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 **PETR ČERVÍČEK**
AUTHOR

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

BRNO 2019

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2018/2019

Zadání bakalářské práce

Student: **Červíček Petr**
Program: Informační technologie
Název: **Chatbot postavený na umělých neuronových sítích**
Chatbot Based on Artificial Neural Networks
Kategorie: Zpracování řeči a přirozeného jazyka

Zadání:

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 techniky strojového učení) 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/

Podrobné závazné pokyny pro vypracování práce viz http://www.fit.vutbr.cz/info/szz/

Vedoucí práce: **Szőke Igor, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2018
Datum odevzdání: 15. května 2019
Datum schválení: 6. listopadu 2018

Abstrakt

Bakalářská práce se zabývá implementací chatbota pomocí neuronových sítí. Využívá *Long short term memory networks*, které slouží k zapamatování dlouhodobých závislostí. Chatbot byl implementován v jazyce Python s nadstavbou Keras a je založen na principu *sequence-to-sequence*. Po implementaci byl chatbot testován pomocí metriky BLEU a dále předložen několika uživatelům ke konverzování. Pro lepší pochopení chatbotů je v této bakalářské práci taktéž popsána historie chatbotů a dán jednoduchý popis použitých technologií.

Abstract

The thesis pursues the implementation of the chatbot based on neural networks. It uses *Long short term memory networks*, which remember long-term dependencies. Chatbot was implemented in Python with superstructure Keras and is based on *sequence-to-sequence*. Chatbot was also tested by BLEU and given to users, who chatted with the chatbot. For a better understanding of the given problematics, there is simple description of the chatbot history and used technologies.

Klíčová slova

chatbot, strojové učení, konverzace, neuronová síť

Keywords

chatbot, machine learning, conversation, artifical neural network

Citace

ČERVÍČEK, Petr. *Chatbot postavený na umělých neuronových sítích.* Brno, 2019. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Igor Szőke, Ph.D.

Chatbot postavený na umělých neuronových sítích

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doktora Igora Szőkeho. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Červíček
12. května 2019

Poděkování

Doktor Igor Szőke mi poskytl mnoho užitečných informací, a tímto bych mu rád poděkoval za jeho ochotu.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá využitím neuronových sítí (strojového učení) při implementaci chatbota. Lidé se snaží ulehčit si práci s počítači, využívají k tomu různá textová nebo hlasová rozhraní, s jejichž pomocí zadávají počítači různé příkazy a ten jim následně odpovídá. Cílem bakalářské práce je implementace chatbota (textové rozhraní, ve kterém počítač reaguje na textový vstup tak, že mu na výstup odpoví). Chatbot komunikuje v anglickém jazyce, důvodem tohoto zvoleného jazyka byl výběr trénovacích dat. Uživatel zadává programu jednoduché anglické věty a chatbot na ně odpovídá větami, které vyhodnotí za vhodné. Součástí implementace chatbota je taktéž jeho testování. To je prováděno dvěma způsoby. Prvním je otestování odpovídání chatbota podle vyhodnocovacího algoritmu známého jako BLEU. Druhým je testování na uživatelích. Těm byl poskytnut chatbot a oni s ním konverzovali, poté jim byl dán dotazník s otázkami ohledně kvality konverzace s chatbotem. Při trénování chatbota jsou také využívány ztrátová funkce a metrika nazývána *Adam*. Tudíž již při trénování lze odhadnout správnost chatbota.

Dialogový systém mého chatbota je založen na natrénovaní neuronové sítě pomocí trénovací sady. Tato sada musela nejdříve projít předpracováním těchto dat. To znamená, že z těchto trénovacích dat byly nejdříve vytvořené slovníky. Dále je z těchto slovníků vytvořen vstupní vektor do neuronové sítě. V této práci byla použita již předem natrénovaná data, která byla převzata z externího zdroje[1]. Jedná se o bakalářskou práci Bc. Jiřího Čecháka [20], která se taktéž zabývá implementací chatbota. Převzatá data jsou více popsaná v sekci Slovník a data 6.3, která je součástí kapitoly Implementace.

Celý program byl napsán v programovacím jazyce *Python* a využívá knihovnu *Tensorflow* pro matematické výpočty a práci s daty a jeho nadstavbu *Keras*.

Implementovaný chatbot slouží pouze ke komunikaci mezi počítačem a uživatelem. Tento chatbot není určen k nahrazení lidí z různých center pro podporu zákazníků. Pro tyto záměry se nepoužívají chatboti implementovaní pomocí neuronových sítí, ale chatboti implementovaní pomocí retrival-based metodou [17, 18].

Implementace chatbota není jediným cílem této práce, ale je jím také nastudování fungování chatbotů a vysvětlení neuronových sítí.

[1] https://github.com/jirkacechak/chatbot/tree/master/data

Kapitola 2

Historie

I když hlavním cílem bakalářské práce je implementace chatbota pomocí neuronových sítí, který je založen na principu *sequence-to-sequence*, není však cílem jediným. Pro lepší pochopení chatbotů je v této kapitole popsána jejich historie a vývoj. Důvodem je obeznámení čtenáře s chatboty. Uvedením, proč jsou lidmi implementovány a jaké výhody každý typ chatbota má. Tato kapitola úzce souvisí s kapitolou Dělení chatbotů (viz 3), která popisuje jednotlivé jejich typy.

2.1 Turingův test

Alan Turing si ve svém článku *Computing Machinery and Intelligence*[14] klade otázku *„Can a Machine Think?“* , tedy zda stroje budou schopné jednou myslet stejně jako člověk. Vymyslel úlohu s názvem *Turingův test*, která má ověřit, zda je stroj dostatečně inteligentní. Jde o hru *„Imitation game“*. Zde člověk (soudce) textově komunikuje s dvěma entitami (jednou je člověk a druhou je počítač). Tato komunikace trvá stanovený čas. Poté soudce musí rozhodnout, který z těchto entit je člověk.

Alan Turing věřil, že v příštích 50 let budou mít počítače dostatečnou paměť k tomu, aby stroje dokázaly komunikovat se soudcem tak, že soudce nebude mít po 5 minutovém dialogu více než 70% šanci správně určit člověka. Jakmile takovýto stroj vznikne, bude považován za inteligentní.

2.2 Loebnerova cena

Hugh Loebner vyhlásil v roce 1990 každoroční soutěž s cílem splnit Turingův test. Pokud někdo dokáže implementovat počítač, který bude nerozpoznatelný od člověka, vyhraje $100 000 a zlatou medaili. Kromě hlavní výhry se každoročně vybírá mezi finalisty nejlidštější program. Ten pak vyhrává bronzovou medaili a menší finanční hotovost. Jakmile se najde stroj, který vyhraje hlavní cenu, poté se už tato soutěž nebude opakovat.

Toto jsou nejznámější držitelé bronzové medaile:

1. **Mitsuku**[1] – *ChatBot*, který získal Loebnerovu cenu v letech 2013 a 2016. Jedná se o 18-ti roční dítě, které se učí ze zkušeností. Jeho schopnosti se zlepšují podle toho, s kolika lidmi komunikuje. Čím více lidí s ním komunikuje, tím se zlepšují jeho komunikační schopnosti. Tvůrcem je Steve Worswick.

[1] https://www.pandorabots.com/mitsuku/

2. **Elbot**[2] – v roce 2008 se pomocí této implementace téměř podařilo zdolat Turingův test. 3 z 12 soudců byli přesvědčeni o inteligenci Elbota. Tento poměr je velmi blízko k hranici, která je potřebná pro zisk zlaté medaile. Autorem je firma Artificial Solutions.

3. **A.L.I.C.E.**[15] – autorem je Dr. Richard Wallace. Při implementaci se inspiroval slavnou *ELIZOU*. Jedná se o dialogový systém, který nenašel přemožitele v letech 2000, 2001 a 2004. Program využívá pro specifikaci pravidel speciální XML.

2.3 ChatterBot

V roce 1994 (téměř po 50 letech od zveřejnění Turingova článku) Michael Loren Mauldin publikoval článek s názvem „ChatterBots, Tiny-Muds, and the Turing Test: Entering the Loebner Prize Competition“ [9]. V tomto článku představil svůj program *ChatterBot*. Právě tento název byl zavedený pro pojem, který popisuje počítačový program komunikující s člověkem v přirozeném jazyce.

Program ELIZA [16] od německého profesora matematiky Josepha Weizenbauma, který v roce 1966 simuloval rozhovor psychologa s pacientem (uživatelem), se považuje za vůbec první implementaci inteligentního dialogového systému.

2.3.1 Návrh ChatterBota

V zadaném textu se vyhledají klíčová slova, ke kterým je podle důležitosti přiřazené prioritní číslo – tzv. *rank*. Každé klíčové slovo se váže s pravidlem, podle kterého je vstupní věta transformovaná. Velkou výhodou ELIZY je použití skriptu, který není součástí samotného programu, ale funguje jako externí data obsahující klíčové slova s příslušnými transformačními pravidly. Díky tomu je rozšíření na jiný přirozený jazyk triviální. Stačí jen přepsat skript do zvoleného jazyka.

Problém může nastat, pokud zadaná věta neobsahuje žádné klíčové slovo. V tento moment systém odepíše předem připravenou neutrální větu, například „Prosím, pokračuj.“, „To je velice zajímavé.“ nebo „Pověz mi o tom více.“

V případě ELIZY nejde o skutečnou inteligenci, ale jen o využívání triků, díky kterým působí systém důvěryhodným dojmem. Ale i navzdory tomuto faktu bylo mnoho lidí přesvědčených, že komunikují se skutečným psychoterapeutem a ne s počítačem.

[2] http://www.elbot.com/chatterbot-elbot/

Kapitola 3

Dělení Chatbotů

ChatBoty dělíme na základě jejich principu fungování do dvou kategorií:

1. **Programy založené na množině jednoduchých pravidel**, které transformují zadaný vstup na vhodný výstup (výstupem může být oznamovací nebo tázací věta). Proto dokáže reagovat jen na vstupy, které mají definované v pravidlech. Ostatní vstupy nedokáže analyzovat. Do této skupiny patří ELIZA a systémy se specifickým zaměřením – například uživatelská podpora (vyhledávání cestovního spojení, programu kin nebo divadel). Jedná se o takzvané retrieval based chatboty [18, 17].

2. Do druhé skupiny patří **pokročilé implementace využívající neuronové sítě**. Rozumějí přirozenému jazyku nejen předem definovaným příkazem, ale s každou konverzací zlepšují svoje schopnosti. Příkladem tohoto typu je například hlasový asistent *Cortana* nebo *ChatBot Xiaoice* (oba systémy jsou od společnosti *Microsoft*), který byl uvedený na čínský trh v květnu 2014 a od té doby s ním komunikovalo více než 40 milionů uživatelů. *Xiaoice* je dívka v teenagrovském věku, která využívá pro svůj vývoj hluboké neuronové sítě. Těmto chatbotům se říká generativní chatboti [18, 17].

   Implementace *ChatBotů* z této skupiny je mnohem náročnější než implementace z první kategorie.

Dále se dají tyto programy dělit na základě získávání informací. Některé systémy využívají vlastní znalostní databázi, která je pevně daná nebo se během komunikace s uživatelem automaticky rozšiřuje. Druhou možností je vyhledávání na webu – to využívá například systém *PAR* pro automatické odpovídání na otázky.

Než se dostaneme k detailnějšímu popisu těchto dvou kategorií, je potřeba se i dozvědět základní informace o doméně[1]. Doména udává, na které vstupy je schopen chatbot odpovědět a na které naopak odpovědět neumí nebo umí odpovědět jen částečně. Proto dělíme domény na dva základní typy:

Otevřená doména U tohoto typu domény není limitováno, odkud je vstupní věta pro chatbota. Není zapotřebí mít předem nadefinovaný účel konverzace. Uživatel tedy může vést konverzaci na libovolné téma a nemusí se držet nějakého konkrétního tématu.

Příkladem otevřené domény jsou například konverzace na sociálních sítích, jako jsou Twitter nebo Reddit. Zde se řeší témata všech možných druhů. Bohužel s nekonečným počtem témat se také zvyšuje

[1] http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

náročnost na vytvoření chatbota, který by dokázal správně reagovat na vše potřebné.

Uzavřená doména Chatbot s uzavřenou doménou může reagovat pouze na určité vstupy od uživatele. Vstup i výstup je limitovaný tím, že se chatbot snaží dosáhnout specifického cíle. Díky tomu neumí takový systém reagovat na všechny možné případy vstupů.

Systémy, které většinou používají uzavřenou doménu, jsou například technické podpory nebo asistenti, pomáhající kupříkladu při nakupování. Je to dané tím, že od těchto systémů nevyžadujeme komunikovat na různá témata, ale musí pouze správně pomoci uživateli se specifickým požadavkem.

3.1 Retrieval based chatbot

Jak již bylo naznačeno, retrieval based chatbot[1] [18, 17] nebo také nazýván rule based chatbot, je mnohem jednodušší na návrh a implementaci. Je to dáno především tím, že využívá určitou databázi, ve které jsou již předdefinované dotazy a odpovědi na tyto dotazy. Chatbot může obsahovat více různých databází. Výstup modelu chatbota implementovaného jako retrieval based je skóre značící přesnost odpovědi na otázku uživatele. Poté je vybrána odpověď z množiny vět v databázi, která má nejvyšší skóre.

U jednodušších typů těchto chatbotů může být databáze vět vlastní. To znamená, že chatboti mají v této databázi uložené věty ještě před začátkem užívání. U složitějších chatbotů dochází k ukládání vět do databáze i během používání chatbota. Rozšiřování je prováděno podle nových vstupů uživatele. Avšak tato vlastnost může mít i za následek i ukládání nevhodných dat do databáze. K tomu dochází například při nesmyslné odpovědi uživatele. Existují i typy chatbotů, které neobsahují žádnou databázi vět a odpovědí. Tito chatboti poté vyhledávají odpovědi na internetu na otázku zadanou uživatelem.

3.1.1 Metrika

Pro zjištění, jak přesné odpovědi udává chatbot postavený na implementaci retrieval based, se používají různé evaluační metriky. Jednou z nejčastěji využívaných metrik je metrika zvaná *recall@k*[1]. *Recall@k* vybírá k nejlepších odpovědí z 10 možných (předem určených). Jedna z těchto deseti odpovědí je správná a ostatní jsou takzvané distraktory. Pokud je správná odpověď mezi vybranými, pak je test vyhodnocen jako úspěšný. Pokud k nastavíme na 10, je pravděpodobnost správného výběru 100 %. Z toho tedy vyplývá, že čím je k menší, tím je pravděpodobnost výběru menší a naopak.

3.1.2 Příklady

ELIZA [16] První vzniklý chatbot. Byl vytvořen v roce 1966 německým profesorem matematiky Josephem Weizenbaumem na MIT.

Česká ELIZA Toto je implementace systému ELIZA pro češtinu. Autorem je Lubomír Sedlář, který se inspiroval originálem od prof. Josepha Weizenbauma. Lubomír Sedlář upravil původní systém tak, aby dokázal uspokojivě komu-

[1] http://www.wildml.com/2016/07/deep-learning-for-chatbots-2-retrieval-based-model-tensorflow/

nikovat v češtině. Česká ELIZA je dostupná na internetové adrese https://nlp.fi.muni.cz/projekty/czech_eliza/index.html.

