SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

EVIDENČNÉ ČÍSLO: FCHPT-5415-61607

Tvorba užívateľského rozhrania pre inteligentné termostaty

Bakalárska práca

2015 **Jokhongir Mamarasulov**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Tvorba užívateľského rozhrania pre inteligentné termostaty

Bakalárska práca

FCHPT-5415-61607

Študijný program: automatizácia, informatizácia a manažment v chémii a potravinárstve
Číslo študijného odboru: 2621
Názov študijného odboru: 5.2.14. automatizácia, 5.2.52. priemyselné inžinierstvo
Školiace pracovisko: Ústav informatizácie, automatizácie a matematiky
Vedúci záverečnej práce/školiteľ: doc. Ing. Michal Kvasnica, PhD.

2015 **Jokhongir Mamarasulov**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2014/2015
Evidenčné číslo: FCHPT-5415-61607

STU FCHPT

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Jokhongir Mamarasulov**
ID študenta:	61607
Študijný program:	automatizácia, informatizácia a manažment v chémii a potravinárstve
Kombinácia študijných odborov:	5.2.14. automatizácia, 5.2.52. priemyselné inžinierstvo
Vedúci práce:	doc. Ing. Michal Kvasnica, PhD.

Názov práce: **Tvorba užívateľského rozhrania pre inteligentné termostaty**

Špecifikácia zadania:

Cieľom práce je vytvoriť pohodlné a prívetíve užívateľské rozhranie určené na ovládanie inteligentných termostatov. Takéto rozhranie pritom dovolí nastaviť žiadanú hodnotu vnútornej teploty ako i vložiť predpokladanú obsadenosť budovy v blízkej budúcnosti

Úlohy:
* kriticky zhodnotiť vhodnosť rôznych prístupov k bezdotykovému ovládaniu termostatov
* pre zvolený prístup implementovať technické riešenie na vkladanie žiadanej teploty a budúcej obsadenosti
* simulačne verifikovať výsledky pri použití inteligentného termostatu založeného na prediktívnom riadení

Rozsah práce: 40

Riešenie zadania práce od: 16. 02. 2015
Dátum odovzdania práce: 24. 05. 2015

L. S.

Jokhongir Mamarasulov
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Poďakovanie

Ďakujem vedúcemu bakalárskej práce doc. Ing. Michalovi Kvasnicovi, PhD. za odborné rady a metodické usmerňovanie pri písaní práce a taktiež Ing. Mariánovi Gallovi, PhD. za výpomoc pri písaní zdrojového kódu.

Jokhongir Mamarasulov

Bratislava, 2015

Abstrakt

Cieľom práce je vytvoriť užívateľské rozhranie pre inteligentné termostaty, komunikujúce prostredníctvom hlasových signálov. Vyvíjané rozhranie je rozdelené na 5 esenciálnych častí a k jednotlivej časti je venovaná kapitola, ktorá uvádza do problematiky vednej disciplíny, opisuje spôsoby jej interpretácie v programovacom jazyku python ako je napríklad inštalácia a implementácia. Každá kapitola je ukončená experimentálnou validáciou, kde sa podrobne rozoberá zdrojový kód programu pre danú fázu rozhrania. Záverom práce je vytvorenie umelej inteligencie, zodpovedajúcej stanoveným cieľom. Úspešnosť vykonávania správnej analýzy vstupných údajov nie je porovnateľná so súčasnými manuálnymi termostatmi, avšak stále vysoká. Spôsob interakcie prostredníctvom hlasovej komunikácie je faktor zjednodušujúci prístup užívateľa k danému zariadeniu.

Kľučové slová: termostat; python; umelá inteligencia; hlasová komunikácia.

Abstract

The objective of the thesis is to create a user friendly interface for intelligent thermostats that communicates with the user through voice signals. The development of the interface is divided into five essential sections. Each individual section is accompanied by a chapter that introduces the scientific disciplines involved and describes the methods of its interpretation in the programming language Python such as installation and implementation. Each chapter ends with experimental validation, where the source code of the program for the given phase of the interface is closely analysed. The conclusion consists of the creation of the artificial intelligence that corresponds to the stated objective. Although the successful analysis of voice input by the interface is still not to the standard of currently used manual thermostats, it is still high. Using voice communication as the means of interaction between user and system is a major factor in simplifying user access to the given device.

Keywords: thermostat; python; artificial intelligence; voice communication.

Obsah

Zoznam obrázkov

Zoznám zdrojových kódov

Kapitola 1

Úvod

Počitačove technológie sa v dnešnom svete dostavajú do popredia a čoraz viac obmedzujú zasahovanie ľudského faktoru v rôznych oblastiach. Zároveň však aj uľahčujú interakciu medzi človekom a elektrotechnikou. Tomu najviac nasvedčujú novodobé telefóny, počítače a automobily. Za posledné desaťročie sa k ním pridala aj bytová technika, ktorá na základe učenia prediktívne riadi domácnosť. Jedným z takýhto príkladov je inteligentný termostat, ktorý vyžaduje čoraz menej užívateľských zásahov. Väčšina z ních je ale nastavená na manuálne riadenie prostredníctvom inteligentných pomôcok, alebo priamym zadavaním príkazov do termostatu. Práve tento poznatok nás viedol k záujmu skúmať možné spôsoby ľudskej interakcii s elektrotechnikou daného druhu.

Jedným z už existujúcich riešení sú inteligentné termostaty, riadené obyčajným potočením kolieska, ktoré predstavuje samotný termostat. Okrem toho sa v dnešnej dobe začínajú vyskytovať termostaty, ktoré s užívateľom komunikujú.

V práci sa zameriame na spôsob interakcie s termostatom prostredníctvom hlasovej komunikácie. Cieľom práce je vytvoriť uživateľské rozhranie, podstata ktorého spočíva v odstránení manuálneho riadenia inteligentných termostatov a v nahradení daného riadenia komunikačným modulom, schopným rozoznať príkazy od užívateľa v anglickom jazyku. Okrem analýzy vstupných príkazov je taktiež účelom programu podávať satisfikujúce výstupy vo forme zmeny teploty, alebo časového plánu, kde podrobne priradí jednotlivým hodinám príslušné teploty.

Na záver, po vytvorení programu usúdime, či takto založené rozhranie má perspektívu vo vývoji, či sa presnosť analýzy hlasových signálov užívateľa môže porovnávať so súčasnými manuálnymi termostatmi a či sa tak vytvorené rozhranie dá aplikovať na extérne použitie do skutočných termostatov.

Kapitola 2

Ciele práce

V súčasnosti inteligentné termostaty riadia teplotu v miestnosti zoskupením algoritmov, ktoré na realizáciu výstupu potrebujú mať užívateľom zadefinovanú žiadanú teplotu a taktiež obsadenosť miestnosti na aspoň najbližších 24 hodín. Definícia vstupných parametrov od užívateľa sa realizuje prostredníctvom riadiaceho panelu, prípadne dialkovým použitím smartphonov. Cieľom práce je uľahčiť užívateľský prístup k termostatom bezdotykovým ovládaním, konkrétne vytvorením rozhrania, komunikujúce prostredníctvom hlasových signálov. Termostat, využívajúci dané rozhranie sa umožní interakciu užívateľa a termostata pomocou hlasovej komunikácie.

Realizácia cieľu práce je vykonávaná vytvorením programu, ktorý bude schopný preložiť vstupné hlasové signály prostredníctvom použitia umelej inteligencie na rozpoznávanie ľudskej reči do podoby digitálneho textu. Vstupné signály predstavujú hlasovú požiadavku užívateľa, ktorá je zameraná buď na okamžitú zmenu teploty, alebo na ohlásenie budúcej obsadenosti miestnosti. Pri analýze digitálnej podoby textu je účelom programu vhodne extrahovať informácie na získanie údajov, potrebných na realizáciu výstupného príkazu.

