH: she ate a little bunny like.
U:  funny you are
CH: yes
U:  are you happy?
CH: i am am sorry about it.
U:  i am sorry too.
CH: just no one.
U:  not even me?
CH: yeah, wants to know it was in me.
U:  i know it was in you
CH: well, you are not.
U:  don't yu trust me?
CH: why not ?
U:  i'm not
CH: no, no
U:  hm, you look confused
CH: yeah, yeah, come on.
U:  where?
CH: yeah, now.
U:  now what?
CH: let me ask it.
U:  i'm listening
CH: it is watch behind you.
U:  watching me?
CH: yes.
U:  ok, now i'm a little nervous.
CH: would you miss anything?
U:  probably the sun
CH: Where are you?
U:  Are you real?
CH: yeah I~am
```

Listing 6.2: Ukážka konverzácie užívateľa s chatbotom

V konečnom dôsledku je kvalita chatbota nízka a preto by nemal praktické využitie. Je to čiastočne spôsobené aj použitým datasetom, ktorý bol z veľkej časti zložený z tituliek filmov a nachádzali sa v ňom vety, ktoré by reálne užívateľ nenapísal, napríklad "Copy. And Lundegaard too." alebo "Yah, the loudmouth. So the whole state has it, Lundegaard and Gustafson?". Použitie lepšieho datasetu zloženého z obyčajných každodenných konverzácií by malo výrazný vplyv na zlepšenie kvality chatbota.

Kapitola 7

Budúce vylepšenia

Tento chatbot bol naprogramovaný modulárne, tak aby doňho bolo možné jednoducho pridávať nové corpusy, nové modely neurónových sietí, a ďalšie postupy spracovávania dát. Existuje mnoho prístupov, ako vylepšiť generatívneho chatbota, založeného na enkodér-dekodér architektúre. V nasledujúcich kapitolách sú popísané niektoré z nich, ktorými sa chcem zaoberať v budúcnosti.

7.1 Beam search

Finálny chatbot, ako aj veľa súčasných chatbotov a generátorov strojového prekladu, ktoré používajú sequence to sequence architektúru, generujú dekodérom slová na základe rozdelenia pravdepodobností nad slovnou zásobou, ktorú systém používa. Konečná vrstva siete obsahuje toľko neurónov, koľko je slov v slovnej zásobe. Aktivačná funkcia softmax potom generuje pravdepodobnosť výskytu vo výstupnej sekvencii slov, pre každé slovo v slovnej zásobe. Greedy search je postup, kedy sa stále pri vygenerovaní pravdepodobností nad slovnou zásobou dekodérom, vezme slovo s najvyššou pravdepodobnosťou. Tento postup je často efektívny, ale nemusí vykazovať najlepšie výsledky. Beam search je postup, ktorý sa javí vo veľa prípadoch ako lepší než greedy search.

Metóda beam search je založená na hľadaní k najpravdepodobnejších sekvencií vygenerovaných dekodérom. Namiesto hľadania najpravdepodobnejšieho slova v každom kroku generovania nového slova dekodérom, vygenerujeme v každom kroku všetky možnosti a ponecháme si len k tých, ktoré sú najpravdepodobnejšie. Greedy search je samostatný prípad metódy beam search, kde $k = 1$. Nevýhodou tejto metódy oproti greedy search je jej zložitejšia implementácia a pomalšie generovanie vety dekodérom. Kvalita viet je však oproti greedy search vyššia [9].

7.2 Attention

V enkodér-dekodér modely funguje enkodér na princípe mapovania vstupnej vety na vektor fixnej dĺžky (throught vector). Dekóder potom generuje slová na základe tohto vektora a minulého vygenerovaného slova. Problém tejto architektúry nastáva pri dlhých vetách, kedy model vykazuje slabú kvalitu odpovedí. Je to pravdepodobne zapríčinené tým, že enkodér nedokáže skomprimovať všetky potrebné informácie do throught vektora, ktorého dĺžka je pevne daná. Attention je mechanizmus, ktorý ako prvý prezentoval Dzmitry Bahdanau, kedy sa dekodér učí, na ktoré časti vstupnej vety má "upriamiť pozornosť", pri každom

kroku generovania nového slova (ktoré časti vety sú relevantné). Takýto vylepšený enkodér-dekodér sa od klasickej architektúry líši tým, že enkodér nekóduje celú vstupnú vetu do jedného vektoru fixnej dĺžky. Namiesto toho kóduje vstupnú vetu na sériu vektorov a potom dekodér pri každom kroku generovania nového slova si z tejto série vyberá niekoľko vektorov, v ktorých sú uložené relevantné informácie. Tento mechanizmus je úspešný v oblastiach strojového prekladu ale aj pri tvorbe zložitejších generatívnych chatbotoch [1].