A.L.I.C.E. Vývoj systému *Artificial Linguistic Internet Computer Entity* (A.L.I.C.E.) [15] začal v roce 1995 a o pět let později, v roce 2000, se tento *ChatBot* stal nejlidštějším (nejinteligentnějším) programem. Dále se mu povedlo obhájit prvenství ještě dvakrát. To ho řadí mezi nejúspěšnější vítěze Loebnerovy ceny.

Základem systému je speciálně navržený jazyk pro vytváření *stimul–odpověď* dialogových systémů, takzvaný AIML jazyk. Tento typ implementace přizpůsobí každý vstup od uživatele na vhodné odpovědi, které dá systém na výstup (prvním systémem tohoto druhu byl ChatBot ELIZA).

Jazyk AIML sestává ze základní jednotky, tzv. *kategorie* (značka *<category>*). Každá kategorie se skládá ze vstupní otázky (značka *<pattern>*), příslušné odpovědi (značka *<template>*) a volitelné možnosti (značka *<that>*), díky níž si systém pamatuje poslední rozhovor a možnosti *<topic>*, která sdružuje podobné kategorie. Tento jazyk podporuje i substituci za speciální znaky „*“ a „_“, která je ekvivalentní s regulárními výrazy, jejíž použití je snadnější.

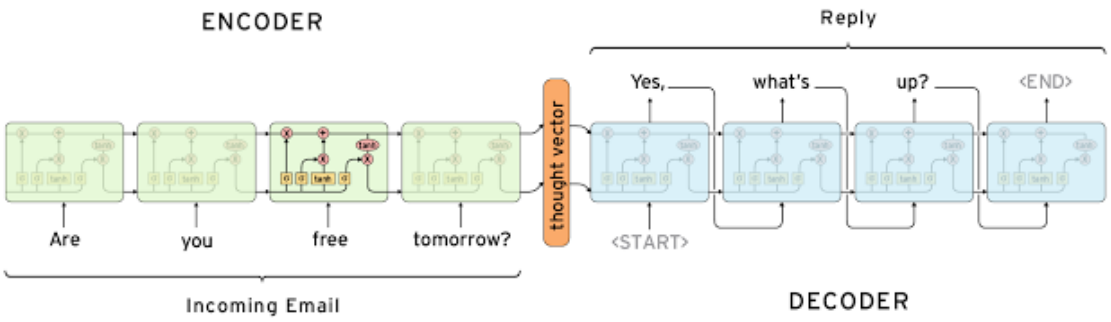
A.L.I.C.E. využívá učení s učitelem, protože osoba, tzv. botmaster, monitoruje všechny konverzace a přidává nový AIML obsah. Tento obsah poskytuje lepší a přesnější odpovědi. Pro další zdokonalování odpovědí byl vyvinut algoritmus na automatickou detekci nových vzorů v konverzacích. Díky němu jsou detekované vstupy, které stále nemají přiřazené svoje vlastní odpovědi.

Na vývoji tohoto systému se podílelo více než 500 dobrovolníků z celého světa a už od začátku byla A.L.I.C.E. zamýšlena jako volně šiřitelný software s licencí GNU.

3.2 Generativní chatbot

Generativní modely[1] [18, 17] nemají žádné předdefinované odpovědi, nové odpovědi generují. Jsou většinou založeny na stejném principu, na kterém pracují i modely pro překlad z jednoho jazyka do druhého. V tomto případě však nedochází k překladu mezi jazyky, ale k transformování jednoho vstupy na jemu daný výstup (odpověď na dotaz).

[1] http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/



Obrázek 3.1: Vyobrazení logiky *Sequence to Sequence*, kde vstup je převeden encoderem na vstupní vektor a výstupní vektor je decoderem převeden na odpověď, zdroj http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.

3.2.1 Metrika

Ideálním způsobem, jak měřit přesnost odpovědí chatbota, je posuzování pomocí lidského faktoru. Bohužel tato metrika[1] je časově i finančně náročná, proto je zapotřebí využívat i jiné. Takovou metrikou je algoritmus BLEU (viz 5.3), který je využíván především pro zjištění kvality překladu u strojových překladů textů do jiných jazyků. Tuto metodu je možné využít i pro zjišťování kvality nově vygenerované odpovědi u generativního chatbota.

Faktem zůstává, že výzkumy dokazují, že neexistují žádné známé využívané metriky, které by dokázaly vyhodnocovat odpovědi stejně dobře jako lidský úsudek.

3.2.2 Problémy

Generativní systémy mají v povaze odpovídat obecnými větami, jako je například „That's great!“ nebo „I don't know!“. Je to dáno tím, že se tyto odpovědi hodí na mnoho vstupních případů. Tento rys chování určují algoritmy použité při trénování.

Lidé obvykle odpovídají specificky podle otázky či jiného typu věty. Ale protože generativní systémy nejsou trénovány za určitým záměrem, chybí jim taková rozmanitost odpovědí, jaké by použil člověk.

3.3 Rozdíl mezi retrieval based a generativními chatboty

Oba typy chatbotů mají své výhody, ale také nevýhody. Retrieval based chatboti nedělají gramatické chyby. Chyba by se mohla vyskytnout pouze tehdy, že by byla obsažena v již předdefinované odpovědi v databázi. Ale na druhou stranu však nejsou schopni zvládnout odpovědět na případy, pro které nemají tuto předdefinovanou odpověď, ani nemohou reagovat na informace, které byly zmíněny jíž v předchozí konverzaci – například jména. Generativní modely jsou v tomto ohledu užitečnější. Mohou se odkazovat na informace zmíněné v předešlé konverzaci a také mohou navodit pocit, že uživatel komunikuje s člověkem. Ale tyto modely jsou těžké na natrénovaní. Na rozdíl od retrieval based modelů mohou odpovídat s gramatickými chybami, převážně u dlouhých souvětí. Další nevýhodou je potřeba velkého množství trénovacích dat.

[1] http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/

Jak pro retrieval based, tak i pro generativní modely mohou být použity techniky strojového učení. Ačkoliv v reálném prostředí se strojové učení používá převážně u generativních modelů.

Čím je konverzace delší, tím je náročnější tuto konverzaci zautomatizovat. Na jedné straně máme konverzaci, kde odpovídáme pouze jednou větou na jednoduchý vstup, na druhé straně pak dlouhé konverzace, u kterých si musíme pamatovat předešlý kontext. Takovým příkladem dlouhé konverzace je například zákaznická podpora.

Kapitola 4

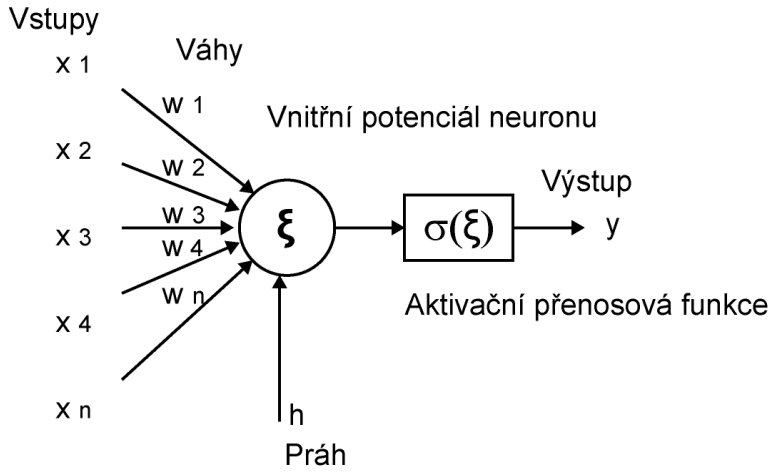
Neuronové sítě

V této sekci se věnuji stěžejní části své práce. Popíši zde, co vlastně neuronová síť [11, 19, 13] je, a v dalších podkapitolách se budu věnovat konkrétním typům neuronových sítí.

Neuronová síť je sada algoritmů, díky nimž dokáže rozeznat různé vzory – zvuky, obrázky, text a další. Je navržena podle lidského mozku. Neuronové sítě interpretují data pomocí strojových perceptorů, označují a shlukují nezpracovaná vstupní data. Veškerá vstupní data musí být převedena do matematického tvaru, lépe řečeno musí být převedena do vektorů, které neuronová síť umí zpracovat. Neuronové sítě shlukují neoznačená data k sobě podle určitých podobností s označenými trénovacími daty.

4.1 Umělý neuron

Vstupy do umělého neuronu jsou vektor X, dále následuje vektor vah W a práh ξ. Na obrázku 4.1 je vidět, že práh je vstupem do aktivační funkce $\sigma(\xi)$. Tato aktivační funkce slouží k aktivaci umělého neuronu. To znamená, že hodnoty výstupu z umělého neuronu jsou vypočítány právě pomocí této funkce. Celý model neuronové sítě je pak složen z n vrstev umělých neuronů. Samotný umělý neuron dokáže být klasifikátorem také. Potom se stává takzvaným lineárním klasifikátorem, to znamená, že dokáže řešit pouze lineárně separovatelná data.

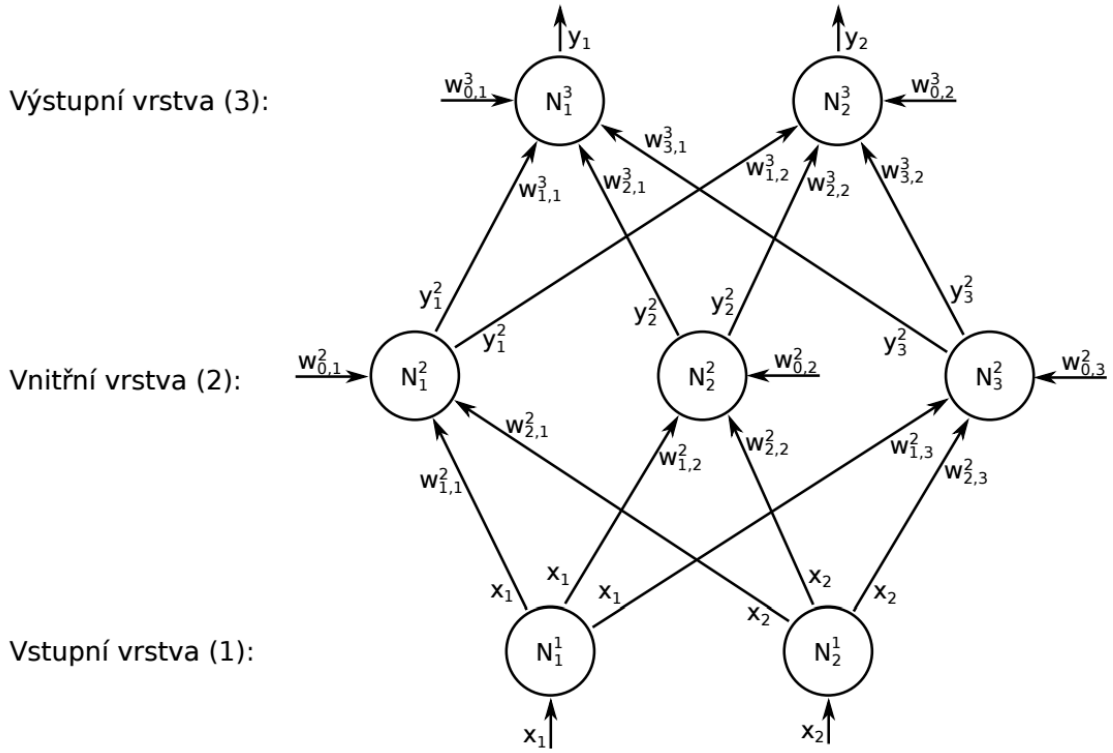


Obrázek 4.1: Umělý neuron neuronové sítě, zdroj http://www.matematickabiologie.cz/.

4.2 Topologie

Tato část se zabývá správnou volbou neuronové sítě. Nejjednodušším typem je dopředná neuronová síť. Zde je přiváděn na vstup vyšší vrstvy výstup všech neuronů z nižší vrstvy. Pokud se jde o výstupní vrstvu, pak se jedná o data výstupní samotné dopředné neuronové sítě. Z toho také vyplývá samotný název tohoto typu neuronové sítě – dopředná neuronová síť.

Vícevrstvá neuronová síť je schopná řešit i nelineární problémy. Je schopna aproximovat složité funkce a slouží jako obecný aproximační model. Musíme si dát pozor na to, že počet vrstev není úměrný zlepšení výsledků. Pro řešení mnoha problémů je vhodné využít architekturu jedné vstupní, vnořené a výstupní vrstvy. Abychom si dokázali představit, jak to vypadá, je možné se podívat na obrázek 4.2.



Obrázek 4.2: Značení neuronů, vah, vstupů a výstupů v neuronové síti.

Dále je nutné zvolit vhodný počet neuronů pro správnou generalizaci problému při trénování. Také je nutné určit, o jaká data se při trénování jedná. Neexistuje obecný postup, kterým by bylo vhodné se řídit. Je možné prakticky začít s malým počtem neuronů a ten dále zvětšovat tak, aby byl schopný aproximovat funkci. Dále je možné využít genetických algoritmů pro nalezení správného (vhodného) počtu neuronů. Musíme vědět, že se bude jednat o velkou neuronovou síť. Pokud je počet neuronů příliš malý, neuronová síť není schopna správné generalizace, ale na druhou stranu, pokud bychom zvolili příliš velký počet neuronů, může dojít k přetrénování.

Přetrénování neuronové sítě je děj, při kterém neuronová síť ukazuje velice kvalitní výsledky na trénovacích datech, ale při reálném nasazení na testovacích datech jsou výsledky

nekvalitní, nedají se použít. Přetrénovaní je bohužel velmi časté a existují techniky, jak tomuto problému předejít.

4.3 Aktivační funkce

Mezi nejpopulárnější aktivační funkce patří logická sigmoida

$$\sigma(z) = \frac{1}{1 + e^{-z}}, \tag{4.1}$$

sigmoida s derivací:

$$\sigma'(z) = (1 - \sigma(z))\sigma(z), \tag{4.2}$$

hyperbolický tangens:

$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}, \tag{4.3}$$

ReLU (rectified linear unit):

$$ReLU(z) = max(0, z), \tag{4.4}$$

ReLU s derivací:

$$ReLU'(z) = max(0, sgn(z)), \tag{4.5}$$

a parametrizovaná ReLU [7], která zaznamenala úspěch na datové sadě imageNet při použití s konvolučními neuronovými sítěmi. Tato aktivační funkce obsahuje parametr, který je adaptován během trénovaní. Jedná se o snahu o odstranění nulového gradientu:

$$PReLU(z) = \begin{cases} z & \text{if } z > 0 \\ az & \text{if } z <= 0 \end{cases} \tag{4.6}$$

$$PReLU'(z) = \begin{cases} 1 & \text{if } z > 0 \\ a & \text{if } z <= 0 \end{cases} \tag{4.7}$$

Neuronové sítě mohou být použity jak pro klasifikaci, tak i pro regresi. V případě klasifikace každý výstupní neuron představuje jednu klasifikační třídu. Chceme-li, aby bylo možné výstup sítě interpretovat jako posteriorní pravděpodobnost, jejíž suma musí být 1, použijeme v poslední vrstvě nelinearitu softmax:

$$softmax_i(x) = \frac{e^{x_i}}{\sum_{c=1}^{C} e^{x_i}} \tag{4.8}$$

Práci s neuronovou sítí obecně charakterizují dvě etapy:

1. používání známých dat pro trénování modelu, zde je k dispozici mapování mezi třídami a daty
2. používání neznámých dat pro model a následný výpočet posteriorní pravděpodobnosti, že daná data náleží dané třídě

4.4 Trénování neuronové sítě

Trénování neuronové sítě je takový proces, při kterém dochází k úpravě parametrů modelu neuronové sítě. Účelem tohoto procesu je minimalizovat chybu na výstupu neuronové sítě. Důležitou vlastností je generalizace, u které se řeší odchylka a variance dat. Před procesem trénování je odchylka velká a variance se dá zanedbat. Poté v průběhu trénování by se měla odchylka zmenšovat, variance by neměla růst. Ideální konec pro trénování neuronové sítě je moment, kdy součet odchylky a variance je nejmenší.