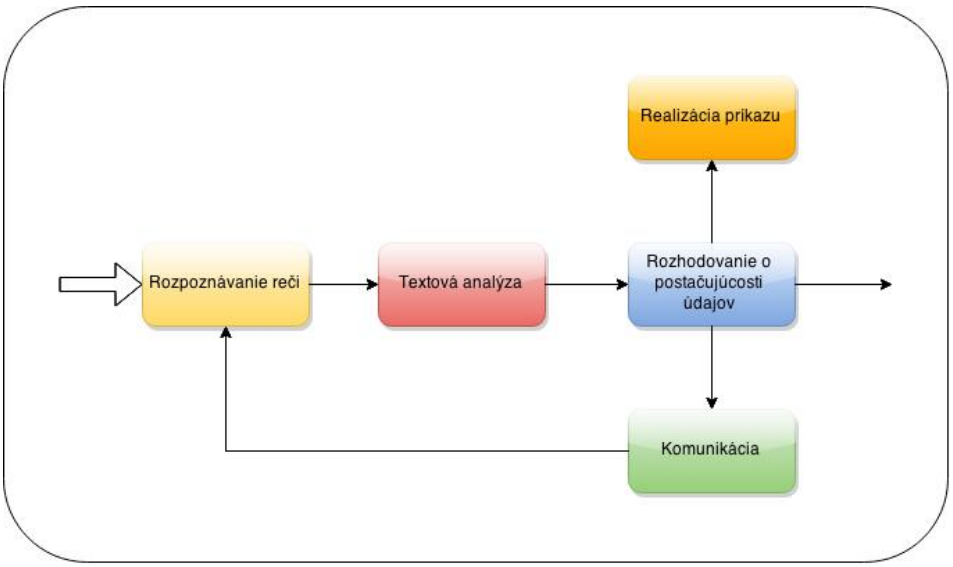
V prípade zámeru uživateľa dosiahnuť okamžitú zmenu teploty v miestnosti sa program orientuje na konkrétne číselné údaje, ktoré sa vo vete vyskytujú v podobe rozdielu, o koľko sa má zmeniť hodnota teploty a taktiež na údaje, či sa má teplota zvyšovať, alebo znižovať. Okrem iného môže nastať situácia, keď užívateľ rovno udáva novú hodnotu žiadanej teploty. V prípade, ak užívateľ oznamuje budúcu obsadenosť miestnosti, text sa analyzuje na prítomnosť časových údajov, ktoré sa vo vstupnej vete reprezentujú

dátumom a časovým intervalom.

Analýza textu sa realizuje prostredníctvom vhodnej kombinácie platform, slúžiacich na preklad ľudskej reči do počítačovej a taktiež použitím knižnice na spracovanie časových údajov.

Po vykonaní analýzy vstupného textu a po priradení číselných alebo časových údajov do premenných sa realizuje výstupná časť programu. Tú predstavuje kombinácia výstupnej vety na obrazovke a výstupných hlasových signálov, v prípade požiadavky na okamžitú zmenu teploty a rozpísanie časového plánu zmeny teplôt na celý deň, v prípade požiadavky na budúcu obsadenosť miestnosti. Výstupné hlasové signály sa realizujú použitím umelej inteligencie na syntézu reči.

Vyššie opísané ciele práce sa dajú schematicky začleniť do jedného celku, z ktorého pozostáva celistvý program. Toto rozdelenie nemusí platiť z hľadiska štruktúry zdrojového kódu, ale skôr ako orientačné celky, na ktoré sa treba zamerať pri jeho tvorbe.



Obr. 2.1: Rozbor procesov

Začiatkom programu je blok **Rozpoznávanie reči**, ktorý má za úlohu spracovať vstupné hlasové signály a premeniť ich do digitálnej formy textu. Ďalej digitálna forma textu sa posiela na spracovanie do bloku **Textová analýza**. Tu sa analyzuje požiadavka užívateľa na zmenu teploty a prepisuje do formy, ktorú je programovací jazyk schopný interpretovať. V bloku **Rozhodovanie o postačujúcosti údajov** má za úlohu program zistiť, či je schopný interpretovať kompletnú informáciu, ktorú dostal od užívateľa, resp., či užívateľ poskytol informáciu dostatočnú na to, aby mohla prebehnúť **Realizácia príkazu**. V prípade, že je toho dosť, prepísané signály postupujú na realizáciu príkazu. V opačnom prípade program pôjde cestou **Komunikácie** a pokusí sa získať chýbajúce, resp. nepresné údaje. Ak sa mu to nepodarí ani na druhý pokus, program ostane bez výstupu.

V nasledujúcich kapitolách sú bližšie opísané jednotlivé bloky schémy, v ktorých sa v krátkosti uvádza do problematiky príslušných vedných odborov, ich historický vývoj a podstata funkčnosti progamu, ktorá bude ukončená experimentálnou validáciou, v ktorej budú opísané príklady použitia zdrojového kódu.

Pred podrobnejším opisom schémy je vhodné zadefinovať pojem **Umelá inteligencia**, ktorá bez pochýb, súvisí s vyvíjaným programom. Bloky na rozpoznávanie a syntézu reči pozostávajú z už napísaných programov, využívajúcich umelú inteligenciu na verifikáciu rôznych šumových signálov. Okrem tých dvoch blokov sem zapadá blok analýza textu, ktorý predstavuje vlastný poskladaný program, využívajúci umelú inteligenciu na verifikáciu textových označení.

Kapitola 3

Umelá inteligencia

Umelá inteligencia (Artificial intelligence, AI) je veda a technika tvorby inteligentných strojov, prevažne inteligentných počítačových programov. Umelá inteligencia súvisí s podobnou úlohou využitia počítačov na porozumenie ľudskej inteligencii, pričom sa nemusí obmedziť na metódy, ktoré sú len biologicky pozorovateľné [1].

3.1 Vznik a vývoj termínu umelá inteligencia

Citovaná v preambule definícia umelej inteligencie, ktorú poskytol John McCarthy v roku 1956 na konferencii na Dartmouthskej univerzite, nie je priamo spojená s porozumením inteligencie u človeka. Podľa McCarthyho AI-výskumníci môžu používať metódy, ktoré nie sú charakteristické pre ľudí, ak to je nevyhnutné pre riešenie konkrétnych problémov.

Vysvetľujúc svoju definíciu, John McCarthy poukazuje na to, že problém pozostáva z toho, že zatiaľ nemôžeme celkom zadefinovať, aké výpočtové postupy chceme nazývať inteligentnými. Rozumieme niektorým mechanizmom inteligencie, ale ostatným zas nie. Preto sa pod inteligenciou v hraniciach tohto odboru chápe len výpočtová časť schopností dosiahnuť cieľov na svete [1].

3.2 História umelej inteligencie

História umelej inteligencie ako učenie o vývoji súčasnej vedy a technológie tvorby intelegentných strojov má svoje počiatky v dávnych filozofických bádaniach ľudskej podstaty a procesu spoznávania sveta, neskôr rozšírených neurofyziológmi a psychológmi ako veľké množstvo teórií, týkajúcich sa práce ľudského mozgu a myslenia. Moderná etapa rozvoja vedy o umelej inteligencii je vývoj základu matematickej výpočtovej teórie - teórie algoritmov a vytvorenie počítačov.

Okrem toho umelá inteligencia už vystupovala v starovekých mýtoch a v beletrii. Mechanický človek a umelé bytosti sa nachádzajú v gréckej mytológii, ako napríklad Hephaestov zlatý robot, alebo Pygmalionova Galatea [2].

V stredoveku sa šírilo veľa povestí o mystických tajomstvách, alebo alchemických metódach uloženia vedomia do hmoty. Príkladmi sú Jabir ibn Hayyan-ov Takwin, Paracelsiov Homunculus alebo Rabbi Judah Loew-ov Golem [3].

K 19. storočiu sa predstava ohľadom umelého človeka a mysliacich strojov vyvíjala v povestiach, ako Frankenstein od M. Shelley, R.U.R. od K. Čapka, alebo Darwin among the Machines od S. Butlera.

