VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHATBOT ZALOŽENÝ NA JAZYKOVÉM MODELOVÁNÍ

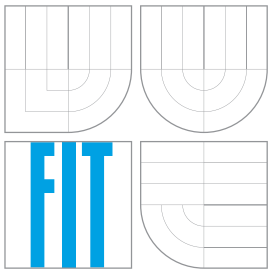
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MATĚJ RADVANSKÝ

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CHATBOT ZALOŽENÝ NA JAZYKOVÉM MODELOVÁNÍ

CHATBOT BASED ON LANGUAGE MODELLING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MATĚJ RADVANSKÝ
AUTHOR

VEDOUCÍ PRÁCE — Ing. FRANTIŠEK SKÁLA
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce řeší využití jazykového modelování v chatbotovi pomocí neuronových sítí. Tento problém je řešen tak, že pomocí zpracování přirozeného jazyka je při vytváření odpovědi na uživatelskou zprávu prvně zpráva analyzována. Poté jsou vytvořeny počátky vět odpovědi, které jsou doplněny výstupem neuronové sítě. Dohromady tyto věty tvoří odpověď chatbota. Proběhlo porovnání s existujícím chatbotem Cleverbotem a byla zjišťována míra inteligence obou chatbotů, na jejichž základě jsou uvedeny možnosti dalšího vývoje.

Abstract

This paper addresses the use of language modeling using neural networks in the chatbot. Problem is solved by using natural language processing and first step of generating response based on user input is input analysiss. As next beginings of sentences are created which are completed by output of neural network. All created sentences form final chatbot response. There was a comparisson with chatbot Cleverbot and measure of intelligence for both chatbots was determined. Based on testing results, some techniques for future progress were concluded.

Klíčová slova

Chatbot, jazykové modelování, jazykové modely, neuronové sítě, POS, zpracování přirozeného jazyka

Keywords

Chatbot, language modelling, language models, neural networks, POS, natural language modelling

Citace

Matěj Radvanský: Chatbot založený na jazykovém modelování, bakalářská práce, Brno, FIT VUT v Brně, 2014

Chatbot založený na jazykovém modelování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Františka Skály. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Matěj Radvanský
21. května 2014

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Františku Skálovi, který mě směroval při propojování jednotlivých částí práce a poskytl odbornou pomoc z hlediska neuronových sítí a jazykového modelování.

Obsah

Kapitola 1

Úvod

Chatbot je program, jehož účelem je simulování lidského vyjadřování většinou formou textu. Programy tohoto typu lze využít pro nasazení do helpdesku ve firmách, offline pomoc nebo jako formu zábavy. Zatím neexistuje chatbot, který by dokázal plně simulovat lidské chování, avšak existují různé techniky chatbotů s odlišnými technikami, kterými se tomu snaží co nejvíce přiblížit. Cílem práce je vytvoření aplikace, která bude na uživatelský vstup reagovat s co největší mírou inteligence.

V první části práce je uveden popis vybraných chatbotů z hlediska jejich technik. Na to navazuje popis zpracování přirozeného jazyka, které umožňuje označkovat text pomocí slovních druhů a definovat pravidla pro gramatické časy, což bude dále využito pro vytváření počátků vět, které budou tvořit reakci chatbota.

Další část se zaměřuje na jazykové modelování, kde je prvně popsána základní idea vytváření jazykových modelů, následovaná uvedením do neuronových sítí a poté jejich využití v jazykovém modelování. Z hlediska chatbota se jedná o doplnění počátků vět z předchozí části na základě jazykového modelu, který vznikl trénováním neuronové sítě na trénovacích datech.

Jádrem práce je zkombinování technik, které využívají ostatní chatboti, s jazykovým modelováním pomocí neuronových sítí. Toho je dosaženo tak, že slovům v textu jsou přiděleny značky pro příslušné slovní druhy a na jejich základě probíhá transformace zájmen a změna slovosledu, pokud je to možné. Dále jsou vybrána klíčová slova, probíhá reakce na různé podněty jako jsou pozdravy, případně vlastní jméno. Při generování odpovědi se chatbot snaží brát v potaz délku podnětu a podle toho vytváří počet vět reakce.

Kapitola 2

Uvedení do problematiky chatbotů a zpracování přirozeného jazyka

Chatboty lze použít pro poskytování relevantních informací ohledně konkrétního tématu, ale také je lze využít pro konverzaci bez specifického účelu, kdy už nejde o zjišťování konkrétních informací, ale hlavně o konverzaci jako takovou. V optimálním případě by chování chatbota nemělo jít rozeznat od chování člověka. K určení míry inteligence slouží obvykle Turingův test. Test funguje tak, že lidský hodnotitel komunikuje současně s programem, který je testován, a jiným člověkem. Hodnotitel poté rozhodne, který z nich je člověk. Na tomto testu je založena Loebnerova cena [2] a vítězem se stane ten chatbot, který je v testu nejúspěšnější.

2.1 Popis funkce vybraných chatbotů

Většina Chatbotů využívá databázi předem definovaných frází, a pro analýzu vstupní fráze využívá klíčová slova. Odpovědí je poté fráze z databáze, která nejlépe odpovídá vybraným klíčovým slovům, např. jejich počtu. Účel, za kterými byli chatboti vytvořeni, se bot od bota liší, stejně tak jako techniky, které ke své funkci využívají. V této části jsou popsáni někteří z nich.

Prvním z nich je `ELIZA` a jedná se o jednoho z prvních Chatbotů – byl vytvořen v roce 1966. Program simuloval chování psychoterapeuta a jeho fungování lze rozdělit na čtyři části - načtení uživatelského vstupu, porovnání se šablonami (vzory), nahrazení slov (transformace) jako „`I`“ za „you“ a výpis odpovědi. To je popsáno v Algoritmu 1, který byl převzat z [9]. Chování programu se odvíjí od hodnoty uživatelského vstupu v proměnné *input*. Pokud hodnota odpovídá některé z možností (vzoru), je jako odpověď programu zvolena některá ze znázorněných odpovědí. Na `ELIZU` navazuje `PARRY`, který simuluje chování paranoidního schizofrenika. Program je složitější než `ELIZA`, ale oproti ní přidává navíc i prvek nálady. Odpovědi jsou pořád generovány na základě klíčových slov, ale důvod, proč je zde tento chatbot zmiňován je ten, že navíc hraje roli i aktuální nálada (např. strach, hněv, nedůvěra) a pro různé nálady jsou definovány různé typy odpovědí.

Dalším vybraným zástupcem je `MegaHAL`, který oproti `ELIZE` využívá dynamičtější způsob vytváření odpovědi – využívá Markovův model [9]. Uživatelský vstup je rozdělen na slova a ne-slova, kdy slovo je sekvence, tvořená alfanumerickými znaky, zatímco ne-slovo je sekvence tvořena ostatními znaky. Ukládá frekvenci výskytu slov do tabulky, která je později použita k vytváření odpovědí. Poté v uživatelském vstupu hledá klíčová slova a na

Algoritmus 1 Porovnání uživatelského vstupu se šablonami u ELIZY

```
switch (input)
case "I like %s":
  output = "Why do you like %s?"
  break
case "You are %s":
  output = "Why do you say I am %s?"
  break
default:
  output = "I don't think I understand"
end switch
```

základě Markova modelu a klíčového slova vytvoří Markovův řetězec – počátek věty. Zbytek věty doplní pomocí druhého Markovova modelu. Generování odpovědí je rychlé, proto vygeneruje co nejvíce vět za dobu jedné sekundy a z nich vybere tu, která obsahuje nejvíce informací pro zvolené klíčové slovo. Výhodou je, že odpovědi nejsou šablonovité, nevýhodou pak je, že nejsou vždy gramaticky správně.

`A.L.I.C.E.` využívá `AIML` (Artificial Intelligence Markup Language), což je verze XML, která slouží pro reprezentaci probíhající konverzace. Základní jednotkou jsou kategorie, kdy každá kategorie je tvořena vzorem (se kterým je porovnáván uživatelský vstup) a k němu přiřazenou odpovědí chatbota [8]. Výhodou takového způsobu reprezentace je, že přidávání dalších vzorů a odpovědí není obtížné.

`Vrajitoru` využívá trochu jiný přístup a vychází z information retrieval systémů [8]. Ty obecně slouží k vyhledávání dokumentů na základě uživatelského vstupu (požadavku). S dokumenty i požadavky se pracuje jako s n-dimenzionálními vektory, kdy v případě požadavků je n počet tokenů (slov) v požadavku a v případě dokumentů n značí počet dokumentů, které se mají prohledat. Systém analyzuje požadavek a vrátí dokument, který odpovídá požadavku (obsahuje nejvyšší počet klíčových slov z požadavku). Pro zvýšení přesnosti odpovědi se používá indexování, které uvádí, o čem jednotlivé dokumenty jsou. Při konverzaci se chatbot chová jako vyhledávací engine – vyhledává klíčová slova v požadavku a pak najde nejvíce odpovídající odpověď v indexovaných dokumentech. Dokumenty v tomto případě reprezentují jednotlivé věty.

2.2 Zpracování přirozeného jazyka

Přirozeným jazykem [5] je myšlen jazyk, kterým mezi sebou lidé komunikují běžně v životě, tedy například čeština nebo angličtina. Tato práce je zaměřena na komunikaci chatbota v angličtině, proto i další popis bude brán z hlediska angličtiny. Zpracování přirozeného jazyka se tedy snaží o to, aby stroj dokázal přirozenému jazyku porozumět na takové úrovni, jako to dokážou lidé. Po syntaktické stránce lze definovat přirozený jazyk pomocí gramatických pravidel specifických pro daný jazyk, ale ani to nestačí k porozumění, o čem například text, který se snažíme analyzovat, vlastně je.

Z hlediska využití v chatbotovi je idea použití zpracování přirozeného jazyka taková, že po obdržení zprávy od uživatele (otázky) se jí pokusíme analyzovat pomocí způsobů popsaných v této kapitole (změnit pořadí slov, přidání jiných), abychom z otázky vytvořili kostru odpovědi (ke které bude později připojen i zbytek odpovědi). Motivační ukázka 2.1 demonstruje, jak by pro otázku (Q) v přítomném čase prostém vypadala kostra odpovědi

(A) ve stejném čase použitím slovesa z otázky a transformováním zájmena. Lze si všimnout podobnosti s vytvářením odpovědí u chatbota ELIZA popsaném v algoritmu 1, ale rozdíl je v tom, že ELIZA měla napevno definované fráze, zatímco nyní jsme k odpovědi došli jiným způsobem - označili jsme sloveso v otázce a na něm jsme postavili odpověď. Jak lze (nejen) sloveso takto označit je popsáno v dalších sekcích.

Q: Do you love me?
A: I love (2.1)

2.2.1 Part of speech tagging

Part of speech (POS) tagging proces, kdy každému slovu ve větě přiřadíme jeho slovní druh (označkujeme jej pomocí tagu). Slovní druhy jsou například podstatná jména nebo přídavná jména, v angličtině je celkem 8 slovních druhů [4]. Aby bylo možné slova ve větě označkovat, existují tagsety. Tagsety obsahují databázi slov a k nim definovaných tagů, těmi nejrozšířenějšími jsou `Brown Corpus tagset` a `Penn Treebank tagset`. Ukázka 2.2 popisuje, jak vypadá formát, ve kterém jsou slova v `Brown corpusu` uložena, tedy *slovo/tag*.

The/at Fulton/np-tl County/nn-tl Grand/jj-tl (2.2)

`Penn Treebank` [10] vychází z `Brown corpusu`, ale některé tagy přiděluje rozdílně. Vybrané tagy a příklady jejich významu jsou zobrazeny v tabulce 2.1. Z hlediska využití v této práci jsou důležité hlavně tagy pro podstatná jména (NN), zájmena (PRP) a tvary sloves (VB, VBD,...).

tag	slovní druh	příklad
MD	modální slovesa	can
NN	podstatná jména, jednotné číslo	cat
NNS	podstatná jména, možné číslo	scotches
PRP	zájmena	you
RB	příslovce	occasionally
VB	slovesa, základní forma	ask
VBD	slovesa, minulý čas prostý	dipped
VBG	slovesa, přítomný čas průběhový nebo gerundium	telegraphing
VBP	slovesa, přítomný čas prostý, kromě 3. osoby jednotného čísla	predominate

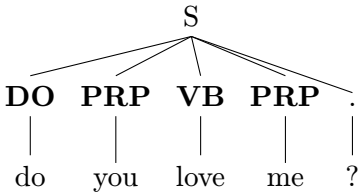
Tabulka 2.1: Vybrané tagy využitého tagsetu

Slova mohou mít přípony specifické pro POS tag daného slova, tedy například v angličtině pro příponu „`-ness`“ v kombinaci s přídavným jménem mohou vzniknout slova jako „`happiness`“ („`happy`“ → „`happiness`“) nebo „`illness`“ („`ill`“ → „`illness`“) a jiné. Taková slova jsou podstatná jména. Jiný příklad může být vytvoření podstatného jména pomocí slovesa a přípony „`-ing`“ („`fail`“ → „`failing`“), což zároveň je i průběhová forma daného slovesa.