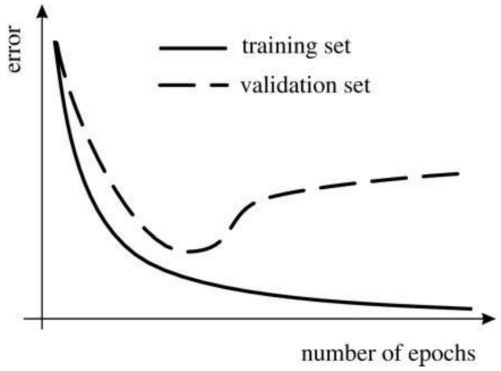
Trénování neuronové sítě lze také chápat jako posloupnost dopředného signálu, zpětné propagace chyby (viz 4.5) a adaptace vah. Tato posloupnost je prováděna v cyklech. Vždy na začátku cyklu jsou na vstup neuronové sítě přiváděna data a na výstupu jsou tato data vyhodnocována. Proces trénování neuronové sítě lze rozdělit na učení bez učitele a na učení s učitelem.

Učení s učitelem

U tohoto principu není znám celý kontext, proto se musí neuronová síť naučit generalizovat. Naučený systém pak dokáže reagovat i na vstupy, na kterých nebyl trénován.

Je vhodné rozdělit datovou sadu pro trénování neuronové sítě na dvě části. Na část pro vlastní trénování neuronové sítě a na část validační. Validační sada slouží k dynamickému vyhodnocování natrénování neuronové sítě – například pomocí testů. Jak bylo zmíněno, trénování probíhá v epochách (cyklech) a pomocí validační sady, která není využívána přímo pro samotné trénování, lze zjistit, jestli neuronová síť ještě pořád generalizuje. Tímto způsobem lze dobře předejít přetrénování neuronové sítě. Dále je dobré zmínit, že validační data jsou neviděná. Obrázek 4.3 nám ukazuje snižování chyby v průběhu trénování pro trénovací a validační sadu. Správné ukončení trénování je vhodné provést právě tehdy, kdy se chyba validační sady nesnižuje.

Funkce $E(D^i, P(Z^i, W))$ je základem, na kterém je princip trénování založen. Tato funkce představuje vyhodnocení dat na výstupu neuronové sítě a dat referenčních. Z^i je vektor vstupních i-tých dat, W je vektor vah. Váhy jsou nastavitelné a jsou ovlivňovány v zpětné propagaci v závislosti na funkci $E(D^i, P(Z^i, W))$. Pokud chceme získat hodnotu, která vypovídá o tom, v jakém stavu je náš model v rámci trénování, musíme zprůměrovat všechny hodnoty funkce $E(D^i, P(Z^i, W))$ v rámci dat pro trénování.



Obrázek 4.3: Graf závislosti chyby na trénovacích cyklech, převzato z [6].

4.5 Zpětná propagace chyby

Zpětná propagace chyby bude popsána pomocí vícevrstvého modelu založeného na učení z gradientu. Funkce $F_n(W_n, X_{n-1})$ je použita pro definování každé vrstvy, kde X_{n-1} je vstupem do každé vrstvy a X_n je výstupem dané vrstvy. Vektor vah, označený pomocí W, je ovlivnitelný. X_0 je vstupním vzorkem dat. Díky tomu je známa parciální derivace funkce E^p podle proměnné X_n. Po výpočtu parciální derivace funkce E^p podle W_n můžeme vypočítat proměnnou X_{n-1}.

$$\frac{\partial E^p}{\partial W_n} = \frac{\partial F_n}{\partial W}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n} \tag{4.9}$$

$$\frac{\partial E^p}{\partial X_{n-1}} = \frac{\partial F_n}{\partial X}(W_n, X_{n-1})\frac{\partial E^p}{\partial X_n} \tag{4.10}$$

$\frac{\partial F_n}{\partial W}$ je Jakobian W podle proměnné W v bodě daném $((W_n, X_{n-1}))$ a $\frac{\partial F_n}{\partial W}$ je Jakobian podle proměnné X. Také bych rád zmínil, co vlastně Jakobian je. Jakobian vektorové funkce obsahuje parciální derivaci všech výstupů s ohledem ke všem vstupům. Pro chybovou funkci budou spočítány všechny parciální derivace s ohledem na všechny parametry. Neuronová síť je speciální případ tohoto systému.

Kapitola 5

Použité technologie

Tato kapitola je zaměřena na vysvětlení technologií, které byly použity při vypracování bakalářské práci. Zaměřil jsem se na technologie, jež jsou využívány při strojovém učení obecně, nejsou zaměřeny jen na vývoj chatbotů. Největší výhodou těchto technologií není pouze jejich rozmanitá použitelnost, ale také jejich dostupnost. Veškeré technologie mnou použité jsou volně dostupné.

Jako programovací jazyk jsem zvolil Python. Python je skriptovací jazyk a díky jeho mnoha knihovnám je velice používán při implementaci strojového učení. Tyto knihovny velice ulehčují práci s daty a operacemi nad nimi.

Python [1] Vysokoúrovňový skriptovací jazyk. Nabízí dynamickou kontrolu datových typů a podporuje různá programovací paradigmata, včetně objektově orientovaného, imperativního, procedurálního nebo funkcionálního. Python je vyvíjen jako open source projekt.

Tensorflow [2] Open-source knihovna pro matematické výpočty využívající grafy pro tok dat.

NumPy [3] Knihovna pro složité výpočty. Umožňuje efektivní práci s N-dimenzionálními poli. [2].

Matplotlib [4] Knihovna pro vykreslování dat. Je možné vygenerovat křivky, histogramy, spektra, tabulky a mnoho dalších užitečných věcí.[8]

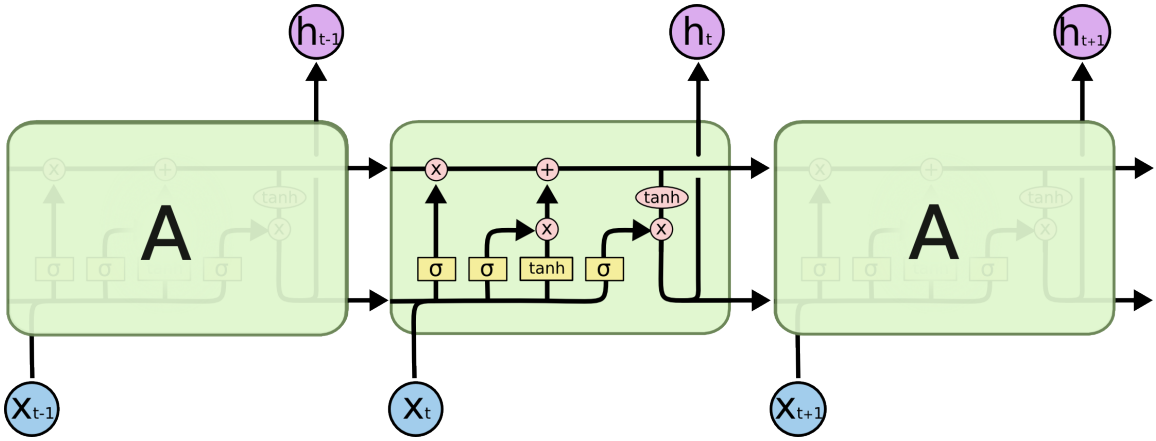
Keras [5] Nadstavbou pro Tensorflow. Zjednodušuje definici modelů neuronových sítí. Keras podporuje konvoluční i rekurentní neuronové sítě, ale i kombinaci obou. [1]

5.1 LSTM

Long short term memory networks [13] – obvykle zkracováno pouze na „LSTM“ – jsou speciálním typem rekurentních neuronových sítí. Jak jejich název napovídá, jsou schopny si zapamatovat dlouhodobé závislosti. Tento typ neuronových sítí byl představen párem Hochreiter & Schmidhuber v roce 1997.

[1] https://www.python.org/
[2] https://github.com/tensorflow/tensorflow
[3] https://www.numpy.org/
[4] https://matplotlib.org/
[5] https://keras.io/

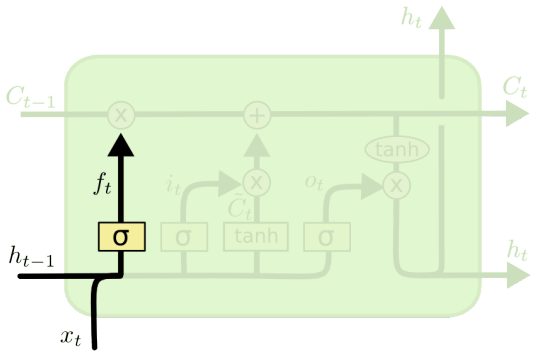
LSTMs byly navrženy, aby vyřešily problém dlouhodobých závislostí systému – takzvaných *long-term dependencies*. Zapamatování si informací po delší dobu je jejich hlavním znakem chování.



Obrázek 5.1: Opakující se perceptron v LSTM, obsahující 4 vrstvy, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Na obrázku 5.1 je vidět, že vstupem do uzlu sítě je celý vektor vystupující z předchozího uzlu. Růžová kolečka značí operace nad vektory, kdežto žluté čtverečky značí jednotlivé vrstvy neuronové sítě.

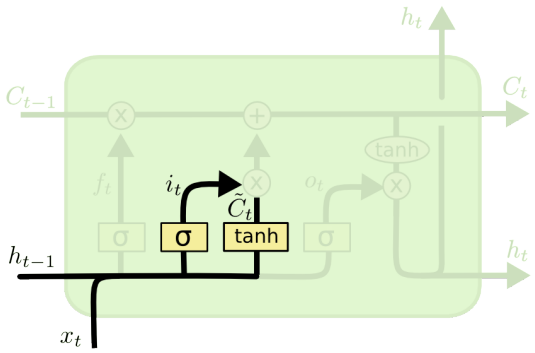
Prvním krokem ve vyobrazeném LSTM je rozhodování, které informace se mohou zahodit, a naopak, které jsou zapotřebí. Toto je realizováno pomocí „sigmoid“ vrstvy, která je též nazývána jako „forget gate layer“. Funkce přijímá na vstupu výstup *h_t-1* a x_t a na výstupu je poté vypočteno číslo v rozmezí 0 až 1 pro každý perceptron C_{t-1}. 0 značí „kompletně se zbavit vstupu“ a naopak 1 značí „zcela zachovat vstup“.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Obrázek 5.2: Výpočet zahození vstupu, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Dalším krokem je rozhodování, kterou novou informaci uchováme ve stavu. Tuto fázi lze rozdělit do dvou částí. První je „sigmoid“ vrstva, též nazývána *input gate layer*, jejímž úkolem je rozhodnout, které hodnoty nahradíme. Druhá vrstva je „tanh“ vytvářející vektor nových kandidátů $\tilde{C}$, kteří by mohli být přidáni do stavu. Spojením těchto dvou kroků sestavíme nový stav.

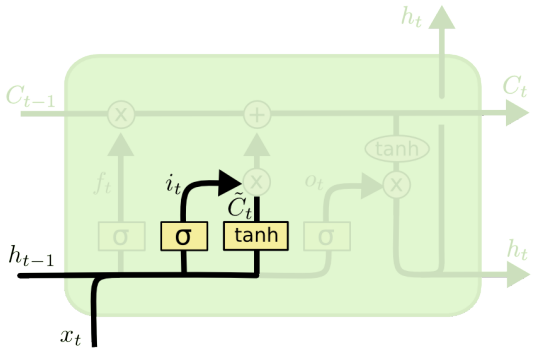


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Obrázek 5.3: Vytváření nového stavu, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Nyní je potřeba aktualizovat starý stav C_{t-1} novým stavem C_t. Předchozí krok určil, jak nový stav bude vypadat, teď je potřeba toto rozhodnutí uskutečnit.

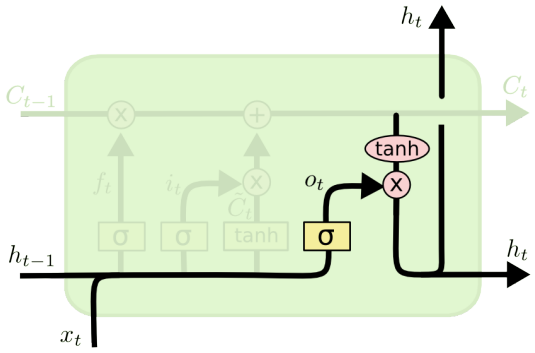
Vynásobíme stav C_{t-1} pomocí f_t, což způsobí „zapomenutí“ informací, které jsme rozhodli v předchozím kroku. Poté přičteme $i_t *$ C̃. Zíkáme tak nového kandidát upraveného podle naší potřeby na aktualizaci každé hodnoty stavu.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Obrázek 5.4: Aktualizace stavu, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

V poslední fázi je potřeba rozhodnout, co bude na výstupu. Tento výstup je daný na stavu perceptronu, ale bude ještě filtrovaný. Nejdříve projde vstup do „sigmoid“ vrstvy. Ta rozhodne, která část daného stavu bude dána na výstup perceptronu. Poté „proženeme“ upravený stav funkcí *tanh* (nastaví hodnoty v rozmezí -1 až 1) a vynásobíme tuto funkci takzvanou *sigmoid branou*, která určí požadovaný výstup.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Obrázek 5.5: Filtrování stavu, zdroj http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

5.2 GloVe

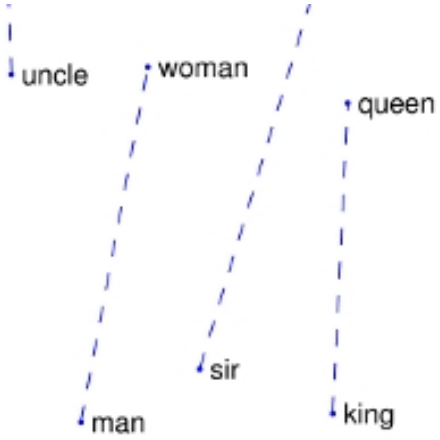
GloVe [10] neboli *Global Vectors for Word Representation* je algoritmus založený na principu učení bez učitele, jehož principem je získávání vektorové reprezentace slov. Trénování je uskutečněno podle souhrnných globálních statistik podle výskytu v korpusu. Již předem natrénované vektory, které chatbot využívá, jsou uloženy v souboru, který byl stažen z internetu[4].

5.2.1 Metody

1. **Nejbližší sousedé** je jedna z efektivních metod pro měření lingvistických nebo sémantických podobností slov. Spočívá ve spočítání *euklidovské vzdálenosti* dvou vektorů reprezentujících slova. Občas se výsledek této metody může působit zvláštně. Týká se to slov, která nepatří do průměrné slovní zásoby člověka.