Science fiction sa v súčasnosti opiera o komixové a filmové diela, ako napríklad komixový Ultron, alebo plátnový Terminátor. V oboch prípadoch ide o stroje, ktoré sa v konečnom dôsledku snažia zničiť ľudstvo. Avšak časom, uprostred deštruktívnych predstáv vystupuje I. Asimov s tromi zákonmi robotiky [4].

1. Robot nesmie ublížiť človeku alebo svojou nečinnosťou dopustiť, aby mu bolo ublížené.

2. Robot musí poslúchnuť človeka, okrem prípadov, keď je to v rozpore s prvým zákonom.

3. Robot sa musí chrániť pred poškodením, okrem prípadov, keď je to v rozpore s prvým alebo druhým zákonom.

3.2.1 Turingov test

K vývoju porozumenia umelej inteligencie taktiež výrazne prispieva Turingov test. A. Turing v roku 1950 publikoval článok, v ktorom popísal efektívny spôsob testovania inteligencie strojov [5]. Tento test sa tiež niekedy označuje ako imitačná hra, pri ktorej, v jej jednoduchej verzii, človek pomocou počítača kladie otázky (akéhokoľvek zamerania) v normálnej ľudskej reči dvom hráčom. Jeden z hráčov je človek, druhý je počítač. Ak človek nie je schopný rozoznať, ktorý z hráčov je počítač, počítač prešiel Turingovým testom.

Za veľmi jednoduchý príklad súčasného Turingovho testu sa môže považovať vynález Captcha [6]. Ten reprezentuje reťazec rôzne deformovaných písmen na obrázku, ktoré človek musí vpísať do boxu pred tým, ako sa formulár odošle. Týmto spôsobom sa zabraňuje automatickému odosielaniu formulárov internetovými robotmi. To je jedným z príkladov imitačnej hry, kde rozhodujúcim faktorom je počítač a musí uvážiť, či odosielateľ formulára je človek alebo program.

3.2.2 Umelá inteligencia v počítačových hrách

V roku 1951, použitím stroja Ferranti Mark 1 na Univerzite Manchestra, vytvoril C. Strachey program pre dámu a D. Prinz pre šach. Šachový program Arthura Samuela, vyvíjaný v polovici 50. a začiatkom 60. rokov časom nadobudol dostatočné zručnosti na to, aby porazil skúseného amatéra. Neskôr umelá inteligencia exponenciálne rástla so zväčšujúm sa dopytom po video hrách až po dnešné dni, kde len jeden inteligentný modul môže obsahovať niekoľko gigabytov pamäte.

3.2.3 Argument čínskej izby

V roku 1980 známy filozof mysle a umelej inteligencie J. Searle navrhol myšlienkový experiment, ktorý bez pochyb ukončil naše úvahy o tom, či počítače dokážu mať myseľ a či dokážu rozumieť tomu, čo vlastne robia. Týmto zároveň aj vytvoril pomerne silný argument proti Turingovmu testu. Experiment je známy, ako argument čínskej izby [7].

Jeden z výkladov jeho experimentu interpretoval P. Kostelník v knihe Umelá inteligencia a kognitívna veda I. Izba, rozdelená na dve časti, ktoré medzi sebou môžu anonymne komunikovať. V jednej časti izby je Číňan, ktorý posiela vety v činštinedo druhej časti izby. V druhej časti izby je Searle, ktorý nevie po čínsky ani slovo, ale má knihu pravidiel

napísanú v jazyku, ktorému Searle rozumie. Kniha pravidiel predstavuje návod, ako prepisovať čínske znaky na iné čínske znaky. Kniha vo svojej podstate simuluje inštrukcie, ako spracovať vstupné údaje a ako z nich vygenerovať údaje výstupné. Searle simuluje mechanizmus počítača, ktorý s týmito inštrukciami narába. Takže, Searle dostane vetu po čínsky, prepíše túto vetu podľa pravidiel v knihe na inú čínsku vetu. Vetu pošle späť bez toho, aby mal najmeňšiu predstavu o tom, čo sa ho pýtajú a aká je odpoveď.

Číňan sa výborne porozpráva a Searle si utrie pot zo svojho čela. Searle a kniha pravidiel vytvárajú dohromady systém, komunikujúci po čínsky bez toho, aby po čínsky rozumel. Takýto systém by zrejme napriek všetkému prešiel Turingovým testom. A na tomto sú založené dnešné počítače. Na rozdiel od ľudí použitý systém nerozumie obsahu a významu symbolov, s ktorými pracuje. Program nemá myseľ a inteligenciu ako ľudia. Len sa správa, ako by ho mali.[8]

Kapitola 4

Rozpoznávanie reči

V danej fáze programu sú dostupné hlasové vstupy užívateľa a úlohou tejto časti je spracovanie týchto údajov a ich prevedenie do formy digitálneho textu. Danej problematike sa vo všeobecnosti venuje vedná disciplína rozpoznávanie reči.

Rozpoznávanie reči (Speech recognition) v počítačovej terminológii znamená proces prekladu signálu ľudskej reči do digitálnej informácie (napr. textové dáta).

4.1 História vývoja rozpoznávania reči

Prvé zariadenie na rozpoznávanie reči vzniklo v roku 1952, ktoré mohlo rozoznať vyslovené čísla. V roku 1964 na veľtrhu počítačových technológií v meste New York bolo predstavené zariadenie IBM Shoebox.

Komerčné programy na rozpoznávanie reči sa začínajú objavovať začiatkom 90. rokov minulého storočia. Zvyčajne ich používajú ľudia, ktorí, následkom zranenia ruky, nie sú schopní dlhodobo používať klávesnicu. Tieto programy (napr. Dragon NaturallySpeaking, VoiceNavigator) prekladajú hlas používateľa do textu, čím napomáhajú odľahčiť jeho únavu. Spoľahlivosť takýchto programov, zrejme nie je vysoká, ale časom sa postupne zlepšuje.

Zvýšenie výpočtových výkonov mobilných zariadení umožnilo vytvoriť programy s funkciou rozpoznávania reči aj pre nich. Určite stojí za spomenutie aplikácia Microsoft Voice

Command, ktorá umožňuje pracovať s mnohými inými aplikáciami prostredníctvom hlasu. Napríklad sa dá zapnúť hudba, alebo vytvoriť nový dokument.

Popularita použitia rozpoznávania reči stále rastie v rôznych oblastiach podnikania. Napríklad lekár v ambulancii môže vysloviť diagnózy, ktoré budú okamžite zadané do elektronického preukazu. Alebo iný príklad. Asi každý aspoň raz za život túžil hlasom zhasnúť svetlá alebo otvoriť okno. V poslednej dobe sa v interaktívnych aplikáciach telefónu začali používať softvéry automatického rozpoznávania reči a syntézy reči. V tomto prípade sa komunikácia s hlasovým portálom stáva prirodzenejšia, pretože voľba môže byť vykonaná nielen prostredníctvom oznamovacieho tónu, ale aj pomocou hlasových príkazov.

4.2 Metódy a algoritmy rozpoznávanie reči

Metódy a algoritmy, ktoré sa používajú pri tvorbe softvérov na rozpoznávanie reči sa dajú rozdeliť na veľke triedy:

Klasifikácia metód rozpoznávania reči na základe porovnávanie so šablonou.

- Dynamické programovanie - časové dynamické algoritmy (Dynamic Time Warping)

Kontextovo-orientovaná klasifikácia. Pri jej realizácii z reči vyčnievajú oddelené slovné prvky - fonémy a allofóny, ktoré sa neskôr spájajú do slabík a morfém.