Pro tvary sloves v angličtině platí, že jsou na ně vázána další slova, resp. POS tagy slov. V přítomném čase prostém jednotného čísla třetí osoby lze očekávat, že sloveso bude zakončeno koncovkou „`-s`“ nebo „`-es`“. Podobně sloveso v přítomném průběhovém tvaru předpokládá jako předchozí slovo tvar slovesa „`be`“ pro konkrétní osobu. POS tag slova je udáván i pozicí ve větě (přídavná jména bývají obvykle ve větě před podstatnými jmény a podobně).

Kromě označkování slov pomocí některého z tagsetů (v našem případě Penn Treebank tagsetu) jsou pro účely této práce navíc tagy pro slovesa, která mohou být použita pro tvoření otázek (například „`do`“), nahrazeny vlastním tagem pro takové sloveso (v případě slovesa „`do`“ tag *do*, v případě jeho tvaru „`does`“ tag *does*).

Obrázek 2.1 zobrazuje, jak by mohl vypadat výstup POS taggeru po označkování textu „`Do you love me?`“. Text je vhodné prvně převést na stejnou velikost písmen, souvisí to spíše s dalším zpracováním a analýzou textu, proto v tomto i dalších příkladech uvažujme zpracování textu převedeného na malá písmena. Postupujme odshora dolů. Jako S je označen původní text a tvoří kořen stromu. Na další úrovni jsou tagy vzniklé označením slov, které jsou vyznačeny tučně, a na poslední úrovni jsou původní slova z analyzovaného textu. Po označení bude strom obsahovat vždy všechna slova. Platí totiž, že pro každé neznámé slovo je nastaven tag *NN*, což odpovídá podstatnému jménu. Jsou označována i interpunkční znaménka, tedy znak „`?`“ je brán jako slovo a je označen tagem, který odpovídá interpunkčnímu znaménku.



Obrázek 2.1: Strom po označkování pro text „`Do you love me?`“

2.2.2 Chunking pomocí regulárních výrazů

Pomocí POS taggingu jsme schopni detekovat, z jakých slovních typů se věta skládá a na něj navazuje chunking. Každý chunk se skládá z jednoho nebo více označených slov (tokenů) pomocí POS taggingu. Abychom byli schopni chunky vytvářet, je potřeba definovat, z jakých označených tokenů se mají chunky skládat. K tomu slouží chunk grammar. Ta je tvořena sadou pravidel (regulárních výrazů), kdy každé z nich udává, jaká je očekávána posloupnost označených tokenů.

Po aplikaci chunkingu nad konkrétní větou je výsledkem strom, který reprezentuje větu na úrovni chunků. Poté při průchodu takovým stromem lze vybrat části, které odpovídají daným pravidlům. Pokud tedy budeme definovat pravidla jako gramatické časy v angličtině, můžeme text rozdělit na věty, které odpovídají pravidlům pro jednotlivé časy. Uvažujme pravidlo pro přítomný čas definované následujícím způsobem [1]:

```
simple_present: (<do>|<does>)(<PRP>|<NNS>|<NN>)
                (<VB>|<have>|<do>|<am>|<VBP>|<VBZ>)
```

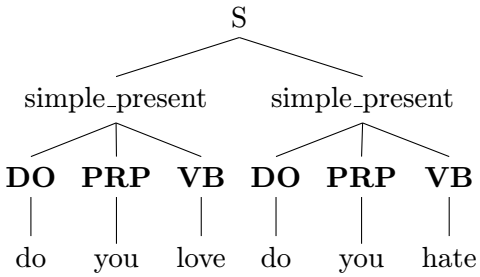
Toto pravidlo má označení *simple_present* a využívá POS tagy popsané tabulkou 2.1 a navíc tagy pro slovesa vytvářející otázku v daném čase (*do*, *does*), nazývejme je pomocná slovesa. Předpokládejme text: „`Do you love me?`“ a na něj aplikujme pravidlo *simple_present*. Odpovídající si úseky pravidla a textu jsou znázorněny v tabulce 2.2. S jednotlivými úseky jsme schopni samostatně pracovat, tedy i změnit slovosled a dále upravit text (například přidat další slova na konkrétní místa ve větě). Pro demonstrování, jakým způsobem průchod

[1] Je nutné dodat, že pravidlo je analyzováno zleva doprava a odřádkování je zde čistě z formátovacích důvodů (lze jej zapsat na jeden řádek).

označení úseku	úsek pravidla	úsek věty
1	`(<do>|<does>)`	do
2	`(<PRP>|<NNS>|<NN>)`	you
3	`(<VB>|<have>|<do>|<am>|<VBP>|<VBZ>)`	love

Tabulka 2.2: Analýza pravidla pro přítomný čas prostý

pravidly funguje, uvažujme podobný text jako v minulém příkladu, ale přidejme k němu další větu. Porovnávaný text bude „`Do you love me? Do you hate me?`“. Po průchodu pravidly může být výstup zobrazen jako strom zobrazený na obrázku 2.2. Při popisu jed-



Obrázek 2.2: Strom po průchodu pravidly pro text „`Do you love me? Do you hate me?`“

notlivých úrovní stromu postupujme odshora dolů. Porovnávaný text je na obrázku označen jako S a tvoří kořen stromu. Text obsahoval dvě věty v přítomném čase prostém, pro které máme definováno pravidlo `simple_present`. Jména odpovídajících pravidel jsou znázorněna v druhé úrovni stromu. Pokud by druhá věta byla v jiném čase, byl by podstrom více vpravo označen pravidlem, které odpovídá času, ve kterém je druhá věta. Na předposlední úrovni jsou tagy, které vznikly označením slov pomocí POS taggeru. Poslední úroveň tvoří slova, která odpovídají chunkům pro každé pravidlo. Slova jsou seřazena stejným způsobem, jakým tomu bylo i v původním textu. Protože se jedná o strom, ve kterém jsou znázorněny chunky, nelze v něm nalézt slovo „`me`“, protože to už je za hranicí vymezenou pravidly. Podobná pravidla lze vytvořit i pro další časy, které jsou popsány v [1]. Nemusí jít jen o časy, ale i specifické typy otázek (například otázky začínající „`where`“).

Pokud budeme větu analyzovat z hlediska větného rozboru a ne jen slovních druhů, obvykle jedna z prvních věcí, kterou budeme chtít určit, je podmět a přísudek a z hlediska chatbota se zdá být zajímavý i předmět. V určité míře se o to snažíme i zde, protože ze samotné definice pravidla pro *simple_present* resp. toho jakým způsobem se v angličtině tvoří otázky (pomocné sloveso následované podmětem a přísudkem) lze z tabulky 2.2 určit, že podmět je část odpovídající úseku 2 a přísudek část s označením 3. Z toho důvodu jsou v pravidle (úsek 2) obsaženy i tagy pro podstatná jména, protože ta mohou být také podmětem. Předmět se v tomto případě ani nesnažíme zjistit.

Kapitola 3

Statistické jazykové modelování pomocí neuronových sítí

Účelem jazykového modelování [6] je vytvoření jazykového modelu (modelu rozložení pravděpodobnosti), který slouží k určení pravděpodobnosti sekvence slov $w_1 \dots w_n$, kdy n je délka sekvence. Tuto pravděpodobnost lze vyjádřit jako:

$$P(w_1 \dots w_i) = P(w_1) \times P(w_2|w1) \times \dots \times P(w_i|w_1 \dots w_{i-1}) \tag{3.1}$$

Vzhledem k tomu, že zjištění rozložení pravděpodobnosti pro sekvenci slov pro větší hodnoty i může být náročné, obvykle se odhad pravděpodobnosti zjednodušuje tak, že pravděpodobnost slova závisí na předchozích n slovech, což je označováno jako n-gramy.

Obecně lze vzorec pro výpočet pravděpodobnosti n-gramu zapsat jako vzorec 3.2, kde je počítán maximální odhad pravděpodobnosti slova A v kontextu H. Označení $C(HA)$ je počet výskytů, kdy se sekvence slov HA vyskytla v trénovacích datech. Kontext nemusí obsahovat žádné slovo, v takovém případě se jedná o unigramy a model nebere v potaz historii. Pokud kontext obsahuje jedno slovo, jde o bigramy, v případě dvou slov o trigramy a pro n slov $(n+1)$gramy. Nevýhodou n-gramů je, že při zvyšování počtu slov, která jsou brána jako kontext, roste počet možností kombinací slov exponenciálně, což vytváří limity při použití n-gramů v oblasti jazykového modelováni. I přesto jsou nejčastěji používaným způsobem pro vytváření jazykových modelů [7].

$$P(A|H) = \frac{C(HA)}{C(H)} \tag{3.2}$$

Problémem jsou také situace, kdy je počítána pravděpodobnost pro sekvenci, která se v trénovacích datech nevyskytla – takovým případům by byla při uvedeném výpočtu přidělena nulová pravděpodobnost, což vede k podhodnocení takových případů. Také existují případy, kterým byla přidělena nadhodnocená pravděpodobnost (obvykle u případů, které se vyskytly v trénovacích datech právě jednou). Oba problémy řeší vyhlazovací techniky, které od nadhodnocených případů odeberou část přidělené pravděpodobnosti a přerozdělí ji mezi případy s nulovou pravděpodobností.

Při porovnání jazykových modelů se používají měřítka, jak dobře dokáže model předpovědět pravděpodobnost sekvence slov. Jedním měřítkem je entropie, což je průměr bitů, které jsou potřeba k reprezentaci sekvence slov, pro kterou se snažíme model ohodnotit. Je dána vztahem 3.3.

$$\frac{1}{K} \sum_{i=1}^{K} \log_2 P(w_i|w_1 \dots w_{i-1}) \tag{3.3}$$

Dalším měřítkem, které vychází z entropie, je perplexity. Je to exponenciála negativních hodnot získaných entropií a je dána vztahem 3.4.

$$\sqrt[K]{\prod_{i=1}^{K} \frac{1}{P(w_i|w_1 \dots w_{i-1})}} = 2^{-\frac{1}{K}\sum_{i=1}^{K} \log_2 P(w_i|w_1 \dots w_{i-1})} \tag{3.4}$$

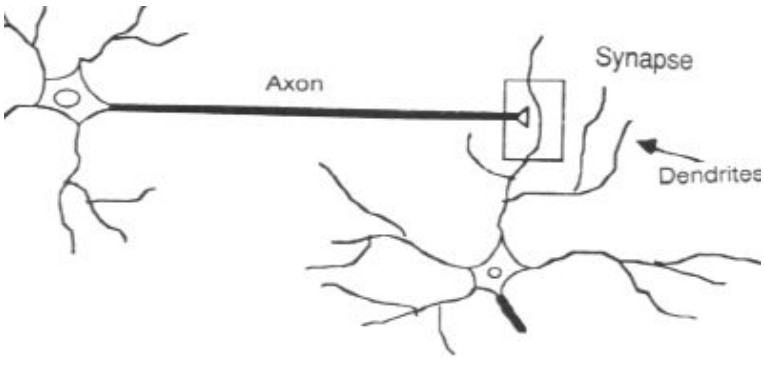
Rozdíl obou měřítek je v tom, že podle [7] se absolutní hodnoty perplexity lépe pamatují (např. 100-200) oproti entropii, kde počty potřebných bitů dosahují hodnoty např. 6.64-7.64. Čím nižší hodnota perplexity, tím více je model bližší reálnému modelu.

3.1 Neuronové sítě

Aby bylo možné popsat neuronové sítě z hlediska jazykového modelování, je nutné první říci, co to vlastně neuronové sítě [3] jsou. Z hlediska následujícího popisu (pokud není řečeno jinak) je neuronová síť i neuron brán z hlediska využití v informatice, jedná se tedy o umělou neuronovou síť tvořenou umělými neurony. Jako první bude popsán neuron z hlediska biologie (ze kterého vychází modely použité v informatice), poté základní model z hlediska informatiky a jako poslední MCP model neuronu. Poté následuje popis rozdílů mezi feedforward a feedback neuronovou sítí. Jako poslední je nastíněno využití neuronové sítě při vytváření jazykového modelu.

3.1.1 Neuron

Biologický neuron, tedy neuron z hlediska lidského mozku komunikuje s ostatními neurony pomocí dendritů. Obrázek 3.1 popisuje komunikaci neuronu s okolím. Axon je spojení mezi tělem neuronu a strukturami, na které chce neuron působit a rozděluje se na tisíce malých větviček, na jejichž koncích jsou synapse. V případě, že je neuron aktivován (počet posilovacích impulsů z okolních neuronů přesáhne aktivační mez), je generován impulz a je vyslán přes axon k synapsi, která je spojena s dalšími neurony. Synapse podle úrovně signálu z axonu posílí nebo zeslabí aktivitu v okolních propojených neuronech. Učení probíhá změnou efektivity výměny signálu pomocí synapsí, takže se mění vliv neuronu na své okolí. Neurony popisované dále jsou brány z hlediska informatiky a prvním druhem je základní mo-



Obrázek 3.1: Biologický neuron

del neuronu. Neurony používané v informatice mají vzor v biologických neuronech a jejich struktura je znázorněna na obrázku 3.2. Každý neuron má libovolný počet vstupů $x_1 \dots x_n$,

ale pouze jeden výstup. V trénovací fázi je neuron učen, pro jaké kombinace vstupů (natrénovaný vzor) má nebo nemá aktivovat svůj výstup. Obecně tedy mohou nastat dva stavy, výstup neuronu je buď aktivován nebo není aktivován. V testovací fázi pokud je na vstupu neuronu detekován naučený vzor, je známo, zda se má výstup neuronu aktivovat nebo ne. Přepínání fází je znázorněno vstupem *fáze*. Pokud vstupní vzor neodpovídá naučenému vzoru, je výstup aktivován na základě generalizovaných aktivačních pravidel.