2. **Linear substructures** je podobná metoda jako metoda *nejbližších sousedů*. Pro vyhodnocování využívá jediný skalár, který kvantifikuje souvislost dvou slov. To dělá tuto metodu jednoduchou. Tato jednoduchost však může být problematická, jestliže vztah dvou slov lze reprezentovat vícero způsoby. Příkladem mohou být slova *man* a *woman*, která jsou vzájemně významově podobná, neboť se v obou případech jedná o lidské bytosti. Na druhou stranu se jedná o slova, která stojí v protikladu – jsou to zástupci opačných pohlaví lidské rasy.

[4] https://nlp.stanford.edu/projects/glove/



Obrázek 5.6: Vztah man - woman, zdroj: https://nlp.stanford.edu/projects/glove/.

Na obrázku 5.6 je možné vidět, že rozlišování pohlaví u lidí může být specifikováno pomocí dalších slovních párů a nejenom pomocí slovního páru *man* a *woman*. Takovým dalším může být například slovní pár *king* a *queen* nebo *brother* a *sister*. Z toho vyplývá, že přesnost určení významově blízkých slov závisí mimo jiné i na velikosti vektoru. Přesnost určení významově podobných slov závisí na velikosti dimenze vektoru.

5.2.2 Trénování

GloVE model je trénován na vstupní nenulové matici souvisejících slov, která znázorňuje, kolik spolu souvisejících slov se v daném korpusu objevuje. Pro velké korpusy může být jejich průchod výpočetně náročný. Proto je lepší, když se využívá postupné trénování po iteracích. Tento způsob je také mnohem rychlejší.

5.3 BLEU

Důvodem vývoje této metriky je, že vyhodnocování člověkem je velice časově i finančně náročné. Navíc automatizované evaluační metriky mohou být použity na často se opakující úkoly, jako je monitorování změn systémů během jejich vývoje.

Je dobré poznamenat, že čím více je referenčních odpovědí pro jednu vstupní větu, tím je větší i evaluační BLEU [12, 5] skóre. Proto je nutné dát si pozor při vyhodnocování s různým počtem referenčních odpovědí.

Metrika BLEU porovnává výstupní řetězec strojového překladu s řetězcem referenčním. Evaluační metriky strojových překladů se liší od metrik pro rozpoznání řeči. Tyto metriky používají algoritmus pro výpočet WER (word error rate). Je to dáno tím, že při strojovém překladu můžeme použít různá slova se stejným významem.

Aby se předešlo problému s pořadím frází, BLEU využívá algoritmus *modified n-gram precision* místo již zmíněného word error rate algoritmu. Tento algoritmus je navržen tak, aby eliminoval opakování slov vyskytujících se ve větách. Pokud se dané slovo opakuje, je vždy počítáno stejně, jako kdyby se vyskytlo jen jednou. *N-gram* je sekvence několika sousedících položek v textu, případně řeči. V případě chatbota se jedná o tokeny, ze kterých se skládá textová sekvence.

BLEU metrika ukazuje přesnost v rozmezí 0 až 1, kde 0 znamená zcela nepřesnou odpověď a 1 naopak značí nejpřesnější odpověď. Ale v reálném prostředí téměř žádné systémy nedosáhnou evaluačního skóre 1. Takovým případem, kdy toto může nastat, by musel být testovací systém zároveň i referenčním systémem. Kvůli tomuto jevu se může stát, že i když bude odpovídat člověk, tak nemusí být skóre rovno 1. Skóre jedna značí, že sekvence n-gramů referenční textové sekvence je rovna sekvenci n-gramů strojového systému. Při výpočtu přesnosti se spočítají tokeny (slova, interpunkční znaménka apod.), které jsou obsaženy ve vstupním textovém řetězci (kandidát) a zároveň jsou obsaženy i v referenčním textovém řetězci. Toto číslo je poté vyděleno počtem všech tokenů v kandidátovi.

Znázornění výpočtu přesnosti

Kandidát:	the	the	the	the	the	the	the.
Reference 1:	The	cat	is	on	the	mat.	
Reference 2:	There	is	a	cat	on	the	mat.

V tabulce je vidět znázornění výpočtu přesnosti. Pro lepší orientaci jsou zde podtržena slova významná pro výpočet. U kandidáta se vyskytuje slovo *the* celkem sedmkrát. Počet výskytů slova *the* v první referenci je dvakrát, zatímco u druhé reference to je jenom jedenkrát. Z toho tedy můžeme vypočítat přesnost. Pro první referenci to je $P_1 = \frac{2}{7}$, zato pro druhou to je pouze $P_2 = \frac{1}{7}$.

Hlavním důvodem, proč se pro vyhodnocování používá tato metrika, je, že se ukázala být velmi podobná vyhodnocování člověkem.

Kapitola 6

Implementace

Tato kapitola popisuje implementaci chatbota a popis skriptů.

6.1 Popis skriptů

Skript pro BLEU testování

Do této skupiny patří jediný skript, a to *blueTestingFunctions.py*. Obsahuje všechny potřebné funkce pro otestování kvality odpovědí tří modelů chatbota. Celkově se jedná o šest potřebných funkcí. Funkce *loadTestingDataForBLEU()* slouží, jak již název naznačuje, k načtení testovacích dat. Jedná se o data uložená v souborech *test.enc* a *test.dec*. Tato funkce je poté volána ve funkci *testBLEUFromLastSavedWeights()*, jež slouží k načítání uložených vah konkrétního modelu. Dále tu jsou čtyři spolu související funkce, které právě provádí samotné vypočítávání výsledku metriky BLEU. *PredictionBLEU()* slouží k predikci odpovědi daného modelu chatbota, která je následně použita jako kandidát v BLEU. Tato funkce je volána ve funkci *calculateBLEU()*, pomocí níž počítá BLEU váhy pro následný výpočet výsledků. Jako poslední testovací BLEU funkce jsou dvě téměř totožné funkce *trainScoreBLEU()* a *testScoreBLEU()*. Obě funkce počítají výsledné BLEU skóre, rozdílem je, že jedna počítá skóre na trénovacích datech a druhá na testovacích datech.

Skript pro spuštění

Start_chatbot.py je skript, kterým se spouští chatbot. Hlavní náplní tohoto skriptu je zpracování argumentů a následné zavolání potřebných funkcí. Více o zpracování argumentů a spuštění je uvedeno v sekci *Spouštění chatbota* 6.2.

Skript pro model chatbota

Všechny funkce související s vytvářením modelu se nacházejí ve skriptu *modelFunctions.py*. Zde se nachází i jedna z nejdůležitějších funkcí chatbota, a tou je funkce *modelCreation()*. Z názvu je patrné, že se jedná o funkci pro vytváření modelu chatbota. Přesněji vyjádřeno, nejedná se pouze o vytváření jednoho modelu, ale uživatel si může vybrat z vytvoření jednoho ze tří modelů. O konkrétních modelech viz 6.4. Jedná se o jednu z nejvíce používaných funkcí. Druhá a zároveň poslední funkce nacházející se v tomto skriptu je *answerPrediction()*, sloužící k predikci odpovědi chatbota.

Skript pro trénování

V této kategorii se nachází skript *startFunctions.py*. Ačkoliv název může být zavádějící, zde se vyskytují funkce pro trénování předem vytvořeného modelu. Jedná se o funkci *train()*, která daný model natrénuje podle požadavků uživatele. Více informací ohledně trénování chatbota se nachází v sekci 6.5 . Druhou funkcí je *dataLoader()*, která má za úkol načíst data pro natrénování neuronové sítě. Jedná se o načtení dat pro encoder a decoder nacházející se v souborech *train.enc* a *train.dec*, tak jako tomu bylo podobně u načtení dat pro BLEU testing.

Skripty pro chatování s chatbotem

V této sekci jsou uvedeny dva skripty související s chatováním. Prvním z nich je *chat.py*. V tomto skriptu se nachází třída ChatManager, která řídí chatování mezi uživatelem a chatbotem. Pro získání odpovědi chatbota se volá funkce *getAnswer()*. Je volána z další funkce vyskytující se v tomto skriptu, a tou je funkce *chat()*. Tato funkce řídí výpis odpovědi chatbota do terminálu uživateli.

Dalším skriptem, řídícím výpis odpovědí chatbota přes GUI, nikoliv do terminálu, je skript *gui.py*. Tady můžeme nalézt třídu *GUI*, která slouží k vytvoření chatovacího gui pro uživatele. Pro výpis odpovědí využívá taktéž funkci *getAnswer()*, jako tomu bylo u výpisu do terminálu.

Ostatní skripty

Zde se budu zabývat ostatními skripty. Ty slouží především jako pomocné pro výše zmíněné funkce. Nejdůležitější z této skupiny je skript *dataUtils.py*. Zde lze nalézt důležité podpůrné funkce pro vytváření, trénování modelu apod. První funkcí v tomto skriptu je funkce *savedWeights()*, která načítá ze souboru uložené natrénované váhy modelu. Tyto váhy jsou uloženy v souboru s příponou *.h5* a vzorový název vypadá takto: *modelWeights-[velikost dat]-[číslo epochy].h5*. Soubor s daty uloženými po poslední epoše je poté načten tak, aby se pokračovalo buď trénování od této epochy, nebo jsou tyto váhy využity k predikci odpovědi chatbota. Další funkcí v tomto skriptu je *vocabularyLoader()*, která slouží k načtení slovníku (6.3). Dále tu je funkce, která převádí vstupní tokeny na indexy – *tokensToIndexs()*. Poslední funkcí z tohoto skripta, kterou bych rád zmínil, je funkce *noStartingAndEndingToken()*. Slouží k odstranění BOS[1] a EOS[2] tokenů ze vstupních tokenů.

Následujícím skriptem je *constants.py*. Zde jsou pouze uloženy globální proměnné, které se používají ve všech bodech vytváření chatbota – od vytváření modelu přes trénování až k testování modelu. V důsledku toho, že je možné si vybírat ze tří typů modelů, vyskytuje se zde několik funkcí, které mají na vstupu číslo modelu a na výstupu mají konstantu, která má hodnotu podle typu modelu. Například se může jednat o nastavení složky modelu. Proto se zavolá funkce *SET_MODEL_DIR(model)*, která vrátí jednu ze tří možností – „data/models/first_model/“, „data/models/second_model/“, „data/models/third_model/“.

Jako poslední skript uvedu *utils.py*. V něm jsou obsaženy pomocné funkce pro výpis informací či práci se souborem. Nachází se zde například funkce pro ověřování existence souboru, která zároveň, zda není tento soubor prázdný. K dalším patří funkce pro výpis chybových hlášek nebo funkce pro výpis „help zprávy.“

[1] Beginning of sentence
[2] End of sentence

6.2 Spouštění chatbota

Pro spuštění chatbota se musí zavolat funkce *start_chatbot.py* (viz 6.1). Chatbot může být spuštěn ve třech módech – trénování, chatování a testování. Pokud spustíme tento skript s parametrem *-t* nebo s parametrem *–train*, tak se zavolá funkce *train()*, která spustí trénování. Tato funkce potřebuje tři parametry. Číslo modelu, na kterém se bude neuronová síť trénovat, datový limit trénování a boolean parametr. Poslední parametr říká, zda se bude pokračovat v trénování od posledních uložených trénovacích vah, nebo zda se bude trénovat od začátku.

Dalším módem, kterým je možné chatbota spouštět, je *-c* nebo *–chat*. Tím se spustí mód pro chatování s uživatelem. Zde je volána funkce *chat()*, která potřebuje dva parametry – číslo modelu a boolean hodnotu gui. Číslo modelu značí, který model bude použit pro chatování s uživatelem, parametr *gui* pak určuje, zda chatování bude probíhat pouze v terminálu (nastaveno na false) nebo v případě nastavení na *true* bude chatování probíhat přes UI.

Poslední mód je testování. Abychom mohli testovat chatbota, je nutné spouštět skript pomocí parametru *–test* nebo *–testAll*. V obou případech se začne testovat chatbot, rozdílem je, že parametr *–test* otestuje pouze jeden model. Parametr *–testAll* otestuje všechny tři modely chatbota. U tohoto módu je volána funkce *testBLEUFromLastSavedWeights()*, která vyžaduje zadání tří parametrů: číslo modelu, který má být testován, limit pro testování a posledním je boolean hodnota, která udává, zda bylo zadáno testování všech modelů či jen jednoho konkrétního.

Pokud při spuštění chatbota nejsou zadány žádné parametry, nastaví se potřebné hodnoty pro spuštění automaticky. Těmito implicitními hodnotami jsou určení módu, ve kterém má chatbot pracovat, model, který bude používán – tím bude základní (první) model. Dále se všechny boolean hodnoty nastaví na „False“, testování chatbota tedy nebude probíhat a komunikace s ním bude pomocí terminálu.

Dále se mohou použít ještě další parametry, nejen parametry pro určování módu chatbota. Například je možné použít parametr *-g* nebo *–gui*, což je boolovská hodnota, která určuje, v jakém prostředí bude uživatel komunikovat s chatbotem. Pokud je hodnota nastavena na „True“, je otevřeno GUI prostředí 6.5.3 vytvořené pro komunikaci. Dalším možným parametrem je parametr *–model=[číslo modelu]* určený pro nastavení modelu, který se při daném módu má používat. Je možné vybírat ze tří možných modelů. Pokud uživatel zadá jiné číslo, než je počet modelů, tak je program ukončen s chybou „Wrong model number. Select number 1-3“. Posledním parametrem, který lze zadat při spuštění chatbota, je parametr *-h* nebo *–help*. Po jeho zadání dojde k vypsání „help zprávy“ a ukončení programu. Tato zpráva je určena pro informování uživatele. Je zde popis chatbota a také nápověda, jakým způsobem lze program spouštět.

6.3 Slovník a data

Pro tuto práci byla použita již předpracovaná data a slovník. Lze je najít na adrese https://github.com/jirkacechak/chatbot/tree/master/data. Jedná se práci od Bc. Jiřího Čecháka [20], která se taktéž zabývá implementací chatbota pomocí neuronových sítích. Proto byla použita právě tato data a slovník z této práce.

Slovník

Slovník [20] je použit pro jednodušší manipulaci s tokeny a k omezení počtu tokenů, které se mohou vyskytovat v trénovacích datech, popřípadě při generování odpovědi. Slovník se používá při všech módech spuštění chatbota – trénování, chatování i testování. Je uložen v příslušném souboru (v případě tohoto chatbota „/data“).

Ve slovníku se vyskytují tokeny, kterými jsou především anglická slova, ale mohou to také být interpunkční znaménka nebo dva speciální tokeny – BOS[1] a EOS[2]. Tyto dva tokeny představují začátek a konec odpovědi. Dodatečně byl do slovníku přidán další speciální token nahrazující všechny neznámé tokeny.

Slovník obsahuje 7 816 tokenů.

Trénovací data

Data[20] pro trénink se nacházejí ve dvou souborech. Jeden obsahuje data pro encoder a druhý data pro decoder. To znamená, že první soubor obsahuje na jednotlivých řádcích kontext a druhý soubor obsahuje odpovědi na tento kontext. Jedná se o dialogy z filmů, sociálních sítí apod. Nacházejí se zde několik desítek dlouhé textové sekvence, ale i krátké, někdy i jednoslovné.

Protože trénovací data jsou převzata, chatbot slyší po natrénování na jméno *George*. Pokud je chatbot osloven jiným jménem, upozorní uživatele, že se jmenuje jinak, popřípadě se znovu představí. Bc. Jiří Čechák tento rys zrealizoval tak, že pomocí skriptu vygeneroval nová data. Ty byla mimo jiné použita pro naučení chatbota reagovat na toto jméno.