- Skryté Markovove Modely (Hidden Markov Model)
- Neurónové siete (Neural networks)

4.2.1 Dynamické programovanie - časové dynamické algoritmy

Dynamická časová deformácia (Dynamic Time Warping) je algoritmus, umožňujúci vyhľadať optimálnu zhodu medzi časovými postupnosťami. Tento algoritmus bol prvýkrát predstavený práve v rozpoznávaní reči, kde bol použítý na určenie toho, ako dva rečové signály predstavujú jednu a tú istú pôvodnú frázu.[9]

Časové algoritmy je rozšírený druh údajov, stretavajúci sa, fakticky, v každej vedeckej oblasti. Hlavnou úlohou takýchto algoritmov je porovnanie dvoch postupností. Na vyčíslenie výchylky je často dostatočné jednoduché meranie vzdialenosti medzi komponentami dvoch postupností (Euklidova vzdialenosť). Často však majú takéto dve postupnosti približne rovnaké vlastnosti, až na os X. Na určenie podobnosti takých postupností sa zavádza "deformácia"časovej osi jednej (alebo oboch) postupností.

4.2.2 Skryté Markovove Modely

Skrytý Markovovy model je štatistický model, ktorý simuluje činnosť procesu, podobného Markovemu procesu s neznámymi parametrami a úlohou procesu je odhadnutie neznámych parametrov na základe pozorovateľných.

Markovov proces je náhodný proces, vývoj ktorého, po zadaní ľubovoľnej hodnoty časového parametra t, nie je závislý od vývoja, ktorý predchádzal času t za podmienky, že hodnota procesu v danný moment je známa. Čiže "budúcnosť"procesu nezávisí od "minulosti", ak poznáme "súčasnosť".

Prvé poznatky o takychto modeloch publikoval Baum v 60-tých rokoch a neskôr v 70-tých sa poprvýkrát použili pri rozpoznávani reči. Od druhej polovice 80-tých sa používajú pri analýze biologických postupností, ako napríklad DNA.

4.2.3 Neurónové siete

Umelá neurónová sieť - matematický model, taktiež jeho softvérove a hardvérové prevedenie, založené na princípe organizácie a činnosti biologických neurónových sietí - sietí nervových buniek živeho organizmu. Tento pojem vznikol pri skúmaní procesov, prebiehajúcich v mozgu a taktiež pri pokuse o modelovanie týchto procesov. Poprvýkrát tento pokus uskutočnili W. McCulloch a W. Pitts. Po vytvorení algoritmov na analýzu tieto modely začali používať v praktických cieľoch: pri ulohách prognozovania, rozpoznávaní reči, alebo riadenia atď.

Umelé neurónové siete predstavujú systém prepojených a vzájomne pôsobiacich nekomplikovaných procesorov (umelých neurónov). Tieto procesory su zvyčajne dosť jednoduché

(hlavne v porovnaní s procesorami, ktore sa používajú v počítačoch). Každý procesor podobnej sieti ma dočinenia len so signálmi, ktoré periodicky dostáva a so signálmi, ktoré periodicky odosiela iným procesorom. Ale vzhľadom k tomu, že takéto lokálne jednoduché procesorz sú zapojené do primerane veľkej sieti s riadenou interakciou, spolu dokážu vykonávať celkom zložité úlohy. Tak je tomu aj pri hlasovom rozpoznávani.

4.3 Google Speech Recognition API

Pri voľbe rozhrania, shopného realizovať rozpoznávanie reči sa hlavne zohľadňovala jeho dostupnosť. Jedným z najpoužívanejších je rozhranie Google Speech Recognition API, ktoré je vyvíjané spoločnosťou Google ako voľne dostupný projekt.

Google Speech Recognition API je programové rozhranie poskytované spoločnosťou Google, ktoré pozostáva z databázy, umiestnenej na internete, vďaka čomu sa neustále obnovuje. Podporuje rozpoznanie hlasu 40 jazykov, jedným z ktorých je aj slovenčina. Spoľahlivosť správneho rozpoznávania a zatriedenia sa výrazne líši od tej, ktorá sa vyskytovala začiatkom 90. rokov, čo taktiež poslúžilo dôvodom k rozhodnutiu zamerať sa práve na toto rozhranie.

Tento modul pozostáva z kombinácií časových dynamických algoritmov a taktiež neurónových sietí.

4.3.1 Dostupnosť programového rozhrania

Používanie, resp. vyskúšanie tohto rozhrania je dostupné pre všetkých na stranke

https://www.google.com/intl/en/chrome/demos/speech.html

4.3.2 Inštalácia a požiadavky

Najednoduchšia cesta pre inštaláciu modulu je použitie príkazu:

pip install SpeechRecognition #príkaz v cmd-line/termináli

V opačnom prípade sa dá zdrojová distribúcia stiahnúť na stránke:

https://pypi.python.org/pypi/SpeechRecognition/

Po stiahnutí treba zložku extrahovať a spustiť inštaláciu súboru *setup.py*

Požiadavky:

- Python 2.6, 2.7, alebo Python 3.3+
- V prípade ak plánujete používať vstup z mikrofónu, tak je potrebné naištalovať *PyAudio*, v opačnom pripade bude program hlásiť chybu (dá sa nainštalovať jednoduchým príkazom v termináli: *sudo apt-get install python-pyaudio* alebo *python3-pyaudio*)
- FLAC kodér na kodovanie audio údajov a posielanie do API. (je predinštalovaný pre systemy založené na x-86 Windows/Linux)

4.3.3 Implementácia v Python

V Python je Google Speech Recognition API používané prostredníctvom modulu SpeechRecognition, ktorý vyvoláva url adresu stránky s umiestneným rozhraním, čiže použitie modulu bez prístupu na internet nie je možné.

Knižnica sa v Python privoláva pomocou príkazu

import ***speech_recognition*** *as* ***sr***

a v tomto prípade sa ukláda do premennej **sr**

4.3.4 Príklady použitia

Listing 4.1: Rozpoznávanie reči zo vstupu mikrofónu

```
import speech_recognition as sr
r = sr.Recognizer()
with sr.Microphone() as source:
        audio = r.listen(source)
try:
    print("You said" + r.recognize(audio))
except LookupError:
    print("Could not understand audio")
```

Listing 4.2: Prepísanie WAV audio súboru

```
import speech_recognition as sr
r = sr.Recognizer()
with sr.WavFile(test.wav) as source:
        audio = r.listen(source)
try:
    print("You said" + r.recognize(audio))
except LookupError:
    print("Could not understand audio")
```

V prípade kódu z 4.1 sa ako vstupný zdroj používa mikrofón a v kóde 4.2 je zdrojom súbor WAV. Vypočutie vstupu a priradenie do premennej **audio** sa uskutočňuje príkazom `audio = r.listen(source)`. Po rozpoznáni reči výstupom na obrazovku bude vyslovená veta a ak sa modulu nepodarí reč rozpoznať, vystupom bude veta *Could not understand audio*.

4.4 Experimentálna validácia

Pri aplikácii modulu SpeechRecognition sa musia najprv zvoliť vhodné nastavenia na vstupy, ktoré by zapadali do štandardu práve pre program. Jedným z takých je parameter **energy_threshold**, ktorý zodpovedá za minimálnu energiu audia. Pri prekročení dannej energie modul považuje hlasový vstup za reč, ktorú ma prekladat do digitálnej formy. Ďalším významným parametrom je **pause_threshold**, ktorý predstavuje čas, vyjadrený v sekundách. Je to maximálny čas, po prekročení ktorého si systém myslí, že je vstupná veta ukončená. Čiže to nemôže byť príliš krátke, aby uživateľ mohol premýšľať počas rozhovoru, ale zas ani príliš dlhé, aby uživateľa neunavilo čakanie na odpoveď. Taktiež je dosť dôležitým prvkom zvoliť si jazyk a použiť validný kľuč na rozlúštenie jazyka. Ako bolo spomenuté, danné rozhranie má schopnosť rozoznať až 40 jazykov, vrátane slovenčiny, ale napriek tomu sa pri tvorbe programu použije americkú angličtinu. Dôvodom tomu je to, že angličtina je jednoduchšia z hľadiska gramatiky a syntaxu (hlavne čo sa týka rozboru viet).