7.3 Ďalšie vylepšenia

Na kvalitu chatbota vplýva taktiež počet neurónov v LSTM vrstvách enkodéra a dekodéra, počet skrytých vrstiev a dimenzionalita word embeddingu. Môjho chatbota som tvoril tak, aby bolo možné s týmito parametrami ľahko manipulovať. Experimentovaním so zmenou parametrov môžeme dôjsť k zlepšeniu kvalitatívnych výsledkov chatbota. Použitie metódy bite pair encoding spolu s predtrénovaným embeddingom pre jednotlivé tokeny by mohlo viesť tiež k zlepšeniu odpovedí a trénovania chatbota.

Dôležité je aj to, ako bude vyhodnocovaná kvalita odpovedí chatbota. Použitím veľkého datasetu, ktorý by mal pre každú otázku niekoľko referenčných odpovedí by viedlo k lepšiemu vyhodnocovaniu toho, či je daná odpoveď chatbota zmysluplná.

Kvalitu odpovedí chatbota by tiež zlepšilo zavedenie takzvanej koherentnej osobnosti (coherent personality), kedy by chatbot na rovnaké, ale ináč položené otázky odpovedal konzistentne. Napríklad na otázky ”Where do you live?” a ”In which city do you live now?” by odpovedal rovnako. [14]

Kapitola 8

Záver

Táto práca bola venovaná tvorbe generatívneho chatbota postaveného na umelých neurónových sieťach. Hlavnou úlohou bolo navrhnúť a implementovať jednoduchého generatívneho chatbota, ktorý bude vedieť odpovedať na krátke otázky užívateľa.

Pre tvorbu chatbota som zvolil známy model sequence to sequence. V práci popisujem jednotlivé etapy, ktoré súvisia s tvorbou generatívnych chatbotov ako napríklad tvorba datasetu, preprocessing dát, tokenizing, padding, word embedding a návrh a trénovanie neurónovej siete. Súčasťou práce je aj niekoľko experimentov, ktoré porovnávajú jednotlivé modely. Na základe ich výsledkov som sa snažil nájsť model chatbota, ktorý by generoval čo najkvalitnejšie odpovede.

Implementáciu chatbota tvorí súbor modulov v rámci jediného programu, ktorý je ovládaný pomocou argumentov. Program poskytuje automatické vygenerovanie datasetu s ľubovoľnou dĺžkou viet, preprocessing dát a prípravu dát na trénovanie. Ďalej umožňuje automatické trénovanie chatbota, výber medzi rôznymi modelmi neurónovej siete a použitie metódy teacher forcing pri trénovaní neurónovej siete.

Ďalšou časťou práce bolo vyhodnocovanie kvality odpovedí chatbota. Kvalita je vyhodnocovaná pomocou BLEU metriky. V práci taktiež popisujem problém, kedy BLEU metrika môže označiť aj relatívne dobrú a zmysluplnú odpoveď chatbota za úplne chybnú. Pre získanie objektívnej spätnej väzby som otestoval chatbota na niekoľkých užívateľoch.

Výsledkom práce je chatbot, ktorý dokáže odpovedať na veľa otázok zahrnutých v trénovacích dátach. Na otázky, ktoré pri trénovaní k dispozícii nemal, väčšinou odpovedá nezrozumiteľne, ku krátkym otázkam občas dokáže vygenerovať zmysluplnú vetu. V praxi je tento chatbot kvôli relatívne slabej kvalite odpovedí použiteľný len ako forma zábavy. Modulárna implementácia však ponúka jednoduché doplňanie nových modelov neurónových sietí a postupov spracovania dát. Tento chatbot môže byť vďaka svojej univerzalite a jednoduchému manipulovaniu použitý pri experimentovaní s novými modelmi a vylepšeniami. Disponuje funkciami na zbieranie štatistík z priebehu trénovania a testovania a vykresľovanie grafov z týchto údajov.

Literatura

[1] Bahdanau, D.; Cho, K.; Bengio, Y.: Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, ročník abs/1409.0473, 2014, 1409.0473.
URL http://arxiv.org/abs/1409.0473

[2] Bayan Abu Shawar, E. A.: Chatbots: Are they Really Useful? 01 2007: s. 29–49.
URL http://www.jlcl.org/2007_Heft1/Bayan_Abu-Shawar_and_Eric_Atwell.pdf

[3] Britz, D.: Deep Learning for Chatbots, Part 1 – Introduction. 04 2016.
URL http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

[4] Britz, D.: Deep Learning for Chatbots, Part 2 – Implementing a Retrieval-Based Model in Tensorflow. 07 2016.
URL http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/

[5] Chinea, A.: Understanding the Principles of Recursive Neural networks: A Generative Approach to Tackle Model Complexity. *CoRR*, ročník abs/0911.3298, 2009, 0911.3298.
URL http://arxiv.org/abs/0911.3298

[6] Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; aj.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR*, ročník abs/1406.1078, 2014, 1406.1078.
URL http://arxiv.org/abs/1406.1078

[7] Chollet, F.: A ten-minute introduction to sequence-to-sequence learning in Keras. Navštívené dňa: 05.12.2017.
URL https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

[8] Davydova, O.: 7 types of Artificial Neural Networks for Natural Language Processing.