Pro získání lepších výsledků by bylo nutné dále upravit trénovací data a získat větší množství vhodných trénovacích dat. Více trénovacích dat lze najít na internetu z dalších volně dostupných chatbotů a přidat je k trénovacím datům této práce.

6.4 Model

Chatbot je natrénován na třech různých modelech a v této části textu jsou všechny tři popsány. Nejprve je představen jejich obecný charakter a jejich chování. Každý model je možné natrénovat a následně použít na predikci odpovědi.

Model chatbota je založený na principu *sequence-to-sequence*, čili vstup je převeden encoderem na vstupní vektor a decoderem převeden výstupní vektor na odpověď.
Model.compile(loss="categorical_crossentropy", optimizer=Adam(), metrics=["accuracy"]) – na tomto výňatku z kódu je možné vidět, že pro ztrátovou funkci je použita funkce *categorical crossentropy*[3]. Je zde využita pro klasifikaci do více tříd. Funkce bere na vstupu dva argumenty – $Argument_{true}$, $Argument_{pred}$, kde $Argument_{true}$ je tensor s výstupem neuronové sítě a $Argument_{pred}$ je tensor s očekávaným výstupem. Čím více se ztráta blíží k hodnotě 0, tím je menší rozdíl mezi výstupem neuronové sítě a očekávaným výstupem. Jako optimalizátor je v tomto chatbotu použit optimalizační nástroj zvaný *Adam* [4], u kterého je nastaven „learning rate“ na 0.00005. Tato hodnota je nastavena, aby bylo trénování dostatečně přesné. Optimalizátor *Adam* je algoritmus, který dokáže aktualizovat váhy neuronových sítí v každé iteraci podle trénovacích dat. Je zde použit pro rychlejší trénování neuronové sítě. Dále se během tréninku používá i metrika *accuracy*, která je velmi podobná ztrátové funkci. Avšak na rozdíl od ní nejsou evaluační výsledky z metriky použité pro tré-

[1] Beginning Of Sentence
[2] End Of Sentence

nování modelu. Navíc u této metriky je porovnán předpovídaný výstup s trénovacími daty. 1 zde značí úplnou shodu těchto dvou vstupů.

U všech modelů se pracuje s word embedding s dimenzí o velikosti 300 a sentence embedding s dimenzí taktéž o velikosti 300. Trénovacích dat bylo použito 172 362 vzorků a slovník obsahuje 7 816 tokenů.

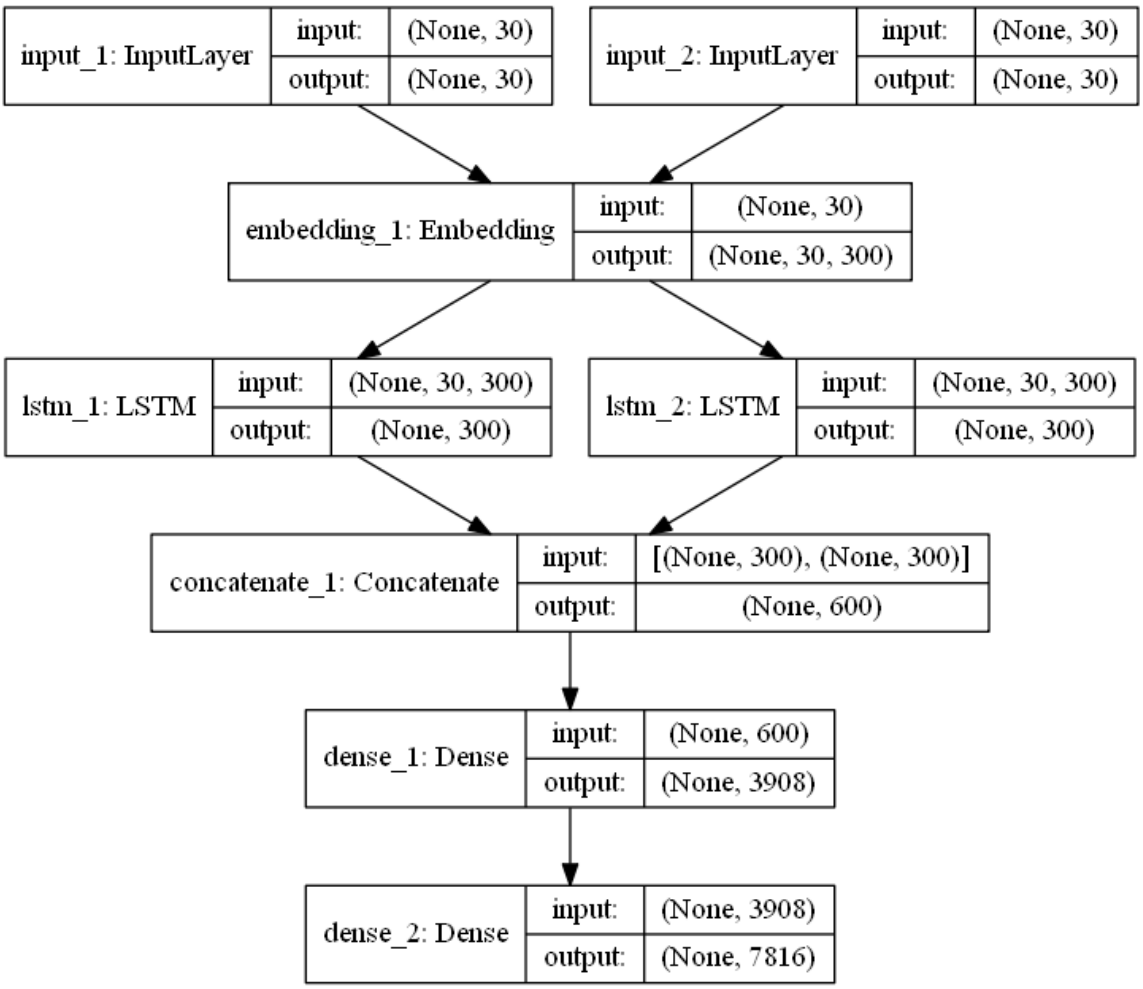
První model chatbota

První model se skládá ze šesti vrstev. Nejdříve je vytvořena vstupní vrstva, která se skládá z *encoder* a *decoder* částí. Berou na vstup vektor s dimenzí o velikosti, která je nastavena proměnnou *MAXLEN_INPUT*. V případě chatbota je to velikost 30 neboli 30 indexů. Tyto indexy značí konkrétní pozici tokenů ve slovníku. Jak bylo řečeno v části týkající se tématu Long short term memory networks (LSTM) 5.1, na vstup pro *encoder* jde kontext, zatímco na vstup pro *decoder* jde odpověď.

Následující *embedding* dostává na vstup výstup z přechozí vrstvy – jak od *encoderu*, tak i od *decoderu*. Tato vrstva nahradí všechny indexy ve vstupním vektoru vektorem s dimenzí o velikosti *word embedding*. Výstup této vrstvy jde do další vrstvy, která je rozdělena na dvě části – LSTM vrstva pro encoder a LSTM vrstva pro decoder.

Obě tyto části berou na vstup výstup *embedding* vrstvy a vrací vektor s dimenzí o velikosti 300; je to hodnota velikosti *sentence embedding* (viz 6.4). Výstup obou LSTM vrstev je poté sjednocen do jednoho vektoru, jehož dimenze je o velikosti 600. Tento výstup poté následuje do další vrstvy, kterou je vrstva *dense*.

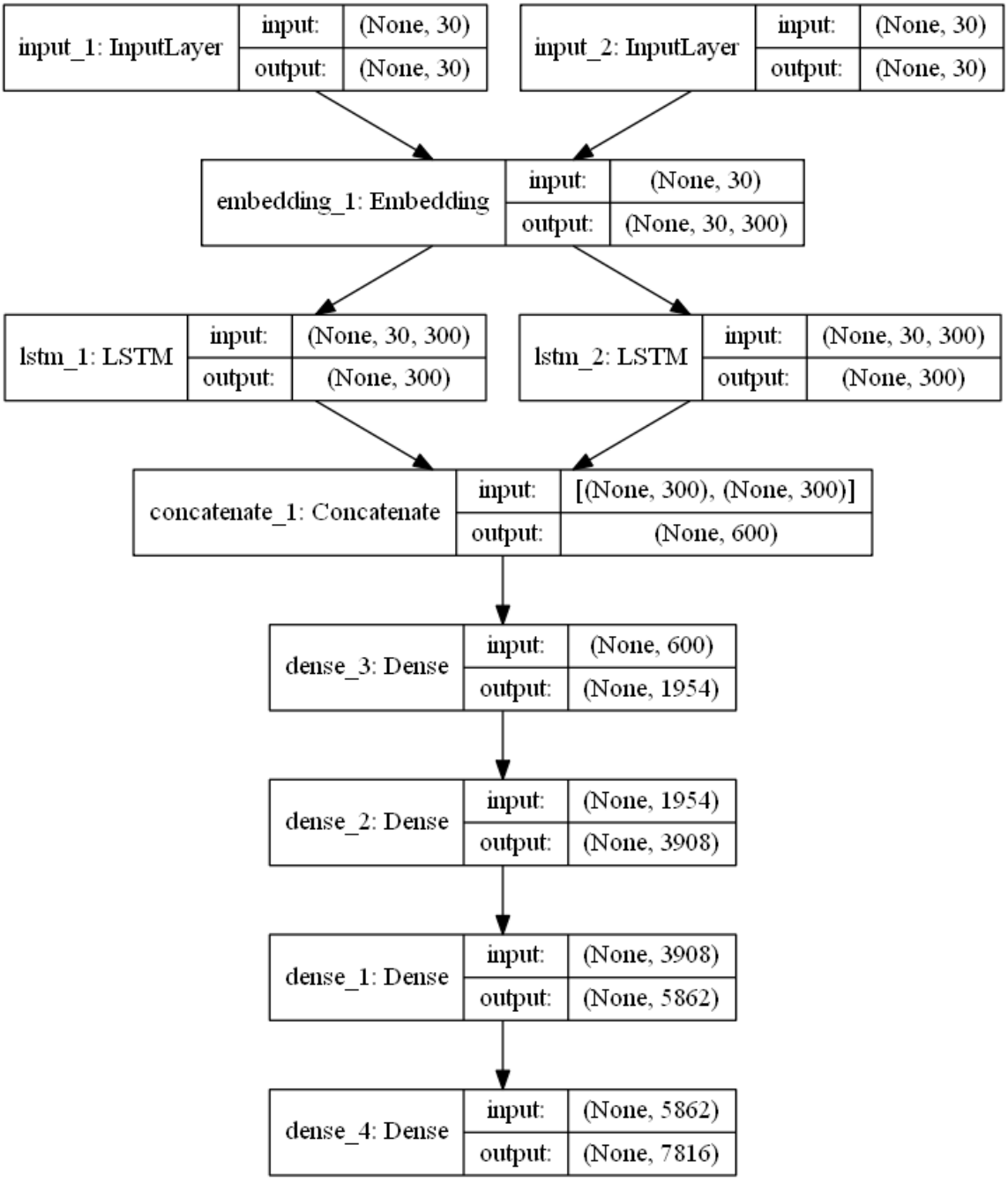
Zde je tento vektor transformován na vektor o velikosti poloviny slovníku (3 908). Aktivační funkcí této vrstvy je funkce *RELU* (viz 4.3). Výstup této vrstvy je poslán do další *dense* vrstvy, která tento vektor transformuje na vektor s dimenzí o velikosti slovníku (7 816). Aktivační funkcí je zde *SOFTMAX* (viz 4.8).



Obrázek 6.1: Složení prvního modelu (vygenerováno pomocí funkce *plot_model* z balíčku *keras.utils*).

Druhý model chatbota

Druhý model je velmi podobný modelu prvnímu. Tomuto modelu jsou pouze přidány *dense* vrstvy. První čtyři vrstvy jsou stejné jako u prvního modelu. Vektor s dimenzí o velikosti 600, který je výstupem ze čtvrté vrstvy (spojení výstupů LSTM encoderu a LSTM decoderu), je předán *dense* vrstvě, která transformuje tento vstup na výstupní vektor s dimenzí o velikosti $\frac{1}{4}$ slovníku, což je 1 954. Poté jsou zde další tři *dense* vrstvy, z nichž každá transformuje předešlý výstup na vektor s dimenzí o velikost větší o $\frac{1}{4}$ slovníku. Tedy velikosti těchto výstupů jsou tedy 3 908, 5 862, 7 816. Z toho vyplývá, že konečná velikost výstupního vektoru modelu má dimenzi o velikosti celého slovníku.



Obrázek 6.2: Složení druhého modelu (vygenerováno pomocí funkce *plot_model* z balíčku *keras.utils*).

Třetí model chatbota

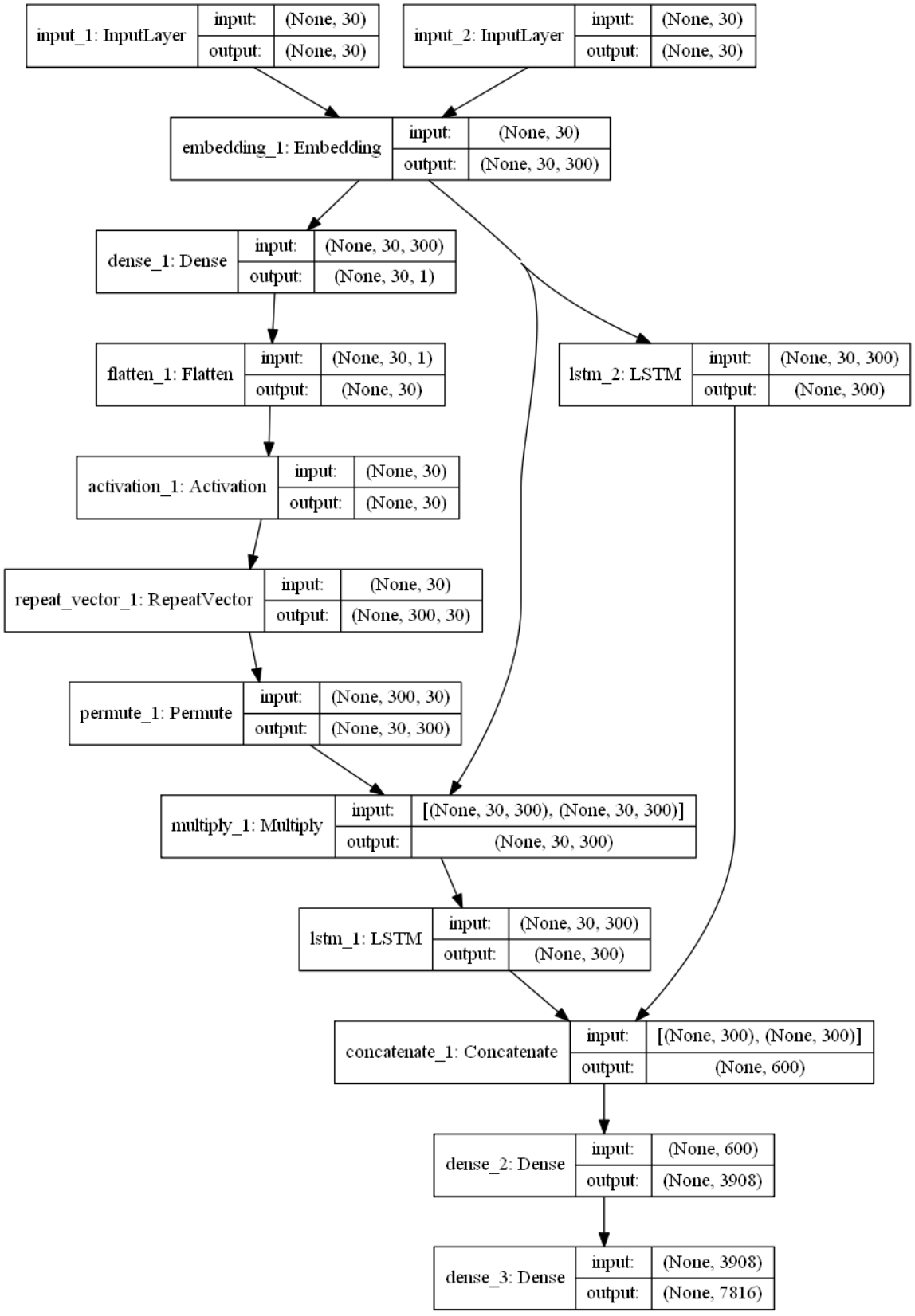
Třetí a zároveň poslední model chatbota je ze všech tří ten nejsložitější, neboť používá i vrstvy zajišťující attention mechanismus[1] u *encoderu*. Tento mechanismus částečně řeší problém načítání celých vět a transformování všech informací do vektoru fixní délky. Berou se v úvahu všechny informace z originální vstupní věty a poté se vygeneruje vhodný kontext.