Vyššie opísané parametre sú dostupné na modifikáciu v tele hlavnej funkcie **__init__.py**, ktorá sa nachádza zložkách knižnice SpeechRecognition. Pri nastavení týchto parametrov boli zvolené následujúce modifikácie.

language = ën-US", **key** = ÄIzaSyBOti4mM-6x9WDnZIjIeyEU21OpBXqWBgw"
self.**energy_threshold** = 300
self.**dynamic_energy_threshold** = True
self.**dynamic_energy_adjustment_damping** = 0.15
self.**dynamic_energy_ratio** = 1.5
self.**pause_threshold** = 2.5
self.**quiet_duration** = 0

4.4.1 Opis zdrojového kódu

Oporným telesom pri použíti modulu SpeechRecognition v tele programu slúží dokumentácia Python [10].

Vzhľadom k tomu, že nie je potrebná komplikovaná funkcia na rozpoznávania hlasu (stači obyčajný program na prepísanie reči), do tela programu môže byť použitý už existujúci príklad v dokumentácii s meňšími úpravami.

Listing 4.3: Funkcia na prepísanie hlasovej reči do digitálnej formy

```
def recognition():        #speech input
    global r,audio
    r = st.Recognizer()
    with sr.Microphone() as source:
        audio = r.listen(source)
    try:
            print(r.recognize(audio))
        except LookupError:
            print("Sorry, could you repeat, please?")
            recognition()
```

Hlasové rozpoznávanie je definované a deklarované ako funkcia s názvom **recognition()**. Tá funkcia má v princípe takú istú úlohu, ako v prvom príklade na Google Speech Recognition API s tým, že ak nastane chyba charakteru "nerozpoznateľná reč", potom funkcia vyvolá samú seba a požiada uživateľa o opakovanie vety. Čiže výstupom tejto funkcie pri úspešnom preklade je **r.recognize(audio)**, čo predstavuje vetu, ktorú uživateľ vyslovil. Neskôr sa tá veta prepisuje do reťazca, s ktorým pracuje ďalšia časť programu.

4.4.2 Spoľahlivosť

Ako bolo spomenuté vyššie, nie je možne používať modul SpeechRecognition bez prístupu na internet. Odhaliť však dannú chybu je veľmi obtiažne, vzhľadom k tomu, že sa funkcia začne zacyklovať a stále dookola žiadať uživateľa o opakovanie vety.

Záujímavým faktom pri preklade je, že úspešnosť rastie so zväčšujúcou sa vetou, pretože pri rozoberaní konkretných slov sa opiera o jej kontext. Čiže vety s jedným, alebo dvoma slovami majú častejšiu náchylnosť byť nesprávne preložené. To isté sa týka čísloviek. Pokiaľ číslovka je vyslovená bez podtextu k tomu, ktorý dáva zmysel, alebo sú vyslovené o samote, tak riziko nesprávneho prekladu je veľmi vysoké. Čísla od 0 po 10 sa rozpoznávajú perfektne, ale ako náhle to prekračuje hlanicu 200, 5000 a 10000 nadobúdajú chybu 3%, 7% a 45%.

Pri práci sa bude predpokladať, že hlasové vstupy od uživateľa nebudú veľmi jednoduché (jednočlenné), preto spoľahlivosť prekladu je postačujúca.

Kapitola 5

Textová analýza

Po úspešnom rozpoznaní reči sa spracovaná časť vstupu premieňa na digitálnu formu textu a postupuje do fázy jej analýzy. Vstupom do tejto sekcie sú reťazce reči, z ktorých sa analyzuje, či je požiadavka užívateľa zameraná na okamžitú zmenu teploty, alebo na budúcu obsadenosť miestnosti. Následne, po stanovení zámeru užívateľa je úlohou danej fázy správne vyextrahovať potrebné číselné, resp., časové údaje. Problematike tejto sekcii sa venuje vedná disciplína analýza textov.

Inteligentná analýza textov (text mining) je smer v oblasti umelej inteligencie, cieľom ktorého je získanie informácií zo zbierok textových súborov na základe uplatnenia efektívnych metód počítačového učenia a spracovania prirodzeného jazyka v praxi. Názov ïnteligentná analýza textov"súvisí s pojmom ïnteligentná analýza údajov"(data mining), čo vyjadruje podobnosť ich cieľov, postupov k spracovaniu informácie a oblastí použitia. Rozdiel sa prejavuje až v konečných metódach, a taktiež v tom, že analýza údajov ma dočinenia s úložiskami a databázami údajov, a nie s elektrónnymi knižnicami a štruktúrami textov.

5.1 História vývoja textovej analýzy

Výskumy manuálnej analýzy textu sa prvýkrát objavili v polovici 60. rokov, ale technologický pokrok umožnil sa intenzívne venovať tejto oblasti až počas posledného desaťročia. Textová analýza je interdisciplinárny odbor, ktorý spočíva vo vyhľadaní informácií, datovej analýze, počítačového učenia, štatistiky a počítacovej lingvistike. Vzhľadom k tomu,

že drvivá väčšina informácií (podľa spoločných odhadov viac ako 80%) je v súčasnosti uložená vo forme textu, textová analýza má veľkú komerčnú potenciálnu hodnotu. V poslednej dobe sa zvyšuje záujem o viacjazyčnú analýzu dát: schopnosť získať informácie skrz jazyky a klastery podobných parametrov z rôznych lingvistických zdrojov vzhľadom na ich význam.

Vývoj textovej analýzy vo svojej súčasnej podobe pochádza z preorientovania výskumu druhej polovici 90. rokov z algoritmickej podstaty do jej aplikácie, ako popísal Prof. Marti A. Hearst v novinárskom článku Untangling Text Data Mining:

Za takmer desať rokov počítačová lingvistická komunita študovalá veľké textové zbierky ako prostriedok na využitie výroby lepších algoritmov textovej analýzy. V tomto článku som sa pokúsil navrhnúť nový dôraz: využitie veľkých internetových textových zbierok na objavenie nových faktov a trendov, týkajúcich sa samotného sveta. Domnievam sa, že na to, aby sme dosiahli pokrok nepotrebujeme dokonalú umelú inteligenciu analýzy textu; skôr, zmes výpočtovo-riadených a užívateľsky sprevádzaných analýz môže otvoriť dvere k novým vzrušujúcim výsledkom.[11]

5.2 Funkcia a úlohy textovej analýzy

Kľúčovými úlohami inteligentnej textovej analýzy sú: kategorizácia textov, extrakcia informácií a jej vyhľadavanie, spracovanie zmien v zbierkach textov, a taktiež vývoj prostriedkov pre poskytnutie informácií používateľovi.

Kategorizácia súborov sa zakláda na pridelení súborov zo zbierky k jednej, alebo viacerým skupinám (triedam, klasterom), ktoré sa medzi sebou podobajú (napr. podľa témy, alebo štylu). Kategorizácia sa môže uskutočňovať ako za účasti človeka, tak aj bez neho. V prvom prípade ide o tzv. klasifikácii súborov. Systém inteligentnej analýzy textov musí prideliť texty k už zadefinovaným (vhodným pre seba) triedam. Z hľadiska počítačového učenia je nevyhnutné realizovať učenie s vyučujúcim, na čo uživateľ musí poskytnúť systemu čo najviac tried, ale aj vzory súborov, patriacich do týchto tried.

Druhý prípad kategorizácie sa nazýva klasterizáciou súborov. Klaster (cluster - zhluk) je zlúčenie niekoľkých homogénnych prvkov, považujúce sa za samostatnú jednotku, ktorá indisponuje určitými vlastnosťami. V tomto prípade system musí sám určiť sadu klasterov,

podľa ktorých môžu byť texty rozdelené. V počítačovom učení sa danná úloha nazýva učeniem bez vyučujúceho. Tu uživateľ musí informovať systém, na koľko klasterov by chcel rozdeliť spracovanú zbierku (predpokláda sa, že procedúra výberu vlastností je už vložená do algoritmu programu).

