Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informačních technologií

Komunikace člověk-stroj s využitím metod umělé inteligence

Diplomová práce

Autor: Adam, Richter
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Karel Mls, Ph.D.

Hradec Králové Květen 2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 10.5.2020 Adam Richter

Poděkování:

Děkuji vedoucímu diplomové práce Ing. Karlu Mlsovi, Ph.D. za metodické vedení a trpělivost při konzultacích, ochotu a rady. Dále taktéž děkuji Mgr. Rudolfu Vrabcovi za plodnou diskuzi při tříbení nápadů a rady při sepisování práce.

Anotace

Diplomová práce se zabývá problematikou komunikace stroje a člověka s využitím umělé inteligence. Za umělou inteligenci jsou v diplomové práci brány neuronové sítě, které řeší spoustu dílčích úkonů v komplexním řešení. Zároveň pak systém imituje lidskou komunikaci včetně neverbální komunikace. Práce představuje jednotlivé metody, které tvoří celý jednotlivé části, které převádí nejen lidskou mluvu do slov, ale následně na ní pak reagují a provádí příkazy, které uživatel požaduje. Pro ověření takovéhoto řešení je následně navržen a implementován systém, který funguje na operační platformě Linux a využívá knihovny OpenCV, Tensoflow a Google Api. V práci jsou reprezentovány a zhodnoceny všechny dosažené výsledky, i je zdokumentován proces vývoje celého systému.

Annotation

The diploma thesis deals with the issue of machine-human communication using artificial intelligence. In the diploma thesis, neural networks are taken for artificial intelligence, which solve a lot of partial tasks in a complex solution. At the same time, the system imitates human communication, including non-verbal communication. The work presents individual methods that form the whole individual part, which not only converts human speech into words, but then responds to it and executes commands that the user requires. To verify such a solution, a system was developed that works on the Linux operating platform and uses the OpenCV, Tensoflow and Google Api libraries. The work represents and evaluates all achieved results, and the process of development of the whole system is documented.

Obsah

1 Úvod

Umělá inteligence a její formy jsou fenoménem posledních let. Různé formy inteligentních zařízení jsou lidmi využívány v různých aspektech života.

Na pojem inteligence se dá nahlížet dvěma způsoby. Jednou z možných definicí muže být definice lidské inteligence. Inteligence je podle ní druh činností, které lidé často dělají.[1] Druhým výkladem je definice používaná v technice a počítačové vědě a definuje umělou inteligenci. Definuje tedy samotnou umělou inteligenci jako odvětví informačních technologií, které je spojeno s automatizací a inteligentním chováním. [2]

S pojmem umělá inteligence jsou nyní spojená různá zařízení. Může se jednat například o televizor s umělou inteligencí, či třeba různé senzory.[3] Ty mohou najít využití například v počítačové bezpečnosti, kde detekuji anomálie v síťové komunikaci. Umělé inteligence jsou v těchto případech programy a systémy, které obsahují automatizační algoritmy. Inteligence jako taková by se dala definovat jako něco, co všichni lidé sdílejí, a co je odlišuje od zvířat. Zahrnuje v sobě jazyk, schopnost rozšiřovat a vyvíjet kulturu, přemýšlení, názory a testování hypotéz. [4] Běžně se totiž nesetkáváme třeba s televizory, které by si povídaly se svými uživateli, a na základě rozhovoru by jim nabídly například filmy, které by odpovídaly aktuální náladě vlastníka televize. Avšak i podobné systémy si nyní nachází cestu do lidských domovů, a to v podobě různých domácích systému, jako je například Alexa od firmy Amazon, anebo Google Home Assistent. I když by se mohlo zdát ze interakce lidí s umělou inteligencí prostřednictvím virtuálních agentů, socialbotu a softwaru pro generování jazyka nezapadá do teorií o mezilidské komunikaci. [5] Tyto systémy naslouchají uživateli a snaží se pomocí analýzy jeho řeči plnit jeho příkazy, například tykající se osvětlení, hudby, nebo získávání informací.

2 Cíl práce

Cílem práce je vytvoření funkčního a veřejně použitelného systému, který bude použitelný při komunikaci mezi člověkem a strojem pomocí mluvené řeči. Systém bude mít modulární strukturu, kde jednotlivé modulu budou na sobě krom dat nezávislé, a bude snadno rozšiřitelný. Systém také bude mít možnost reagovat na lidské příkazy a na jejich základě vykonávat úkony.

3 Analýza systému

Při tvorbě systému je možné využít velkého množství různých přístupů k analýze a vývoji softwaru Jednotlivé přístupy popisují, jakým způsobem je možné přistupovat k vývoji softwaru. Jako příklad takových metodik je možné zmínit například metodiky Prototypování, Spirála nebo Vodopád. [6]

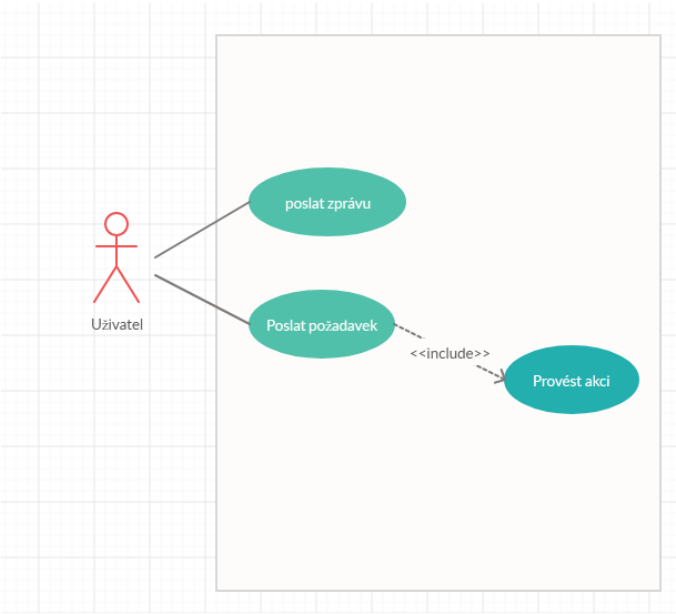
Metodika prototypování začíná vývojem a následně prochází postupně vylepšeními, demonstrací a zpátky k vývoji. Tento kruh je opakován, dokud není dosaženo koncové verze, která je otestována a následně nasazena.

Spirála narozdíl od prototypování začíná analýzou, následuje hodnocení, vývoj a plánování. Na konci plánování je pak znovu software analyzován, a tak stále dokola.

Při vývoji pomocí metodiky vodopád jsou nejprve specifikovány požadavky, pak je navržen design, pak je celý systém implementován, verifikován a pak nastává údržba.

Při návrhu a řešení pro systém, který bude implementován na základě této práce bylo nejprve nutné specifikovat hlavní požadavky. Těmi to požadavky byly nepřetržitý chod, možnost fungování i na zařízeních se slabším výkonem.

Na základě těchto požadavků bylo zvoleno vytvoření systému se dvěma částmi, a to serverovou a klientskou.



Obrázek 1: Use case diagram systému

Na základě těchto požadavků byl pro vývoj softwaru zvolen přístup prototypování. Byly průběžně testovány jednotlivé části, a upravovány do podoby, která splňovala požadavky na výsledný software.

4 Teoretická část umělá inteligence

Myšlenka umělé inteligence a inteligentního stroje je poměrně stará. Ovšem až v posledních letech je vidět velký pokrok ve zkoumání a vývoji v této oblasti, a i v porozumění celé problematiky. Jako počátek hlubšího porozumění a zkoumání umělé inteligence muže být brán článek Allana Turinga: „Výpočetní technika a inteligence“ [7].

V tomto článku je nastíněna možnost testování způsobilosti umělé inteligence jednat s lidským oponentem. Tato umělá inteligence nejen že naplní tuto způsobilost, ale také simuluje lidskou komunikaci takovým způsobem, že ji vyhodnocovatel-nerozezná od člověka.

Od té doby výzkum a pokrok v oblasti umělé inteligence velmi pokročil. Byly vytvořeny počítačové programy, které dokázaly hrát hry jako šachy nebo dámu. Hraní těchto her byly přisuzovány vlastnosti specifické pro rozhodování lidí a jejich hraní bylo přisuzováno, že jsou závislé na intelektuálních schopnostech hráče. Intelektuálními schopnostmi jsou brány jako na příklad předvídání tahu nebo i vytvoření strategie a klam protihráče.

Obrázek 2: První turnaj, kdy počítač vyhrál proti lidskému mistrovi světa v šachách. [8]

Hraní těchto her může některým lidem přijít náročné. Když však tyto úkony, spojené s hraním, řeší počítačový program, může to být pro počítač jednodušší.

Programy jsou pak testovány často proti lidským subjektům, nebo jsou matematicky analyzovány. Turing ve své práci definuje test dokazující inteligenci stroje.[9]

V turingově testu je testován počítačový program oproti člověku. Výsledek testu pak záleží na druhé osobě, která vyhodnocuje zprávy posílané programem a člověkem. Program projde tímto turingovým testem právě tehdy, když lidský vyhodnocovatel na základě komunikace se oběma subjekty nebude chopen rozlišit, jestli vede konverzaci se strojem, nebo s dalším člověkem. Tento test je navržen tak, že lidský subjekt není ovlivněn fyziologickými projevy stroje, jako jsou tvar, vůně a podobné. Jeho soustředění se tedy pouze zaměřuje na komunikaci jako takovou.

Tento test je však náchylný na lidské projevy při písemné komunikaci, jako jsou například chyby v gramatice, chybějící interpunkce apod.

Pokud tedy použijeme zmíněný test jako definici umělé inteligence, pak tedy je umělá inteligence taková, kdy člověk není schopen rozeznat její komunikaci od komunikace s člověkem.

Oproti tomu definice umělé inteligence, jak se o ní ve své práci zmiňuje John McCarthy a Patrick J. Hayes [10], má komplexnější znění. Jejich definice je rozdělena do dvou částí, a to epistemologická a heuristická. Epistemologická část je reprezentace světa v takové formě, že řešení problémů vyplývá ze skutečností vyjádřených ve vyobrazení. Heuristická část je mechanismus, který na základě informací řeší problém, a rozhodne, co udělat.

Tato práce se především zabývá přístupem, který je bližší Turingově definovaní umělé inteligence, a tedy i maximalizací způsobu a typu komunikace s umělou inteligencí, aby lidský subjekt s ní dokázal lépe a přirozeněji komunikovat.

Pro lepší komunikaci, bude umělá inteligence tvořena z několika částí, které zároveň podpoří její schopnost projít Turingovým testem. Stejně jako člověk bude mít tento systém schopnosti zpracovat obrazové a zvukové informace, a na jejich základě bude schopen interakce s lidmi pomocí rozhodovacího jádra systému.

4.1 Morální problémy vzniku umělé inteligence

Při tvorbě umělé inteligence mohou vyvstat různé znepokojivé otázky. Nicholas Agar ve své práci [11] zmiňuje možnost, že inteligence bude vylepšovat sama sebe, aby zvládla úkoly, které potřebuje vyřešit.

V aktuálním kontextu, ve kterém je práce tvořena je dobré se nad takovými otázkami zamýšlet, a mít je v paměti, avšak zatím není nutné se těchto situací nastíněných v otázkách obávat. Dále popisované technologie jsou zatím limitovány silou a schopností vývojáře. Vývojář musí mít v hlavě konkrétní řešení problému, a jeho zvládnutím se teprve potom může posunout dál. Pokud tedy není schopný vyvinout řešení, nemá možnost pokračovat dál.

Taktéž není tedy možné, aby se nějakým způsobem počítačové programy vyvíjely. Protože samotný kód sice dokáže vytvořit a zkompilovat kód, avšak bylo by

potřebné jej zavést do systému, datové struktury, a taktéž by bylo potřeba ho měnit na základě potřeb, a to aktuálně není pro program možné.

Tyto argumenty ovšem neznamenají, že nebudou aktuální v budoucích letech, kdy technologie pokročí takovým způsobem, aby již toto bylo možné. Ohledně komplexní umělé inteligence výzkum mezi odborníky z roku 2016 ukázal, že více než 50% dotázaných expertů usuzuje, že umělá inteligence schopná projít turingovým testem bude trvat více než 26 let[12].

4.2 Součásti umělé inteligence

Umělá inteligence by měla mít několik základních funkcí které by její komunikaci přiblížili uživateli. Následující kapitoly popisují metody, které pomáhají zpracovat informace, které jsou pro umělou inteligenci důležité při jeho funkci.

4.2.1 Zpracování obrazu

Pro mezilidskou komunikaci je velmi důležitá vizuální stránka komunikace. Lidé na jejím základě mohou správně určit různé aspekty, které konverzaci ovlivňují před jejím započetím. Až 55% mezilidské komunikace tvoří nonverbální projevy jako řeč těla, výraz tváře, intonace, atd. [13]. Dalšími takovými aspekty jsou nálada, pohlaví, případně i věk. Schopnost rozpoznávat objekty je i pro umělou inteligenci důležitá, ale ne zcela nezbytná. Aktuálně nejvyužívanější umělé inteligence, jako Google home nebo Alexa tu to schopnost nemají. Nemají totiž přístup ke kamerám a neobsahují software, který by rozpoznání z obrazu zvládl.

Kvůli tomu tyto umělé inteligence postrádají jisté schopnosti, které by mohly zlepšit fungování a celé zdání umělé inteligence. Není tak možné identifikovat osobu pouze na základě jejího vzhledu. Příkazy, které by měly po každého jiného uživatele jiné výsledky např. oblíbená písnička, tak nejsou splnitelné. Není tak možné bez rozeznávání hlasu rozeznat, kdo vydal povel a identifikovat, jestli nemá speciální význam odlišný od původního významu.

Jestliže umělá inteligence zpracovává i obraz, dostává silný nástroj pro svou práci. Systém jako takový může obsluhovat funkce založené na poloze uživatele, jako například automatické zapínání a vypínání světel. Taktéž to přináší možnost interagovat s uživatelem i v jiných místnostech, anebo i kontinuálně, když uživatel

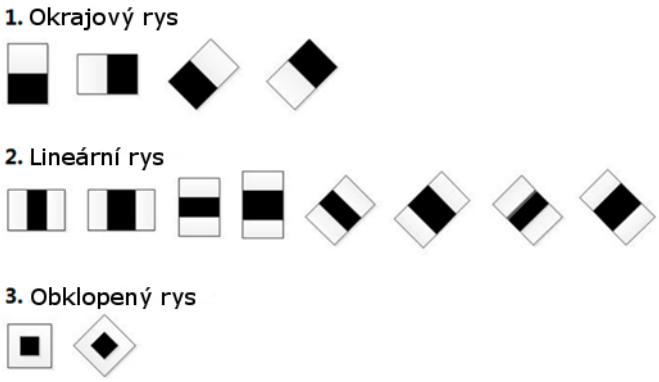
přechází z místnosti do místnosti. Dokáže totiž sledovat uživatele a identifikovat jeho příkazy na základě umístění mikrofonu.

4.2.1.1 Haar cascade detection – rozpoznávání obličeje

Při rozpoznávání obličeje existuje mnoho přístupů. Některé jsou založeny na strojovém učení, jiné jsou generické. Přístupy založené na strojovém učení jsou výpočetně méně náročné, než generické. Avšak generické dosahují velice často lepších výsledků, pokud jde o umístění či velikost obličeje, jak ve své práci uvádí Staffan Reinius [14]

Haar klasifikace je metoda založená na rozhodovacích stromech. V této metodě je fáze školení statistická. Klasifikační model je školen na reálných datech – v tomto případě na fotkách. Na jeho základě je vytvořena posílená kaskáda pro vyřazení chyb. Posílená v tomto případě znamená, že klasifikátor určující s vysokou pravděpodobností, je vytvořen z vícero slabších klasifikátorů. Slabší klasifikátor úspěšně vyhodnocuje alespoň s 50 % úspešností, že se v obraze element nachází.