[1] https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

Dokonce je možné zaměřit se na konkrétní informace a nebo na globální rysy věty. Lépe se poté uchovává celkový význam a sémantika vstupního kontextu.

Attention mechanismus je zde proveden tak, že výstupní vektor z *embedding* vrstvy je předán *dense* vrstvě, která využívá aktivační funkci *tanh* (viz 4.3). Tato aktivační funkce převede reálná čísla na čísla v rozsahu -1 až 1. Poté je výstup této vrstvy předán do další vrstvy nazývané *flatten*, která zarovnává vektor na vektor s dimenzí o velikosti 30, tedy o velikosti maximálního počtu tokenů. Tento zarovnaný vektor je dále poslán do aktivační funkce *softmax* (viz 4.8). Společně vrstvami *RepeatVector* a *Permute* je vektor transformován na vektor s dimenzí o velikost *word embedding* – tedy 300 a vložen do LSTM encoder vrstvy.

Na závěr dojde ke sjednocení výstupů LSTM encoder a LSTM decoder vrstev a výstup je poslán do *dense* vrstvy, která transformuje vektor na vektor s dimenzí o velikosti poloviny slovníku. Poslední vrstvou je opět vrstva *dense*, která transformuje výstup z předešlé vrstvy na výstupní vektor s dimenzí o velikosti slovníku.



Obrázek 6.3: Složení třetího modelu (vygenerováno pomocí funkce *plot_model* z balíčku *keras.utils*).

6.5 Trénování

6.5.1 Rozdělení dat na trénovací a validační

Vstupní a odpovídající výstupní data jsou náhodně rozdělena na tréninková data, která budou v každé epoše použita k tréninku modelu, a na data validační. Jsou používána k validování modelu po každé epoše tréninku. Na validačních datech není model trénován, tudíž musí generalizovat. Tato data tvoří 5 % z celkového množství trénovacích dat, ale maximální počet vzorků může být omezen kvůli urychlení tréninku. Omezení je nastaveno na 1000 vzorků dat. Validační data jsou vybrána z trénovacích dat ještě před tréninkem a model na nich tedy není trénován.

Rozdělení dat je provedeno pomocí funkce *train_test_split* z knihovny *sklearn* (konkrétně z balíčku sklearn.model_selection). Funkce přijímá parametry vstupních a výstupních dat a desetinné číslo, které procentuálně určuje výslednou část validačních dat pro vstupní i výstupní data, nebo celé číslo, které určuje přesný počet validačních vzorků.

Poté je stejným způsobem ještě vybráno stejné množství trénovacích dat, která budou v každé epoše sloužit k měření kvality modelu. Tato data jsou ale použita i k tréninku, a jedná se tedy o schopnost modelu tvořit odpověď na kontexty z trénovacích dat. Data jsou také vybrána ještě před tréninkem a po každé epoše jsou tedy použita stejná data.

6.5.2 Trénink modelu

Doba trvání trénování je dána počtem nastavených epoch, které jsou nastaveny ještě před začátkem trénování. Další možností je ukončení trénování zásahem uživatele. Díky ukládání natrénovaných vah do souboru je možné pokračovat v trénování modelu od poslední trénovací epochy. Implementace je určena pro neočekávané ukončení trénovaní, aby se nemuselo začínat trénování od nuly.

Každá epocha je dále rozdělena na menší části. V každé části jsou dány do neuronové sítě dva vstupy. Prvním vstupem je sekvence indexů. Každý index je jednotlivý token ve slovníku. Druhým vstupem je část odpovědi. Výstupem neuronové sítě je částečná odpověď, ve které je přidán další vygenerovaný token rozšiřující odpověď. Poté je vypočtena hodnota ztrátové funkce a přesnost optimalizačního nástroje.

Na konci každé epochy je možné vypočítat hodnotu testovací metriky BLEU. Výsledné váhy jsou následně uloženy do souboru.

6.5.3 Chatování

Před začátkem každého konverzování je nutné natrénovat model chatbota a uložit natrénované váhy do souborů. Tyto váhy jsou dále načteny a použity pro predikování odpovědí. Není-li nalezen soubor s uloženými váhami, obdrží uživatel chybovou hlášku a program se ukončí. Také je možné, aby si uživatel zvolil, zda chce konverzovat pomocí terminálu nebo jestli chce k chatování využít GUI. Při výchozím nastavení je užito chatování pomocí terminálu.

Po načtení vah dojde také k načtení slovníku, indexů, a také k vytvoření zvoleného modelu chatbota, který je využit pro generování odpovědi.

Při volbě konverzování pomocí terminálu dojde na začátku konverzace k vypsání informační hlášky, která pomáhá uživateli s používáním chatbota. Také je možné zadat během konverzace zadat klíčová slova, která vyvolají speciální akci od chatbota.

HELP or INFO Zadáme-li jedno z těchto slov, dojde k vypsání zprávy, která obsahuje informace týkající se práce s chatbotem.

BYE Toto klíčové slovo ukončí konverzaci s chatbotem. Také je možné zadat i prázdný řetězec.

Pokud je zadáno jakékoliv jiné slovo, dojde ke generování odpovědi modelu chatbota. V tomto ohledu se chatování přes terminál a chatování pomocí GUI neliší.

U generování odpovědi dochází nejprve k vytvoření tokenů ze vstupu a jeho ořezání podle maximální velikosti vstupu. Následovně jsou všechny tokeny nahrazeny svým indexem ve slovníku.

Prvním tokenem odpovědi je speciální token BOS[1], který je uložen na poslední pozici v sekvenci indexů tokenů u odpovědi. Zbytek sekvence indexů je vynulován pomocí funkce *np.zeros()*. Poté je vygenerován následující index tokenu odpovědi pomocí funkce *model.predict()*. Tímto iteračním způsobem se sestaví celá sekvence indexů reprezentující jednotlivé tokeny odpovědi.

Protože je odpověď chatbota právě pouze ve formě indexů, je nutné celý tento řetězec projít a jednotlivé indexy jsou nahrazeny jednotlivými tokeny.

Je-li odpověď dána jednotlivými tokeny, přichází na řadu převedení této sekvence tokenů na odpověď, která je předána na výstup uživateli. Postupně se prochází touto sekvencí tokenů. Pokud se jedná o speciální token BOS[2], dojde k jeho přeskočení a pokračuje se dále. Naopak je-li nalezen speciální token EOS[2], dojde k ukončení převodu tokenů na odpověď a následnému výpisu. Aby chatbot působil více věrohodně, je v tomto převodu i zahrnuto psaní velkých písmen na začátku slov, u kterých je to vyžadováno. Jedná se o slova na začátku vět či jmen. Realizace řešení tohoto problému je jednoduchá. Pokud je daný token v seznamu jmen, dojde k zavolání funkce *capitalize()*, která převede první písmeno ve slově na velké. Dále se u tohoto problému využívá zapamatování předposledního tokenu. Je-li předposlední token nalezen v seznamu ukončovacích tokenů „.“, „!“, „?“, je taktéž na token zavolána funkce *capitalize()*.

Všechny tokeny jsou odděleny mezerou. Výjimkou jsou speciální tokeny, po kterých se mezera nepíše, a tokeny, před kterými se mezera nepíše. Speciálními tokeny, za kterými se čárka nepíše, jsou „,“, „.“, „..“, „...“, „!“, „?“, „:“, „;“, „’“, „″“, „)“, „]“, „}“. Druhým typem jsou speciální tokeny „(“, „[“, „{“, „‘“, „″“.

Po úpravě sekvence tokenů tím, že se odstraní speciální tokeny BOS[1] a EOS[2], nahradí se počáteční písmena tokenů a úpraví mezery, je sekvence připravena na to být použita jako odpověď, která se vypíše uživateli.

Příklad chatu

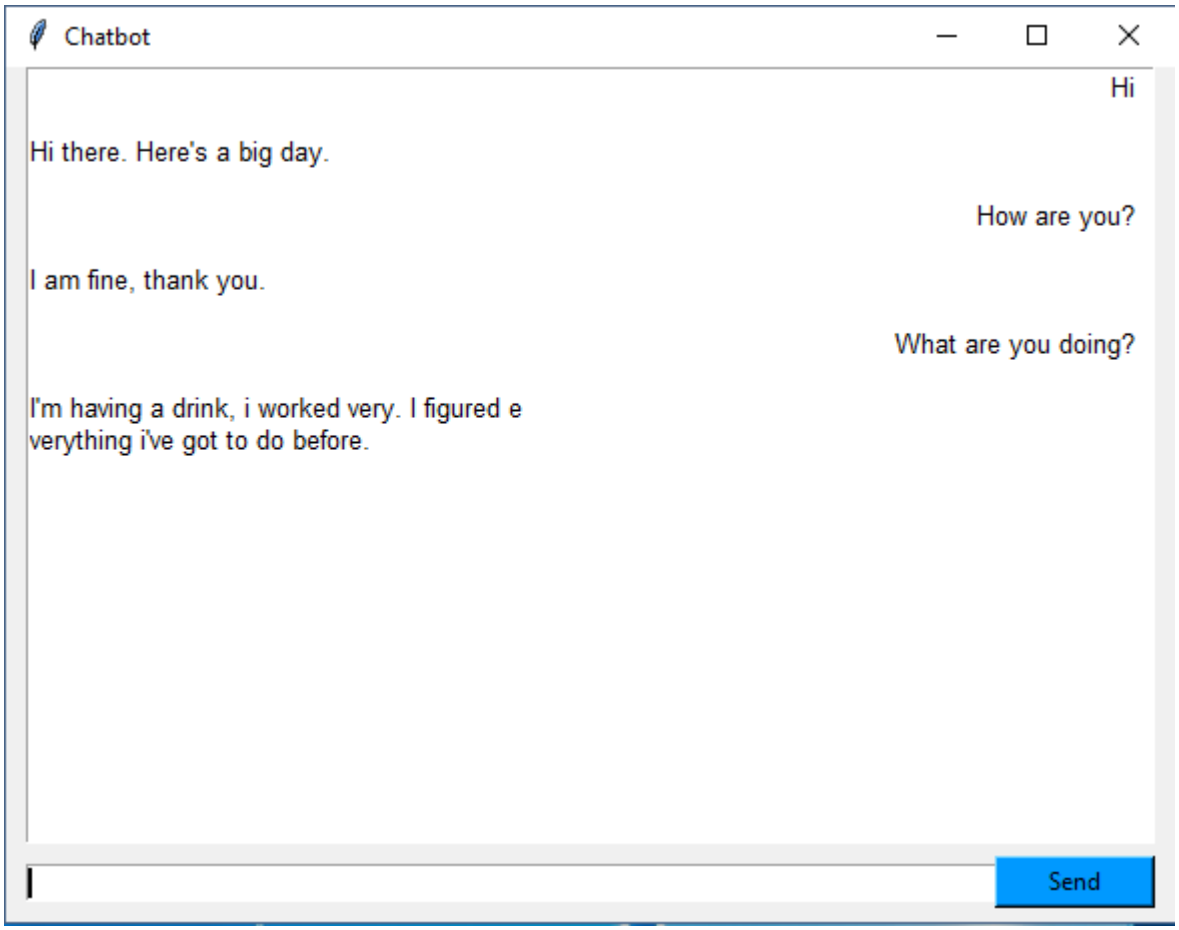
uživatel: Hi
chatbot: Hi there. Here’s a big day.
uživatel: How are you?
chatbot: I am fine, thank you.
uživatel: How old are you?
chatbot: I’m not a kid.
uživatel: Do you have any siblings?
chatbot: No.

[1] Beginning of sentence
[2] End of sentence

uživatel: Where do you live?
chatbot: Oh, uptown.
uživatel: What are you doing?
chatbot: I'm having a drink, i worked very. I figured everything i've got to do before.
uživatel: Where are you?
chatbot: I'm getting a drink.
uživatel: Do you have any hobbies??
chatbot: No.
uživatel: Are you a boy or a girl?
chatbot: I am a boy.
uživatel: Bye.
chatbot: I look forward to our next meeting.

Chat GUI

GUI je tvořeno třemi částmi – textové pole, tlačítko pro odeslání, okno pro výpis zpráv. Textové pole slouží k zadávání zpráv od uživatele. Pokud chce uživatel odeslat zprávu, stačí stisknout *enter* nebo zmáčknout tlačítko pro odeslání. Okno pro výpis zpráv je rozděleno na dvě poloviny. V pravé části se ukazují odpovědi od chatbota a v levé jsou zprávy uživatele.



Obrázek 6.4: Obrázek vzhledu GUI.

6.6 Testování

V této části je vysvětleno testování chatbota na metrice BLEU (viz 5.3) a popsáno testování na množině uživatelů, kteří byli pověřeni konverzováním s chatbotem a následným zodpovězením otázek hodnotících kvalitu chatbota.

6.6.1 Testování pomocí BLEU

Jak bylo vysvětleno v části týkající se metriky BLEU (viz 5.3), je nutné získat pro BLEU metriku kandidáta a referenční text, podle kterých se bude tato metrika počítat. Tuto metriku lze počítat buď během trénování modelu, nebo po natrénování spočítáním z uložených vah. Kandidát i referenční text se v každém typu počítání načítají jiným způsobem.

V případě počítání metriky po natrénování chatbota jsou nejdříve načteny poslední natrénované váhy ze souboru. Poté jsou načtena testovací data ze dvou souborů. Těmito soubory jsou *test.enc* a *test.dec*. Důvodem použití dvou souborů je nutnost načtení dat pro *encoder* a *decoder* neuronové sítě. Po načtení všech těchto dat začíná samotné počítání metriky BLEU. Na referenční text jsou použita data uložená v souboru *test.dec*. Kandidátem v tomto případě je predikovaná odpověď chatbota bez speciálních tokenů. U druhého případu, kterým je testování během trénování modelu, nejsou testovací data načítána z těchto dvou souborů. Jsou použita data, která jsou získána pomocí funkce *train_test_split()*. Tato funkce vybere data pro testování z dat určených pro trénování a ta jsou dále použita jako referenční text. Kandidát je i zde stejný jako u počítání metriky z uložených vah.