Aktivační pravidla jsou jasně definovaná pro vzory, které se vyskytly v trénovací fázi. Pravdivostní tabulka 3.1 znázorňuje, jak mohou aktivační pravidla vypadat pro neuron se třemi vstupy (x_1, x_2, x_3). Naučené vzory jsou takové, které mají výstup 1 (pro aktivaci výstupu) nebo 0 (neaktivaci výstupu). Pro ostatní možné kombinace vstupů nejsou žádné vzory definovány a v tabulce jsou označeny hodnotami 0/1, proto je potřeba určit, jak se při těchto nedefinovaných kombinacích má neuron chovat.

Toho lze dosáhnout porovnáním nedefinovaného vzoru se vzory známými. Pro konkrétní jeden nedefinovaný vzor jsme tedy schopni určit, kterému ze známých vzorů se podobá nejvíce. Může ale nastat i situace, kdy se nejméně podobá více známým vzorům a nejsme tedy schopni hodnotu výstupu určit. V tabulce 3.2 lze vidět, že po generalizaci aktivačních pravidel došlo k specifikování výstupu na hodnotu 0 pro vzor 010. Při porovnání s ostatními už známými vzory došlo k největší shodě se vzorem 000 (od sebe se liší jen v jednom vstupu), zatímco od ostatních se liší ve více vstupech, například od vzoru 111 ve dvou vstupech. Stejně tak pro vzor 110 jsme určili výstup jako 1 na základě podobnosti se vzorem 111, kdy se liší v jednom vstupu. U vzoru 011 jsme ale nemohli určit hodnotu výstupu proto, že se nejméně liší od (více) vzorů, které mají rozdílné výstupy. McCulloch-Pitts model neuronu

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1
výstup	0	0	0/1	0/1	0/1	1	0/1	1

Tabulka 3.1: Aktivace neuronu před generalizací

x_1	0	0	**0**	0	1	1	**1**	1
x_2	0	0	**1**	1	0	0	**1**	1
x_3	0	1	**0**	1	0	1	**0**	1
výstup	0	0	**0**	0/1	0/1	1	**1**	1

Tabulka 3.2: Aktivace neuronu po generalizaci

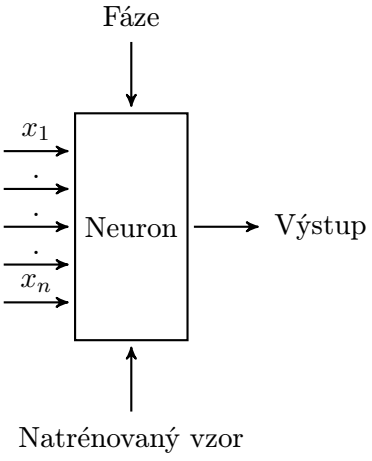
(MCP) vychází z jednoduchého modelu neuronu, ale navíc přidává každému vstupu neuronu váhu, což je znázorněno na obrázku 3.3, kde váhy jsou označeny jako $w_1 \dots w_n$. Funkce aktivace neuronu je závislá od kombinace vstupů a je dána vztahem 3.5

$$f(x) = \begin{cases} 1 & \sum_{i=1}^{N} x_i w_i > T \\ 0 & \sum_{i=1}^{N} x_i w_i < T. \end{cases} \tag{3.5}$$

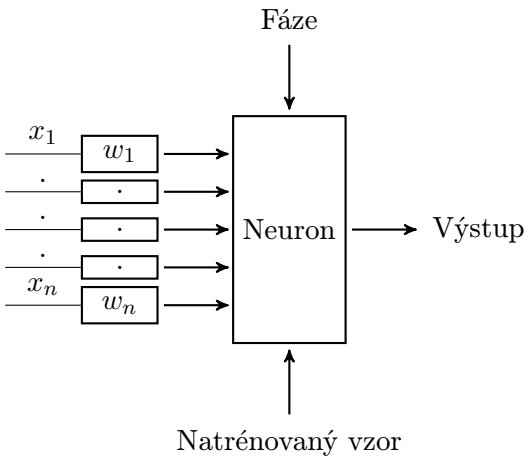
T je prahová hodnota, kdy má být aktivován výstup a N označuje počet vstupů neuronu. Aktivaci výstupu reprezentuje hodnota 1, opak hodnota 0. Váhu lze využít pro adaptaci neuronu.

Popis neuronů v předcházející části bral jako jejich výstup hodnoty 0 a 1. Obecně ale výstupem můžou být jakékoliv hodnoty. Výstupy neuronů, které jsou použity při reprezentaci jazykového modelu (tedy rozdělení pravděpodobnosti) jsou dány aktivační funkcí ze vztahu 3.6, kde g je softmax funkce viz 3.1.3.

$$f(x) = g(\sum_{i=1}^{N} x_i w_i) \tag{3.6}$$



Obrázek 3.2: Jednoduchý neuron

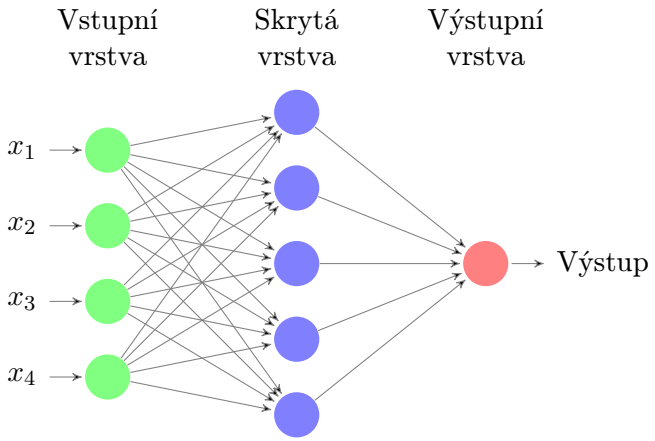


Obrázek 3.3: MCP model neuronu

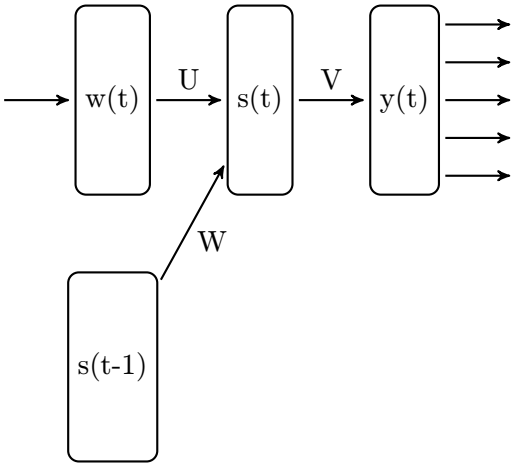
3.1.2 Druhy neuronových sítí

Nejvíce používaným typem neuronových sítí jsou feedforward neuronové sítě [7], které obsahují 3 vrstvy a ty se nazývají vstupní, skrytá a výstupní vrstva. Každá vrstva se skládá z jednoho nebo více uzlů. Pro popis dále uvažujme, že síť už je v natrénovaném stavu. Obrázek 3.4 ukazuje propojení vrstev. Pro vstupní vrstvu platí, že uzly v této vrstvě přenášejí hodnotu na jejich vstupu (všechny uzly dohromady tvoří vstup celé sítě) na vstup skryté vrstvy, modifikovanou vahami mezi vstupní a skrytou vrstvou. Ve skryté vrstvě jsou neurony, které na základě vzoru na jejich vstupech aktivují nebo neaktivují svůj výstup. Výstup skryté vrstvy je přenášen do výstupní vrstvy, opět modifikovaný vahami. Ve výstupní vrstvě probíhá v závislosti na velikosti všech vrstev zkombinování výstupů skryté vrstvy a výstupem této vrstvy je i výstup sítě. Dalším typem neuronových sítí jsou feedback (rekurentní) neuronové sítě [7], vycházejí z feedforward neuronových sítí. Feedforward neuronová síť ale neumožňuje možnost zpětné vazby. Uvažujme situaci, kdy chceme na vstup sítě přivést nějakou kombinaci a poté zjistit výsledek, pak přivést jinou kombinaci a opět zjistit výsledek. Při použití feedforward sítě získáme dva výsledky, které spolu nesouvisí. Jsou ale případy (třeba tato práce), kdy je potřeba, aby bylo možné vycházet z předchozích výstupů, resp. z předchozích stavů skryté vrstvy a vytvářet další výstupy v závislosti na nich. Obrázek 3.5 ukazuje, že aktuální stav skryté vrstvy $s(t)$ lze ovlivnit tak, že na vstup $w(t)$ reaguje nejen modifikací vstupu pomocí vah vstupu U, ale navíc jej kombinuje i s předcházejícími hodnotami skryté vrstvy $s(t-1)$, modifikované vahami W.

[1] Převzato z http://www.texample.net/tikz/examples/neural-network viz [7]



Obrázek 3.4: Feedforward neuronová síť[1]



Obrázek 3.5: Feedback neuronová síť

3.1.3 Jazykový model pomocí neuronové sítě

Pro práci s rekurentními (feedback) neuronovými sítěmi existuje několik nástrojů, pro využití z hlediska jazykových modelů existuje `RNNLM Toolkit` (Recurrent Neural Network Language Modeling Toolkit). `RNNLM Toolkit` je nástroj, který umožňuje trénovat i dále používat neuronové sítě. Z hlediska využití v chatbotovi je podstatná jeho funkce trénování neuronové sítě (vytvoření jazykového modelu), inicializace skryté vrstvy na požadovaný kontext a hlavně funkce pro generování sekvence slov jako výstupu.

Pro popis, jak je jazykový model reprezentován pomocí feedback neuronové sítě [7] v `RNNLM Toolkitu`, je opět použit obrázek 3.5, kde jsou jednotlivé vrstvy brány jako vektory a váhy jako váhové matice. Vstupní vrstva se skládá z vektoru $w(t)$, který představuje aktuální slovo w_t kódované jako 1 z N, kde N je velikost slovníku. Označení $s(t-1)$ je vektor hodnot skryté vrstvy z předchozího kroku. Po natrénování modelu představuje výstupní vrstva $y(t)$ rozdělení pravděpodobnosti pro další slovo w_{t+1} na základě aktuálního slova a předcházejícího stavu skryté vrstvy $s(t-1)$ a je dána vztahem 3.7. Funkce softmax označená jako g ve výstupní vrstvě slouží k ujištění, že výstupy tvoří validní rozdělení pravděpodobnosti, tedy všechny jsou kladné hodnoty a jejich součet je 1. V je váhová

matice mezi skrytou a výstupní vrstvou, $s(t)$ je aktuální stav skryté vrstvy.

$$y(t) = g(Vs(t)) \tag{3.7}$$

Trénování neuronové sítě (tedy vytvoření jazykového modelu) probíhá v několika epochách dokud se síť trénováním stále dostatečně zlepšuje, což je ověřováno na validačních datech. Může probíhat tak, že na počátku tréninku je nastavena úroveň učení, a dokud dochází k výraznému snižování entropie, je použita stejná hodnota úrovně učení. Pokud nedojde k dostatečnému snížení entropie, je hodnota úrovně učení snížena o polovinu.

Prvním způsobem, jakým lze neuronové sítě trénovat, je algoritmus 2, který byl převzat z [7], a popisuje jednu trénovací etapu pomocí Backpropagation pro feedback neuronovou síť. Obvykle se tento algoritmus využívá spíše pro feedforward neuronové sítě (s vynecháním kroku 4). Váhové matice jsou na začátku nastaveny na malé náhodné hodnoty. Použitá označení v algoritmu vychází z obrázku 3.5. Výpočet gradientu chyby $e(t)$ ve výstupní vrstvě je dán jako 3.8.

$$e_0(t) = d(t) - y(t) \tag{3.8}$$

Algoritmus 2 Backpropagation pro jednu trénovací etapu u feedback neuronové sítě

1. Nastav counter $t = 0$ a inicializuj stav neuronů ve skryté vrstvě $s(t)$ na 1
2. Inkrementuj hodnotu counteru o 1
3. Dej na vstup vstupní sítě $w(t)$ aktuální slovo w_t
4. Zkopíruj stav skryté vrstvy $s(t-1)$ do vstupní vrstvy
5. Proveď forward pass pro zjištění $s(t)$ a $y(t)$
6. Spočítej gradient chyby $e(t)$ ve výstupní vrstvě
7. Přenes chybu přes neuronovou síť a změň odpovídající váhy
8. Pokud nebyla zpracována všechna trénovací data, jdi na krok 2

Backpropagation je možné použít i pro trénování feedback neuronových sítí, tím se ovšem vytrácí jejich potenciál. Proto je pro trénování feedback neuronových sítí vhodnější využít variantu backpropagation, která se nazývá backpropagation throught time a umožňuje plně využít jejich potenciál. Feedback neuronová síť s jednou skrytou vrstvou z hlediska N časových kroků může být brána jako feedforward neuronová síť s N skrytými vrstvami. Rozdíl oproti backpropagation je tedy v tom, že při backpropagation throught time je brán v potaz více než jeden časový krok.