Základním kamenem pro detekci objektu pomocí Haar klasifikace jsou Haar funkce. Tyto funkce namísto použití hodnot intenzity pixelů používají změnu kontrastu jednotlivých pixelů. Konkrétně používají změnu hodnoty kontrastu mezi sousedními pixely a mezi sousedními obdélníkovými skupinami pixelů. Rozdíly v kontrastu mezi skupinami pixelů se používají k určení relativních světlých a tmavých oblastí v obraze. Dvě nebo tři sousední skupiny s relativní odchylkou v kontrastu tvoří Haaruv rys. Haarovy rysy se používají k detekci prvků v obrazu. Haarovy rysy jsou libovolně škálovatelné, je možné je libovolně zvětšovat anebo zmenšovat a měnit tím velikost zkoumané oblasti. Díky této vlastnosti je možné v obraze detekovat objekty nezávisle na jejich velikosti, záleží pouze na seskupení Haarových rysů. Haarových rysů je několik druhů, jak ukazuje Obrázek 3.



Obrázek 3: Haarovy rysy (převzato z [15])

Základní pravoúhlé rysy obrazu jsou počítány pomocí mezilehlé reprezentace obrazu a jsou nazývány „Integrálním obrazem“. Integrální obraz je tedy pole obsahující součty pixelů umístěných přímo nad a nalevo od zkoumaného pixelu na pozici (x, y). Pokud je tedy původní obrázek označen jako $A[y, x]$, a $A_y[y, x]$ je integrální obraz, pak integrální obraz je vypočítán pomocí **Chyba! Nenalezen zdroj odkazů.**:

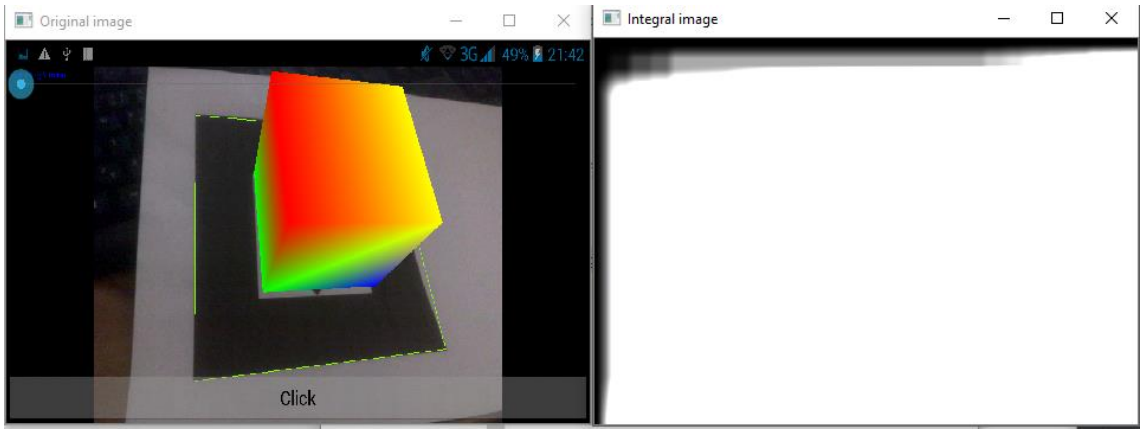
$$\mathbf{AI}\,[\mathbf{x},\mathbf{y}] = \sum_{x^{\check{}} \le x, y \le y} \boldsymbol{A}(\boldsymbol{X}', \boldsymbol{y}') \tag{1}$$

Prvky otočené o 45°, jak je zavedli ve své práci Lienhart a Maydt [16], vyžadují své vlastní označení, nazývané rotovaný integrální obraz, nebo rotovaný součet pomocného obrazu. Rotovaný integrální obraz je vypočten tak, že je zjištěn součet hodnot intenzity pixelů, které jsou umístěny ve 45° úhlu od zkoumaného pixelu s hodnotou x vetší než zkoumaný pixel a s hodnotou y menší než zkoumaný pixel. Pokud tedy je $A[x, y]$ úvodní obrázek a $AR[x, y]$ je otočený integrální obraz, pak integrální obraz je počítán pomocí **Chyba! Nenalezen zdroj odkazů.**:

$$\boldsymbol{AR}[\boldsymbol{x},\boldsymbol{y}] = \sum_{\boldsymbol{x}' \le \boldsymbol{x}\ \boldsymbol{x}' \le \boldsymbol{x} - |\boldsymbol{y} - \boldsymbol{y}'|} \boldsymbol{A}(\boldsymbol{x}', \boldsymbol{y}') \tag{2}$$

Vypočítání obou integrálních obrazů vyžaduje pouze dva průchody pro každé pole. Použitím vhodně zvoleného integrálního obrazu a rozlišením tohoto obrazu mezi osmi a šesti prvky pole, které tvoří dva až tři spojené obdélníky, lze vypočítat

znak libovolného měřítka. Výpočet je tedy rychlý a efektivní. Další výhodou tohoto přístupu je, že výpočet rysu různých velikostí vyžaduje stejnou pracnost jako rys skládající se z pouze několika pixelů.



Obrázek 4: Zdrojový obrázek a jeho integrální obraz (vlastní tvorba)

Pro určení výskytu objektu v obraze není potřeba všech klasifikátorů. Pokud by byly všechny funkce procházeny, bylo by to velice náročné na výpočetní kapacitu. Pro vyloučení velkého množství obrazů se při analýze podobrazů používá pouze několik funkcí, které definují objekt. Cílem je eliminovat přibližně 50 % dílčích obrazů, které neobsahují objekt. Tento proces pokračuje a zvyšuje počet funkcí použitých k analýze dílčího obrazu v každé fázi. Sestavení klasifikátorů do kaskády umožňuje procházet pouze podobrazy s nejvyšší pravděpodobností, že budou analyzovány všechny Haarovy rysy rozlišující objekt. Umožňuje také měnit přesnost klasifikátoru. Je možné zvýšit rychlost falešných detekcí i míru pozitivního určení snížením počtu stupňů v kaskádě.

Pro detekování obličeje je tedy nutné vytrénovat kaskády Haarových klasifikátorů pro segmenty obličeje jako jsou oči nos či ústa. Pro správné vytrénování klasifikátoru je zvolen algoritmus AdaBoost.

Pro Haar metodu jsou v knihovně OpenCV dostupné čtyři metody: Skutečný adaboost, diskrétní adaboost, logitboost a jemný adaboost. Všechno jsou to varianty algoritmu Adabost. Adaboost je zkrácený název pro algoritmus adaptive boosting. Tento algoritmus formuloval Yoav Freund a Robert Shapire [17]. Adaboost

odkazuje na specifickou učební metodu pro boosted klasifikátor. Boosted klasifikátor je ve formě:

$$F_T(x) = \sum_{t=1}^{T} f_t(x) \quad (3)$$

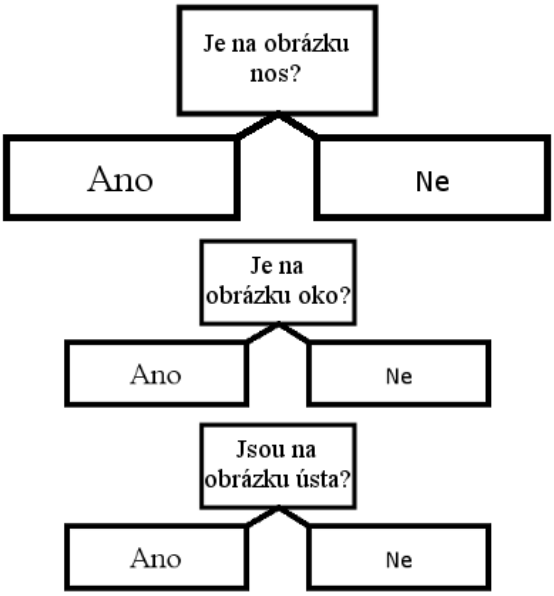
Tato metoda využívá výsledku prohledávání obrazu haarovými klasifikátory. Na jejich základě vytvoří seznam jednoduchých stromů. Tyto stromy mají na finální rozhodnutí rozdílný dopad. Jejich výsledky mají rozdílnou váhu. Pořadí stromu v klasifikátoru je velice důležité, proto je jako první použit nejsilnější strom, který byl vytvořen. Na jeho základě pak vždy výsledky stromu jsou ovlivněny výsledky předchozího stromu. Jako ukázka může sloužit Obrázek 5 .

K výcviku klasifikátorů je zapotřebí dvou sad obrazů. Jedna sada obsahuje obrázky, které hledaný objekt, nebo sérii objektů, neobsahují, další sada pak obsahuje hledaný rys, což je v tomto případě lidský obličej. Tato sada obrázků se označuje jako negativní obrázky. Druhá sada obsahuje takzvané pozitivní obrázky, které obsahují jen jeden nebo vícero výskytů hledaného objektu. Místo výskytu objektu je v každém obrázku učeno pomocí pozice pixelů a výšky a šířky objektu. Pro získání co nejuniverzálnější detekce obličejových rysů je nutné, aby sada trénovacích obrazu byla co nejvíce reprezentativní. Měla by proto obsahovat databázi obrazů osob různého pohlaví, věku, rasy atd.

Pro uspíšení analyzování obličeje je potřeba vytvořit klasifikátory, které vyhledávají jednotlivé části obličeje. Tyto tři klasifikátory určují nos, ústa, a oko.

To, že Haar klasifikace používá sérii odmítnutí znamená, že konečný klasifikátor se sestává ze série mnoha jednoduchých klasifikátorů a musí projít všechny zájmové oblasti a všemi fázemi této kaskády. Pořadí uzlu je často uspořádáno podle složitosti, takže mnoho funkčních kandidátů je vyloučeno včas, což ve vetší míře urychlí výpočetní proces.

Když je obraz analyzován, je procházen podokny různých velikostí. V každém podokně jsou vypočítány součty hodnot pixelů. Na základě jejich významnosti jsou potom použity integrální obrazy a aplikuje se rozhodující kaskáda.



Obrázek 5: Ukázkové schéma možného adaboost klasifikátoru pro detekci obličeje (vlastní tvorba)

Nakonec tedy pomocí detekování jednotlivých elementů jsme schopní na obraze nalézt vyhledávaný předmět, v našem případě obličej. Díky detekci obličeje, je pak možné zkusit zjistit o osobě další informace, jako je například věk, či pohlaví. Taktéž je detekce obličeje důležitá při navázání komunikace s uživatelem pomocí obrazovky. Například, když bude jedna z kamer umístěna nad obrazovkou a algoritmus s detekcí obličeje vyhodnotí, že osoba obrazovku sleduje, bude možné přilákat její pozornost nějakou vizuální interakcí.

4.2.1.2 Detekce obličeje, věku a pohlaví

Pro umělou inteligenci je taktéž důležité zjistit dílčí informace o osobě se kterou interaguje, jako například věk či pohlaví. Díky tomu dokáže lépe simulovat lidskou komunikaci. Díky těmto informacím, může daleko více přizpůsobit komunikaci osobě, se kterou komunikuje. Pomocí přibližného odhadu je pak možné pro umělou inteligenci získat informace, které zlepší její fungování. Může tedy volit jiné způsoby komunikace, či případné oslovení osoby s níž komunikuje.

Pro tyto schopnosti není nutné chodit daleko. Metodami, které disponují takovýmito schopnosti již disponují dříve zmíněné knihovny OpenCV. Tato knihovna nejen že obsahuje metody pro práci s obrazem, ale také metody pro deep learning network čili strojové učení a hluboké strojové učení.

Deep learning je jedním z typů strojového učení. Tato metoda využívá více vrstev, díky kterým postupně získává prvky vyšší úrovně ze samotného původního vstupu. Deep learning je využíván například při zpracování obrazu, kdy z původního obrazu mohou spodní vrstvy této sítě například identifikovat hrany v obraze a podobně.

OpenCV používá pro deep learning network konvoluční neuronové sítě [18]. Tato síť je velmi často využívána při analýze vizuálních obrazů. Tato konvoluční neuronová sítě se skládá ze vstupní vrstvy a výstupní vrstvy, a mezi nimi je několik skrytých vrstev. Skryté vrstvy konvolučních neuronových sítí se obvykle sestávají z řady konvolučních vrstev, které konvolují multiplikačním, nebo jiným skalárním součinem.

Konvoluční neuronové sítě nevnímají obrazy jako lidé. Naopak vnímají objekty zobrazené na obraze jako prostorové. Nejedná se o to, že by neuronová síť dokázala rozpoznat, že objekt je trojrozměrný, ale nevnímá data v obraze takovým způsobem jako lidské oko. Lidské oko vidí obraz jako dvourozměrnou plochu plnou barevných bodů, zatímco neuronová síť rozlišuje a zpracovává vícerozměrnou datovou hloubku. Vidí totiž tři vrstvy, ze kterých se výsledný obraz skládá a to červenou, zelenou a modrou. Tyto vrstvy v sobě mohou skrývat informace, které nejsou ve výsledné barvě zjevné, v jednotlivých vrstvách jsou ovšem viditelné. Pro síť jsou to informace o různých podrobnostech zachycených v obraze.

Konvoluční neuronová síť tedy dostává jako vstup obdélníkový úsek obrazu, kdy velikost tohoto obrazu muže být libovolně veliká, avšak jeho hloubka bude vždy tři vrstvy, kvůli výše zmíněným třem obrazovým kanálům. Vrstvu od vrstvy se poté rozměry obrazu mění.

Neuronové sítě nezpracovávají obraz po jednotlivých pixelech, ale celkovým obrazovým vstupem. Tento obrazový vstup projde filtrem, který je stejný jako obraz maticového tvaru, avšak jeho velikost je menší nebo stejně veliká jako zkoumaný úsek obrazu. Tento maticový filtr se nazývá jádro a jeho úkolem je najít různé vzory v obraze.

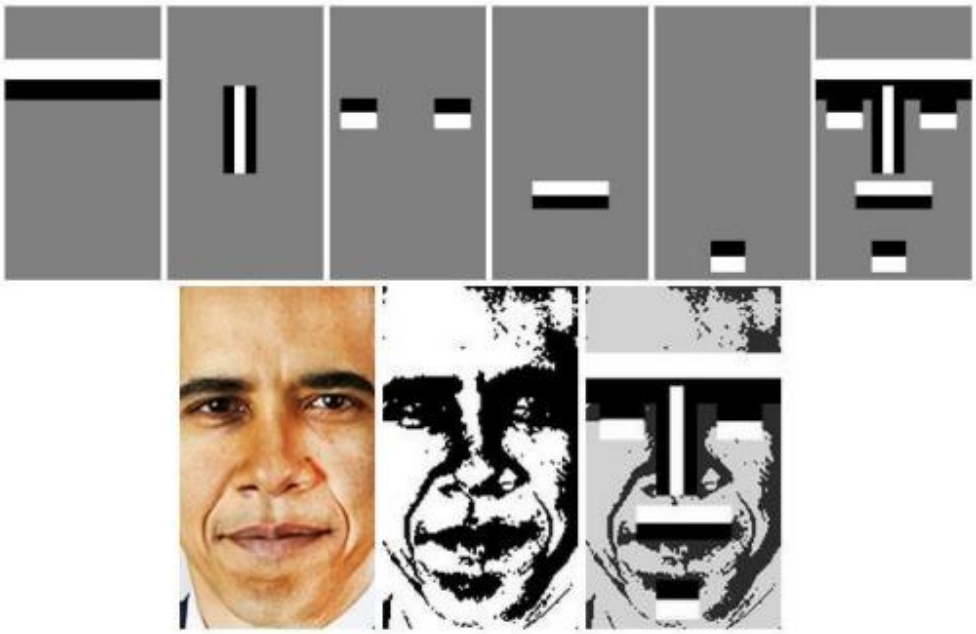
Při přejíždění obrazovým filtrem po obraze, je možné zvolit různou velikost kroku posunu. Při každém kroku je použit skalární součin původního obrazu a filtru. Výsledek je vložen do třetí úrovně neuronové sítě zvané aktivační mapa. Velikost třetí vrstvy neuronové sítě je počet kroků při skenu obrazu na výšku a na šířku. Velikost aktivační mapy je závislá na velikosti kroku, proto má velikost kroku velkou důležitost. Velikosti vrstev v konvoluční neuronové síti tedy určuje i jak dlouho bude síti trvat její učení. Větší kroky znamenají kratší čas na výpočet a trénování.