5.3 Natural Language Toolkit

Pri voľbe prostriedku na analýzu textu sa hlavne zohľadňovalo množstvo informácii, s ktorými sa v tomto prostriedku dá narabať. Pre pohodlné narábanie so synonymami a vkladaním do databázy bola zvolená platforma NLTK.

Natural Language Toolkit (NLTK) je vedúca platforma na budovanie programov v jazyku Python na spracovanie údajov ľudskej reči. Poskytuje ľahko použiteľné rozhranie vo viac ako 50 korpusov a lexikálnych prostriedkov, ako napríklad WordNet, spolu s balíkom knižníc na spracovanie textu pre klasifikáciu, tokenizáciu, označenie, rozbor a semantické uvažovanie.

Vďaka praktickému manuálu, uvádzajúcemu základy programovania podľa kapitol v počitačovej lingvistike, NLTK je vhodný pre lingvistov, inženierov, študentov, pedagógov, výzkumníkov a pre podobné odvetvia. NLTK je dostupný pre Windows, Mac OS X a Linux ako otvorený pre verejnosť projekt.

NLTK prezvali "skvelým nástrojom na učenie a na prácu s ním, počítačovou lingvistikou, využívajúcu Python."[12]

5.3.1 Implementácia v Python

Na používanie danného modulu v programovacom jazyku Python nie je potrebné vykonať inštaláciu. Balíčky NLTK sú už predinštalované spolu s programom Python.

Privolanie modulu NLTK sa uskutočňuje pomozou príkazu

import **nltk**,

alebo triedením a vytiahnutím konkretného balíčku, napríklad:

from **nltk.corpus** *import* **wordnet**

5.3.2 Príklady použitia

Listing 5.1: Tokenizácia a označenie niektorého textu

```
>>> import nltk
>>> sentence = "At eight o'clock on Thursday morning
...Arthur didn't feel very good."
>>> tokens = nltk.word_tokenize(sentence)
>>> tokens
['At', 'eight', "o'clock", 'on', 'Thursday', 'morning',
'Arthur', 'did', "n't", 'feel', 'very', 'good'", '.']
>>> tagged = nltk.pos_tag(tokens)
>>> tagged[0:6]
[('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'), ('on', 'IN'),
('Thursday', 'NNP'), ('morning', 'NN')]
```

Listing 5.2: Identifikácia pomenovaných entít

```
>>> import nltk
>>> entities = nltk.chunk.ne_chunk(tagged)
>>> entities
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'JJ'),
           ('on', 'IN'), ('Thursday', 'NNP'), ('morning', 'NN'),
       Tree('PERSON', [('Arthur', 'NNP')]),
           ('did', 'VBD'), ("n't", 'RB'), ('feel', 'VB'),
           ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

5.4 TextBlob

Ako druhý prostriedok na spracovanie textových údajov sa zvolil TextBlob pre dobrú schopnosť používať označovanie reťazcov. TextBlob poskytuje jednoduché programovacie rozhranie na riešenie obyčajných úloh spracovania prirodzeného jazyka, ako je napríklad označenie časti textu, extrakcia podstatných mien z fráz, analýza sentimentu, klasifikácia, preklad, čiže podobné úlohy ako pri module NLTK.[13]

5.4.1 Inštalácia a implementácia

Knižnica je voľne dostupná na stiahnutie na oficiálnej stránke

http://textblob.readthedocs.org/en/dev/,

alebo vykonaním príkazov v terminali:

pip install -U textblob

python -m textblob.download_corpora

Privolanie knižnice sa uskutočňuje zadaním príkazu

import **textblob**,

alebo triedením a vytiahnutím konkretného balíčku, napríklad:

from **textblob** *import* **Word**

5.4.2 Príklady použitia

Listing 5.3: Gramatické opravovanie

```
>>> from textblob import Word
>>> w = Word('falibility')
>>> w.spellcheck()
[('fallibility', 1.0)]
```

Listing 5.4: Preklad

```
>>> en_blob = TextBlob(u"Simple is better than complex.")
>>> en_blob.translate(to="es")
TextBlob("Simple es mejor que complejo.")
```

5.5 Datetime

Modul datetime podporuje triedy na manipuláciu s datumami a časmi jednoduchým aj komplexným spôsobom. Kým je datum a čas podporený aritmitickými operáciami, implementácia sa sústredí na učinnú extrakciu atribútov na formatovanie a manipulovanie výstupov.[10]

5.5.1 Implementácia

Používanie daného modulu sa realizuje príkazom

import **datetime**,

alebo triedením a vytiahnutím konkretného balíčku, napríklad:

from **datetime** *import* **timedelta**

Taktiež požívanie modulu nepožaduje extérnu inštaláciu, pretože sú predinštalované spolu s Python.

5.5.2 Príklady použitia

Na následujúcom obrázku možeme vidieť jednoduché matematické operácie s knižnicou datetime. V danom prípade príkaz **timedelta** vyjadruje zmenu času, s ktorou by sa dalo narábať.

Listing 5.5: Jednoduché matematické operácie s dátumami

```
>>> from datetime import timedelta
>>> year = timedelta(days=365)
>>> another_year = timedelta(weeks=40, days=84, hours=23,
... minutes=50, seconds=600)  # dokopy 365 dni
>>> year.total_seconds()
31536000.0
>>> year == another_year
True
>>> ten_years = 10 * year
>>> ten_years, ten_years.days // 365
(datetime.timedelta(3650), 10)
>>> nine_years = ten_years - year
>>> nine_years, nine_years.days // 365
(datetime.timedelta(3285), 9)
>>> three_years = nine_years // 3;
>>> three_years, three_years.days // 365
(datetime.timedelta(1095), 3)
```

```
>>> abs(three_years - ten_years) == 2 * three_years + year
True
```

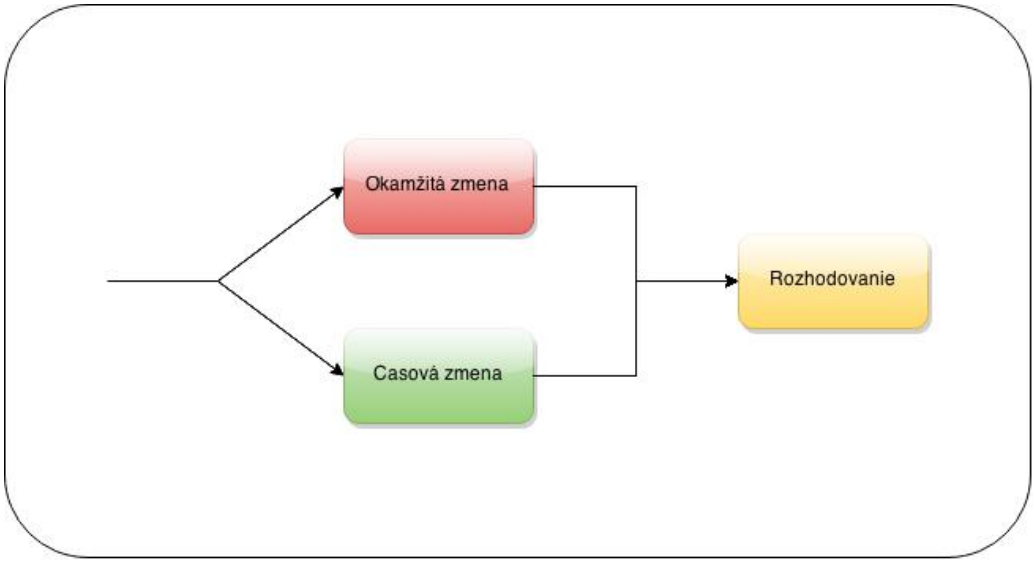
Listing 5.6: Jednoduché matematické operácie s dátumami