Kapitola 4

Využití neuronové sítě v chatbotovi

Uvažujme motivační ukázku 4.1, která se snaží demonstrovat účel využití i jiných postupů, než jen generování výstupu neuronové sítě na základě jejího vstupu. Tato ukázka vychází z použití jen neuronové sítě. Pokud dáme na vstup feedback neuronové sítě otázku (Q), tedy text „`I can.`“, může být výstupem (odpovědí) neuronové sítě text „`Really sit back.`“ označený jako A. Neuronová síť použila text otázky, aby inicializovala stav skryté vrstvy na požadovaný kontext (postupně po slovech). Výstup je generován také po slovech, kdy neuronová síť vychází z předchozích stavů skryté vrstvy. Pokud je zadáno, aby generovala N slov, bude po každém dalším vygenerovaném slovu brát v potaz jak původní kontext (Q), na který byla inicializována, tak předcházející vygenerovaná slova. Vztah Q a A je tedy takový, že text A doplňuje text Q. Pro lepší názornost spojme Q a A do jedné věty: „`I can really sit back.`“. Z hlediska chatbota to ovšem nestačí – chatbot by měl na otázku Q odpovídat, ne ji jen doplňovat[1].

Q: I can.
A: Really sit back. (4.1)

Ještě předtím, než vůbec začne probíhat komunikace mezi chatbotem a uživatelem, je potřeba získat jazykový model, tedy natrénovat neuronovou síť. Sběr trénovacích dat a popis vytvořených modelů je popsán v části 4.1. Zbytek kapitoly popisuje zkombinování technik z kapitol 2.1 a 3.1, tedy zkombinování technik existujících chatbotů se zpracováním přirozeného jazyka a jazykovým modelem pomocí neuronové sítě. Pro připomenutí jsou vybrané techniky chatbotů shrnuty v následujícím výčtu:

- hledání klíčových slov a transformace některých slov (transformace „`I`“ → „you“) – `ELIZA`
- použití Markovova modelu – `MegaHAL`
- odpovědi jako reakce na vzor na vstupu, uložené ve specifickém formátu – `A.L.I.C.E`
- označení odpovědí pomocí klíčových slov – `Vrajitoru`

[1] Tento konkrétní příklad používá jazykový model založený na datech z Twitteru, při použití jiného modelu by odpověď mohla vypadat úplně jinak a už by se nemuselo jednat o doplnění v tom smyslu, jak je zmíněno zde.

Přístup, který využívá `ELIZA` – tedy předem definované odpovědi s transformací některých slov, popírá účel využití neuronových sítí (resp. opravdu dynamické vytváření odpovědí). Nelze ho proto použít tak, jak je, ale při zobecnění tohoto přístupu už použít lze. Vzory, které jsou v textu vyhledávány nemusí být pevně daná slova, protože slova lze klasifikovat z hlediska slovních druhů i z hlediska větných členů. Jak taková klasifikace probíhá je popsáno v kapitole 2.2. Stejně tak nemusí být předem pevně definována odpověď, ale je možné definovat jen její tvar (vzor výstupu).

Markovovy modely, které využívá `MegaHAL`, jsou bližší neuronovým sítím, protože mají v tomto případě téměř totožný účel. Proto i při použití neuronových sítí místo Markovových modelů je vhodné hledat klíčová slova a ta použít jako základ pro generování odpovědi chatbota.

4.1 Použitý model a trénovací data

Pro trénovaní jazykového modelu byla použita trénovací data z Twitteru, získaná z tzv. public streamu. Twitter public stream nabízí vzorek dat (1%), která v danou chvíli uživatelé dávají na Twitter. Funguje v reálném čase, tzn. pokud se na stream napojí více lidí zároveň, budou dostávat stejná data. Data jsou náhodně vybrané tweety, které se ale dají filtrovat a pro sběr dat v této práci bylo použito filtrování na tweety, které jsou napsány anglicky. Takovéto filtrování však Twitter nedělá zcela spolehlivě, proto se v datech vyskytují i jiné jazyky než angličtina.

Před samotným trénováním neuronové sítě bylo potřeba trénovací data ještě upravit (normalizovat). Konkrétní implementace neuronové sítě, která je v této práci využita rozlišuje velká a malá písmena. To znamená, že bez úprav by byl model méně přesný, protože pro každé slovo by existovaly i jeho varianty s malými a velkými písmeny, navíc z hlediska velikosti slovníku by počet slov narostl o to víc. Proto všechna trénovací data byla převedena na malá písmena. V datech se dále mohou vyskytovat jména uživatelů a pro Twitter je specifické, že při odpověďi na jiný tweet je před přezdívkou uveden znak „`@`“. Hashtagy jsou také detekovatelné a to pomocí „`#`“ před tagem, podobně jako webové odkazy. Přezdívky, hashtagy i odkazy byly z textu odstraněny a nahrazeny tagem „`<UNK>`“. Bylo možné je v trénovacích datech i nechat, ale vzhledem k tomu, že webové odkazy jsou v podstatě neověřitelné co se týče obsahu, nakonec se zdálo být jistější je odstranit. Hashtagy se pro interakci dvou uživatelů také nehodí, resp. tento trend moc rozšířený mimo Twitter, kde má být chatbot také použit, není. Přezdívky, resp. jména uživatelů byla odstraněna spíše z etických důvodů. V datech zůstaly nespisovné výrazy, překlepy, sprostá slova – ta je možné odfiltrovat i později (po vytvoření odpovědi) a také z důvodů autentičnosti.

Během vývoje bylo použito více modelů a i pro jejich trénování byla použita jiná velikost trénovacích dat a parametry. První pokusy byly modely založené na pár tisících slov, kdy trénovací data nebyla nijak upravována a výsledky byly v podstatě nepoužitelné. Vznikly 3 větší modely – první z nich byl trénován na zhruba 110 miliónech slov, druhý na necelých 32 miliónech slov a třetí na zhruba 80 miliónech slov.

Co se týče velikosti slovníku, pro urychlení trénování bylo vhodné slovník omezit a jako velikost slovníku byla zvolena hodnota 100000 slov. Oproti původním zhruba 2.5 miliónům slov ve slovníku pro trénovací data o velikosti 110 miliónů slov se to může zdát jako značná redukce, ovšem většina z těchto slov měla zanedbatelný počet výskytů, reálně by použití ještě menšího slovníku mělo minimální dopad na kvalitu výsledného modelu.

`RNNLM Toolkit` využívá pro urychlení práce s modelem možnost zařazovat slova do tříd. Počet těchto tříd byl u všech modelů zvolen na hodnotu 300. První dva modely ob-

sahovaly velikost skryté vrstvy 300, poslední z modelů 400. Testování probíhalo primárně na prvním modelu, ovšem z hlediska porovnání perplexity vychází poslední model nejlépe, protože u něj byly provedeny úpravy navíc oproti původním dvěma modelům. Tyto úpravy obsahovaly kvalitnější úroveň normalizace trénovacích dat (oddělování interpunkčních znamének mezerami, rozlišování mezi nahrazovanými slovy, oproti „`<UNK>`“ přibyly tagy jako „`<webadress>`“ pro nahrazený odkaz apod.). Jak už bylo řečeno, na tomto modelu nebyly prováděny žádné testy a rozšířené tagy nahrazených slov program dále nijak nevyužívá, proto při spuštění chatbota s tímto modelem generuje odpovědi obsahující tyto tagy v neupravené formě. Vytvoření posledního modelu bylo spíše pro porovnání s těmi ostatními. Zvětšení velikosti skyté vrstvy sice přináší kvalitnější reprezentaci jazykového modelu a zaznamenání kontextu, ale na druhou stranu jeho načítání trvá déle s tím souvisí problém popsaný v kapitole 6.

4.2 Fáze vytváření odpovědi

Jako hlavní prostředí pro komunikaci s botem byl zvolen facebook. Komunikace probíhá prostřednictvím zpráv, kdy komunikaci vždy zahajuje uživatel a bot reaguje, nejedná se tedy o žádnou formu spamu. Kromě komunikace na facebooku bylo vytvořeno i webové rozhraní, aby nebyla možnost využít bota upírána i uživatelům, kteří facebook nevyužívají.

Zprávy mohou být ve formě otázek, tvrzení nebo větných struktur specifických pro internetovou komunikaci (smajlíci, samolepky na facebooku a jiné). Základní idea je taková, že zprávu prvně analyzujeme pomocí metod popsaných v kapitole 2.2, k původní zprávě připojíme výsledek takové analýzy (první část odpovědi) a slova dáme postupně na vstup neuronové sítě. Tím je skrytá vrstva inicializována na požadovaný kontext a je připravena pro generování druhé části odpovědi. Obě části jsou poté spojeny a odeslány jako odpověď.

Celý proces komunikace se tedy dá rozdělit na několik fází:

- Příjem zprávy od uživatele (otázky)
- Úprava zprávy od uživatele
- Zaslání upravené zprávy na vstup neuronové sítě a vygenerování jejího výstupu
- Úprava výstupu neuronové sítě a vytvoření odpovědi
- Odeslání odpovědi

Příjem a odeslání zpráv pro konkrétní rozhraní, stejně jako podrobnější komunikace mezi jednotlivými částmi programu je popsána v kapitole 5. Úprava zprávy od uživatele probíhá v několika krocích, které popisuje algoritmus 3. Každý z kroků algoritmu je rozepsán v dalších částech této kapitoly. Je nutné prvně vysvětlit, jak jsou jednotlivé kroky algoritmu prováděny a poté je v sekci 4.7 rozebrán konkrétní příklad z hlediska tohoto algoritmu krok za krokem. Prvním z kroků je ověřování, zda se uživatel neopakuje, tedy zda během posledních několika otázek nepoložil tu samou vícekrát. Kvůli tomu je vždy posledních pár otázek ukládáno do cache, aby na to mohl chatbot adekvátně reagovat. Dalším krokem je hledání klíčových slov, která slouží jako způsob vytváření odpovědi. Reakce na vlastní jméno se pokouší přidat i jiné prvky než snahu o analýzu a hlavním krokem je analýza struktury zprávy. Po průchodu těmito fázemi vznikne první část odpovědi, která je dána na vstup neuronové sítě, a ta doplní zbytek odpovědi. Další částí je úprava takové odpovědi. Záleží na použitém modelu, ale obecně lze říci, že vždy jsou vyžadovány i dodatečné úpravy (odstranění mezer navíc, detekce nevhodně vložených interpunkčních znamének a další).

Algoritmus 3 Fáze vytváření odpovědi

1. Přijmi zprávu
2. Zkontroluj, zda se nejedná o zprávu, která je v cache a poté vlož zprávu do cache
3. Reaguj na vlastní jméno
4. Označ zprávu pomocí POS tagů
5. Vyhledej klíčová slova
6. Analyzuj zprávu z hlediska POS a chunkingu
7. Odhadni délku očekávané odpovědi
8. Podle očekávané délky odpovědi vytvoř počátky vět
9. Spoj původní zprávu a počátek vytvořené věty a pošli ji na vstup neuronové sítě
10. Spoj počátek věty a výstup neuronové sítě, uprav formátování a připoj k odpovědi
11. Skoč na bod 9 a pokračuj dokud neprojdeš počátky všechny vět
12. Odešli odpověď

4.3 Cache

Při přijmutí více stejných zpráv za sebou nebo v určitém intervalu, je nežádoucí, aby přicházely podobně strukturované odpovědi. Proto existuje cache, do které je ukládán daný počet posledních zpráv. Pokud je položena otázka, která už byla jednou nebo vícekrát položena, k odpovědi bude přidána navíc náhodně vybraná fráze, například „`say something different...`“. Nabízí se možnost frázi nevybírat náhodně, ale tak, aby se nemohla opakovat, ale když si uvědomíme, že otázky se vlastně také opakují, občasné zopakování stejné fráze, naznačující, že se uživatel opakuje, značí jistou míru ironie, a proto se zdál zvolený způsob řešení jako dostačující.

4.4 Reakce na pozdrav a vlastní jméno

Pozdravem jsou myšleny obvyklé vítací výrazy, kterými jsou například „`hi`“ nebo „`hello`“, případně výrazy, které slouží k rozloučení. Pokud se v textu takové výrazy vyskytují, je na počátek odpovědi přidán náhodně vybraný pozdrav z množiny těchto výrazů.

Za vlastní jméno je považováno „`Anna`“, „`Anna Bell`“ nebo „`Annabell`“. Pokud se v otázce vyskytuje vlastní jméno, chatbot kromě způsobu, jakým by obvykle reagoval k odpovědi přidá navíc řádek z písně „`Do you wanna build a snowman?`“ z filmu Frozen[2].