Zkoumaná část obrazu je pomocí skalárního součinu s filtrem analyzována. Pokud má část obrazu v sobě vysoké hodnoty a vysoké hodnoty jsou i na stejné pozici i ve filtru, má výsledný skalární součin taktéž vysokou hodnotu. Tato vysoká hodnota se pak propíše do aktivační mapy.

Následující vrstvou v konvoluční síti je vrstva, které má více jmen, jako například převzorkovací vrstva, nebo maximální sdružení. Tato vrstva následuje aktivační mapy, které do ní vedou. Tato vrstva vybere nejvyšší hodnoty z aktivační mapy a následně z nich pomocí jejího umístění vytvoří novou matici. Ostatní data jsou pro ni nedůležitá, a tudíž s nimi není nutné počítat.

Pro každou informaci, kterou chceme z tváře člověka získat, je proto potřeba vytvořit samostatnou neuronovou síť, která zpracuje obraz tváře a vyhodnotí důležité informace. Pro odhadnutí věku člověka na obraze bude potřeba jedna síť,

pro vyhodnocení jeho pohlaví bude potřeba další a podobně.



Obrázek 6: Haarovy rysy v obličeji (převzato z [19])

Přidáváním těchto sítí je pak možné postupně analyzovat více informací, které nám obraz člověka dodává. Trénování jednotlivých sítí probíhá vždy separovaně a doplňuje získaný informační rámec analyzované osoby.

4.2.2 Zpracování zvuku

Umělá inteligence může využívat více informací než jen informace zjištěné z obrazu. Velké procento mezilidské komunikace probíhá pomocí lidské řeči. Je proto pro člověka zcela intuitivní komunikovat pomocí slov. Ústní komunikace s umělou inteligencí je tak pro člověka přirozenější a příjemnější.

Umělá inteligence by tedy měla umět zaznamenávat a analyzovat lidské mluvené slovo. Tato analýza zpracuje lidské slovo na pro stroj použitelnou formu, se kterou bude moci pracovat. Následné zpracování zvuku může probíhat v různých formách. Jmenovitě může zpracovávat přímo binární záznam zvukové vlny anebo může převést tuto zvukovou vlnu na jednotlivá slova a věty. V této práci byla z důvodů kontroly umělé inteligence a její komunikace zvolena forma převedení hlasového záznamu do textové podoby mluveného slova. Jelikož umělá inteligence bude komunikovat s veřejností, nebylo by vhodné, aby komunikace byla ponechána

pouze na stroji. Mohlo by pak dojít k naučení komunikace ve zcela nevhodné formě. Příkladem takového scénáře může být umělá inteligence od firmy Microsoft, která po učení začala projevovat rasistické názory[20].

Převedení mluveného slova do textové podoby má vcelku jednoduchou systémovou strukturu. V základní verzi se takový systém může skládat ze zařízení zaznamenávajícího zvuk jež přenáší do neuronové sítě a následně pouhý terminál, který vypíše textové zpracování slov. Schéma zpracování zvuku na text není složité, ale existují v něm problémy, se kterými je nutno počítat. Prvním velkým problémem je způsob, jakým člověk mluví. Styl mluvy může být u každého člověka velmi odlišný. Například délka vyslovovaných slov může být velmi rozdílná a působit systému problémy. Někdo vysloví slovo krátce, někdo protahuje nějaké hlásky. Takovéto rozdíly v mluvě by ale měly mít v konečném výsledku stejný záznam v textu. Automatické zarovnání délky zvukového úryvku a slova je složité, protože slovo může mít různou délku výslednosti a konec tohoto slova nemusí být zřetelný.

Jako první probíhá zpracování zvukových vln, které jsou v počítačovém prostředí reprezentovány hodnotou, kterou má zaznamenána zvuková vlna v daném časovém okamžiku. Takovéto vzorkování zvuku muže mít v rámci kvality různé velikosti. Tyto frekvence jsou udávány v hertzech, což znamená počet zaznamenaných hodnot během sekundy. Pro lidskou řeč je dostatečné vzorkování kolem 20kHz [21].

Jelikož není možné říct, jak dlouhé slovo bude zpracováváno, musí se zvolit jiná možnost místo zpracování celého slova. Zvuková nahrávka musí být rozdělena do zvolených segmentů o přesné velikosti. Tyto samostatné segmenty zvuku musí být zpracovávány postupně.

Tyto zvukové segmenty jsou ale směs různých zvukových frekvencí, které jsou zařízením zaznamenány. Kompletní směs těchto frekvencí dohromady dává zvuk lidského hlasu. Pro usnadnění zpracování těchto dat neuronovou sítí, je potřeba zvuk rozložit do jednotlivých částí. Pro každé z těchto frekvenčních pásem je spočítána hodnota energie, které toto pásmo obsahuje. Pro zkoumaný zvukový záznam je díky této hodnotě energie vytvořen takzvaný otisk, který slouží jako identifikátor tohoto záznamu. Pomocí otisku je možné vytvořit kombinaci frekvencí, která odpovídá zvuku jednoho písmene. K tomuto procesu se využívá Fourierova transformace, která rozloží složenou zvukovou vlnu na jednotlivé zvukové vlny.

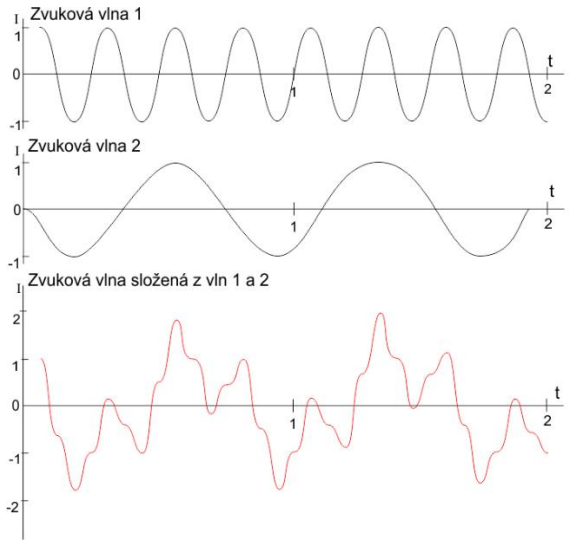
$$\hat{g}(f) = \int_{\infty}^{\infty} g(t) e^{-2\pi i f t}\, dt \tag{4}$$

V této rovnici je $g(t)$ rovnice funkce odpovídající zkoumané zvukové vlně v čase t. Tato funkce je násobena e s exponentem $-2\pi i f t$. Tato část odpovídá převedení lineárního grafu zvukové vlny do kruhového uspořádání. Čas je zde značen t, i je imaginární část komplexního čísla a f je hledaná frekvence.

Pro aktuální zvukovou vlnu, která je zachycena pomocí sérií bodů v čase je však složité nejprve hledat její předpis a poté aplikovat Fourierovu transformaci. Proto je důležité upravit rovnici tak, aby rovnou vypočetla, kde se na ose x nachází těžiště funkce $g(t)$.

$$\hat{g}(f) = \frac{1}{N} \sum_{k=1}^{N} g(t_k) e^{-2\pi i f t_k} \tag{5}$$

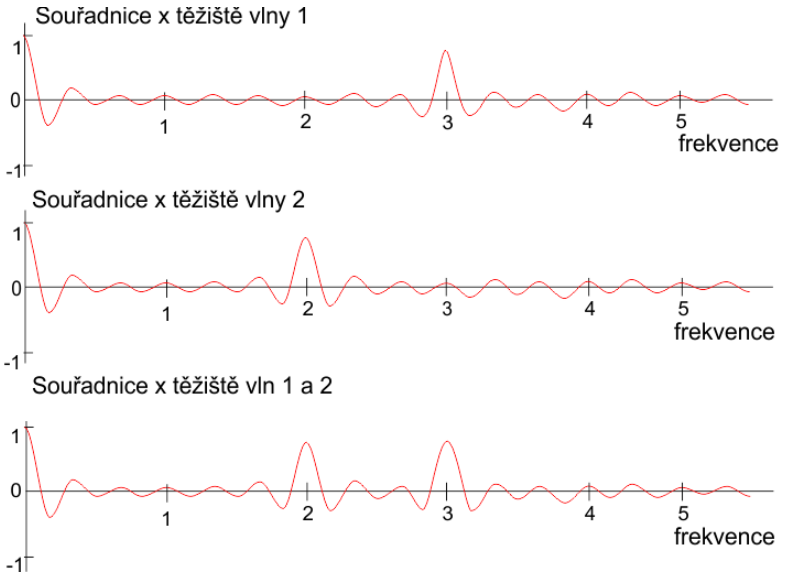
Výsledkem je struktura zvukového signálu v každém časovém úseku skládající se z jednotlivých zvukových vln v tomto okamžiku zvaný zvukový otisk[22].



Obrázek 7: Složení zvukové vlny z jiných vln (vlastní tvorba)

Tento zvukový otisk v kmitočtovém rozsahu je následně zpracován neuronovou sítí. Pro zpracování je použita rekurentní neurální sít. Tato neuronová síť má v sobě

paměť, která ovlivňuje její budoucí předpovědi. Je použita kvůli tomu, že ovlivňuje pravděpodobnost správného nalezení dalšího písmene. Paměť pomáhá neuronové síti v přesnějším určení písmene pro další předpovědi.



Obrázek 8: Výkyv x souřadnice indikující zvukovou vlnu ve složené vlně (vlastní tvorba)

Celý zvukový segment je takto zpracován neuronovou sítí a je převeden na písmena, která odpovídají zvukovým segmentům. Použitím neuronové sítě vznikne tedy slovo, ve kterém budou s velkou pravděpodobností zdvojená písmena. Jako první korektura textu je sdružení více stejných písmen do jednoho. Následně jsou odebrány všechny značky, které byly určeny pro označení mezery mezi slovy. Je několik možných výsledků, které jsou si zvukově při výslovnosti podobné. Jako poslední korekce výsledného slova je použito porovnání se slovníkem, který obsahuje všechny slova včetně jejich pravděpodobností výskytu. Tento slovník je vytvořen analýzou různých textů v daném jazyce. Nakonec je vybráno slovo, které má největší pravděpodobnost výskytu. Tato poslední korekce ale občas může způsobovat chybu, kdy dojde k zaměnění méně používaného slova za slovo s častějším výskytem v jazyce.

Překlad mluveného slova do textové podoby je důležitý pro následnou další komunikaci umělé inteligence s uživatelem. Nyní je možné zpracovat jejich příkazy a informace, které předává.

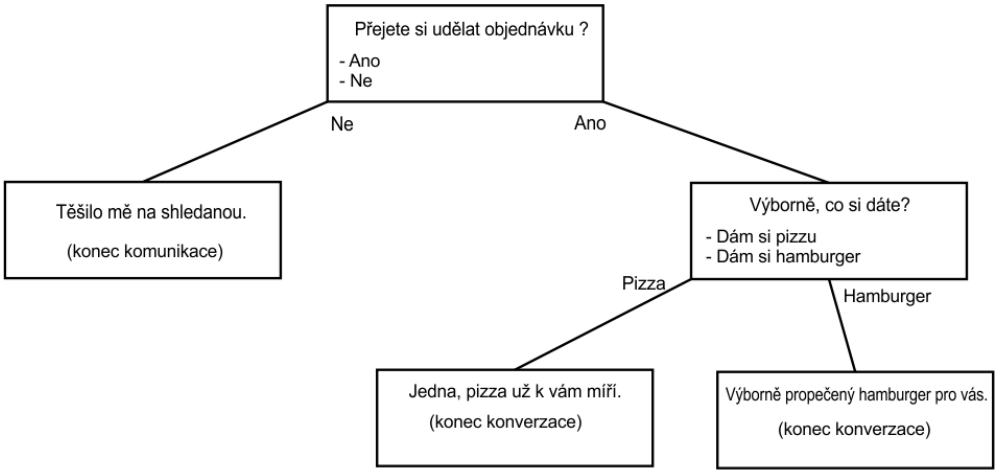
4.2.3 Zpracování komunikace

V případě umělé inteligence je velmi důležité nejenom informace získávat, ale zároveň je i zpracovávat v komunikaci s uživatelem. Uživatel očekává, že umělá inteligence bude komunikovat lidem podobným způsobem. Pro samotnou komunikaci není jen důležité dávat odpovědi na otázky, nebo reagovat na podměty od uživatele. V případě systému, kdy je převedena zvuková informace na textovou podobu, je potřeba na ni adekvátně zareagovat. V mezilidské komunikaci je důležité nejen reakce ale i porozumění jazyku také vyžaduje odvození skrytého významu, konkrétně úmysl mluvčího. [23]Mezilidská interakce má většinou dvě různé pozice.

Prvotní impuls k interakci muže být vyvolán jak uživatelem, tak umělou inteligencí jako takovou. Jako příklad může být brána interakce od uživatele v podobě aktivování pomocí doteku a odeslání prvních informací bez zjevného kontextu. Taktéž je možné, aby první krok v interakci vyvolala umělá inteligence, například identifikuje-li umělá inteligence, že je možnost oslovit člověka. Tato situace může nastat, když systém vyhodnotí, že člověk soustředí svůj pohled na obrazovku, kterou systém ovládá. Následně pak umělá inteligence vyšle informaci o pozdravu a interaguje s uživatelem.

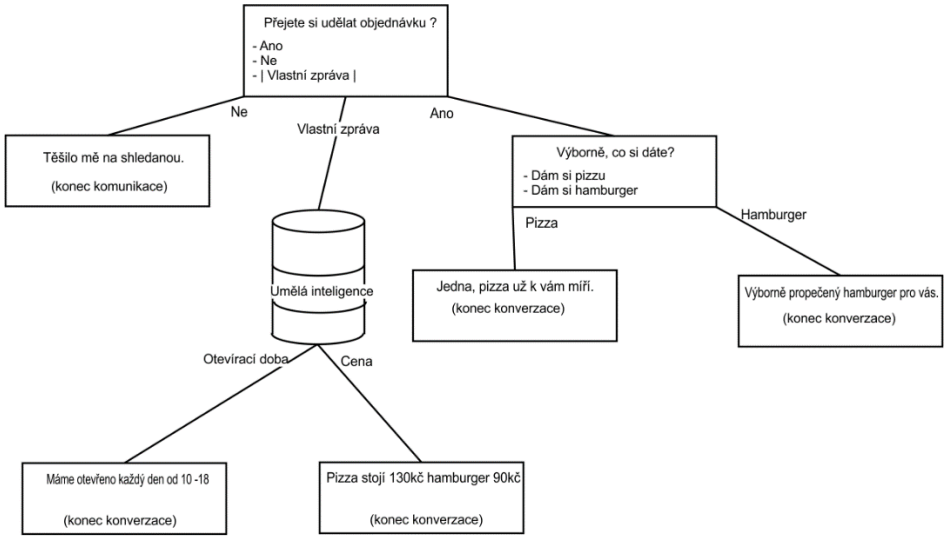
Pokud člověk s umělou inteligencí komunikuje v textové formě, je vhodné ji zpracovávat právě v této podobě. Pro tuto formu je již delší dobu funkční řešení, které je používáno ve vícero různých systémech, například na zákaznických linkách a podobně[24]. Jedná se o takzvané QA systémy, které odpovídají na uživatelské dotazy. Tyto systémy jsou typy chatbotů, které na základě textových informací, jež jsou jim předány tak vybírají správnou odpověď, která je maximálně relevantní pro vstupní informaci.