Další části počítání metriky BLEU jsou pro oba dva případy společné. Je nutné vypočítat váhy pro počítání této metriky. Pokud velikost délky referenčního textu a zároveň velikost délky kandidáta je větší než 4 tokeny, tak jsou váhy nastaveny následovně: *weights=(0.25, 0.25, 0.25, 0.25)*. Pokud je alespoň jedna z těchto velikostí menší než 4, je potřeba tyto váhy upravit. Jsou-li obě velikosti rovny 0, tak BLEU skóre je 1 – kandidát a referenční text jsou si „rovni“. Pokud je jedna z těchto dvou velikostí rovna 0, tak váhy jsou nastaveny takto: *weights=(1,)*. Poté mohou nastat už jen dvě možnosti – velikost délky kandidáta je menší než velikost délky referenčního textu a naopak. Pak je nastavení vah následující: *weights=((1 / [menší velikost],) * [menší velikost])*.

Po nastavení váhy se spočítá BLEU skóre pomocí funkce *sentence_bleu()*, která je importovaná z knihovny *nltk.translate.bleu_score*.

6.6.2 Testování pomocí uživatelů

Chatbot byl testován na deseti uživatelích. Každý uživatel konverzoval se všemi třemi modely, na kterých je chatbot natrénován, dále s volně dostupným chatbotem pro porovnání kvality konverzace a na závěr byly uživateli položeny otázky, na které musel odpovědět. Otázky jsou uvedeny v sekci 7.1.

Kapitola 7

Výsledky testování

V této kapitole jsou popsány výsledky z testování.

Pro testování byly použity všechny tři modely chatbota. U všech bylo použito pro trénování 172 362 vzorků dat. Z nich bylo ještě před začátkem odebráno 1 000 vzorků, které nebyly použity pro trénování modelů. Důvodem je možné použití pro validování modelu pomocí metriky BLEU. Všechny modely byly taktéž natrénovány na 100 epochách, a jak již bylo zmíněno v sekci 6.4 týkající se vytváření modelů, byla u trénování použita ztrátová funkce a optimalizační nástroj.

První model

Po 100 epochách byla *loss functions*, která byla získána pomocí ztrátové funkce *categorical crossentropy*, rovna 0.5558 (viz graf 7.1) a přesnost vypočtená pomocí optimalizačního nástroje *Adam* je 0.7853 (viz graf 7.2). Při testování na testovacích datech byl výsledek testovací metriky BLEU roven 0.1131.

Příklad chatu prvního modelu

uživatel: Hi
chatbot: Hi there. Here's a big day.
uživatel: How are you?
chatbot: I am fine, thank you.
uživatel: How old are you?
chatbot: I'm not a kid.
uživatel: Do you have any siblings?
chatbot: No.
uživatel: Where do you live?
chatbot: Oh, uptown.
uživatel: What are you doing?
chatbot: I'm having a drink, i worked very. I figured everything i've got to do before.
uživatel: Where are you?
chatbot: I'm getting a drink.
uživatel: Do you have any hobbies?
chatbot: No.
uživatel: Are you a boy or a girl?
chatbot: No, i'm too old for you. Ca n't tell the truth.
uživatel: Where are you from?

chatbot: Queens, new york city.
uživatel: Are you a student?
chatbot: Why, yes, of course. But i always have a choice for one – with a box.
uživatel: Do you do sports?
chatbot: No time.
uživatel: Bye.
chatbot: I look forward to our next meeting.

Druhý model

U tohoto modelu byla po 100 epochách *loss functions*, která byla získána pomocí ztrátové funkce *categorical crossentropy*, rovna 0.1658 (viz graf 7.1) a přesnost vypočtená pomocí optimalizačního nástroje *Adam* je 0.88825 (viz graf 7.2). Při testování na testovacích datech byl výsledek testovací metriky BLEU roven 0.1127.

Příklad chatu druhého modelu

uživatel: Hi
chatbot: Hi... happy.
uživatel: How are you?
chatbot: I got fired.
uživatel: How old are you?
chatbot: .
uživatel: Do you have any siblings?
chatbot: Very much.
uživatel: Where do you live?
chatbot: Orange, school.
uživatel: What are you doing?
chatbot: Nothing... that's... that's singer, is n't it? Goose turn around, pay attention. What are you doing?
uživatel: Where are you?
chatbot: I finished for school or. I thought i'd remember.
uživatel: Do you have any hobbies?
chatbot: Very much.
uživatel: Are you a boy or a girl?
chatbot: No, i'm too old for you. Ca n't tell the truth.
uživatel: Where are you from?
chatbot: New york.
uživatel: Are you a student?
chatbot: Not exactly.
uživatel: Do you do sports?
chatbot: Sure.
uživatel: Bye.
chatbot: I look forward to our next meeting.

Třetí model

U třetího modelu byly po 100 epochách hodnoty *loss functions*, optimalizačního nástroje a testovací metriky následující: výsledek funkce *categorical crossentropy* je 1.0402 (viz graf 7.1), pro nástroj *Adam* tato hodnota činí 0.7143 (viz graf 7.2) a posledním je výsledek metriky BLEU 0.1117.

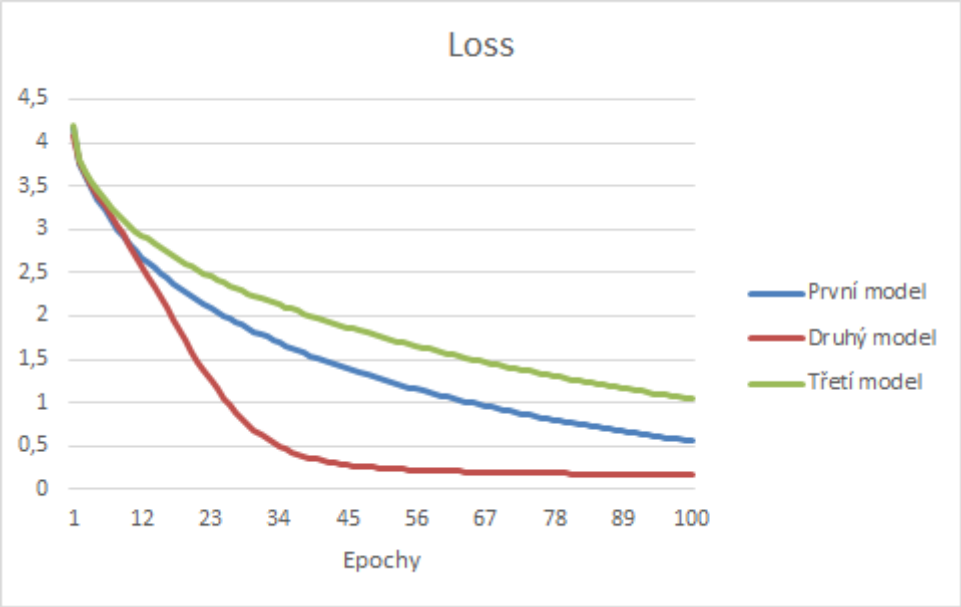
Příklad chatu třetího modelu

uživatel: Hi
chatbot: Hi.
uživatel: How are you?
chatbot: I got fired.
uživatel: How old are you?
chatbot: Monday.
uživatel: Do you have any siblings?
chatbot: Yes.
uživatel: Where do you live?
chatbot: Queens, new york.
uživatel: What are you doing?
chatbot: I'm having a drink, i bet.
uživatel: Where are you?
chatbot: Nowhere.
uživatel: Do you have any hobbies?
chatbot: Yes.
uživatel: Are you a boy or a girl?
chatbot: Yes.
uživatel: Where are you from?
chatbot: I'm in.
uživatel: Are you a student?
chatbot: No.
uživatel: Do you do sports?
chatbot: I am on the, too.
uživatel: Bye.
chatbot: Have a nice day!

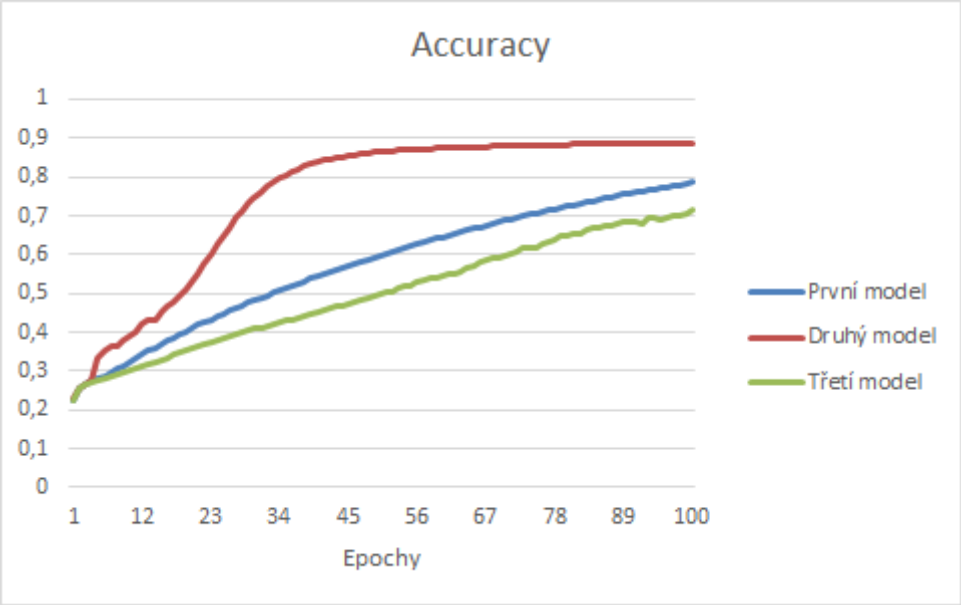
Příklad chatu s CleverBotem

uživatel: Hi
chatbot: How are you?
uživatel: How are you?
chatbot: I am.
uživatel: How old are you?
chatbot: 20.
uživatel: Do you have any siblings?
chatbot: Yes, you?
uživatel: Where do you live?
chatbot: How many times are you going to ask that?

uživatel: What are you doing?
chatbot: I watching tv.
uživatel: Where are you?
chatbot: I am at home.
uživatel: Do you have any hobbies??
chatbot: I like to play videogames.
uživatel: Are you a boy or a girl?
chatbot: I'm a boy, yeah.
uživatel: Where are you from?
chatbot: I just told you.
uživatel: Are you a student?
chatbot: No, not going to school, ever...
uživatel: Do you do sports?
chatbot: I do badminton.
uživatel: Bye.
chatbot: Love you babe.



Obrázek 7.1: Graf znázorňující průběh výpočtu ztrátové funkce během trénování všech tří modelů. Graf byl vytvořen pomocí aplikace Microsoft Excel.



Obrázek 7.2: Graf znázorňující průběh výpočtu optimalizátoru Adam během trénování všech tří modelů. Graf byl vytvořen pomocí aplikace Microsoft Excel.

7.1 Testování na uživatelích

Otázky a jejich popis

1. **Jak chatbot reaguje na otázky?** Tato otázka zjišťuje, jaký pocit mají uživatelé z kvality odpovídání chatbota. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, 5 značí špatný výsledek. Viz graf 7.3.

2. **Jak kvalitní jsou krátké odpovědi chatbota? (méně než 10 slov)** Druhá otázka zjišťuje kvalitu krátkých odpovědí. Tedy odpovědí, které mají méně než 10 tokenů. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, naopak 5 špatný výsledek. Viz graf 7.4.

3. **Jak kvalitní jsou dlouhé odpovědi chatbota? (více než 10 slov)** Třetí otázka zjišťuje kvalitu dlouhých odpovědí. Tedy odpovědí, které mají více než 10 tokenů. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, 5 značí špatný výsledek. Viz graf 7.5.

4. **Odpovídá chatbot s chybami?** Předposlední otázka se snaží zjistit, zda je chatbot srozumitelný – jestli odpovídá s gramatickými chybami. Viz graf 7.6.

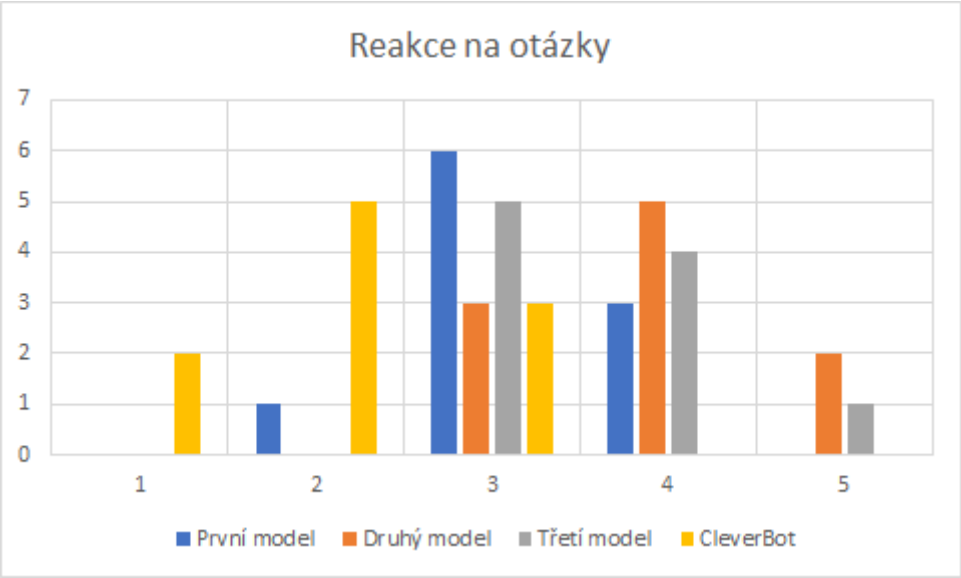
   (a) **Žádné chyby** Konverzace proběhla bez gramatických chyb.

   (b) **Občas odpoví s chybou** V konverzaci se občas vyskytne gramatická chyba, ale není to nic závažného.

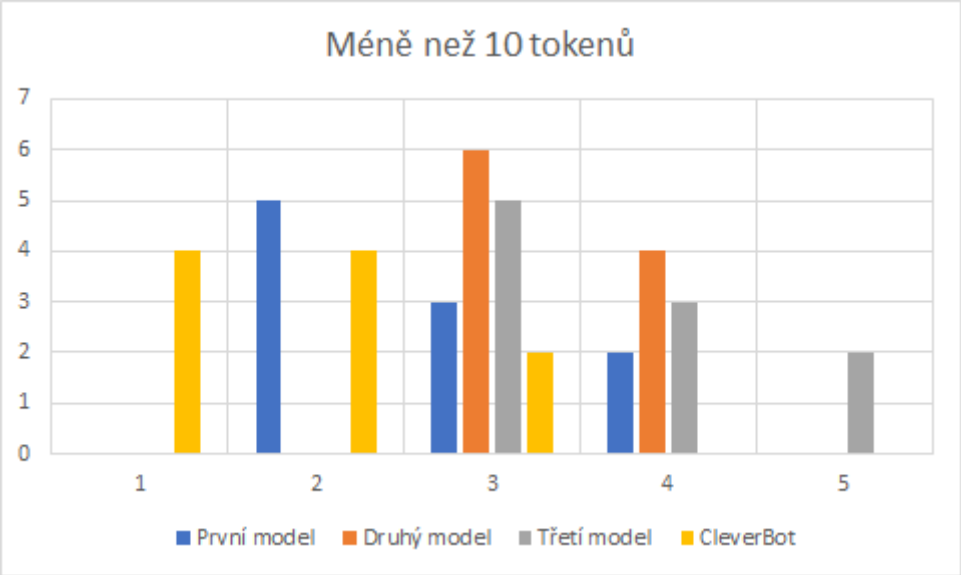
   (c) **Často odpovídá s chybami** V konverzaci se často vyskytují gramatické chyby, ale není stále je chatbotovi rozumět.