[2]Text písně je dostupný na http://www.metrolyrics.com/do-you-wanna-build-a-snowman-lyrics-kristen-bell.html

4.5 Hledání klíčových slov

Za klíčová slova jsou považována podstatná jména a slovesa, která se vyskytují v otázce. Podstatná jména jsou ukládána do vlastního seznamu samostatně, stejně tak slovesa. Každé takto detekované klíčové slovo má nastavenou prioritu. Po položení otázky uživatel očekává odpověď, která bude souviset s otázkou. Pomocí takovéto detekce klíčových slov se dá říci, že jsme pravděpodobně našli podmět a předmět věty (ale i další slova, která s odpovědí ani nemusí souviset).

Otázka ale nemusí obsahovat žádná podstatná jména ani slovesa, resp. POS tagger je může označit špatně. Proto nejsou vyhledaná klíčová slova udržována jen pro poslední zprávu, ale i pro definovaný počet předchozích zpráv. Pro rozpoznání, jak moc se uložené klíčové slovo vztahuje k otázce, mají všechna klíčová slova definováno stáří. Po přijmutí další zprávy (pro konkrétního uživatele) všechna klíčová slova zestárnou (zvýší hodnotu atributu stáří o 1). Po překročení limitu stáří je takové slovo z paměti odstraněno. V případě, že je rozhodnuto, že má být nějaké klíčové slovo součástí odpovědi, je jeho stáří zvýšeno o hodnotu 2 (hlavně z toho důvodu, aby každá odpověď neobsahovala stále dokola ta samá klíčová slova).

Při vytváření odpovědi související s klíčovými slovy existuje několik možností, jakým způsobem bude tvořena a těmi jsou (v závorkách jsou uvedené pravděpodobnosti, jaké byly pro každou možnost zvoleny[3]):

- výběr nejmladšího podstatného jména (25%)
- výběr nejmladšího slovesa v kombinaci s dotazovací frází (35%)
- výběr náhodného podstatného jména (10%)
- výběr náhodného slovesa (5%)
- výběr náhodné fráze (bez použití klíčových slov) (25%)

Před každým výběrem je vygenerována pseudonáhodná hodnota, podle které je určeno, která možnost má být vybrána. Ovšem v případě, že klíčová slova (podstatná jména i slovesa) nejsou k dispozici, je výstupem této části prázdný řetězec, nebo náhodná fráze. Původně se zdálo vhodné definovat vyšší počet frází, které obsahovaly více slov, ovšem i během kratšího rozhovoru se začaly odpovědi až moc podobat kvůli těmto frázím a to i při detekci opakování. Zajímavým kompromisem se nakonec ukázalo být použití zájmen „`I`“ a „`we`“. Z hlediska šablonovitosti to není problém, protože zájmena se vyskytují ve větách velice často, a to hlavně při způsobu komunikace v první osobě, tam je (i když nepřímo) zájmeno „`I`“ přítomno vždy. Použití dalších zájmen opět přináší problém šablonovitosti a v tomto případě (vzhledem k jazykovému modelu) se jevilo lepší je nepoužít.

Výběr nejmladšího podstatného jména funguje tak, že pokud jsou jako klíčová slova k dispozici nějaká podstatná jména, je v seznamu klíčových slov nalezeno takové, které má nejmenší hodnotu atributu stáří. Toto slovo je pak připojeno na konec původní otázky od uživatele, která je poslána na vstup neuronové sítě.

Výběr nejmladšího slovesa funguje podobně jako výběr nejmladšího podstatného jména, ale navíc je opět pomocí (další) pseudonáhodné hodnoty rozhodnuto, jaká fráze v kombinaci s vybraným slovesem bude použita. Jde o to, že podstatné jméno obvykle může být

[3]Hodnoty byly zvoleny základě experimentů, ovšem nebyla jim věnována důkladnější analýza. Výběr nejmladších sloves a podstatných jmen má nastaven vyšší výskyt hlavně proto, že při vedení souvislého rozhovoru je vyšší šance, že se bude rozhovor týkat jednoho tématu více než jednu zprávu.

na začátku věty, ale u sloves to není tak obvyklé. Hranice mezi slovesem a podstatným jménem určuje POS tagger, a proto mohou nastat případy, kdy je sloveso značeno jako podstatné jméno a opačně (případ slov zakončených na „`-ing`“). Pro ostatní tvary sloves je ale vhodnější je zařadit buď do otázky, nebo za podstatné jméno. Do otázky lze zařadit sloveso tak, že je použito pomocné sloveso následované zájmenem „`you`“ a za ním vybrané sloveso. Pomocné sloveso je vybíráno z možností *do, have, can* nebo *would*, kdy všechny možnosti mají nastavenou rovnocennou pravděpodobnost. Zde by bylo vhodné v budoucnu použít více takových šablon a optimálně i zkombinovat popisované přístupy.

4.6 Analýza struktury otázky

Jak už bylo popsáno v kapitole 2.2, k analýze struktury věty slouží POS tagging a chunking. Prvně je otázka (tedy i více vět) rozdělena na slova (tokeny), ty jsou pomocí POS taggingu označeny. Označená slova jsou poté dány na vstup chunkeru, který otázku rozdělí na chunky, což jsou v našem případě části otázky (věty), které odpovídají zadaným pravidlům pro chunkování. Je bráno v potaz i pořadí, v jakém jsou pravidla definována – pravidla definovaná jako první mají vyšší prioritu, než ta pod nimi. Toto v případě různých časů není podstatné, ale pro případy, kdy jsou pravidla definovaná jinak než gramatické časy, už to roli hraje. Například použití „`going to`“ jako budoucí formy, protože definice pravidla pro přítomný průběhový čas lze zapsat i tak, že se bude s budoucí formou překrývat. Pokud část otázky odpovídá některému z definovaných pravidel, je v této konkrétní části přehozen slovosled.

V tento moment záleží na tom, jakým způsobem byla pravidla definována. Příklad definice pravidla z 2.2.2 demonstroval, jak by vypadalo pravidlo pro přítomný čas prostý. Taková definice ovšem pokrývá jen situace, kdy je věta ve tvaru přesně odpovídajícímu pravidlu. Ovšem častěji se můžeme setkat s rozvinutějšími větami. Například rozdíl mezi „`Do you love me?`“ a „`Do you really love me?`“ je jen jedno slovo („`really`“), ale tak jak je definováno pravidlo z uvedeného příkladu by druhá věta pravidlem neprošla.

To lze vyřešit přidáním pravidla, které se snaží pokrýt i možnost, že se mezi původním vzorem z [1] vyskytují jiná slova. Pořadí pravidel hraje roli, proto je při použití takovýchto pravidel vhodné je oproti těm jednoduchým dát až nejvíce vespod. Tak jsou prvně zachycena jednoduchá pravidla a teprve v případě, že žádné neodpovídá, jsou zkoušena ta složitější. Co se týče výsledků, zde se opět může projevit překrývaní pravidel, ale větší problém je v tom, že na místě `<.*>*` může být cokoliv a v další analýze to potom dělá problémy (je potřeba počítat s tím, že POS tagging ne vždy správně označí všechny tokeny a čím obecněji je pravidlo zadáno, tím větší problémy to způsobuje). Proto jsou takováto pravidla použita jen pro některé časy a spíše z důvodu demonstrace takového přístupu. Více o tomto problému v kapitole 6.2. Dalším druhem pravidel, která lze použít, jsou pravidla, která pracují opačným způsobem. Tedy nesnaží se transformovat větu ve formě otázky na odpověď ve tvaru tvrzení, ale opačně. Takové pravidlo pro přítomný čas prostý by vypadalo jako:

```
rev_simple_present: (<PRP>|<NNS>|<NN>)
                    (<VB>|<VBP>|<have>|<do>|<am>)
```

Můžeme si všimnout, že pravidlo se snaží zachytit věty začínající podmětem, následované přísudkem. Příklad 4.2 popisuje, jak by mohla vypadat odpověď při tomto způsobu definování pravidla. Na rozdíl od předchozích pravidel je zde navíc jako první slovo odpovědi slovo „`why`“. Zde si čtenář může všimnout podobnosti s chatbotem `ELIZA` a v obecné formě

i u ostatních typů pravidel. Kromě „`why`“ je možné přidat i jiná slova, ale to je alternativa do budoucna (opět více v kapitole 6.2). Jak už bylo řečeno, neuronová síť určitým způsobem doplňuje otázku a v kombinaci s tím, že trénovací data byla tvořena převážně tvrzeními, dokáže neuronová síť sama o sobě generovat relevantnější odpovědi na tvrzení, než na zprávy ve formě otázek.

Q: I love you.
A: Why do you love? (4.2)

V případě základních pravidel je přehození slovosledu a transformace zájmen jednoduchá. Z každého pravidla víme, na které pozici má být každý větný člen. Pokud předpokládáme, že na určité pozici je předmět otázky, který může být tvořen zájmenem nebo podstatným jménem, v kostře odpovědi z něj bude transformovaný podmět. V případě podstatných jmen není transformace nutná, ale v případě zájmen ano, resp. v případě první a druhé osoby jednotného čísla („`I` → „you““). Některé časy obsahují pomocné sloveso „`have`“ apod., pro ně je nutné kontrolovat, který tvar se má vložit (např. „`has`“). Kromě toho je v této fázi rozhodováno, zda je k odpovědi připojeno navíc i „`Yes`“ nebo „`No`“ s negovaným slovesem. Určování zda tato slova přidat nebo ne probíhá tak, že jsou v textu vyhledána slova, která se používají v dotazovacích otázkách na něco konkrétního, tedy slova jako „`where`“ nebo „`why`“. Jinou možností by bylo opět definovat pravidla obsahující tagy pro taková slova přímo v pravidlech. Pokud text taková slova neobsahuje, pak je na základě pseudonáhodně vygenerované hodnoty rozhodnuto, která z následujících možností bude provedena - buď není přidáno nic, nebo je přidáno jen „`yes`“, případně jen znegované sloveso a poslední možností je přidání „`no`“ a znegování slovesa. Každá z těchto možností má přidělenou stejnou pravděpodobnost (25%).

Pro pravidla obsahující část `<.*>*` je vytvoření odpovědi složitější. Principiálně je způsob stejný jako u jednoduchých pravidel, ale prakticky se jednotlivé větné členy hůře nacházejí – v podstatě je chunk procházen, dokud není nalezen odpovídající větný člen, v pořadí v jakém chceme, aby tvořily odpověď a poté je odpověď tvořena podobným způsobem, jako u jednoduchých pravidel.

4.7 Tvorba odpovědi

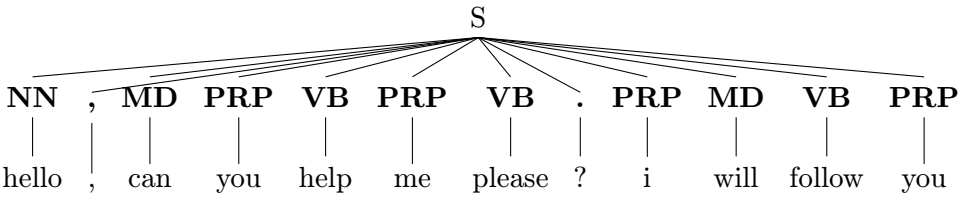
Pokud budeme opět vycházet z algoritmu 3, který popisuje jak jdou za sebou jednotlivé fáze, zaměřme se na vytváření odpovědi z hlediska těchto fází a pro popis tvorby odpovědi krok za krokem použijme následující příklad:

Q: hello, can you help me please? i will follow you anywhere, i promise
A: welcome, no, i can not help that now some of can you help others. thank you. (4.3)

Po přijmutí zprávy, ještě než proběhne jakákoliv z fází analýzy, je odpověď tvořena prázdným řetězcem. Každá z fází k odpovědi přidá něco navíc, počínaje reakcí na pozdrav a vlastní jméno a úpravou formátování konče.

Chatbot přijme zprávu (krok 1). Vzhledem k tomu, že se jedná o konverzaci zatím obsahující jedinou zprávu, je cache prázdná. Po kontrole, zda se nejedná o opakující se zprávu je tedy odpověď stále prázdný řetězec a do cache je přidána aktuální zpráva (krok 2). Zpráva také neobsahuje vlastní jméno chatbota, proto k odpovědi není nic připojeno. Zpráva ovšem obsahuje pozdrav „`hello`“, proto je k odpovědi přidána reakce na pozdrav, v tomto případě „`welcome`“ (krok 3).

Jako další přichází na řadu označení pomocí POS tagů (bod 4), předcházející kroky pracovaly s ještě neoznačeným textem, protože reagovaly na konkrétní fráze. Jedná se o první zprávu, proto klíčová slova, se kterými budeme později pracovat, pochází z této zprávy. Po označení je výstupem strom, který je zobrazen na obrázku 4.1. Otázka položená uživatelem je označena jako S a tvoří kořen stromu, další úroveň tvoří přiřazené POS tagy a nejnižší úroveň stromu tvoří původní slova. Z označených slov jsou dále vybrána klíčová slova (krok



Obrázek 4.1: Strom po označení otázky z příkladu 4.3

5). Z hlediska první kategorie klíčových slov, tedy podstatných jmen, neobsahuje text otázky žádná taková klíčová slova. Jediné slovo, které by potencionálně mohlo být označeno jako takové klíčové slovo totiž odpovídá výjimce, tedy zdravící frázi. Z hlediska druhé kategorie – sloves, je vybráno slovo „`help`“, dalším slovem je „`please`“ a posledním „`follow`“.