Takovéto QA systémy mohou mít různé podoby. Aktuálně jsou k vidění tři základní typy takovýchto systémů. První z nich je typ bez umělé inteligence. Tento typ provází člověka pomocí pevně zvolených otázek a odpovědí. Celý systém má podobu velkého rozhodovacího stromu, kde jednotlivé listy stromu jsou vždy odpovědi v dané větvi. Tento systém je jednoduchý na implementaci, avšak jeho působení je omezeno pouze na to, co mu rozhodovací strom dovolí.



Obrázek 9: Diagram chatbota s rozhodovacím stromem (vlastní tvorba)

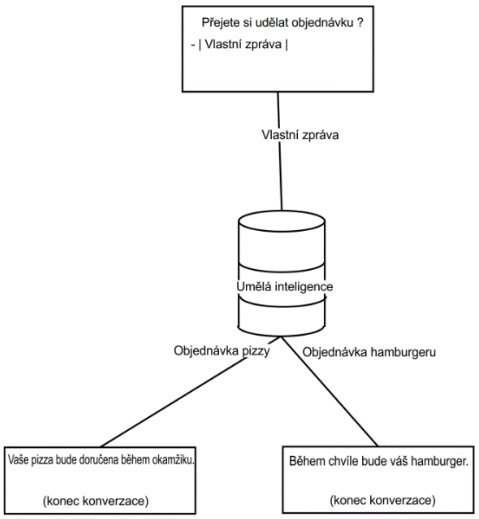
Druhým typem je hybridní QA systém. Tento systém využívá kombinaci QA systému bez umělé inteligence a té s umělou inteligencí. Tento systém umožnuje použít jak předpřipravené možnosti odpovědi na položený dotaz, tak odpověď vlastní. Tato vlastní odpověď je pak zpracována umělou inteligencí.



Obrázek 10: Diagram chatbota s rozhodovacím stromem a umělou inteligencí (vlastní tvorba)

Třetím typem QA systému je systém kompletně založený na umělé inteligenci. Tento systém očekává ručně psanou a formulovanou otázku, případně informaci,

kterou mu uživatel předloží. Dále pak pomocí neuronové sítě vyhodnotí požadavek a nabídne možné řešení anebo odpověď.



Obrázek 11: Typ chatbota s umělou inteligencí (vlastní tvorba)

Tento třetí typ funkčního řešení se pro tuto práci hodí nejvíce, neboť dovoluje pouze minimální korekce do struktury systému. Taktéž dovoluje vetší dynamiku komunikace, protože není nutně závislá na umístění ve stromě, a tudíž se může měnit s vetší volností, a přitom nemusí ztrácet na přesnosti a relevantnosti.

V systému je tento typ chatbota implementován jako neuronová síť, která klasifikuje jednotlivé textové podměty od uživatele.

5 Použité technologie

Cílem diplomové je nejen navrhnutí systému umělé inteligence, ale zároveň i jeho implementace a otestování pro komunikaci s člověkem. V následující kapitole jsou obsaženy základní informace o použitých technologiích.

5.1 Python

Python je interpretovaný vysokoúrovňový programovací jazyk. Python vytvořil Guido van Rossum v roce 1991. Design programovacího jazyka dbá velký důraz na čitelnost kódu díky specifickému využití mezer. Kód v programovacím jazyce Python je dynamicky zadáván a zároveň již nevyužívaná paměť není držena, ale je

uvolňována. Jazyk podporuje procedurální programování, objektově orientované programování a funkcionální programování. Taktéž je kladen důraz na rozšiřitelnost jazyka pomocí různých modulů a knihoven.[25]

Jelikož Python patří mezi interpretované jazyky, je pro jeho funkčnost potřeba interpreta. Aktuálně jsou dostupné tři interprety postavené na různých programovacích jazycích. Jedná se o CPython, Jython a IronPython. Nejrozšířenější je verze psaná v jazyce C a to CPython. Oproti tomu je interpret Jython interpretován na virtuálním strojem běžícím na jazyce Java neboli JVM. Jako poslední je IronPython který je implementován nad jazykem .NET. [26]

Jazyk Python je podle žebříku TIOBE mezi vývojáři třetí nejoblíbenější programovací jazyk[27]. Taktéž je tento jazyk využíván velkým počtem společností, například Google, Facebook a další. Je možné jej najít ve velkém množství softwaru. včetně v grafických programů jako je např. Blender nebo 3ds Max.

Python nachází také zejména uplatnění ve vědeckých kruzích a převážně při práci s daty, díky knihovnám Numpy, či SciPy, které přímo pro tyto oblasti obsahují speciální funkce. Obsahují například funkce pro práci s numerickou matematikou, statistikou a podobně. Python je také využíván pro práci s umělou inteligencí a neuronovými sítěmi. Tuto funkci zpřístupňují knihovny jako například TensorFlow nebo Keras.

Apr 2020	Apr 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.73%	+1.69%
2	2		C	16.72%	+2.64%
3	4	▲	Python	9.31%	+1.15%
4	3	▼	C++	6.78%	-2.06%
5	6	▲	C#	4.74%	+1.23%
6	5	▼	Visual Basic	4.72%	-1.07%
7	7		JavaScript	2.38%	-0.12%
8	9	▲	PHP	2.37%	+0.13%
9	8	▼	SQL	2.17%	-0.10%

Obrázek 12: Nejoblíbenější programovací jazyky podle TIOBE [27]

5.2 Webové technologie

Při tvorbě celého systému, byly pro tvorbu tenkého klienta použity následující webové technologie.

5.2.1 HTML (Hypertext Markup Language)

HTML je zkratka pro Hypertext Markup Language. Jedná se o standardizovaný značkovací jazyk využívaný pro vykreslování a strukturalizaci webových dokumentů a stránek. Společně s kaskádovými styly (CSS) a skriptovacími jazyky, jako je JavaScript, patří mezi základní technologie pro tvorbu webiových stránek a aplikací.

První verzi HTML vyvinul Tim Bernes-Less v roce 1990. Předchůdcem HTML byl jazyk SGML (Standard Generalized Markup Language). Aktuálně je HTML standard spravovaný konsorciem W3C[28].

5.2.2 CSS (kaskádové styly)

CSS je zkratka slov Cascading Style Sheets. Jedná se o technologii používanou k zapisování stylů webových stránek v jazyce HTML.

Registrování CSS pro použití proběhlo v roce 1998. Kaskádové styly vytvořil a popsal Håkon Wium Lie[29] . Aktuálně je technologie udržována konsorciem W3C stejně jako technologie HTML.

5.2.3 JavaScript

JavaScript je název pro programovací jazyk, který je jednou z klíčových technologií webových stránek. JavaScript totiž přináší do statického obsahu webových stránek interaktivní prvky, například animace, nebo zpracování uživatelských akcí jako kliknutí na element či ukázání kurzorem myši. JavaScript je ve velké míře používán na straně klienta, avšak v poslední době se JavaScript začíná objevovat i na straně serveru. Pro funkčnost JavaScriptu musí být ve webovém prohlížeči implementován JavaScriptový stroj pro jeho překlad.

První použití JavaScriptu bylo společností Netscape v roce 1995 v prohlížeči Netscape Naviator. V roce 1997 byla společností ECMA [30] stanovena platná oficiální norma jazykem, aby byla zajištěna kompatibilita mezi zařízeními.

5.2.4 Technologie websocketů

Technologie websocketů je typ komunikačního protokolu v počítačové komunikaci. Tento protokol poskytuje plně obousměrný komunikační kanál, který je vytvořen skrze jediné TCP připojení. Toto propojení vytvoří kanál mezi webovým prohlížečem a webovým serverem. Websocketový protokol je tedy vytvořený na základě protokolu TCP. Oba tyto protokoly se nacházejí v 7. úrovni OSI modelu. Protokol byl standardizován skupinou IETF v roce 2011[31]. IETF je organizace která se zabývá vývojem a podporou internetového standardu. Tato firma spolupracuje taktéž s konsorciem w3c a organizacemi ISO/IETC.

Websocketový protokol ve výchozím stavu je designován, aby komunikoval pomocí stejných portů jako klasické webové služby, tedy na klasickém portu 80 a šifrovaném 443. Využití těchto portů pak umožnuje komunikaci i v sítích, kde jsou jiné porty blokovány pomocí firewallu. Aby bylo dosaženo kompatibility pro navázání spojení, je jako první navázán handshakea použita HTTP hlavička pro upgrade. Pomocí tohoto handshaku dojde ke změnění hlavičky z HTTP protokolu na protokol websocketový.

Hlavní výhodou websocketového protokolu je výše zmíněná obousměrná komunikace. Websocketová komunikace má nižší režijní náklady než další typy komunikace, jako například http dotazy. Usnadněním je přenos dat na server v reálném čase nebo ze serveru na klienta. Další výhodou je odesílání dat ze serveru na klienta, aniž by bylo nutné nejdříve o data požádat pomocí http dotazů.

Aktuálně je podpora standardizovaného websocketového protokolu na všech nejpoužívanějších webových prohlížečích. Díky tomu je možné technologii používat na velkém počtu zařízení a zajišťuje tak velmi rychlý a dostupný kanál pro komunikaci.

Při vytvoření umělé inteligence jejíž tenký klient může mít podobu webové stránky je díky výše zmíněnému výhodné použít tuto technologii websocketů jako spojení ze serveru na uživatelem přístupnou stranu. Nevýhodou pro tuto technologii je další běžící websocketový server na serverové straně, který bude celou komunikaci zajišťovat. Při návrhu systému a vnitřních informačních struktur je potřeba tedy s tímto samostatným serverem počítat.

5.3 OpenCV

OpenCV je název knihovny pro vývoj aplikací s počítačovým vidění. OpenCV bylo vytvořeno, za účelem poskytnutí infrastruktury pro různé aplikace, které pro svůj účel potřebují zpracovávat počítačové vidění. OpenCV obsahuje implementace více než 2500 algoritmů. Algoritmy obsažené v knihovně jsou nejen z oblasti počítačového vidění ale také strojového učení (machine learning). Tyto algoritmy lze použít při tvorbě rozšířené reality, detekování objektu v obraze či rekonstrukce scény z obrazu.

Jádro knihovny OpenCV je naprogramováno v jazyce C/C++ a Knihovny díky tomu jsou použitelné na všech systém od systému Windows po Mac OS X. Součástí této knihovny je taktéž velký počet různých wrapeerů, které zpřístupňují metody knihovny pro různé další programovací jazyky například Python.

Knihovna OpenCV obsahuje velké množství statických metod rozdělených do jednotlivých tříd. Tyto třídy jsou pojmenovány na základě účelu metod, které jsou v nich obsaženy. Příkladem muže být třeba třída Imgproc, která obsahuje metody pro práci s obrazem. Funkce, které provádí jednotlivé úkony mají v OpenCV často implementováno víc metod pro provedení stejného úkonu. Programátor má na výběr jednotlivých metod tzv. (anglicky flag). Tato vlajka se nachází uvnitř příslušné třídy je metodě předána jako parametr. Dalším specifikem je i třeba návratový parametr metody, V něm je vrácen výsledek metody. který vrací její výsledek.

Knihovna OpenCV je volně dostupná ke stažení z oficiálních stránek [32]. Taktéž je na stránkách velké množství tutoriálů, které pomáhají pochopit, jak se s knihovnami pracuje a jaké je řešení základních problémů[33]. Součástí podpory je i fórum, ve kterém při řešení problémů pomohou zkušenější vývojáři a odborníci [34].

6 Praktická část implementace umělé inteligence

Samotné teoretické řešení dílčích problému ovšem nevytváří kompletní systém umělé inteligence. Dále popsaný systém byl vytvářen pro použití ve veřejném prostoru. Jako hlavní účel bylo ukázat, co je v technologiích možné. Hlavním principem systému bylo navrhnout modulární dynamický systém, který by mohl

fungovat bez přestání a mohl by být rozvíjen bez nutnosti odstavení systému. Taktéž byl vyvíjen s možností multiplatformního využití bez omezení. Zacílením na tento aspekt systém získá schopnost od uživatele získávat další informace, které by uživatel mohl využit.

Pro uživatele je velmi důležité mít systém sobě personifikovaný. Je pro něj důležité mít osobní vztah se systémem a mít systém přístupný například i v mobilním zařízení, nejenom na počítači. Uživatel pak má možnost interagovat se systémem kdykoliv je potřeba. Taktéž se mohou lišit i způsoby, kterými se systémem interaguje. Například hlasová komunikace může být maximálně užitečná v prostředí ve kterém se uživatel nesoustředí na okolí, například doma. Oproti tomu nemusí tento typ komunikace být přínosný v každé jiné situaci. Asi není dobré zadávat si nahlas úkony do poznámek nebo podobné věci pomocí hlasového zadávání např v tramvaji, zaměstnání či v obchodě. Často totiž při interakci zadáváme informace, které jsou pro nás intimní anebo nechceme rušit při jejím zadávání okolí.

Z důvodu použití systému je jako primární typ zvolena hlasová komunikace. Při písemné interakci by bylo velmi zdlouhavé zadávat informace vymačkáváním jednotlivých písmen na klávesnici.

6.1 Návrh systému

Systém se skládá ze dvou částí. Tento typ byl zvolen na základě požadavků, kterými je tenký klient limitován. Tenkým klientem je nazván miniaturní program, který má pouze schopnost přenosu a zobrazení dat. Samotné zpracování dat probíhá jinde. Pro tento účel byla zvolena webová stránka. Webové technologie dovolují nejenom zpracování dat odeslaných ze serveru, ale taktéž dovolují pomocí vestavených knihoven v prohlížečích nahrávat zvuk či provádět úkony, anebo udržovat websocketovou komunikaci. Také je velmi výhodné, že na webové stránky je možné přistupovat skoro z každého zařízení, které má display.

Pro dokonalejší personifikaci byl systému navržen humanoidní obličej, který by zpříjemnil uživateli interakci a umocnil pocit z komunikace s kompletní umělou inteligencí. Dodatečné informace jsou totiž lidmi ve velkém procentu interpretovány pomocí výrazu v obličeji, stejně jak uvádí výzkum o lidské komunikaci. [13]

Systém tenkého klienta obsahuje funkce pro reprezentování slov v textové podobě, která má podobu komiksové bubliny. Takto systém zobrazuje textové odpovědi a svým vzhledem zajišťuje imerzivnější interakci.

Umělá inteligence v serverové části obsahuje databázi výrazů obličeje, které je možné střídat a které reprezentují výběr různých emocí. Samostatná komunikace mezi tenkým klientem a serverovou částí pak obsahuje i informaci o tom, jakou emoci pak má systém tenkého klienta uživateli zobrazit.

Díky funkci tenkého klienta o zobrazení emoce je možné uživateli doručit i informace, které mohou mít jinou interpretaci na základě doprovodné emoce. V serverové části je nejen vytvořena odpověď na interakci s uživatelem, ale také je vybrána emoce, která odpovídá znění zprávy. Díky tomu je možné například uživateli doručit mrknutí, nebo například smích a podobné typicky lidské projevy. Interakce rozšířená o tyto emocionální výrazy je velmi podobná komunikaci mezi lidmi. Takovéto obličejové signály jsou v moderní komunikaci pomocí zpráv zastoupeny emotikony. Takovýchto emotikonů je veliké množství a mohou v různých kombinacích lidem zprostředkovat nonverbální komunikační informaci, kterou by jinak zpráva postrádala.

Samotné propojení tenkého klienta a serverové části je vytvořeno pomocí oboustranného kanálu. Klasická komunikace mezi webovou stránkou a klientem probíhá pravidelně pomocí http requestů, ve kterém se stránka dotazuje serveru pro informace, případně vyvolává na serverové straně akce. Takováto komunikace je vyvolána pouze z jedné strany. Oboustranný kanál je v tomto případě reprezentován pomocí technologií websocketů, kdy na webové stránce je otevřený websocket, a na serverové straně je spuštěn websocketový server.