URL https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

[9] Freitag, M.; Al-Onaizan, Y.: Beam Search Strategies for Neural Machine Translation. *CoRR*, ročník abs/1702.01806, 2017, 1702.01806.
URL http://arxiv.org/abs/1702.01806

[10] Goldberg, Y.: *Neural Network Methods in Natural Language Processing (Synthesis Lectures on Human Language Technologies)*. Morgan & Claypool Publishers, 2017, ISBN 9781627052955.

[11] Goodfellow, I.; Bengio, Y.; Courville, A.: *Deep Learning (Chapter 10)*. MIT Press, 2016, http://www.deeplearningbook.org.

[12] Haykin, S.: *Neural Networks and Learning Machines*. Prentice Hall, 2008, ISBN 978-0131471399.

[13] Khan, R.; Das, A.: *Build Better Chatbots: A Complete Guide to Getting Started with Chatbots*. Apress, 2017, ISBN 978-1-4842-31111-1.

[14] Li, J.; Galley, M.; Brockett, C.; aj.: A Persona-Based Neural Conversation Model. *CoRR*, ročník abs/1603.06155, 2016, 1603.06155.
URL http://arxiv.org/abs/1603.06155

[15] Liu, C.; Lowe, R.; Serban, I. V.; aj.: How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. *CoRR*, ročník abs/1603.08023, 2016, 1603.08023.
URL http://arxiv.org/abs/1603.08023

[16] Mikolov, T.; Sutskever, I.; Chen, K.; aj.: Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, ročník abs/1310.4546, 2013, 1310.4546.
URL http://arxiv.org/abs/1310.4546

[17] Olah, C.: Understanding LSTM Networks. 2015.
URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[18] Papineni, K.; Roukos, S.; Ward, T.; aj.: BLEU: a Method for Automatic Evaluation of Machine Translation. 2002.
URL https://www.researchgate.net/publication/2588204_BLEU_a_Method_for_Automatic_Evaluation_of_Machine_Translation

[19] Pennington, J.; Socher, R.; Manning, C. D.: GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, s. 1532–1543.
URL http://www.aclweb.org/anthology/D14-1162

[20] Ram, S.: Chatbots with Seq2Seq. 2016.
URL http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/

[21] Schuster, M.; Paliwal., K. K.: *EEE Transactions on Signal Processing*, ročník 45, kapitola Bidirectional recurrent neural networks. IEEE Journals & Magazines, 1997, str. 2673–2681.

[22] Sennrich, R.; Haddow, B.; Birch, A.: Neural Machine Translation of Rare Words with Subword Units. *CoRR*, ročník abs/1508.07909, 2015, 1508.07909.
URL http://arxiv.org/abs/1508.07909

[23] Shridhar, K.: Rule based bots vs AI bots. 2017.
URL
https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa

[24] Štefanko, M.; Kvasnička, V.; Beňušková, L.; aj.: *Úvod do teórie neurónových sietí*. Iris, 1997, ISBN 9788088778301.
URL https://books.google.cz/books?id=0lI6AAAACAAJ

[25] Bidirectional LSTMs. Navštívené dňa: 09.02.2018.
URL http://omqwn4oyr.bkt.clouddn.com/201801261523_682.png

[26] LSTM enkóder-dekóder model. Navštívené dňa: 11.11.2017.
URL http://cfile30.uf.tistory.com/original/24FA50335965BB032FC623

[27] LSTM neurón. Navštívené dňa: 09.11.2017.
URL http://www.stratio.com/wp-content/uploads/2017/10/6-1.jpg

[28] Model jednoduchej neurónovej siete zloženej z jedného rekurentného neurónu. Navštívené dňa: 08.12.2017.
URL https://s3-ap-south-1.amazonaws.com/av-blog-media/wp-content/uploads/2017/12/06022525/bptt.png

[29] Viacvrstvová neurónová sieť. Navštívené dňa: 05.11.2017.
URL
https://cdn-images-1.medium.com/max/800/1*DW0Ccmj1hZ0OvSXi7Kz5MQ.jpeg

[30] Word embedding. Navštívené dňa: 08.02.2018.
URL
https://qph.fs.quoracdn.net/main-qimg-e8b83b14d7261d75754a92d0d3605e36

[31] Word embedding. Navštívené dňa: 01.03.2018.
URL https://raw.githubusercontent.com/rohan-varma/paper-analysis/master/word2vec-papers/models.png

[32] Řehůřek, R.: Making sense of word2vec. 2014.
URL https://rare-technologies.com/making-sense-of-word2vec/

Příloha A

Obsah priloženého DVD

Priložené DVD obsahuje nasledujúcu adresárovú štruktúru:

- *bakalarskapraca.pdf* – elektronická verzia tejto práce.
- Program – adresár obsahujúci program chatbot.
- Latex – adresár obsahujúci latex súbory pre vygenerovanie pdf.
- Plagát – Propagačný plagát k tejto práci
- Video – Propagačné video k tejto práci