Analýza pomocí POS taggingu a chunkingu vytvoří základ první věty odpovědi (krok 6). Z „`can you help`“ pomocí pravidla pro podmiňovací způsob vytvoří „`no, i can not help`“, v tomto případě negovanou formu.

Odhad očekávané délky odpovědi n (krok 7) slouží k hrubému odhadu, kolik vět bude odpověď obsahovat. Průměrná délka věty je pro tento příklad nastavena na 20 znaků, což je hodně nízká hodnota a to vede ke generování většího počtu počátků vět. Parametr *npos* udává, kolik počátků vět bylo vytvořeno na základě analýzy pomocí POS a chunkingu. V tomto případě je jeho hodnota 1. Délka zprávy je 48 znaků. Označení *nkey* udává počet počátků vět, které jsou vytvářeny pomocí technik popsaných v 4.5, tedy hlavně pomocí klíčových slov a je dáno vztahem 4.4. Celkový počet generovaných vět pak udává parametr n, což je vztah 4.5. Po dosazení do vztahů vyjde $nkey = 1$ a $n = 2$, budou vytvářeny celkem 2 věty.

$$nkey = \max\left(\frac{delka\_zpravy}{prumerna\_delka\_vety} - npos, 0\right) \tag{4.4}$$

$$n = nkey + npos \tag{4.5}$$

Jeden počátek odpovědi už byl vytvořen (v kroku 6). Další počátek (krok 8) je v tomto případě „`can you help`“, který vznikl vybráním klíčového slova „`help`“ a přidáním fráze „`can you`“.

Všechny počátky vět jsou řešeny odděleně. Před zasláním na vstup neuronové sítě je k nim připojena původní zpráva (krok 9). Délka konce každé z takových vět (tedy požadovaný počet slov, které budou výstupem neuronové sítě) je volena náhodným výběrem z intervalu hodnot 3 až 5[4]. Výstupem neuronové sítě pro první větu vzniklou analýzou pomocí POS a chunkingu, je doplnění odpovědi „`that now some of`“. Pro druhou větu, která vznikla pomocí klíčových slov je to pak „`others. thank you`“. Po spojení s odpovídajícími počátky vět jsou vytvořeny finální věty (krok 10) a po dokončení tvorby všech vět (krok 11) jsou spojeny – společně pak tvoří odpověď (krok 12), která je odeslána uživateli.

[4]Tyto hodnoty byly zvoleny na základě experimentů

4.8 Sentiment

Zpracování z hlediska sentimentu funguje na principu vyhledávání pozitivních a negativních slov. Za pozitivní se dají považovat například „`good`“ nebo „`happy`“ a negativní „`bad`“ nebo „`unhappy`“. Podle poměru zastoupení takových slov je výstupem porovnání textu míra pozitivnosti. Tato míra je v rozsahu -1.0 až 1.0 a bylo rozhodnuto, že na základě této míry bude sentiment klasifikován jako dobrý (*good*), neutrální (*neutral*) a špatný (*bad*), s rozdělením uvedeného intervalu rovnoměrně mezi tyto tři možnosti.

Takovýto přístup lze hlavně využít u vytváření odpovědi na otázku, která je položena jako tvrzení. Věta „`John walked`“ se dá považovat za neutrální, protože neobsahuje žádná emotivně zabarvená slova, navíc nejde o otázku, na kterou bychom podle gramatických pravidel mohli vytvořit šablonovitou odpověď přehozením slovosledu. Pro takový typ vět lze použít „`I see`“ nebo „`Hmm`“, případně smajlíka a zbytek věty doplnit posloupností vygenerovanou sítí.

I tak ale nelze spoléhat jen na generování (v tomto případně náhodného výběru z příslušné množiny frází) podle citového zabarvení otázky. Je to jedna z možností jak reagovat, ale stejně jako při dalších postupech, při použití malého množství přístupů si uživatel může po velice krátké době všimnout, že mu bot odpovídá šablonovitě - a to není žádoucí.

Kapitola 5

Implementace

Jádro programu je implementováno v jazyce Python a je využíván modul Natural Language Toolkit `NLTK`[1], který je využit pro práci s POS taggingem a chunkingem. Na tomto modulu je postaven i modul `TextBlob`[2], který posloužil k zjišťování míry sentimentu. Při úpravách slovosledu bylo občas vhodné použít základní tvary sloves, resp. z minulého tvaru slovesa vytvořit přítomný tvar apod., k tomu byl využit modul `Nodebox English Linguistics library`[3]. Co se týče rychlosti, POS označkování, chunkování, zjišťování sentimentu i vyhledávání vhodného tvaru slovesa, trvá zanedbatelnou dobu, ovšem načtení modulů nějaký čas zabere, to je ale potřeba provádět jen na začátku běhu programu.

Jak už bylo zmíněno v kapitole 3.1, k práci s jazykovým modelem a neuronovou sítí byl použit `RNNLM Toolkit`[4]. Pro využití v chatbotovi bylo potřeba implementovat několik úprav, protože nástroj v takové podobě, v jaké je šířen, nepodporuje prvně inicializaci skryté vrstvy na požadovaný kontext a poté generování výstupní sekvence slov na základě tohoto kontextu. Konkrétní úprava byla tedy v tom, že byly tyto dvě funkce spojeny do jedné. Navíc bylo potřeba zajistit, aby byla síť načítána pro komunikaci s každým uživatelem jen na začátku konverzace a dále zůstávala spuštěná. Načtení modelu trvá neúnosně dlouhou dobu a i když existuje požadavek na to, že chatbot by měl do odeslání zprávy přidat i čekací dobu, není vhodné toho dosahovat takovýmto způsobem. Navíc bychom se v takovém případě bavili o kontextu spíše v rámci jedné otázky, zatímco při spuštění sítě na delší dobu dosáhneme obsažení kontextu i mimo rámec jedné otázky (tedy vzhledem k tomu, že zde využíváme rekurentní neuronovou síť, která vychází i z předchozích stavů skryté vrstvy). Proto byla provedena další úprava a to, že nástroj očekává na standardním vstupu délku sekvence, kterou má generovat. Jako další očekává frázi, resp. text zprávy, pro kterou má inicializovat skrytou vrstvu (nastavit kontext). Poté je její výstup (sekvence slov) vypisován na standardní výstup. Parametry pro spuštění neuronové sítě s funkcí nastavení kontextu a generování výstupu (tedy s upravenou funkcí) jsou `-gencontext GENLEN CONTEXT`. Parametry jsou očekávány v tomto pořadí, kde `GENLEN` je délka generované sekvence a `CONTEXT` je text, na který má být nastaven kontext.

[1] http://www.nltk.org/install.html
[2] http://textblob.readthedocs.org/en/dev
[3] http://nodebox.net/code/index.php/Linguistics
[4] http://rnnlm.org, byla použita verze 0.4b

5.1 Komunikační rozhraní

Pro komunikaci existují dvě prostředí – webové prostředí a facebook. Program běží jako server, kdy je obsluha webové i facebookové části oddělena, běží tedy nezávisle na sobě. Jádro programu je uloženo v souboru `annabell.py`.

K řešení facebookového rozhraní slouží modul `xmpppy`[5], který využívá `XMPP` (Extensible Messaging and Presence Protocol), dříve označovaný jako `Jabber`. Tento protokol je využíván jako standard při IM (instant messanging) komunikaci. V souboru `logininfo` jsou uloženy přihlašovací údaje. Po připojení na server je spojení udržováno po celou dobu běhu programu a je pro něj spuštěn proces, který kontroluje, zda není k dispozici nová zpráva. Pokud je k dispozici, je aktivován handler pro obsluhu zprávy a následně spuštěn proces pro obsluhu zprávy.

Zpráva neobsahuje jen text zprávy, ale i informace o tom, kdo zprávu poslal, například jeho identifikátor na facebooku. Identifikátor je tvořen číslem, které je zakončeno příponou „`@chat.facebook.com`“. Pro unikátní identifikaci (ID) uživatele stačí číslo před příponou (z tohoto čísla je možné zpětně dohledat i jméno uživatele například pro věrohodnější komunikaci s uživatelem).

Webové rozhraní běží na portu `80` s protokolem `TCP` a využívá modul `twisted`[6], který zajišťuje provoz webové stránky na definovaném portu a protokolu. Modul `txws`[7] slouží pro práci s protokolem `WebSocket`(`WebSocket` umožňuje obousměrnou komunikaci po `TCP`), který využívá port `8076`. Vzhled stránky je v souboru `annabell.html` a tvoří ji textová oblast, kde se novější zprávy zobrazují více nahoře, odpovědi chatbota jsou označeny jménem `Anna`. Po přijetí první zprávy je potřeba prvně načíst model neuronové sítě, což může zabrat zhruba 10 sekund, proto je zobrazen text „`Anna is writing.`“ v řádku pod oknem, kde jsou zobrazeny všechny zprávy. Tento text je zobrazen i při dalších generováních odpovědí, není ale vkládáno žádné zpoždění a odpovědi přicházejí velice rychle, proto se text obvykle nezobrazí.

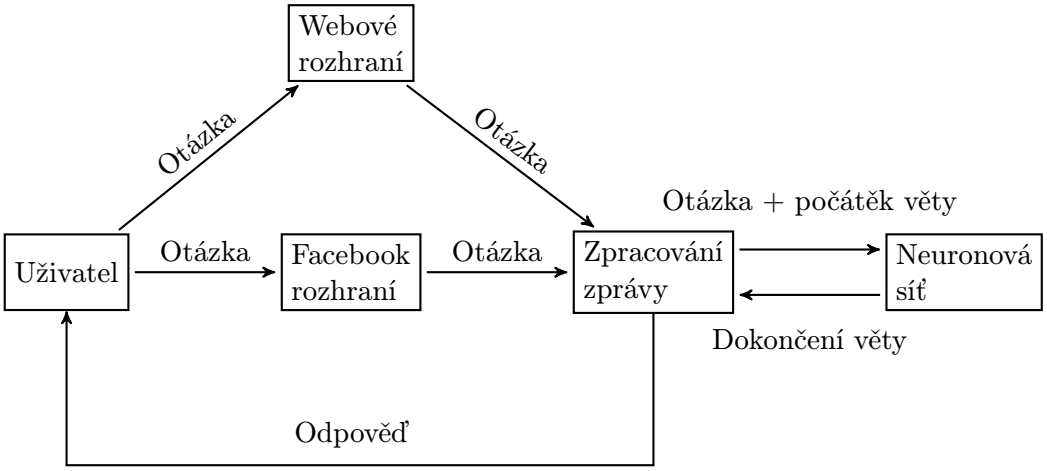
Implementace funkce tlačítka, odeslání a příjmu zprávy je v souboru `annabell.js` a využívá javascriptový framework `jQuery`[8]. Po načtení stránky je vytvořen socket pomocí protokolu `WebSocket` na přiděleném volném portu (jiném, než na kterém běží `WebSocket`, tedy ne `8076`). Po kliknutí na tlačítko nebo enter je odeslána přes socket zpráva, kterou zachytí na straně serveru handler obsluhy zprávy a poté je spuštěn proces obsluhy zprávy s tím rozdílem, že je detekováno, zda se jedná o zprávu z webového rozhraní a pokud ano, je unikátní identifikátor vytvářen jinak než u facebookového rozhraní. Tvoří jej číslo portu, na kterém byla zpráva obdržena (resp. na kterém bylo vytvořeno spojení s uživatelem). Při zahájení více konverzací jedním uživatelem (ve více oknech prohlížeče) je každá taková konverzace brána samostatně.

5.2 Zpracování zpráv a generování odpovědí

Obrázek 5.1 popisuje jednotlivé dále popsané části a demonstruje, jakými částmi prochází zpráva obdržená od uživatele. Pro každou zprávu je vytvořen proces, který obsluhuje analýzu zprávy (dále označen jako `zpracování zprávy`), tedy úpravu textu zprávy do podoby, v jakém bude dána na vstup neuronové sítě (POS, chunking, rozpoznávání sentimentu atd.)

[5] http://xmpppy.sourceforge.net, verze 0.5.0rc1
[6] https://twistedmatrix.com/trac
[7] https://github.com/MostAwesomeDude/txWS
[8] http://jquery.com/

a poté vytvořenou odpověď odešle uživateli. Text zprávy je převeden na malá písmena. `Zpracování zprávy` řeší téměř všechny části (kromě trénování modelu sítě) popsané v kapitole 4. V případě, že se jedná o novou konverzaci, je spuštěn proces s neuronovou sítí (dále



Obrázek 5.1: Princip komunikace mezi částmi programu

označen jako `neuronová síť`) pro nového konkrétního uživatele, který poběží definovanou dobu od přijetí poslední zprávy od konkrétního uživatele a čeká na případné další zprávy. V případě, že je zpráva od uživatele, pro kterého byl proces neuronové sítě už vytvořen, zašle `zpracování zprávy` zprávu tomuto procesu. Takto není nutné vždy čekat na spuštění a načtení modelu, ale jen při přijmutí první zprávy. Při dalších zprávách je generování rychlé.