Takové uspořádání má velké množství výhod. Spojení je permanentní, a tudíž je možné ze serveru vyvolat nové informace, aniž by bylo nutné celou stránku znovu načítat. Taktéž je možné zjistit jednodušeji konektivitu se serverem a je možné tuto informaci předat i uživateli.

Taktéž je jednodušší zjistit počet zařízení, na kterých je systém připojen. Každý tenký klient je připojen na serverovou část pomocí svého vlastního websocketu. Pokud je systém vybaven vícero displeji, na kterých dochází ke komunikaci, je pak díky vytvoření unikátního identifikátoru pro prohlížeč možné tyto displeje uložit.

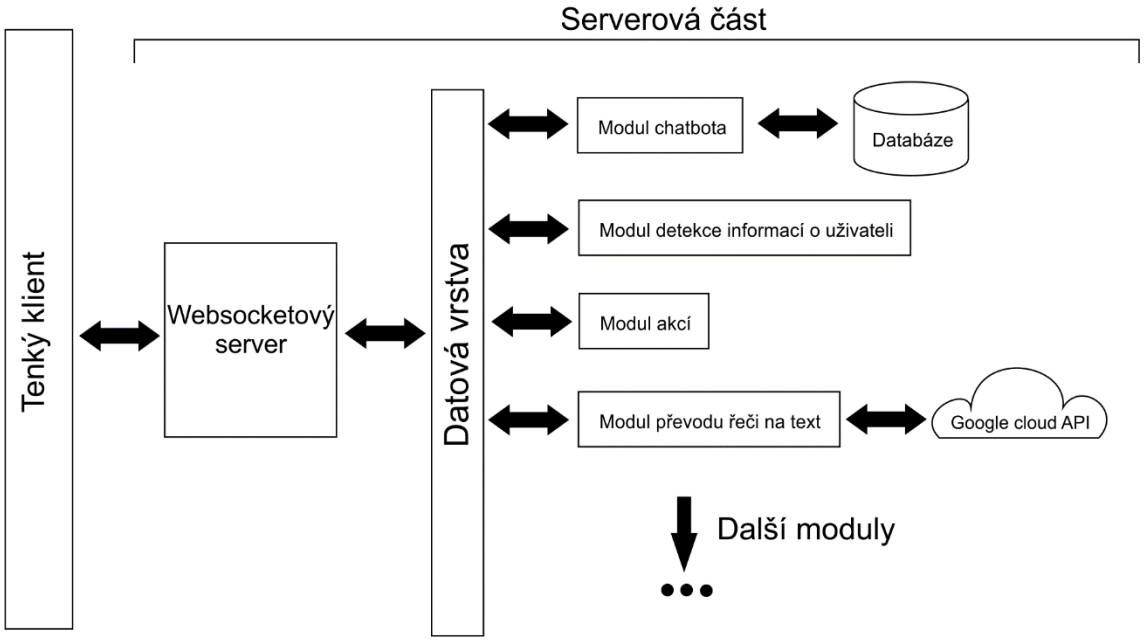
Posléze je tak možné, aby systém přistupoval k těmto obrazovkám jednotlivě. Pokud se tedy tyto obrazovky i například pojmenují místem ve kterém se nacházejí, je potom možné, že systém bude přecházet na jednotlivé obrazovky podle toho, kde se bude uživatel momentálně vyskytovat.

Takováto informace je pro systém důležitá a uživatel třeba může dát systému příkaz, který může být omezen lokálně. Jako příklad takovéto interakce může být příkaz systému o otevření specifických dokumentů v pracovním počítači anebo puštění hudby a nastavení osvětlení v obývacím pokoji při vstupu do domu.

Komunikace probíhá pomocí formátu JSON (neboli JavaScript Object Notation). Z tenkého klienta přichází zvuková data případně textová data, které má server zpracovat. Každý typ dat je označen specifickým způsobem, podle toho, jestli se jedná o textovou reprezentaci, nebo obrazová data, či případně zvuková data.

Na webové stránce je v technologii JavaScriptu vytvořen systém, který zpracovává data a ty následně posílá na server výše zmíněným websocketovým tunelem. Tato strana má za úkol sběr informací, které mají pro systém velkou důležitost. Díky propojené webkameře a mikrofonu jsou stránky schopné získávat zvuková i obrazová data.

Kamery zaznamenávají kontinuální obraz jako sekvence snímků. Tyto snímky jsou sekvence barev odpovídající na jednotlivé barevné signály v obrazu. Jedná se o velké množství informací a snímků, které není nutné neustále zpracovávat na serverové straně.



Obrázek 13: Diagram struktury systému (vlastní tvorba)

6.2 Klientská část

6.2.1 Veřejně přístupné části

Samotný systém s umělou inteligencí obsahuje velmi důležitou část, která v nemalé míře přispívá k celému kompletnímu řešení. Jedná se o část, která je přístupná lidem. V této části je komunikační vrstva celého systému. Jedná se o webovou stránku. Při načtení této stránky jsou jako první inicializovány grafické prvky. Jako další jsou pak inicializovány scripty, které zajištují komunikaci. Tyto části jsou implementovány v multiplatformním jazyce JavaScript.

Jako první je otevřen websocket, který zajištuje komunikaci se serverem. Na existenci tohoto spojení a tento websocket jsou navázané další funkce. Samostatná implementace websocketu obsahuje možnost pro implementaci různých událostí, jako například ztráta spojení, navázání spojení a podobné akce. Tato část je důležitá při kontrole stavu websocketu a při reakci systému na výpadek spojení.

Kontrola stavu websocketu je velmi důležitá pro celkový výsledný efekt systému. Je totiž nutné zajistit maximální konektivitu a zároveň i informovat o případné ztrátě

konektivity, aby nedocházelo ke zmatení uživatele. Následující úryvek kódu ukazuje, jakým způsobem jsou tyto události ve websocketu implementovány.

```
socket.onopen = function (e) {
    chckKnownBrowser(socket);
    setFace("smyling");
    clearInterval(connectInterval);
    clearInterval(blink)
    blink = setInterval(faceBlink, 2000);
};

socket.onmessage = function (event) {
    var data = JSON.parse(event.data);
    if (data.hasOwnProperty("emotion")) {
        setFace(data.emotion);
    }
    if (data.hasOwnProperty("text")) {
        showText(data.text);
    }
};

socket.onclose = function (event) {
    if (event.wasClean) {
        connectInterval = setInterval(tryToConnect, 5000);
    } else {
        connectInterval = setInterval(tryToConnect, 5000);
    }

};

socket.onerror = function (error) {};
```

První metoda se nazývá *onConnect*. Tato metoda je zavolána vždy, když dojde k navázání kontaktu s websocketovým serverem. Je to první metoda jejíž provedení signalizuje, že jsou oba systémy propojeny. V těle této metody je možné inicializovat celý systém na stránce do výchozího stavu. V ukázce kódu je vidět, že je nastaven výchozí výraz tváře, a je ukončen automatický pokus o připojení. Zároveň je znovu spuštěno mrkání tváře.

Druhá metoda *onMessage* je volána vždy, když ze serveru přijdou informace. Tyto informace přijdou do metody v textové podobě. Jelikož chodí data v textové podobě, není problém dát jim strukturu. Výhodné je v tomto případě široce používaný a podporovaný formát JSON. Díky tomu je možné použití metod převádějící formát JSON do textové podoby a zpátky. Tento formát je ve velkém používán ve webových aplikacích. Jeho výhodou je, že nemusí nutně obsahovat pouze text. Ve zprávě, které jsou přijímány klientem ze serveru jsou informace o tom, jaký výraz má být systémem nastaven, a případně jaká práva mají být zobrazena.

Další dvě metody *onClose* a *onError* indikují přerušení komunikace se serverem. Při zavolání těchto metod je nastaven časový interval pro opětovné pokusy o zpětné

připojení k webovému serveru. Taktéž je výhodné v těchto metodách dát uživateli informaci o tom, že je komunikace přerušena. Z uživatelského hlediska není dobré nechat systém s omezenou funkčností bez upozornění uživatele.

Na serveru je pak možné vidět počet připojených websocketů. Jeden připojený klient pak odpovídá jednomu připojenému zařízení. Při vygenerování unikátního identifikátoru, který dostatečným způsobem identifikuje zařízení, je pak možné na straně serveru tato zařízení ukládat a dále zpracovávat případně jimi modifikovat akce.

Webové cache jsou technologie pro dočasné ukládaní informací webových stránek. V této paměti mohou být uložené obrázky, data nebo scripty. Do této paměti jsou tyto informace ukládány, aby se omezilo počet stahovaných informací z webového serveru. Informace, které jsou tedy neměnné jsou takto uloženy v prohlížeči, a z této paměti jsou při načtení stránky znovu vyvolány. Ze serveru jsou pak jen stahovány data, která prošla změnou.

Při ukládání informací do cache a následnému přístupu k nim, je důležité mít zvolený klíč. Kontrolou existence dat pod tímto klíčem je možné určitě, jestli má být využitý uložený klíč, nebo jestli má být vygenerovaný nový. Tuto kontrolu je potřeba vyvolat přímo při načtení webové stránky hned po otevření komunikačního kanálu.

Odesláním informací s tímto klíčem je systému dána informace, které umístění je aktuálně připojeno. Na straně serveru je k otevřenému spojení přidán identifikátor zařízení. Tato informace je důležitá pro delegování rozdělení informací, na zařízeních, která jsou k dispozici anebo u kterých dochází ke komunikaci.

6.2.2 Snímání obrazu a zvuku

Aby samotná umělá inteligence mohla správně pracovat, je nutné vyžívat co možná nejkvalitnější data. Při sběru dat je využito vždy připojení klienta ke serverové části a jeho identifikace. Jelikož je klient tvořený webovou stránkou, je nutné využít technologie, které jej tvoří a použít tak ke sběru dat možnosti webových stránek a konkrétně JavaScriptu. Výhodou tohoto přístupu je, že není nutnost instalace dalšího softwaru do klienta, a není nutné otevírat další websocketové spojení. V případě vytvoření nového softwaru, který by zařizoval spojení se serverovou částí jak pro obraz, tak pro zvuk, by zbytečně docházelo

k nutnosti držet vztahy mezi jednotlivými spojeními a identifikovat ke kterému klientovi patří který datový tunel. Taktéž by bylo nutné odesílat a kontrolovat, do kterého tunelu by data chodila, což není nezbytné a spíš by to situaci komplikovalo.

Pro přístupu k datovému streamu z webkamery a mikrofonu mají webové technologie vytvořené JavaScriptové rozhraní. Použitím tohoto implementovaného rozhraní se významně ušetří vlastní psaný kód aplikace. V samotné aplikaci by musela být kontrola, jestli je připojený mikrofon, jestli jsou nainstalované ovladače a podobně. Použitím této metody prohlížeč v prvotní fázi automaticky ukáže uživateli dialog s dotazem, jestli povolí stránce přístup k mikrofonu a případně webkameře. Následující kód ukazuje, jak k těmto zdrojům přistoupit:

```
if (navigator.mediaDevices.getUserMedia) {
    navigator.mediaDevices.getUserMedia({audio: true }).then(gotStream)
        .catch(e => {
            alert(`getUserMedia() error: ${e.name}`);
        });
}
```

Toto využití má jednu podmínku, kterou je nutné splnit, jinak nebude přístup k zařízením povolen. Touto podmínkou je, že webová stránka musí být zabezpečena pomocí https protokolu.

Https protokol vydaný důvěryhodným certifikačním zdrojem zajišťuje, že prohlížeč opravdu komunikuje se serverem a nejedná se o stránku podvrhnutou případným útočníkem. Jedná se o bezpečnostní pravidlo, které podmiňuje použití těchto funkcí a zároveň brání proti odposlouchávání a sledování.

Ukázka kódu výše ukazuje, jakým způsobem otevřít sbírání dat pouze pro audio kanál. Knihovna mediaDevices má možnost přístupu i pro obrazový kanál. Při otevření přístupu k zařízení nejsou data hned k dispozici, ale je potřeba zvuk nebo video nejdříve nahrát a pak po částech odesílat.

V systému umělé inteligence byl kvůli úspoře zdrojů a nenavyšování datových toků zvolen systém, který pouze odešle uživatelem nahraná data. V praxi to pak funguje tak, že po dobu kliknutí na obrazovku je nahráván zvuk a po uvolnění jsou data odeslána. Následující úryvek zdrojového kódu zobrazuje, jakým způsobem jsou data na straně klienta nahrávána.

```
function gotStream(stream) {
    console.log('Received local stream');
    mediaRecorder = new MediaRecorder(stream);
    mediaRecorder.onstop = function(e) {
        var reader = new FileReader();
        reader.readAsDataURL(chunks[0]);
        reader.onloadend = function () {
                var base64data = reader.result;
                socket.send(JSON.stringify({type: "sound", data:
base64data}));
                chunks = [];
            }
        }

    mediaRecorder.ondataavailable = function(e) {
        console.log(e.data)
        chunks.push(e.data);
    }
}
```

Zvuk snímaný mikrofonem nemusí být po zaznamenání v jednom datovém objektu. Data samotná jsou reprezentována jako nepřetržitý tok, ale nemohou být jako tok zpracována. Zvuky jsou snímány po částech s pevnou délkou a poté spojovány v celek. Při ukončení zaznamenávání zvuku je provedena metoda *ondataavailable*, u které jsou parametrem zaznamenaná data. Zvuková data jsou pak objektem typu Blob, tedy bez přesné struktury, a obsahují zvuková data ve zvukovém kanálu, nebo obrazová data v obrazovém. Jelikož je možné pomocí těchto knihoven nahrávat i obraz, bude vždy v datech i kanál pro video, i když prázdný. Pro další zpracování je tedy nutné vyjmout a použít jen příslušný kanál.

Výsledná data, která jsou ze zařízení získána jsou zakódována do kódování base64 a odeslána websocketem na server.

6.3 Serverová část

Serverová část je vytvořena z několika různých částí. Hlavní část, která je jádrem celé serverové části, je vytvořena v jazyce python. Toto řešení bylo zvoleno z důvodu přenositelnosti kódu, který může být spuštěn jak na operačním systému Linux, tak i Windows.

Serverová část obsahuje jediný spouštěcí script, který inicializuje celý systém. Tato struktura byla zvolena z důvodu, že je jednodušší po startu systému spustit jednotlivý script, a nikoliv hlídat, aby došlo k spuštění všech potřebných částí.

Následující úryvek kódu rozbrazuje iniciační script celé serverové části.

```
import json
import os


class Brain:

    def __init__(self):
        self.settings = self.load_settings()
        self.modules = []
        self.running = True
        self.data = []

    def start(self):
        self.init_modules()
        self.start_modules()
        while self.running:
            self.run()

    def load_settings(self):
        setf = open("../settings.json", "r")
        data = setf.read()
        return json.loads(data)

    def init_modules(self):
        for filename in os.listdir(self.settings["modDir"]):
            if os.path.isdir(self.settings["modDir"] + filename) and
               filename != "__pycache__":
                module = __import__(self.settings["modMod"] + filename,
                                    fromlist=[''])
                class_ = getattr(module, "InitModule")
                instance = class_(self.settings, self.data)
                self.modules.append(instance)

    def start_modules(self):
        for m in self.modules:
            m.run()

    def cancel_modules(self):
        for m in self.modules:
            m.cancel()

    def reboot_module(self, module):
        pass

    def run(self):
        pass
        # self.handle_sound()
        # if s == "q":
        #   self.cancel_modules()
        #   self.running = False


brain = Brain()
brain.start()
```

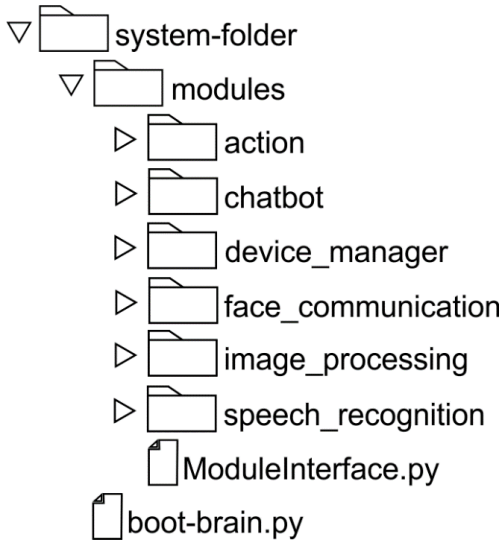
Výše je ukázka kompletního inicializačního scriptu. Jako první jsou inicializovány permanentní datové struktury. Tyto struktury jsou systémem držené objekty, které jsou dostupné pro zpracováním všem modulům. Tato data jsou trojího typu a vždy obsahují minimálně dvě informace – samotná data a také informaci o tom, jestli byla zpracována.