```
>>> import time
>>> from datetime import date
>>> today = date.today()
>>> today
datetime.date(2007, 12, 5)
>>> today == date.fromtimestamp(time.time())
True
>>> my_birthday = date(today.year, 6, 24)
>>> if my_birthday < today:
...     my_birthday = my_birthday.replace(year=today.year + 1)
>>> my_birthday
datetime.date(2008, 6, 24)
>>> time_to_birthday = abs(my_birthday - today)
>>> time_to_birthday.days
202
```

Na obrázku 5.6 vidíme zrojový kód, ktorý má vypočítať množstvo dní, ktoré ostavajú do narodenín v tomto roku, a ak náhodou už v tomto roku boli, potom má vyrátať až do budúceho roku.

5.6 Experimentálna validácia

Na to, aby analýza uloženého textu prebiehala správne, musí sa najprv vedieť orientovať, čo vstupná veta od uživateľa požaduje. Preto je vhodné rozštiepiť blok texová analýza na ďalšie tri podradené bloky podľa následujúcej schémy.



Obr. 5.1: Zjednodušenie bloku textová analýza

Vstupná veta prichádza do blokov Ökamžitá zmenaä "Časová zmena", kde program podľa syntézy jednotlivých členov vety zisťije informácie a zároveň podľa toho priraďuje rôzne hodnoty do lokálnych premenných. Potom, ako náhle prebehne proces priraďovania, signál poputuje do bloku "Rozhodovanie", v ktorom umelá inteligencia porovná množstvo priradených premenných medzi jednotlivými blokmi a na základe toho sa rozhodne, na akú zmenú má uživateľ požiadavku.

5.6.1 Okamžitá zmena

Podstata okamžitej zmeny spočíva v požiadavke konktrétnej teploty priamou, alebo nepriamou metódou. Pri prvej pôjde o konkrétne číslo, o ktoré, alebo na ktoré uživateľ chce zmeniť teplotu v miestnosti. V druhom prípade sa bude skrývať informácia o stave človeka a taktiež údaje, ktoré by mali charakterizovať intenzitu daného stavu. Napríklad veta "Je mi trochu zima"by mala programu oznámiť, že uživateľ žiada o "mierne"zvyšenie teploty. Na začiatku si program zistí, do ktorého smeru sa ma uskutočniť zmena, neskôr, keď to bude známe, začne analýzovať vetu na prítomnosť prísloviek, ktoré by mu umožnili uložiť konkrétne číslo (zmenu teploty) do premennej.

Pre vyhľadavanie stavu a jeho intenzite sa použijú kľučové slova, ako napríklad zima, teplo, trochu, veľa, ktoré sa vďaka modulu NLTK neskôr obohatia o celú databázu syno-

nymov a vďaka modulu TextBlob budú môcť porovnať so vstupnou vetou.

Listing 5.7: Porovnanie slov zo vstupu so slovom **cold**

```
from textblob import TextBlob
from nltk.corpus import wordnet as wn
a = TextBlob('I am getting cold')
for tag in a.tags:
        for s in wn.synsets('cold'):
                if tag[0] in s.lemma_names():
                        posch() #pozitivna zmena
```

Tento príklad zdrojového kódu po priradení vstupnej vety do premennej **a** ju rozčlení na jednotlivé slova a porovnáva ich so všetkými synonymami slova **cold**. Po úspešnom prebehnutí porovnania sa vykoná funkcia **posch()**, ktorá upozorní program na to, že vykonávana zmena je pozitívna.

Intenzita stavu bola zvolená na stupnici od 0.5 po 2.0 a pre jednotlivé prívlastky následujúce hodnoty:

- little = 0.5
- few = 1
- very = 2

Tieto 3 slová samozrejme nereprezentujú celú databázu slov. Ku slovu **little** napríklad tiež zapadá **slightly**, alebo k slovu **very** patrí **extremely** atď.

Listing 5.8: Analýza vstupnej vety

```
from textblob import TextBlob
from nltk.corpus import wordnet as wn
a = TextBlob('I am getting a little bit cold')
for tag in a.tags:
        for s in wn.synsets('cold'):
                if tag[0] in s.lemma_names():
                        posch() #pozitivna zmena
```

```
for tag in a.tags:
        for s in wn.synsets('little'):
                if tag[0] in s.lemma_names():
                        c = 0.5 #hodnota zmeny teploty
```

Na obrázku 5.8 už môžeme sledovať zdrojový kód, úlohou ktorého je nielen zistiť smer zmeny teploty, ale aj jej hodnotu. Potom, ako program vyhľadá slovo **little** vo vstupnej vete, priradí do premennej **c** hodnotu 0.5 a tým pádom aj vie pri rozhodovaní, o akú zmenu ide.

5.6.2 Časová zmena

Podstata časovej zmeny spočíva v tom, že uživateľ vo vstupnej vete udáva časový údaj o obsadenosti miestnosti. Čiže vstupná veta by sa mala skladať z údajov, v aký deň miestnosť nebude obsadená a taktiež v časový interval toho dňa.

Pri práci s datovými údajmi sa bude používať modul **Datetime** a pri vyhľadavaní týchto údajov vo vtupnej vete sa použije modul TextBlob, ktorý bude vetu skúmať na prítomnosť časových údajov, čo predstavuje buď deň týždňa (pondelok, utorok, ...), alebo konkrétny dátum (24. Máj) a taktiež interval času toho dňa.

Listing 5.9: Analýza časových údajov

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
from array import array
a = TextBlob('I am going to the cinema on Monday from 18 to 21')
ar = array('b')
for tag in a.tags:
        if tag[0] == 'Monday':
                day = 1
        if tag[1] == 'CD':
                ar.append(int(tag[0]))
```

Na obrázku 5.9 je uvedený príklad analýzy časových údajov. V tomto prípade sa veta skúma na prítomnosť slova **Monday** a v momente, keď sa vyhľadá, priradí do premmenej **day** číslo 1. Ďalším objektom skúmania sú číselné údaje času. Číslovky cez knižnicu **TextBlob** sú značené ako reťazec **CD**, preto program najprv skúma vetu na prítomnosť čísloviek.

Neskôr sa používa knižnica **array**, ktorá všetky číslovky vo vete prepíše do jednorozmerného poľa.

Knižnica **array** nepotrebuje extérnu inštaláciu a je predinštalovaná spolu s Python. Jej privolanie sa uskutočňuje pomocou príkazu

from **array** *import* **array**

Listing 5.10: Analýza časových údajov (podľa dátumu)

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
a = TextBlob('I will not be at home on 21st of July')
for tag in a.tags:
        if tag[0] == '21st':
                day = 21
        if tag[0] == 'July':
                month = 7
```

Ďalším príkladom je analýza časových údajov, ktorá sa zameriava na konkrétny dátum. Dátumy sa reprezentujú mesiacom, pri objavení ktorého sa mesiac prepíše do premennej **month**, ako jeho poradie. Pre poradia dňa platí to isté a prepisuje sa do premennej **dd**.

Pri rozhodovaní, ktorú zmenu predstavovala vstupná veta sa teda program bude opierať o vyššie uvedené premenné.

Kapitola 6

Rozhodovanie a realizácia príkazu. Experimenálna validácia

Po textovej analýze sú dostupné priradené premenné, ktoré slúžia ako vstupné signály do danej sekcii. Úlohou tejto fázy je spracovanie a správne prevedenie týchto vstupov na výstupy vo forme vypísania na obrazovku teplotného kalendára a taktiež hlasového výstupu. Pred tým ale má časť **Rozhodovanie o postačujúcosti údajov** usúdiť, či je vstupných signálov dosť na ich prevedenie.

Môže sa totížto stať, že požiadavka uživateľa je zameraná na okamžitu zmenu teploty, avšak vstupných údajov nie je dosť a tým pádom nie všetkým premenným bude priradená svoja hodnota. Táto situácia môže nastať pri nekonkrétnom vstupe od uživateľa, alebo v dôsledku nekomplexnosti zdrojového kódu. Napriek tomu, že databáza údajov je veľká, predsa ide o počítač, ktorý nemôže pochopiť prirodzenú logiku človeka.

Preto, úlohou bloku **Rozhodovanie o postačujúcosti údajov** je zistiť, či sú všetky premenné pre danú zmenu zadefinované.

6.1 Podmienka je splnená. Realizácia príkazu

V prípade splnenia všetkých požiadaviek posiela blok signály na spracovanie daných premených do bloku **Realizácia príkazu**. Napríklad z obrázku 5.8 sa do premennej **c** uložila hodnota 0.5 a zároveň sa vykonala funkcia **posch()**. Pokračovanie toho kódu by bolo následujúce.

Listing 6.1: Vykonanie príkazu na okamžitú zmenu teploty

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
def negch():              #negativna zmena
        global b
        b = -c
        execution()
def posch():              #pozitivna zmena
        global b
        b = c
        execution()
def execution():                  #okamzita zmena teploty
        global ntemp
        ntemp = temp + b
        print('Temperature was changed to', ntemp)
c = 0.5
temp = 21
posch()
# >>> Temperature was changed to 21.5
```