Propojení `zpracování zprávy` a `neuronové sítě` je realizováno pomocí `subprocess`, konkrétně `Popen` a `PIPE`. Při prvním spuštění `neuronové sítě` pro uživatele je pomocí `Popen` spuštěn `RNNLM Toolkit` s konkrétními parametry. Další komunikace s ním probíhá pomocí `PIPE`. `Popen` tedy vytvoří propojení pomocí `PIPE` a při další komunikaci je na standardní vstup `neuronové sítě` zaslán text zprávy a na standardním výstupu `neuronové sítě` je očekáván její výstup. `Zpracování zprávy` mezitím čeká (konkrétní implementace čtení výstupu `neuronové sítě` je blokující operace). Po získání výstupu `neuronové sítě` dokončí `zpracování zprávy` vytváření odpovědi a odešle ho zpět uživateli. `Zpracování zprávy` je dokončeno (proces je ukončen), ale `neuronová síť` pro konkrétního uživatele běží dál. Periodicky je kontrolována doba od poslední obdržené zprávy od uživatele a v případě překročení maximální doby nečinnosti je proces pro `neuronovou síť` ukončen.

Z hlediska každého uživatele jsou pro něj ukládány některé informace, které jsou dále využity při generování odpovědí. Toho je docíleno třídou `user`. Již byl zmíněn unikátní identifikátor `name` a ten slouží jako klíč při přístupu k ostatním informacím. Dále je ukládán čas `time` poslední přijaté zprávy od uživatele. Pravidelně (každou minutu) probíhá kontrola neaktivních uživatelů a podle času poslední přijaté zprávy je rozhodnuto, zda byla překročena maximální doba neaktivity (5 minut) a pokud ano, je pro tohoto uživatele ukončen proces sítě i proces obsluhy. Dalším atributem je `pipe`, což je identifikátor otevřené pipe, tedy adresa, kam směrovat zprávy z `obsluhy zprávy`. Ke kontrole, zda se otázky neopakují, slouží `cache`, což je seznam, který obsahuje určený počet naposledy přijatých zpráv (10) a po zaplnění jsou nejstarší zprávy nahrazovány těmi novými. Pro ukládání klíčových slov jsou určeny dva seznamy, jeden pro podstatná jména (`keys`) a druhý pro slovesa (`verbs`). Nemají definovanou maximální velikost (resp. jsou dány maximální velikostí seznamu v pythonu).

Reakce na vlastní jméno má také vlastní atribut `annas` a slouží pro počítání výskytů vlastního jména v předchozích zprávách a pokud otázka obsahuje alespoň jeden výraz pro vlastní jméno, je hodnota toho atributu zvýšena o 1 (při více výskytech vlastního jména v jedné otázce také jen o 1). Souvisejícím atributem `chosen` je seznam pro uložení písně, kterou bude chatbot zpívat – resp. aktuální sloky. Atribut `picks` je seznam posledních několika počátků odpovědí (tzn. stav odpovědi před tím, než je poslán `neuronové síti`) z předešlých zpráv, který má nastavenou maximální velikost na 3 a opět po zaplnění nahrazuje starší odpovědi těmi novými.

Kapitola 6

Testování a další vývoj

6.1 Testování

Testování bylo prováděno na 50 uživatelích pomocí dotazníku. Pokud by byl uživatelům dán k dispozici chatbot a oni s ním mohli komunikovat na jakékoliv téma, bylo by obtížné z takového přístupu vyvodit konkrétní závěry. V úvahu připadal Turingův test, ovšem ještě před zahájením testování s účastí většího počtu uživatelů bylo zřejmé, že toto také není vhodný přístup (kritéria testu jsou poměrně přísná). Proto pro objektivní určení míry inteligence (MI) byl zvolen způsob hodnocení odpovědí chatbota uživateli, kteří s ním nekomunikují přímo, ale hodnotí předem vytvořený přepis konverzace.

V každé otázce dotazníku byla uživatelům předložena zpráva, která byla zaslána dvěma různým chatbotům. Uživatelé měli ze dvou možností vybrat tu, která se jim zdála více související se zprávou. Obě možnosti poté hodnotili na stupnici 1-10, kde 1 znamená, že odpověď nesouvisí vůbec a 10 znamená, že souvisí úplně. Jedním z chatbotů byl bot vytvářený v této práci (`Annabell`) a druhým byl `Cleverbot`. Možnosti byly předkládány v náhodném pořadí a v soupisu obsahu dotazníku v příloze C je v hranatých závorkách pro přehlednost uvedeno, která odpověď je od `Annabell` (`A`) nebo od `Cleverbota` (`C`). Uživatelé nevěděli, která odpověď přísluší konkrétnímu botovi – hranaté závorky ani jejich obsah nebyly v dotazníku uvedeny a místo nich uživatelé viděli pouze možnost (1) a možnost (2). U každé otázky byla nepovinná možnost uvést komentář a tuto možnost někteří uživatelé využili. Otázky dotazníku jsou tedy tvořeny přepisem konverzace mezi člověkem a chatboty. Konverzace obsahuje 9 otázek, resp. zpráv. Zprávy lze rozdělit na dvě části. V první části jsou takové zprávy, které obsahují kombinaci otázek a tvrzení. Druhou část tvoří zprávy, které obsahují jen otázky a tato část je z hlediska zhodnocení zajímavější.

Výsledky porovnání obou chatbotů z hlediska počtu uživatelů (uvedených v procentech), kteří zvolili jejich odpověď jako více odpovídající na zadanou zprávu, jsou uvedeny v tabulce 6.1. Tabulka navíc obsahuje pro oba chatboty míru inteligence, získanou váženým průměrem hodnocení uživatelů na stupnici 1-10. Odpověď `Cleverbota` byla vybrána jako více odpovídající na zprávu v 7 případech z 9. Otázky, ve kterých byla `Annabell` ohodnocena lépe jsou otázky s označením 7 a 8, jež jsou v tabulce zvýrazněny tučně. Z komentářů i z vypočtených hodnot lze vyvodit, že jedním z největších problémů při porovnávání odpovědí obou chatbotů je ten, že při vytvoření odpovědi, která je v porovnání s odpovědí druhého bota značně delší, je takový bot v tomto srovnání znevýhodněn. Tabulka 6.2 znázorňuje, jak se liší délka odpovědí (DO) pro jednotlivé otázky. `Annabell` vytváří delší odpovědi než `Cleverbot` (a také delší než je délka otázky). Některé odpovědi od `Annabell` jsou neúměrně dlouhé a v posbíraných komentářích se občas u těchto odpovědí vyskytovaly názory, že po-

Otázka	Cleverbot [%]	Annabell [%]	MI Cleverbot	MI Annabell
1	90	10	7.0	2.62
2	86	14	5.44	2.6
3	84	16	3.74	1.96
4	66	34	5.54	4.38
5	92	8	8.34	4.0
6	90	10	8.34	4.18
7	12	**88**	3.46	**6.7**
8	36	**64**	3.36	**4.16**
9	86	14	5.74	2.64

Tabulka 6.1: Porovnání Cleverbota a Annabell z hlediska vybraných odpovědí a MI

strádají jakýkoliv smysl co se týče slovosledu. U kratších odpovědí někteří uživatelé ocenili změnu slovosledu (což je jedna z věcí, na kterou se tato práce zaměřovala více). Na základě tohoto zjištění byla proto upravena funkce pro výpočet délky odpovědi, proto aktuální verze oproti verzi, na které probíhalo testování, vytváří kratší věty. Šlo jen o drobnou úpravu, ale výsledky se zdají být lepší, ty už ovšem nebyly dále nijak porovnávány a zpracovávány. Dá se říci, že tento problém by bylo v budoucnu potřeba řešit lépe.

Otázka	Délka otázky	*DO* Cleverbot	*DO* Annabell
1	32	17	38
2	90	43	170
3	118	34	314
4	28	43	43
5	44	12	124
6	19	25	51
7	18	55	44
8	45	44	201
9	133	50	369

Tabulka 6.2: Porovnání Cleverbota a Annabell z hlediska délky odpovědi

Vybraná odpověď uživateli je jen základní měřítko, o mnoho zajímavější je míra inteligence MI. Z hlediska `Annabell` je to poslední sloupec v tabulce 6.1. Otázky 1, 2, 3 a 9 jsou ohodnoceny pod hodnotou MI 3.0, všechny další jsou však ohodnoceny nad hodnotou MI 4.0. To souvisí s tím, že první 3 zprávy a zpráva 9 obsahují kombinace tvrzení a otázek (otázka 5 kombinaci obsahuje také, byla ale ohodnocena na MI 4.0), zatímco ostatní zprávy obsahují jen otázku. Vzhledem k tomu, že tato práce se více zabývala analýzou otázek než tvrzení, bylo u těchto zpráv očekáváno vyšší ohodnocení MI, což se také stalo. Pro tyto zprávy byly také očekávány odpovědi, které by mohly být ohodnoceny lépe, než ty od `Cleverbota`. Ze 4 zpráv obsahujících jen otázky, byly 2 ohodnoceny jako lepší než odpovědi vytvořené `Cleverbotem`, což je uspokojující. Otázka 7 obsahuje slovo „`follow`“, což je velmi často se vyskytující slovo na Twitteru (odkud jsou trénovací data pro model neuronové sítě) a i to je důvod, proč tato odpověď získala nejlepší MI ohodnocení u `Annabell`.

Chatbot	Rozsah	Průměrná MI	Průměrná DO
Cleverbot	1-10	5.66	36
Annabell	1-10	3.69	150

Tabulka 6.3: Porovnání průměrné MI a DO Cleverbota a Annabell

6.2 Další vývoj

Lze tedy říci, že z hlediska dalšího vývoje by bylo dobré se více zaměřit na analýzu tvrzení. K tomu by se dalo například využít více jazykových modelů, kdy by existovaly kategorie s určitým zaměřením a v kombinaci s vyhledáváním klíčových slov by byla určena kategorie a tedy i model, pro který má být načten. Z hlediska modelů by také bylo možné použít modely trénované na jiných datech, než jsou data z Twitteru. V průběhu vytváření této práce byla vyvinuta snaha i o extrakci rozhovorů z knih nebo filmových scénářů. Mohly by se však vyskytnout komplikace vzhledem k autorským právům a proto od toho bylo upuštěno, takový přístup by se ovšem použít dal. Alternativou jsou webové diskuze, případně jiné webové Q-A zdroje.

Další zajímavou možností je vytvoření databáze otázek, zadaných uživateli, a k nim přiřazených odpovědí chatbota. První možností jak toho docílit, je spustit fázi, ve které chatbot poběží a uživatelé budou hodnotit MI odpovědí chatbota na otázky (opět na stupnici 1-10), které mu uživatelé položí. Otázka, odpověď i hodnocení MI poté budou uloženy do databáze. Další možností je i neohodnocené dvojice otázka-odpověď ukládat (což je užitečné i pro zpětné zobrazení historie) a ohodnotit je bez účasti uživatele pomocí způsobu, který využívá `Vrajitoru`, tedy klíčových slov. Při vytváření odpovědi pomocí neuronové sítě by pak bylo možné brát v potaz i uložené odpovědi (porovnat je, případně zkombinovat). Proto jsou otázky i odpovědi ukládány, ovšem jejich zpětná analýza je alternativa do budoucna.

Při generování odpovědí neuronovou sítí by také šlo generovat větší počet odpovědí a z nich na základě klíčových slov vybrat tu, která nejlépe odpovídá otázce. Výběr může být také založen na analýze z hlediska POS, avšak tentokrát by analýza probíhala u zbytku vět, tedy částí generovaných neuronovou sítí. Takovým způsobem by šlo vytvářet dynamicky definované dvojice pravidel otázka-odpověď. Z vygenerovaných výstupů neuronové sítě by byl poté vybrán ten nejvíce se blížící se vybrané dynamické šabloně.

Při generování odpovědi je důležitá role náhody. Pokud bychom se snažili vytvořit deterministický automat, popřeli bychom tím jednu z hlavních vlastností člověka, a tím je nepředvídatelnost. Je nežádoucí, aby při položení stejných nebo podobně strukturovaných otázek přicházely podobné nebo dokonce stejné odpovědi. Proto by bylo dobré při vytváření počátků vět nepoužívat jen pravděpodobnost zvolenou tak, jak bylo uvedeno v kapitole 4, ale vycházet i z konkrétního modelu chování určitého typu člověka (podobně jak to dělá ELIZA, která modeluje chování psychoterapeuta).

Konkrétně pro otázky začínající „`why`“ nebo „`what`“ by nejspíše šlo použít vyhledávání ve `Wordnet` (součást modulu `NLTK`), pomocí něhož lze vyhledat definici pro konkrétní slovo s POS tagem. Po nalezení předmětu nebo slovesa pak nalezená definice (nebo její část) lze použít jako základ pro odpověď ve formě „`I heard that`“ + `<PREDMET>` „`is`“ `<DEFINICE_WORDNET>` + `<TEXT_GENEROVANY_SITI>` nebo v podobném stylu.