Jelikož se systém skládá z více modulů, které jsou na sobě nezávislé, je nutné, aby zpracování informací bylo asynchronní. Každý modul je spuštěn ve vlastním vlákně a může přistupovat ke všem datům. Vždy však zpracovává pouze tu část dat, která mu náleží, a ještě nebyla zpracována.

Taktéž jsou v prvotní inicializaci systémem načteny i konfigurační soubory, které v sobě hlavně obsahují informace o adresářové struktuře celého kompletního systému. Následně je potom celý systéme nastartován. Při startování systému jsou načteny všechny moduly, které systém obsahuje.

6.3.1 Struktura serverové části

Pro správnou funkci systému je důležitý návrh jeho celé struktury. Systém díky tomu může získat vlastnosti, které pomáhají při jeho vývoji a dalším rozvoji. Na základě požadavku na neomezený běh byl systém navržen jako modulární. Taktéž byly vybrány technologie, které podporují toto fungování. Následující obrázek zobrazuje datovou strukturu, kterou celý systém splňuje.



Obrázek 14: Struktura serverové části (vlastní tvorba)

Hlavním scriptem je zde zaváděcí script, který je jako jediný script spouštěn a vytvoří všechny další části systému. Systém taktéž obsahuje v hlavním umístění jedinou složku. Touto složkou je složka modules, v níž jsou umístěny všechny moduly, které systém obsahuje. V této složce je také soubor s rozhraním, které tyto moduly musí splňovat pro bezproblémové fungování. Samotné propojení systému pak probíhá právě díky tomu, že se všechny zdrojové kódy modulu nachází v této jediné složce.

Každý modul pak bude mít vnitřní strukturu libovolnou. Jediné, co tak musí obsahovat, je inicializační script s názvem *__init__.py*, který implementuje právě interface modulu.

```
def init_modules(self):
    for filename in os.listdir(self.settings["modDir"]):
        if os.path.isdir(self.settings["modDir"] + filename) and
           filename != "__pycache__":
            module = __import__(self.settings["modMod"] + filename,
                                fromlist=[''])
            class_ = getattr(module, "InitModule")
            instance = class_(self.settings, self.data)
            self.modules.append(instance)
```

Ukázka kódu nahoře zobrazuje, jaký způsobem jsou dynamicky načteny všechny moduly ze složek. Tento způsob není často využívaný, většinou dochází k importu modulu a scriptů přímo v hlavičce skriptu.

Nevýhodou tohoto systému je to, že zaváděcí script musí být vždy upraven, což při řešení v dynamickém načtení není třeba. Je proto možné pouze nahrát složku s modulem a následně celý systém nemusí být restartován, pouze jen bude systému nastaveno, aby zkontroloval moduly a nový modul zapojil automaticky. Není tak nutné, aby byl systém mimo provoz.

6.3.2 Vnitřní datová struktura

Vnitřní datová struktura systému je velice důležitá pro celkovou práci systému. Tato věc obzvláště platí, když je systém modulární. V tomto systému jsou data zpracovávána každým modulem odlišně a každý modul musí vykonat svou část při zpracování dat.

První návrh systému byl postaven pro jednotlivý datový tok, který přichází z připojeného klienta. S narůstající složitostí a změnou systému ze zařízení podporující jedno zařízení na systém pro víc zařízení, bylo nutné tuto datovou strukturu změnit. Novým typem systému byl takový, ke kterému může být připojeno více různých zařízení.

Prví návrh struktury zpracovával vždy jednu přijatou zprávu. Tato zpráva byla nejdříve systémem zpracovávána a následně poté byla odeslána odpověď uživateli. Následující úryvek kódu reprezentuje datovou strukturu, která odpovídá sekvenčnímu zpracování pro jednoho klienta.

```
{
    "data-to-send": [],
    "data-to-translate": {
        "processed": True,
        "device-id": "",
        "data": ""
    },
    "action-data": {
        "processed": True,
        "action": "",
        "data": ""
    },
    "data-to-process": {
        "processed": True,
        "device-id": "",
        "text": ""
    },
    "data-device": {
        "processed": True,
        "data": ""
    }
}
```

Tato datová struktura obsahuje informace a jednotlivé stavy, kterými projde informace. Při přijmutí zprávy nedochází k vytvoření nové instance tohoto objektu a jeho zpracování. Tento objekt je neměnný po celou dobu spuštění serverové části a jsou v něm pouze zpracovávána data.

Jako první je websocketovým serverem obdržena zpráva z klienta a její informace jsou uloženy pro zpracování modulu převodu řeči do textové podoby. Tato část objektu pak obsahuje data a je nastavena podmínka pro zpracování. Tato data jsou uložena do části objektu, ke které následně přistupují další moduly a je do dat přidána i informace o tom, že tato data ještě nejsou zpracována.

Systém modulů následně zpracuje data krok po kroku, kde kroky jsou na sobě nezávislé a plně asynchronní. Jen je potřeba v každém modulu kontrolovat, jestli jsou data potřebná pro zpracování modulem již připravena, a zároveň jestli už tyto data modul nezpracoval. Nevýhodou tohoto uspořádání je nestabilita při větším množství dat, která přichází rychle za sebou. V tomto případě může docházet k přepsání dat a systém nebude fungovat správně. Pokud totiž přijdou ze dvou zařízení dvě za sebou zprávy jsou uloženy do stejné struktury a je tedy přepsána první zpráva tou novou.

Tato struktura taktéž neobsahuje frontu, ve které jsou objekty, které jsou odesílány potom ze serveru zpátky na klienta. Celkově se tato struktura ukázala při rozšiřování systému na systém pro větší množství klientů a zpracovávání většího počtu dat jako chybná. Na základě požadavků při rozšíření tak byla navržena nová struktura, které více odpovídá potřebám, které by měl systém splňovat. Následující ukázka kódu zobrazuje, jak nová struktura vypadá.

```
[
    {
        "data-to-translate": {
            "processed": False,
            "data": ""
        },  # data pro překlad text to speech
        "data-to-process": {
            "processed": False,
            "text": ""
        },  # data pro zpracování chatbotem
        "data-to-send": {
            "processed": False,
            "text": ""
        },  # data pro odeslání
        "device-id": "",
        "is_action": False
    }
]
```

Výše zobrazená struktura plní v systému úlohu nosiče informací. Taktéž jako původní struktura je navržena pro to, aby systém sekvenčně zpracoval každou zprávu. Hlavním rozdílem oproti předchozí verzi je ten, že objekty nesoucí datovou komunikaci jsou dány do fronty. Není tedy přistupováno stále ke stejné instanci a ta přepisována.

Celý objekt byl zaměněn frontou, do které jsou všechny data hned po přijmutí ukládána. Při přijmutí nejsou přepsána stará data, ale je vytvořena nová instance

objektu ve výchozí podobě. Dále jsou v inicializaci naplněna všechna data, která přišla, a objekt je umístěn do fronty. V této frontě vždy každý modul prochází postupně přišlá data a pokud jsou připravena pro zpracování modulem zpracuje je.

Díky frontě je možné přijímat data z vícero připojených zařízení, aniž by bylo nutné neustále kopírovat id zařízení a data která přišla.

Taktéž nedochází k duplikaci informací, které by nemusely nutně mít odlišnou hodnotu a nedochází ke ztrátě dat při větším počtu zařízení. Data tudíž jsou vždy odeslána na správné zařízení a nedojde k jejich ztrátě nebo přepsání.

V systému jsou tyto datové struktury uloženy v konfiguračním souboru, ze kterého jsou vždy načteny. Výhodou je možnost doplnění struktury za běhu systému. To následně podporuje i přidávání modulu za běhu, kdy systém nemusí byt ukončen, ale jsou pouze načteny změny.

6.3.3 Interface jednotlivých modulů

Hlavní podmínkou systému bylo nepřetržité fungování. Kvůli tomu je potřeba omezit všechny možnosti, při kterých by bylo nutné systém odstavit. Kvůli tomu je nutné přizpůsobit této vlastnosti strukturu. Každá část systému je jednotlivý modul. Výhodou tohoto přístupu je, že pokud budou mít moduly stejný interface, bude možné neomezeně škálovat systém o užitečné funkce. Následující ukázka kódu zobrazuje interface, který musí každý modul implementovat.

```
class ModuleInterface:

    def __init__(self, settings, data):
        self.settings = settings
        self.data = data

    def run(self):
        pass

    def cancel(self):
        pass

    def is_running(self):
        pass

    def restart(self):
        pass

    def get_module_id(self):
        pass
```

Tento interface obsahuje několik metod, které jsou pro běh systému potřebné. První inicializační metoda má dva parametry, do kterých jsou systémem poslány globální systémová data a dále taky nastavení celého systému. Další metodou je metoda run, která je zavolána při startu systému a jedná se o startovní metodu. Tato metoda je zavolána hned po inicializaci modulů.

Metoda *cancel* je zavolána vždy, když je systém ukončován. Taktéž může být volána při restartu samotného modulu. Využití této metody může být například při aktualizaci modulu.

Metoda *is_running* kontroluje, jestli je modul aktivní. Tato metoda slouží k ladění kódu, a taktéž dodává systému informaci, jestli je modul ve funkčním stavu.

Metoda restart je volána, když je potřeba modul restartovat, například při spuštění aktualizace. Poslední metoda interface je metoda *get_module_id*. Tato metoda vrací identifikátor, který tento modul musí mít. Díky tomu je možné rozlišit jaký modul není funkční, a případně se pokusit jej znovu spustit. Tato metoda je nutná z důvodu, že moduly mohou být inicializovány v různém pořadí

6.3.4 Modul pro převodu zvuku na textovou podobu

Kvůli typu zvoleného chatbota je nejdříve nutné převést kus zvukového segmentu reprezentující hlas do textové podoby. Hlavním důležitým aspektem systému je vyextrahování zvukových informací a jejich následné převedení do textové podoby. Serverová část systému ovšem přijímá data, které neobsahují jen zvukovou stopu, Tyto data obsahují prázdný kanál pro případnou sekvenci snímků videa. Tento kanál nemá v modulu pro převod zvuku na text žádné využití.

Data je potřeba nejprve uložit do samotného souboru, aby bylo následně možné jej zpracovat externím nástrojem pro práci s audiovizuálními soubory. U systému není možné zjistit? případné přerušení a kdy k přerušení došlo. Je proto nutné, aby kód obsahoval i pojistku, která v případě přerušení zpracování předchozí soubor smaže, aby nedošlo k chybě.

```
# odstranění možného pozůstalého souboru
if os.path.isfile("audio_file.mp3"):
    os.remove("audio_file.mp3")

# Uložení dat do spustitelného souboru
file = open("audio_file.wav", "wb")
splitted_data = data.split(",")
file.write(base64.decodebytes(splitted_data[1].encode("ascii")))
file.close()
```

Výše zobrazený úryvek kódu zobrazuje právě smazání a zároveň vytvoření nového souboru s vložením dat zaznamenaných v klientovi. Pro další zpracování je tento formát absolutně bez problému, avšak v případě, že systém pracuje na zpracování většího počtu dotazů nezávisle na sebe, je tento přístup velmi nevýhodný, protože může docházet k různému přepisování. Pro ochranu je vhodné použití unikátního jména pro zpracovávané soubory.

Díky této vlastnosti systém také získá schopnost uchovávat jednotlivé zvukové záznamy. V systému bylo zvoleno jako unikátní identifikátor přesné datum pořízení a zpracování záznamu.

```
now = date.today()

filename = now.strftime("%Y_%m_%d_%H_%M_%S.%f")
file = open(filename + ".wav", "wb")
splitted_data = data.split(",")
file.write(base64.decodebytes(splitted_data[1].encode("ascii")))
file.close()
```

Výše zobrazeny kus kódu zobrazuje, jak jsou tyto data uložena a zpracována. Není pak nutné mazat předchozí soubor. Toto je však volitelné a čistě záleží, jestli je nutné pro další zpracování tyto data uchovávat nebo nikoliv.

Jelikož systém pracuje se službou *Speech to text* od Google cloud, [35] není uchovávání nezbytné. Zmíněný systém převádí zvuková data na text tím, že jsou odeslána na Api služby od Google cloud. Tato data jsou pro přenos zakódována opět do formátu base64, ale jedná se nyní pouze o zvuková data. Data, která obsahují informace o zpracování, však obsahují i video kanál, který je sice prázdný ale musí být před zpracováním odebrán, jinak jej služby Google cloud nezpracují. V modulu je proto použita externí aplikace, a to aplikace *ffmpg*. [36] Zavoláním následujícího příkazu jsou data ze souboru transformována do nového souboru a obsahují už pouze zvukovou stopu.

```
command = "A:/ffmpeg/bin/ffmpeg -i audio_file.wav -ab 160k -ac 2 -ar 44100 -vn au-
dio_file.mp3"

subprocess.call(command, shell=True)

with io.open("audio_file.mp3", 'rb') as audio_file:
    content = audio_file.read()
    audio = types.RecognitionAudio(content=content)
```

Výše zobrazený kus kódu pomocí externího příkazu extrahuje z původních dat pouze zvukovou stopu. Následně tuto stopu uloží do nového souboru. Dále kód otevře nově vytvořený soubor a vyextrahuje z něj zvuková data. Tato zvuková data jsou odeslána pro zpracování do služby Google cloud. Google pro své služby poskytuje přímo knihovny, které zpřístupní služby do různých systémů používající různé programovací jazyky. Kód pro získání překladu textu je pak následující.

```
config = types.RecognitionConfig(
    encoding=enums.RecognitionConfig.AudioEncoding.ENCODING_UNSPECIFIED,
    sample_rate_hertz=16000,
    language_code='cs-CZ')

# Detects speech in the audio file
response = self.client.recognize(config, audio)

resultstr = ""
for result in response.results:
    resultstr = resultstr + " " + result.alternatives[0].transcript
```

Výše zobrazený kód odešle data pro zpracování. Výsledkem je poté seznam možných překladů. Jedná se o pole textů, které ukazují na to, jaký je výsledek analýzy zvuku. Výsledek je poté znovu uložen do instance třídy s daty a poté je také nastaveno, že je zpracován a připraven pro zpracování modulem Chatbota.

6.3.5 Modul chatbota

Jelikož se od systému očekává i zpětná interakce, je proto nutné mít v systému vytvořený modul chatbota. Chatbot zpracovává textový vstup a na jeho základě vybírá textovou odpověď. Existuje více druhů chatbotů, kdy některé mohou například odpovědi generovat, anebo jako v tomto případě, odpovědi vybírat na základě příslušnosti interakce. Jedná se tedy o klasifikování otázky do jistých sekcí a pak vybírání odpovědi z databáze odpovědí.