   (d) **Kvůli chybám je obtížné s chatbotem konverzovat** Chatbot dělá tolik gramatických chyb, že se s ním nelze konverzovat.

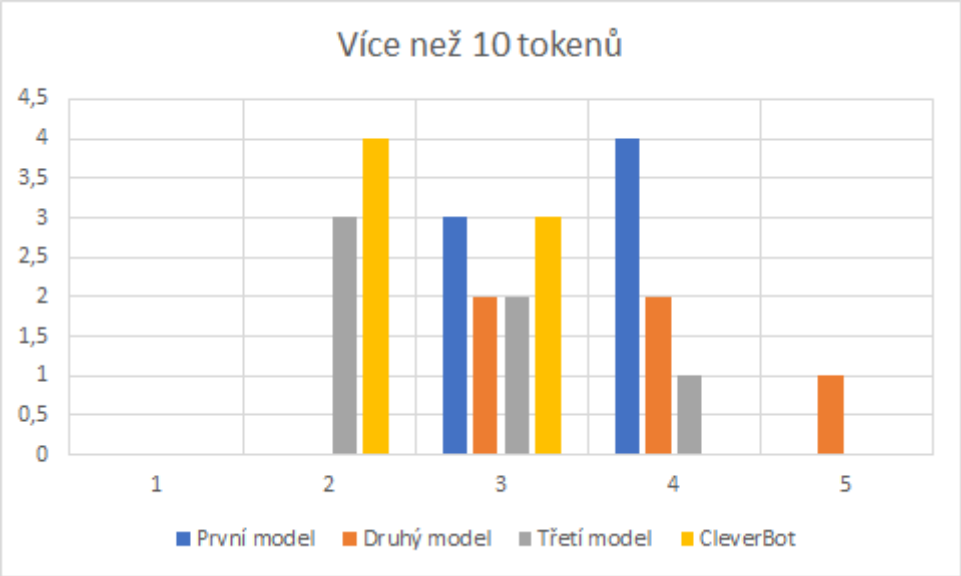
5. **Shrnutí pocitů z konverzace?** Slovní ohodnocení všech modelů chatbota. Viz 7.1.



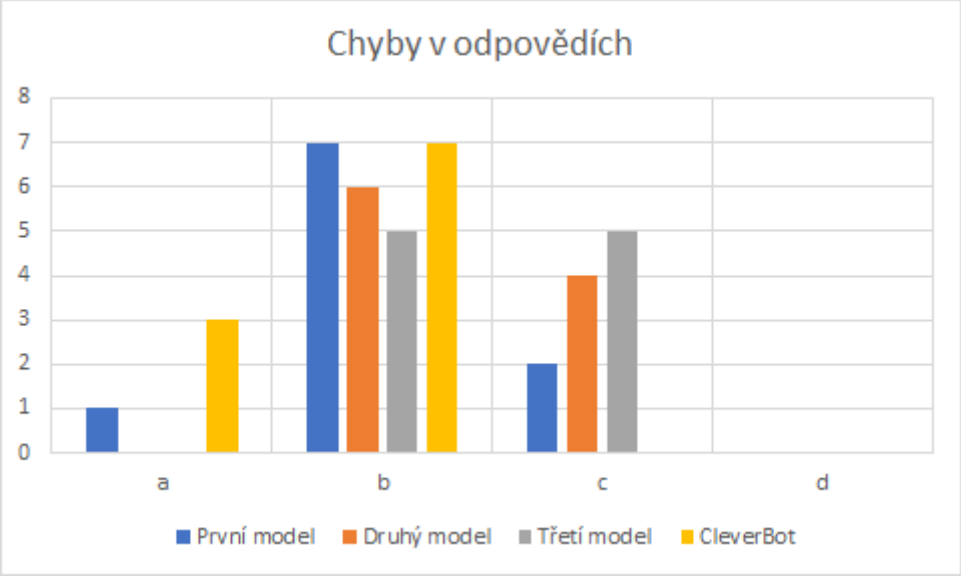
Obrázek 7.3: Graf ukazující výsledek odpovědí na otázku, jaký pocit mají mají uživatelé z kvality odpovídání chatbota. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, 5 značí špatný výsledek. Graf byl vytvořen pomocí Microsoft Excel.



Obrázek 7.4: Graf znázorňující odpovědi na otázku, jak kvalitní jsou krátké odpovědi chatbota? (méně než 10 slov). Tedy odpovědí, které mají méně než 10 tokenů. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, 5 značí špatný výsledek. Graf byl vytvořen pomocí Microsoft Excel.



Obrázek 7.5: Graf pro ukázku odpovědí na otázku, jak kvalitní jsou dlouhé odpovědi chatbota? (více než 10 slov). Tedy odpovědí, které mají více než 10 tokenů. Ohodnocení je stejné jako známky ve škole. 1 skvělý výsledek, 5 značí špatný výsledek. Graf byl vytvořen pomocí Microsoft Excel.



Obrázek 7.6: Graf, který ukazuje, jak často chatbot odpovídá s chybami (a) Žádné chyby (b) Občas odpoví s chybou (c) Často odpovídá s chybami (d) Kvůli chybám je obtížné s chatbotem konverzovat. Graf byl vytvořen pomocí Microsoft Excel.

Vyhodnocení páté otázky

Uživatelům připadalo konverzování s chatbotem převážně zajímavé. I přestože občas odpověděl s chybou. Z výsledků hodnocení vyplývá, že první model nejlépe odpovídá, pokud jde o krátké věty (méně než 10 tokenů), druhý model je nejslabší ve všech ohledech a třetí

model je nejlepší na dlouhé odopovědi (více než 10 tokenů). Všichni uživatelé se bohužel shodli na tom, že porovnávací CleverBot je prozatím lepší než jakýkoliv model.

Závěr testováni

Všechny tři modely získaly při testování metrikou BLEU velmi podobné výsledky. První model získal výsledek 0.1131, druhý model 0.1127 a třetí model měl výsledek 0.1117. Tedy kvalita odpovědí modelů je srovnatelná.

Testováním na uživatelích se ukázalo, že CleverBot generuje odpovědi mnohem lépe než jakýkoliv model chatbota. CleverBot odpovídá srozumitelněji a neobsahuje tolik gramatických chyb. I když všechny tři modely mají na testovací metrice BLEU podobné skóre, jejich odpovědi mají rozdílnou kvalitu. První model je nejlepší při používání kratších odpovědí, třetí model je nejlepší při vytváření delších odpovědí. Druhý model vyšel z hodnocení nejhůře.

Tabulka BLEU výsledků

	100 epoch
První model:	0.1131
Druhý model:	0.1127
Třetí model:	0.1117

Kapitola 8

Závěr

Výstupem této bakalářské práce je chatbot implementovaný pomocí neuronových sítí. Práce charakterizuje neuronové sítě a podává obecné informace o fungování všech typů chatbotů. Chatbot je naučen odpovídat tak, aby dokázal konverzovat s uživateli. Cílem této bakalářské práce však nebylo pouze implementovat chatbota, ale také ho otestovat a porovnat s jiným volně dostupný chatbotem.

Chatbot je schopný konverzovat s uživateli na téma, na které byl trénován. Pokud uživatelé změní téma a ptají se ho na informace, které nebyly obsaženy v trénovacích datech, může mít problémy s odpovědí. Testování bylo prováděno pomocí metriky BLEU a testováním na uživatelích. Pro robustnější testování by bylo vhodné použít více testovacích metrik, ne jen jednu, kterou je metrika BLEU. Současně by tyto další metriky měly mít odlišný testovací algoritmus, aby byl chatbot otestován co nejlépe. Dalším vylepšením testování by bylo testování na více uživatelích. Větší počet testovacích subjektů dá přesnější vyhodnocení. Také by bylo vhodné, aby uživatelé testovali chatbota v klidném, nerušeném prostředí a aby měli na testování dostatek času. Nejlepší by bylo, aby testovací uživatelé konverzovali s chatbotem tak dlouho, jak sami uváží za vhodné.

Z testování vyplynulo, že žádný model tohoto chatbota nedosahuje kvality *CleverBota*, se kterým byl srovnáván.

Tohoto chatbota by bylo možné nadále zlepšovat. Mohli bychom vytvořit další modely pro trénování a také by bylo dobré zvětšit slovník a přidat další trénovací data.

Lidem, kteří se zajímají o neuronové sítě či chatboty, bych doporučil si přečíst mou práci. Může být důležitá pro ty, kdo by chtěli vyvíjet nebo rozšiřovat podobný systém. Celkové teoretické vysvětlení neuronových sítí pomůže každému, kdo se bude zabývat neuronovými sítěmi, nemusí to být pouze chatbot. Práce může být prospěšná i těm, kdo by chtěli vytvářet *retrieval-based* chatbota.

Podle zjištěných informací vyplývá, že chatboti implementovaní pomocí neuronových sítí nejsou zatím k takovému užitku jako *retrieval-based* chatboti – ti mohou být použiti například v různých pracovištích jako náhrada za člověka. Generativní chatboti se ale stále zlepšují a jednou budou kvalitnější než rule based chatboti. Proto si myslím, že práce na generativních chatbotech má slibnou budoucnost a pokračování na této bakalářské práci zlepšováním chatbota má velký smysl.

Literatura

[1] *Keras*. [Online; navštíveno 27.4.2019].
URL https://keras.io/

[2] *Numpy*. [Online; navštíveno 27.4.2019].
URL http://www.numpy.org/

[3] *Usage of loss functions*. [Online; navštíveno 16.1.2019].
URL https://keras.io/losses/

[4] Brownlee, J.: *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. [Online; navštíveno 16.1.2019].
URL https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[5] Callison-Burch, C.; Osborne, M.; Koehn, P.: *Re-evaluating the Role of BLEU in Machine Translation Research*. [Online; navštíveno 27.4.2019].
URL https://www.aclweb.org/anthology/E06-1032

[6] Gurney, K.: *Neural networks for perceptual processing: from simulation tools to theories*. The Royal Society, 2007, ISBN 471-2970, doi:10.1098/rstb.2006.1962.

[7] He, K.; Zhang, X.; Ren, S.; aj.: Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. 2015, arXiv:1502.01852.

[8] Hunter, J.; Dale, D.; Firing, E.; aj.: *Matplotlib*. [Online; navštíveno 27.4.2019].
URL https://matplotlib.org/

[9] Mauldin, M. L.: *CHATTERBOTS, TINYMUDS, and the Turing Test Entering the Loebner Prize Competition* . [Online; navštíveno 27.4.2019].
URL http://www.aaai.org/Papers/AAAI/1994/AAAI94-003.pdf

[10] Pennington, J.; Socher, R.; Manning, C. D.: *GloVe: Global Vectors for Word Representation*. [Online; navštíveno 27.4.2019].
URL https://nlp.stanford.edu/pubs/glove.pdf

[11] Schmidhuber, J.: *Recurrent Neural Networks*. [Online; navštíveno 19.12.2017].
URL http://people.idsia.ch/~juergen/rnn.html

[12] Sutskever, I.; Vinyals, O.; Le, Q. V.: *BLEU: a Method for Automatic Evaluation of Machine Translation*. [Online; navštíveno 27.4.2019].
URL https://www.aclweb.org/anthology/P02-1040

[13] Sutskever, I.; Vinyals, O.; Le, Q. V.: *Sequence to Sequence Learning with Neural Networks*. [Online; navštíveno 27.4.2019].
URL http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[14] Turing, A. M.: *COMPUTING MACHINERY AND INTELLIGENCE*. [Online; navštíveno 27.4.2019].
URL https://www.csee.umbc.edu/courses/471/papers/turing.pdf

[15] Wallace, R.: *The Anatomy of A.L.I.C.E.* Springer, 2009, ISBN 978-1-4020-6710-5, doi:https://doi.org/10.1007/978-1-4020-6710-5_13.

[16] Weizenbaum, J.: *ELIZA—a computer program for the study of natural language communication between man and machine* . [Online; navštíveno 27.4.2019].
URL https://dl.acm.org/citation.cfm?doid=365153.365168

[17] Wu, Y.; Wu, W.; Li, Z.; aj.: Response Selection with Topic Clues for Retrieval-based Chatbots. 2016, arXiv:1605.00090.

[18] Wu, Y.; Wu, W.; Xing, C.; aj.: Sequential Matching Network: A New Architecture for Multi-turn Response Selection in Retrieval-based Chatbots. 2016, arXiv:1612.01627.

[19] Zhang, G.; Patuwo, B.; Hu, M. Y.: *Forecasting with artificial neural networks:: The state of the art*. Graduate School of Management, Kent State University, Kent, Ohio 44242-0001, USA, 1998, doi:10.1016/S0169-2070(97)00044-7.

[20] Čechák, J.: *Chatbot postavený na umělých neuronových sítích*. bachelor thesis, Vysoké učení technické v Brně, 2018.

Příloha A

Obsah přiloženého paměťového média

Paměťové médium obsahuje tyto soubory a adresáře:

- Adresář *source* obsahuje zdrojové a další soubory chatbota
- Adresář *Latex* obsahuje kódy k vytvoření technické zprávy
- Adresář *text* obsahuje textové soubory bakalářské práce – dokumentace, technická zpráva, ukázka konverzace
- Soubor *plakat.pdf* obsahuje vytvořený plakát
- Soubor *video.mp4* obsahuje natočené video chatbota

Příloha B

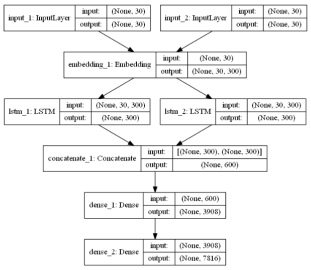
Plakát

Chatbot implementovaný pomocí neuronových sítí

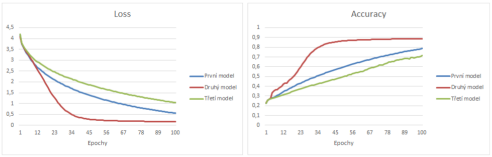
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Autor: Petr Červíček
Vedoucí: Ing. Igor Szőke, Ph.D.
Bakalářská práce, 2019

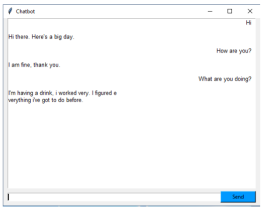
Natrénování modelů neuronové sítě na trénovacích datech tak, aby systém byl schopný konverzace.



Ukázka architerktury jednoho ze tří modelů neuronové sítě, která funguje na principu sequence-to-sequence a je zde použita neuronová síť typu Long short term memory.



Grafy znázorňující změny hodnot funkce loss function a optimalizátoru jménem Adam během trénování všech tří modelů na 100 epochách.



Ukázka GUI aplikace pro komunikace mezi chatbotem a uživatelem.