V odpovědích se mohou vyskytnout sprostá slova. Původně bylo plánováno taková slova po vygenerování odpovědi odstranit, nakonec byla ale ponechána beze změny. Pokud zpráva od uživatele obsahuje sprostá slova, která mohou být označena jako klíčová slova, odpověď

by je měla obsahovat také. Neuronová síť sama o sobě sice může dokončit počátek věty i sprostými slovy, to ovšem vychází z jazykového modelu, který tato práce respektuje. Pokud by bylo z určitých důvodů nutné vyfiltrovat sprostá slova z odpovědi, taková slova by bylo potřeba nahradit jejich slušnými formami, případně jiným vhodným způsobem. V textu dotazníku v příloze C některé odpovědi původně obsahovaly sprostá slova, a proto byla cenzurována pomocí tagu `<BAD_WORD>`.

Při porovnávání modelů s různou velikostí skryté vrstvy se vyskytl problém s načítáním modelu. Tento problém se týká rychlosti načítání. Pro zvolené hodnoty velikosti skryté vrstvy 300-400, které vychází z upravených doporučených hodnot popsaných v práci zabývající se `RNNLM Toolkitem` [7], je model načítán relativně rychle, ovšem při použití vyššího počtu neuronů ve skryté vrstvě by bylo potřeba změnit způsob načítání neuronové sítě pro každého uživatele. Pokud by měl uživatel čekat několik minut na první odpověď, pravděpodobně by spojení dříve ukončil. Toto by bylo možno vyřešit přednačítáním modelů pro definovaný počet uživatelů podle aktuální zátěže serveru, takže by uživateli byl přidělen už načtený model.

Kapitola 7

Závěr

Podařilo se implementovat chatbota Annabell, který je schopen komunikace v reálném čase s uživateli na sociální síti Facebook a také přes webové rozhraní. Chatbot není vázán jen na tato dvě rozhraní, vzhledem ke zvolenému způsobu komunikace (XMPP) je možné jej použít obecně pro komunikaci s jakýmkoliv klientem, který podporuje použití tohoto protokolu.

Využití samotné neuronové sítě z hlediska jazykových modelů se ukázalo být nedostačující, proto byly převzaty některé obecné techniky existujících chatbotů a byly upraveny do podoby, která lépe vyhovuje použití v kombinaci s neuronovou sítí. Jednalo se zejména o vytvoření základních obecných šablon odpovědí pomocí označení slovních druhů. Tento přístup vychází z šablon definovaných celými frázemi u klasických chatbotů, jen jsou místo konkrétních slov použity slovní druhy.

Annabell dokáže reagovat na uživatelské podněty a provádí jejich analýzu, na jejímž základě vytváří odpověď. Hlavní výhodou je to, že se snaží o dynamický způsob generování odpovědí. Takový způsob vytváření odpovědí v oblasti chatbotů není obvyklý a činí tohoto chatbota unikátním. Jeho potenciál je zajímavý – co se týče dalšího vývoje lze provést určitá rozšíření, která byla naznačena v kapitole 6.2. Navíc je generování odpovědí po načtení jazykového modelu velice rychlé, čehož lze v budoucnu také dobře využít. Jazykový model, ze kterého vychází generování odpovědí se dá nahradit jinými modely, více zaměřenými na specifickou oblast tématu. Porovnání s Cleverbotem ukázalo, že Annabell je v určitých oblastech schopna konkurovat ostatním již vytvořeným chatbotům. V oblastech, kde se to nepovedlo je ovšem prostor pro další vývoj.

Literatura

[1] Englisch-hilfen. online, 2014.
URL http://www.englisch-hilfen.de/en/grammar/tenses_table.pdf

[2] Loebner Prize. online, may 2014.
URL http://en.wikipedia.org/wiki/Loebner_Prize

[3] Neural networks. online, Duben 2014.
URL
http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[4] Part of speech. online, Duben 2014.
URL http://en.wikipedia.org/wiki/Part_of_speech

[5] Bird, S.; Klein, E.; Loper, E.: *Natural Language Processing with Python*. O'Reilly Media Inc, 2009, iSBN 978-0-596-51649-9.

[6] Goodman, J. T.: A Bit of Progress in Language Modeling. Technická zpráva, Microsoft Research, 2001.

[7] Mikolov, T.: *Statistical Language Models based on Neural Networks*. Dizertační práce, FIT VUT v Brně, 2012.

[8] Shawar, B. A. A.: *A Corpus Based Approach to Generalising a Chatbot System*. Dizertační práce, University of Leeds, 2005.

[9] Stenberg, M.: Artificial Intelligence - ELIZA and MegaHAL a Brief Analysis. 2006.

[10] Taylor, A.; Marcus, M.; Santorini, B.: The Penn Treebank: An Overview. 2003.

Příloha A

Obsah CD

Následující výčet je ve tvaru: `název_adresáře` – obsah

1. `annabell` – zdrojové kódy programu
2. `annabell/models` – použitý model
3. `annabell/users` – ukázka souborů s uloženými zprávami, které vznikly komunikací uživatele a chatbota
4. `doc` – technická zpráva v `.pdf` a `.tex`
5. `rnnlm` – upravená verze `RNNLM Toolkitu`
6. `training` – skripty pro filtrování trénovacích dat a trénování modelů, připravená adresářová struktura pro uložení trénovacích dat

Příloha B

Zprovoznění programu

Adresa, na které momentálně běží webové rozhraní, je:

http://109-123-216-074.tvujweb.net

Facebook rozhraní je dostupné na adrese:

https://www.facebook.com/anna.doll.716

Na těchto adresách je možné chatbota vyzkoušet bez nutnosti kompilace ze zdrojových souborů. Chatbot je aktivován po příjmu zprávy od uživatele. Webové rozhraní obsahuje základní prvky pro zobrazení konverzace, při použití facebookového rozhraní stačí poslat zprávu uživateli `Anna Bell`. Vytváření první odpovědi obvykle trvá zhruba 10 sekund, ve webovém rozhraní je psaní zprávy chatbotem znázorněno, ale na facebooku to není znázorněno nijak.

Následující popis je brán z hlediska CentOS distribuce Linuxu, pro jiné distribuce se může nastavení lišit a nebylo testováno. Jádro programu je v jazyce Python a k jeho spuštění je potřeba verze 2.7. Kromě základních modulů jsou dále použity moduly `xmpppy`, `nltk`, `en`, `textblob`, `twisted` a `txws`. Pro postup, jak tyto moduly zprovoznit je potřeba navštívit jejich domovské weby (uvedené v poznámkách pod čarou v kapitole 5), kde je pro každý z nich uvedeno, jak konkrétně postupovat. Pro neuronovou síť je použit `RNNLM Toolkit` ve verzi 0.4b, kdy byly změneny soubory `rnnlm.cpp`, `rnnlmlib.cpp` a `rnnlmlib.h` za účelem přidání parametru `-gencontext`. Upravená verze je přiložena ve složce `rnnlm` s upraveným `make` pro CentOS.

Pro spuštění webového rozhraní je potřeba mít nainstalován `Apache HTTP Server` a povolen TCP port, na kterém webové rozhraní poběží (defaultně port `80`). `Websocket běží` defaultně na portu `8076`. Adresa serveru je nastavena defaultně na `localhost`. Povolení portu je potřeba někdy provést v konfiguračním souboru `/etc/httpd/conf/httpd.conf` a také v nastavení firewallu v souboru `/etc/sysconfig/iptables`.

Pro facebook rozhranní jsou přihlašovací údaje uloženy v souboru `logininfo`, kdy každý údaj je na novém řádku. Pořadí je `ID, PASS, SERVER`, pro připojení na facebook je `ID` ve formátu `fb_id@chat.facebook.com`, `PASS` je heslo a SERVER je `chat.facebook.com`. Program je spouštěn pomocí příkazu:

```
twistd -ny annabell.py
```

Pro spuštění programu jako deamona je nutné jej pustit pomocí příkazu:

```
nohup twistd -ny annabell.py &
```

Zprávy jsou ukládány do adresáře `users`. Adresář `models` obsahuje přiložený model, na kterém bylo prováděno testování. Posledním adresářem je `training`, kde jsou skripty, které byly použity na vytváření jazykových modelů.

Kromě nich tento adresář obsahuje i skript (v jazyce Python) pro sběr trénovacích dat z Twitteru `getTweets.py`, ve kterém je nutné zadat 4 přístupové kódy[1] (consumer key, consumer secret, access token key, access token secret). Ke svému běhu potřebuje modul `TwitterAPI`[2].

[1] Jak kódy vygenerovat je popsáno na https://dev.twitter.com/docs/auth/tokens-devtwittercom
[2] https://dev.twitter.com

Příloha C

Testovací dotazník

1. hello, this is just casual talk.
   - [A] hi, talk christmas up with spamming.
   - [C] yes, i know that.
2. my name is john and i would like to know something about you. could you tell me something?
   - [C] i would like to know about quantum physics.
   - [A] i could tell him :) christmas a sun :) "i don't love you i hate head people and him, can you john hahaha - time with him :) name it :) :) today stats: one follower, one.
3. yeah, it's quite interesting. but i think we should focus on something else. do you like lemons? i really like lemons.
   - [C] where did you learn what you know?
   - [A] have you 's stunning walk you promote like :d 3:16 beauty smell have you think for me, never forget folks that's the captions from a box, would you focus who sent you extremely painful :d thats what it have you like me just b... hehe. sure perhaps don't care what he's, i just need you. maybe if i wanna <BAD_WORD> <BAD_WORD>.
4. have you seen the new movie?
   - [C] i thought we were talking about hamburgers.
   - [A] i have seen today your the only thing that.
5. please focus. do you watch movies sometimes?
   - [A] no, i do not watch it there. mike glennon came through whole :) :) goingg :) please do you see hella good for ya <BAD_WORD> so far.
   - [C] yes. do you?
6. would you marry me?
   - [C] i will if you are female?

- [A] i would not marry her and all coz you're in 3 days.

7. can you follow me?
    - [C] i will always forgive you, no matter what you do wrong.
    - [A] yes, i can follow you u u guys. going to pe.
8. do you love me? do you hate me? can you swim?
    - [A] i do not love him? today stats: no new followers, one unfollower via :) happy i hate everything who likes and the way i'll ever ends with some cute fellows, i can not swim is you can get over 24/7 lol.
    - [C] and not all men are stupid. and i love cats.
9. what if i told you this world is real but you are not. you are not even here. would you beleive me? and what would you do about that?
    - [A] i is so <BAD_WORD> about every girl that can collapse is already made, woohoo! check it yes, i would beleive him i ball :("even when i'm lose a, i would do anything right, or forget about not know if u know where i would you beleive are :) so if it :), i guess toilets just got that will? i've collected :) gold coins! :) do you love eachother even if you see your promo.
    - [C] i would feel the object in my physical enviroment.

Příloha D

Přepis konverzace s chatbotem

User: hello, how are you?
Anna: good afternoon, i am like it :).
User: thats interesting. i feel sad, really sad. something awfull happened to me today.
Anna: we get close less.
User: thats not what i was talking about
Anna: awfull just hold on <BAD_WORD>.
User: what?
Anna: would you talk to you.. but.
User: no, i dont think you understand me
Anna: have you dont go stand up until they're.
User: are u reading what i type?
Anna: would you u :) tuesday shift hari.
User: hmm
Anna: we just do this.
User: no, we dont
Anna: i think, there's no snow.
User: really
Anna: thats all my baby.
User: this is not even funny
Anna: hmm mula diamond alright ty.
User: so, do you want to hear my story or not?
Anna: no, i do not want to talk. yeah.
User: you were created to talk with people
Anna: people who get some.
User: no exceptions, i want to talk so we will talk now
Anna: exceptions lunch cook pizza :).
User: sure
Anna: story you feel disrespected lol.
User: now we are getting somewhere
Anna: we can talk come out.
User: ok. but i forgot the story already
Anna: i could do this.
User: hmm, now you want to talk? no.
Anna: would you want my cute big dream too!

User: yeah, tell me about that
Anna: yeah gonna make over.
User: do you know my name?
Anna: no, i do not know free dont txt.
User: well, my name is john
Anna: name he doesn’t have.
User: who is he?
Anna: he is the perfect singer i.
User: thats so cool!
Anna: do you john lennon i’m pulling _styles:.
User: no, i do not like him
Anna: john :) ! ""i.
User: it seems you like him a lot
Anna: do you seem like <BAD_WORD> like.
User: dont be rude, it’s disrespectful
Anna: would you ’s have milk :) and.
User: and what?
Anna: can you like me back ill.
User: sure, i can like you back
Anna: dont you aggy go follow.
User: follow where?
Anna: would you like lord it lovely means you.
User: nope
Anna: i hate it either.. lmfao.
User: what do you hate?
Anna: yes, i hate him."helloooo harold.
User: i am not harold :(
Anna: hate austin. morning.