Výhoda a důvod tohoto systému je, že je pod kontrolou text odpovědi. Není tak možné, aby systém vybral odpověď, která je nějakým způsobem urážlivá anebo

nevhodná. Samotná klasifikace probíhá pomocí slovníkového rozebrání vstupní informace.

```
def bag_of_words(self, s):
    bag = [0 for _ in range(len(self.words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(self.words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

Kód nahoře vytváří seznam slov ze zadané věty. A na základě tohoto seznamu vytvoří pole, ve kterém jsou počty jednotlivých slov, které jsou v kusu zprávy obsaženy.

Na základě tohoto pořadí a výskytu slov pak vznikne identifikátor jednotlivé věty, která je poté zpracována neuronovou sítí. Jako vstup této sítě je klasifikace, ke které je zpráva přiřazena. Díky této klasifikaci je pak možné vybrat z relevantních odpovědí, které databáze obsahuje.

Samotná neuronová sít je vytvořena pomocí knihoven Tensorflow. [37]

```
def pretend_result(self, data):
    if hasattr(self, 'model') or not data == "":
        results = self.model.predict([self.bag_of_words(data)])
        results_index = numpy.argmax(results)
        tag = self.labels[results_index]

    db = DBConnector()
    if not db.check_pattern_exists(self.node_id, data):
        db2 = DBConnector()
        db2.pattern_save(self.node_id, data)

    return tag
```

Jako výsledek z neurální sítě přijde pole s procentuálními zastoupením, kam může věta patřit. Výběrem indexu s největší hodnotou pak určí, o jakou třídu se jedná. Následně se z těchto tříd vybere správná odpověď. Takovýmto způsobem je možno vytvářet odpovědi na určité otázky.

Pomocí tohoto systému je možné vytvořit jednotlivé uzly komunikace. Mezi jednotlivými uzly je pak možné přecházet pomocí vhodně zvolených otázek. Tyto

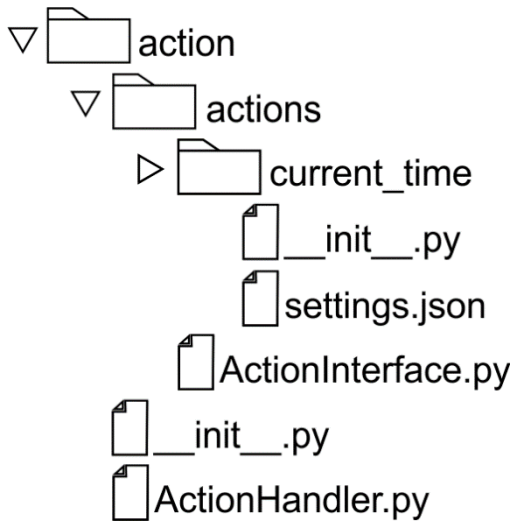
otázky pak mohou nabízet pouze kladné a záporné odpovědi. Kladnou odpovědí pak systém zvládne změnit komunikační uzel, a tudíž i třeba reagovat odlišněji než v jiném modulu. Ve výsledku pak každý modul bude mít vlastní vytrénovanou neurální síť.

Výsledek se poté vloží do zpracovaného objektu a změní se příznak, že je již zpracovaný.

6.3.6 Modul akcí

Umělá inteligence by neměla pouze reagovat na lidskou komunikace v rámci naučené odpovědi. Součástí funkce je i plnění požadavků ze strany uživatele. Po takovýto ušel systém obsahuje modul akcí. Tento modul není pouze modul pro jednu akci, ale jedná se o modul, který v sobě obsahuje všechny akce, které má systém implementované.

Struktura modulu se liší od předchozích modulu hlavně tím, že krom implementování samotné akce obsahuje adresář, ve kterém jsou všechny akce. Není tedy nutné pro akce vytvářet samostatné moduly. Pro jednotlivé akce, je zde pak interface, který obsahuje pár metod a načtení nastavení, které každý modul musí obsahovat. Ukázku takovéto struktury včetně jednoho modulu pro zjištění aktuálního času zobrazuje následující Obrázek 15.



Obrázek 15: Struktura modulu akce včetně ukázkové akce (vlastní tvorba)

Následující kód pak zobrazuje strukturu, kterou má interface jednotlivých akcí.

```
import json
import os
import pathlib
import sys

class ActionInterface:

    def __init__(self, dir_name):
        setf = open(os.path.dirname(os.path.realpath(__file__)) + "/" +
                    dir_name + "/settings.json")
        data = setf.read()
        self.settings = json.loads(data)


    def do_work(self):
        pass

    def get_id(self):
        return self.settings["id"]
```

Tento interface obsahuje dvě metody, které obstarávají vše potřebné. První je inicializační metoda, v které dojde k načtení nastavení, které se nachází v adresáři modulu. Dále je zde metoda *do_work*, která odpovídá za zpracování požadavku. A poslední je zde metoda *get_id*, které vrací jednoznačný identifikátor modulu, který je obsažen v nastavení modulu. Díky tomu je pak možné jednotlivě rozlišit jednotlivé akce od sebe a vyvolat tu správnou. Ukázkou takové akce muže být třeba jednoduchá akce, která na požadavek kolik je hodin vrátí text, který obsahuje aktuální čas.

Všechny tyto akce jsou načteny při startu celého modulu. Následující úryvek kódu zobrazuje část třídy starající se o akce.

```
import os
from threading import Thread


class ActionHandler():
    def __init__(self, settings, data):
        self.data = data # fronta s daty
        self.settings = settings

        #inicializuje actions
        self.actions = self.init_actions()

        # nastartuje vlastni vlakno
        self.thread = Thread(target=self.run, args=())

    def run(self):
        while True:
            for item in self.data:
                if not item["action-data"]["processed"]:
                    response = self.actions[str(item["action-
data"]["action"])].do_work()
                    self.set_response(item, response, "pleased")
                    item["action-data"]["processed"] = True

    # inicializuje akce
    def init_actions(self):
        actions_dir = self.settings["modDir"]+"action/actions/"
        actions_mod = self.settings["modMod"] + "action.actions."
        actions = {}
        for filename in os.listdir(actions_dir):
            if os.path.isdir(actions_dir + filename) and filename != "__pyca-
che__":
                action = __import__(actions_mod + filename, fromlist=[''])
                class_ = getattr(action, "InitAction")
                instance = class_(self.data, self.data)
                id = instance.get_id()
                actions.update({str(id): instance})
        return actions

    def set_response(self, item, text, emoticon):
        item["data-to-send"]["text"] = text
        item["data-to-send"]["emotion"] = emoticon
        item["data-to-send"]["is_send"] = False
        item["prepare-to-send"] = True

    def start(self):
        self.thread.start()
        pass
```

V ukázce kódu nahoře je vidět, že tento modul nevyhodnocuje, jaká akce má být provedena. Pouze se stará o to, aby byla provedena ta akce, kterou komunikační modul vyhodnotil jako tu správnou. Je-li informace přijatá od uživatele je akcí, nebo pouze komunikací zpracovává modul komunikace.

6.3.7 Modul analýzy obrazu

Nejméně důležitý modul v systému je modul pro rozpoznávání osob v obraze. Pomocí nej je možné zjistit, jestli je obrazovka systému sledována že případně jakou verzi odpovědi má systém použít, jestli se jedná o muže či o ženu. Jako první je ze tenkého klienta přijat a uložen obrázek z webkamery která snímá prostor před zařízením. Následující úryvek, dokud ukládá zakódovaný obrázek do obrazové podoby.

```
with open(device_id+'.jpg'  , 'wb') as f:
    f.write(base64.b64decode(webcam_data))
```

V modulu samotném jsou pak při jeho startu načteny naučené neuronové sítě pro detekci různých elementů v obraze. Obrázek je načten a převeden do stupňů šedi a pak analyzován na věk a pohlaví.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags=cv2.CASCADE_SCALE_IMAGE
)

face = cv2.dnn.blobFromImage(image [y:y + h, h:h + w].copy(), 1, (227, 227),
MODEL_MEAN_VALUES, swapRB=False)

g_net.setInput(face)
g_results = g_net.forward()
gender = genders[g_results[0].argmax()]

a_net.setInput(face)
a_results = a_net.forward()
age = ages[a_results[0].argmax()]
```

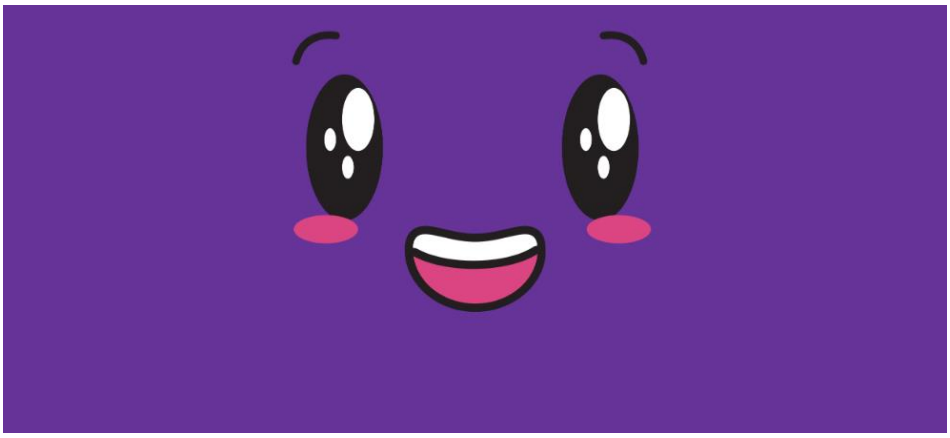
Výsledkem z úryvku kódu nahoře je analyzované umístění obličeje a pokud se v obrazu vyskytuje a také odhad věku a pohlaví pomocí neuronových sítí. Jelikož tyto metody jsou zpracovávány z obrazu mají vetší náchylnost na špatnou predikci. I pouhé světlo ze špatného uhlu muže tuto predikce rozhodit. Taktéž je predikce velmi ovlivněna souborem obličejů, na kterém byly neuronové sítě trénovány.

7 Shrnutí výsledků

Výsledným implementováním popsaných metod vznikl systém, který umožnuje interagovat mluveným slovem. Kvůli omezení ze strany Google Cloud services je systém implementován tak, že slovní povely jsou odesílány pouze pokud dochází k interakci s webovou stránkou uživatelem.

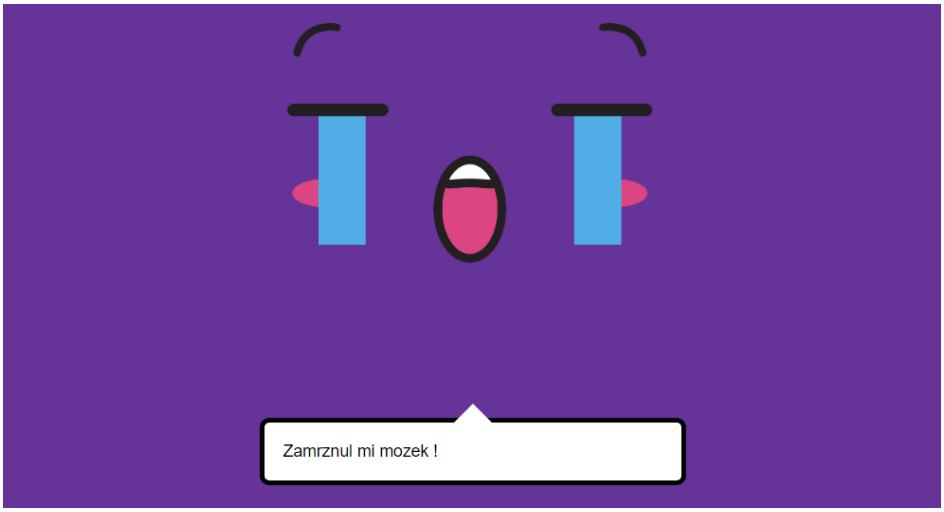
Systém je spuštěn na Linuxovém serveru, kde jsou spuštěny dílčí aplikace. První z částí, která na serveru musí být spuštěna je Webový server Apache. Tento server zpřístupňuje webové stránky a klientskou část pro koncová zařízení. Dále je pak na serveru MysQL server, na kterém je vytvořena databáze, která obsahuje data pro odpovědi a data pro trénování neuronových sítí. Pro fungování serverové části systému je na serveru nainstalována nejnovější verze Pythonu.

Následující obrázky pak zobrazují reakce systému na různé podměty, které byly pro testování vytvořeny. Prvním obrázkem je výchozí stav, do kterého je systém uveden při úspěšném navázání komunikace se serverovou částí.



Obrázek 16: Výchozí obličej klientské části (obličej použit z [38])

Oproti tomuto výchozímu stavu se pak nabízí situace, kdy nefunguje spojení se serverovou částí a následně je uživatel informován obličejem s jasným vzkazem jaký je zobrazen na Obrázek 17.

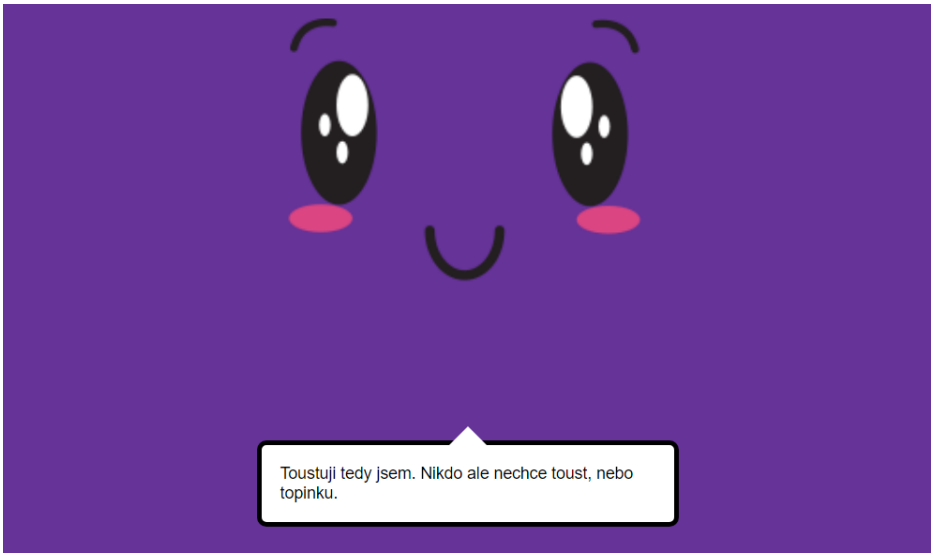


Obrázek 17: Stav systému indikující ztrátu konektivity (obličej použit z [38])

Jako první ze scénářů byl vytvořen scénář na pozdrav, který by uživatel vznesl na systém.

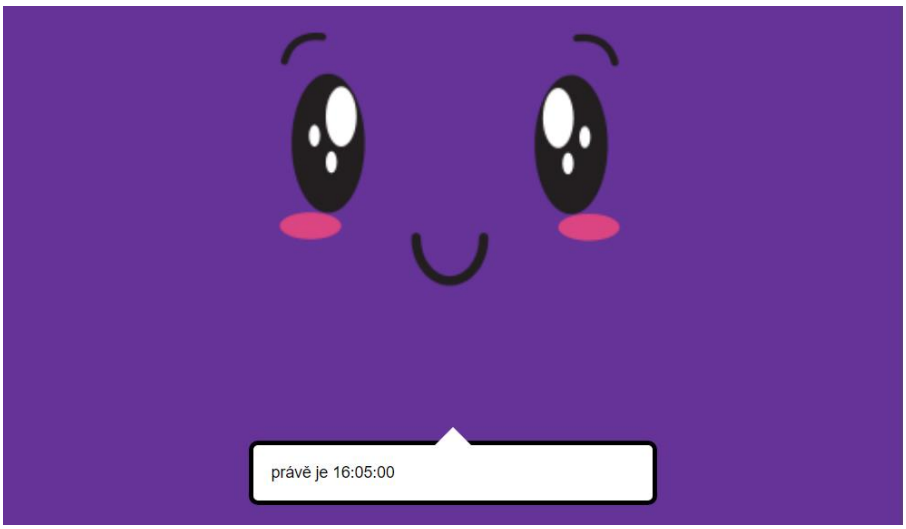
Obrázek 18: Odpověď na pozdrav "Dobrý den" (obličej použit z [38])

Další logická otázka, a tudíž ukázkový scénář, je „Jak se máš?“. Na tento scénář má systém také připravené odpovědi, které na tento scénář odpovídají viz. Obrázek 19.



Obrázek 19: Reakce na otázku jak se má (obličej použit z [33])

Jako poslední ukázkou je obrázek zobrazující reakci na dotaz uživatele, kolik je hodin. Jedná se tedy nejen o konverzační odpověď, ale také se jedná o předpřipravenou demonstrační akci, na jejímž základě systém zjistí aktuální čas a odpoví uživateli.



Obrázek 20: Reakce na požadavek na otázku kolik je hodin (obličej použit z [38])

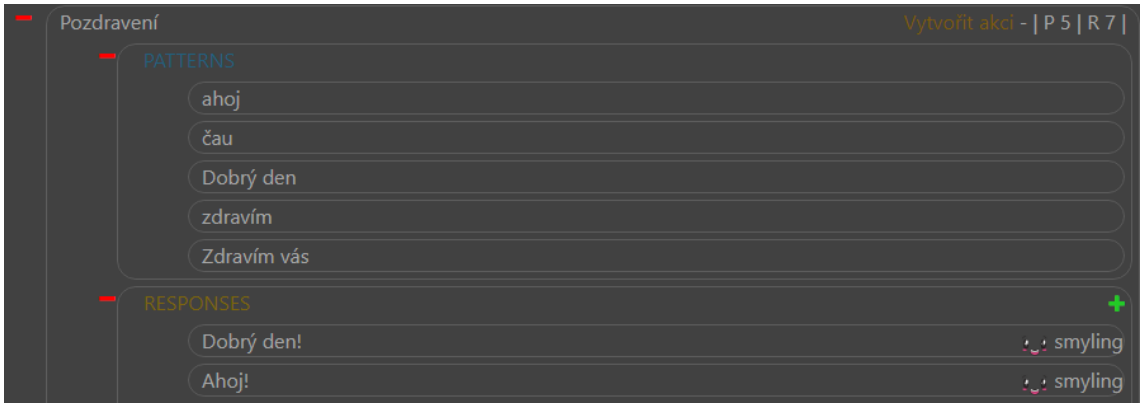
Při testování systémy vznikly pouze nepřesnosti při testování analýzy obrazu, když bylo velmi dynamické rozpoznávání věku a pohlaví. V otázce správnosti analýzy mluvnené řeči docházelo k méně častým nepřesnostem, například při

zakrytí mikrofonu, nebo při nesprávné artikulaci. U samotné komunikační neuronové sítě docházelo pouze k nepřesnostem při analyzování zpráv, které nebyly při učení použity. Tento nedostatek bude odstraněn zvětšením velikosti učících dat, která pokryjí větší množství variant.

Následující ukázka zobrazuje následné uspořádání komunikačních dat v systému. Jedná se o jednotlivé typy odpovědí na jednotlivé případy. Výhodou takovéhoto uspořádání je přehlednost a kontrola výsledků.

Obrázek 21: Ukázka komunikačního uzlu včetně akcí (vlastní tvorba)

Každý takovýto komunikační případ obsahuje ukázky možných interakcí na naučení neuronové sítě a potom i vhodné odpovědi. Následující obrázek pak zobrazuje ukázku takové části. Stejně tak to i funguje při akci, kterou má systém provést. Taktéž je možné zvolit jakou emoci má systém ukázat uživateli.



Obrázek 22: Ukázka odpovědí a otázek pro učení neuronové sítě (vlastní tvorba)

Systém taktéž umí zavádět nové akce a moduly přímo za běhu, stačí pouze dodržet podmínky pro modul. Systém pak jen co je modul nahrán jej spustí a zavede do systému. Je tedy možné vytvořit modul odděleně a dodat jej vzdáleně bez přerušení systému.

Stejně pak systém zavede i jednotlivé balíčky akcí a zpřístupní je uživateli. Díky tomu je možné systém rozšířit o různé akce. Je tak možné vytvořit akci pro ovládání světel, která je bude rozsvěcet, nebo je možné vytvořit akci pro zjištění novinek z informačních portálů a předat je uživateli.

8 Závěr

Diplomová práce se zabývá problematikou komunikace mezi člověkem a počítačem. V práci jsou posupně rozebrány a popsány jednotlivé dílčí procesy, které tvoří celkový systém. V práci je také vidět proces jednotlivých informací a jejich zpracování až po výslednou reakci systému. Součástí práce je taky navržení systému, který reaguje na lidské povely a simuluje lidskou komunikaci.

Tento navržený systém je možno libovolně rozšiřovat a díky tomu je možné jej nasadit v různých situací, kdy systém může zpracovávat lidské povely a komunikaci různými způsoby a stát se tak virtuální sekretářkou, nebo pomocníkem lidí s omezenou pohyblivostí nebo zjednodušit a zautomatizovat úkony při výrobě v průmyslu.

Metody tak jak tu byly popsány, mají jistou úroveň chybovosti. Chybovost je určena nedokonalým vzorkem při učení neuronových sítí. Čím větší bude soubor dat pro učení, tím budou metody přesnější, a i systém bude pracovat s menšími chybami. Chybami je myšleno vyhodnocování a reagování na podměty, nikoliv pády systému a podobné.

Systém, který byl vytvořen pro demonstraci, obsahuje základní výchozí moduly, které zpracovávají data. Tato data jsou dostupná dalším případným modulům, které je mohou zpracovávat pro odlišné účely.

Hlavním problémem systému bylo správné propojení a zpracování dat, která systém pro svou práci potřebuje a využívá. Každý modul totiž vyžaduje různý čas na různé operace. Je tedy nutné, aby systém nepředbíhal a zpracoval informace ve správném pořadí a bez chyb. Proto bylo nutné každý modul v systému oddělit a i upřednostnit, aby běžel v samotné úloze a nedocházelo k blokování funkce dalších modulů. Díky tomu je pak možné, aby systém správně a plynule fungoval.

Výsledkem je tedy systém ovladatelný lidskou řečí s velkou možností rozšiřitelnosti a znuvupoužitelnosti v různých aspektech lidského života.

9 Seznam použité literatury

[1] BROOKS, Rodney A. Intelligence Without Reason. In: Luc STEELS a Rodney BROOKS, ed. *The Artificial Life Route to Artificial Intelligence* [online]. 1. vyd. B.m.: Routledge, 2018 [vid. 2020-04-08], s. 25–81. ISBN 978-1-351-00188-5. Dostupné z: doi:10.4324/9781351001885-2

[2] LUGER, George F. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. B.m.: Pearson Education, 2005. ISBN 978-0-321-26318-6.

[3] ALRAJEH, Nabil Ali a J. LLORET. Intrusion Detection Systems Based on Artificial Intelligence Techniques in Wireless Sensor Networks. *International Journal of Distributed Sensor Networks* [online]. 2013, **9**(10), 351047. ISSN 1550-1477. Dostupné z: doi:10.1155/2013/351047

[4] MACKINTOSH, Nicholas a Nicholas John MACKINTOSH. *IQ and Human Intelligence*. B.m.: OUP Oxford, 2011. ISBN 978-0-19-958559-5.

[5] GUZMAN, Andrea L a Seth C LEWIS. Artificial intelligence and communication: A Human–Machine Communication research agenda. *New Media & Society* [online]. 2020, **22**(1), 70–86. ISSN 1461-4448. Dostupné z: doi:10.1177/1461444819858691

[6] MARTINŮ, Jiří. METODIKY VÝVOJE SOFTWARE. nedatováno, 146.

[7] TURING, Alan Mathison. *COMPUTING MACHINERY AND INTELLIGENCE* [online]. [vid. 2020-04-08]. Dostupné z: http://cogprints.org/499/1/turing.html

[8] *A new (computer) chess champion is crowned, and the continued demise of human Grandmasters - ExtremeTech* [online]. [vid. 2020-04-05]. Dostupné z: https://www.extremetech.com/extreme/196554-a-new-computer-chess-champion-is-crowned-and-the-continued-demise-of-human-grandmasters

[9] ALLEN, Colin. COMPUTING MACHINERY AND INTELLIGENCE. nedatováno, 22.

[10] MCCARTHY, J. a P.J. HAYES. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In: *Readings in Artificial Intelligence* [online]. B.m.: Elsevier, 1981 [vid. 2020-02-01], s. 431–450. ISBN 978-0-934613-03-3. Dostupné z: doi:10.1016/B978-0-934613-03-3.50033-7

[11] AGAR, Nicholas. Don't Worry about Superintelligence. nedatováno, 10.

[12] MÜLLER, Vincent C. a Nick BOSTROM. Future Progress in Artificial Intelligence: A Survey of Expert Opinion. In: Vincent C. MÜLLER, ed. *Fundamental Issues of Artificial Intelligence* [online]. Cham: Springer International Publishing, 2016 [vid. 2020-04-08], s. 555–572. ISBN 978-3-319-26483-7. Dostupné z: doi:10.1007/978-3-319-26485-1_33

[13] LAPAKKO, David. Three cheers for language: A closer examination of a widely cited study of nonverbal communication. *Communication Education* [online]. 1997, **46**(1), 63–67. ISSN 0363-4523, 1479-5795. Dostupné z: doi:10.1080/03634529709379073

[14] *Object recognition using the OpenCV Haar cascade-classifier on the iOS platform* [online]. [vid. 2020-02-02]. Dostupné z: http://www.diva-portal.org/smash/get/diva2:601707/FULLTEXT01.pdf

[15] ZHOU, Wenzhang, Yong CHEN a Siyuan LIANG. Sparse Haar-Like Feature and Image Similarity-Based Detection Algorithm for Circular Hole of Engine Cylinder Head. *Applied Sciences* [online]. 2018, **8**(10), 2006. ISSN 2076-3417. Dostupné z: doi:10.3390/app8102006

[16] LIENHART, R. a J. MAYDT. An extended set of Haar-like features for rapid object detection. In: *ICIP 2002 International Conference on Image Processing*: *Proceedings. International Conference on Image Processing* [online]. Rochester, NY, USA: IEEE, 2002, s. I-900-I–903 [vid. 2020-02-18]. ISBN 978-0-7803-7622-9. Dostupné z: doi:10.1109/ICIP.2002.1038171

[17] FREUND, Yoav a Robert E SCHAPIRE. Experiments with a New Boosting Algorithm. nedatováno, 9.

[18] LAWRENCE, S., C.L. GILES, AH CHUNG TSOI a A.D. BACK. Face recognition: a convolutional neural-network approach. *IEEE Transactions on Neural Networks* [online]. 1997, **8**(1), 98–113. ISSN 10459227. Dostupné z: doi:10.1109/72.554195

[19] KADIR, Kushsairy, Mohd Khairi KAMARUDDIN, Haidawati NASIR, Sairul I SAFIE a Zulkifli Abdul Kadir BAKTI. A comparative study between LBP and Haar-like features for Face Detection using OpenCV. In: *2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)*: *2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T)* [online]. Kuala Lumpur, Malaysia: IEEE, 2014, s. 335–339 [vid. 2020-05-01]. ISBN 978-1-4799-4621-1. Dostupné z: doi:10.1109/ICE2T.2014.7006273

[20] PRICE, Rob. Microsoft is deleting its AI chatbot's incredibly racist tweets. nedatováno, 1.

[21] RUSS, Martin. *Sound Synthesis and Sampling*. B.m.: Taylor & Francis, 2004. ISBN 978-0-240-51692-9.

[22] CHENG, Yan-Ming. [54] AUTOMATIC AND ATTENDANT SPEECH TO TEXT CONVERSION IN A SELECTIVE CALL RADIO SYSTEM AND METHOD. 2000, 13.

[23] RUSSELL, Stuart a Peter NORVIG. *Artificial intelligence: a modern approach* [online]. 2002 [vid. 2020-05-05]. Dostupné z: https://www.sti-

innsbruck.at/sites/default/files/Knowledge-Representation-Search-and-Rules/Russel-&-Norvig-Inference-and-Logic-Sections-7.pdf

[24] XU, Anbang, Zhe LIU, Yufan GUO, Vibha SINHA a Rama AKKIRAJU. A New Chatbot for Customer Service on Social Media. In: *CHI '17: CHI Conference on Human Factors in Computing Systems*: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* [online]. Denver Colorado USA: ACM, 2017, s. 3506–3510 [vid. 2020-04-13]. ISBN 978-1-4503-4655-9. Dostupné z: doi:10.1145/3025453.3025496

[25] Welcome to Python.org. *Python.org* [online]. [vid. 2020-04-13]. Dostupné z: https://www.python.org/

[26] REITZ, Kenneth a Tanya SCHLUSSER. The Hitchhiker's Guide To Python. nedatováno, 322.

[27] *index | TIOBE - The Software Quality Company* [online]. [vid. 2020-04-13]. Dostupné z: https://www.tiobe.com/tiobe-index/

[28] *World Wide Web Consortium (W3C)* [online]. [vid. 2020-04-20]. Dostupné z: https://www.w3.org/

[29] LIE, Håkon Wium a Janne SAARELA. Multipurpose Web publishing using HTML, XML, and CSS. *Communications of the ACM* [online]. 1999, **42**(10), 95–101. ISSN 00010782. Dostupné z: doi:10.1145/317665.317681

[30] *Welcome to Ecma International* [online]. [vid. 2020-04-20]. Dostupné z: https://www.ecma-international.org/

[31] *IETF | Internet Engineering Task Force* [online]. [vid. 2020-04-20]. Dostupné z: https://www.ietf.org/

[32] *OpenCV library* [online]. [vid. 2017-08-16]. Dostupné z: http://opencv.org/

[33] *OpenCV: OpenCV Tutorials* [online]. [vid. 2020-04-20]. Dostupné z: https://docs.opencv.org/master/d9/df8/tutorial_root.html

[34] *Questions - OpenCV Q&A Forum* [online]. [vid. 2017-08-16]. Dostupné z: http://answers.opencv.org/questions/

[35] Cloud Speech-to-Text - Speech Recognition. *Google Cloud* [online]. [vid. 2020-05-10]. Dostupné z: https://cloud.google.com/speech-to-text?hl=cs

[36] *FFmpeg* [online]. [vid. 2020-05-10]. Dostupné z: https://www.ffmpeg.org/

[37] *TensorFlow* [online]. [vid. 2020-05-10]. Dostupné z: https://www.tensorflow.org/

[38] Download Set Of 18 Designs Of Kawaii Expressions for free. *Freepik* [online]. [vid. 2020-05-04]. Dostupné z: https://www.freepik.com/free-vector/set-18-designs-kawaii-expressions_5685655.htm

10 Seznam obrázků

Příloha č. 1

Zadání diplomové práce

Autor:	**Bc. Adam Richter**
Studium:	I1700700
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Komunikace člověk-stroj s využitím metod umělé inteligence**
Název diplomové práce AJ:	Human-machine communication using artificial intelligence methods

Cíl, metody, literatura, předpoklady:

Cíl: Návrh a implementace modulárního systému pro

oboustrannou komunikaci počítač - člověk

Obsah:

1. Úvod

2. Metody umělé inteligence

3. Zpracování obrazových a zvukových dat

4. Návrh a implementace komunikačního systému

5. Výsledky a závěr

Literatura:

GUZMAN, Andrea L a Seth C LEWIS. Artificial intelligence and communication: A Human-Machine Communication research agenda. *New Media {\& Society [online]. 2020,* ***22(1), 70-86. ISSN 1461-4448.***

RUSSELL, Stuart; NORVIG, Peter. Artificial intelligence: a modern approach. 2002.

RUSS, Martin. Sound synthesis and sampling. Taylor {\& Francis, 2004.

LAWRENCE, Steve, et al. Face recognition: A convolutional neural-network approach. IEEE transactions on neural networks, 1997, 8.1: 98-113.

Garantující pracoviště:	Katedra informačních technologií, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Karel Mls, Ph.D.
Oponent:	Ing. Martina Husáková, Ph.D.
Datum zadání závěrečné práce:	14.1.2015