Premenná **temp** z obrázku 6.1 predstavuje súčasnú teplotu v miestnosti a **ntemp** novú teplotu po realizácii príkazu. Po vykonaní funkcie **posch()** sa priradí hodnota 0.5 do premennej **b**, ktorá sa následne, vo funkcii **execution** sčíta so súčasnou teplotou. Výstupom programu teda bude veta *Temerature was changed to 21.5*.

Ako ďalší príklad sa môže rozoberať zdrojový kód z obrázku 5.10. V tomto prípade sa do premennej **dd** uložila hodnota 21 a do premennej **month** hodnota 7. Po takejto analýze sa vyvolá funkcia **set2()**, ktorá vyzerá takto.

Listing 6.2: Vykonanie príkazu na časovú zmenu teploty (podľa dátumu)

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
def set2():                #sorting by date
        global chdate
        today = date.today()
        nd = date(today.year, month, dd)
        chdate = nd.strftime('%A %d. %B')
        print('Heating is turning off on', chdate)
dd = 21
month = 7
set2()
#>>> Heating is turning off on Tuesday 21. July
```

Funkcia **set2()** prepisuje preddefinované premenné do premennej **nd**, ktorá predstavuje deň, kedy má nastať vypnutie kúrenia. Po vykonaní meňších úprav vo formatovaní dátumu, výstupom tejto funkcie je veta *Heating is turning off on Tuesday 21. July.*

Listing 6.3: Program na generovanie výstupu vo forme teplotného kalendára

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
from array import array
def set1():          #sorting by day
    global chdate
    k = 0
    if day <= date.today().weekday()+1:
        k = 6
    else:
```

```
            k = -1
        delta = timedelta(days = day - date.today().weekday()+k)
        change = date.today() + delta
        chdate = change.strftime('%A %d. %B')
        tab(ar[0],ar[1])
def tab(x,y):
        print 'Temperature calendar for', chdate+':\n
... Time || Temperature'
        for i in range(0,24):
            if i >= x and i <= y:
                print i,' || 18'       #vypnutie kurenia
            else:
                print i,' || 21'      #standardna teplota v miestnosti
a = TextBlob('I am going to the cinema on Monday from 18 to 21')
ar = array('b')
for tag in a.tags:
        if tag[0] == 'Monday':
            day = 1
        if tag[1] == 'CD':
            ar.append(int(tag[0]))
set1()
```

Posledný zobrazený kód je pokračovaním kódu číslo 5.9 a zároveň aj príkladom výstupu v podobe teplotného kalendára. Tento kalendár je charakterizovaný časom od 0. po 24. hodinu a ku každej hodine je priradená príslušná teplota. Pri bežnom kúrení je teplota miestnosti 21 a pri vypnutom je 18. Po analýze vety sa spúšťa funkcia **set1()**, ktorá matematicky rozratáva množstvo dní, koľko ostáva do budúceho pondelka, ukladá nový dátum do premennej **chdate** vo formatovanej podobe a vyvoláva funkciu **tab(x,y)**. Tá, následne rozpisuje teplotný kalendár pre zvolený deň. Výstupom takéhoto kódu (s dnešným dátumom Sobota, 23. Mája) je:

Listing 6.4: Výstup z programu číslo 6.3

```
Temperature calendar for Monday 25. May:
Time || Temperature
```

```
0   || 21
1   || 21
2   || 21
3   || 21
4   || 21
5   || 21
6   || 21
7   || 21
8   || 21
9   || 21
10   || 21
11   || 21
12   || 21
13   || 21
14   || 21
15   || 21
16   || 21
17   || 21
18   || 18
19   || 18
20   || 18
21   || 18
22   || 21
23   || 21
set1()
```

6.2 Podmienka nie je splnená

V prípade, ak podmienka nie je splnená, musí program presne vedieť, aké premenné chýbajú na realizáciu príkazu. Pravdivosť splnenia podmienky sa skúša funkciou **try/except**. Napríklad:

Listing 6.5: Pokus o vykonanie funkcie **set2()**

```
from textblob import TextBlob
from datetime import timedelta
from datetime import date
def set2():                    #sorting by date
        global chdate
        today = date.today()
        nd = date(today.year, month, dd)
        chdate = nd.strftime('%A %d. %B')
        print('Heating is turning off on', chdate)
month = 7
try:
        set2()
except:
        print('It is not possible to determine the date of
...execution')
# >>> It is not possible to determine the date of execution
```

V prípade obrázku 6.5 sa program pokúša privolať funkciu **set2()**, ale nemá zadefinovanú premennú **dd**, preto výstupom toho programu je veta *It is not possible to determine the date of execution.*

Kapitola 7

Syntéza reči

Daná fáza programu slúži na hlasovú komunikáciu s užívateľom. Na to má 2 hlavné dôvody. V prvom prípade, okrem už stanovených foriem výstupu, sa realizuje ako ďalšia alternatíva, v druhom prípade slúži na spätnú komunikáciu s užívateľom, ak podmienka nebola splnená. V oboch prípadoch však ide o premenu digitálnej formy textu do zvukovej podoby. Problematikou daného odvetvia sa zaoberá vedná disciplína syntéza reči, ktorá je zároveň aj opakom, už rozobranéj sekcie rozpoznávanie reči.

Syntéza reči je umelá inteligencia, úlohou ktorej je produkcia ľudskej reči. Počítač, ktorý sa používa na tieto účely sa nazýva syntetizér reči. Najčastejšie sa syntéza reči používa pre tzv. system text-to-Speech (TTS), úlohou ktorého je translácia digitálneho textu do zvukovej formy. Najznámejšia osobnosť, používajúca syntézu reči na komunikáciu je Stephen Hawking

7.1 História vývoja syntézy reči

Prvé syntetizéry reči zneli celom neprirodzene a často nebolo rozumieť frázam, ktoré vytvaral. Avšak kvalita tvorenej reči sa neutále zlepšovala a niekedy, reč, ktorú generuje syntetizér je ťažké odlíšiť od reálnej ľudskej. Napriek všetkým úspechom elektronických syntetizérov reči, výzkumy v oblasti tvorby mechanických syntetizérov sa stále vedú. Napríklad japonskí vedzi z Takanishi Laboratory Waseda University vyvíjajú antropomorfický model hovoriaceho robota. Model Waseda Talker No.5 - má balíček všetkých hlasových inštrumentov: pľúca, hrtaň, jazyk, zuby, ústa atď . Dokopy všetky tieto orgány

majú 18 stupňov voľnosti.

7.2 Pyttsx

Python text-to-speech x-platform je balíček v **Python**, podporujúci syntetizér obyčajnej reči pre Mac OS X, Linux a Windows.

7.2.1 Inštalácia a implementácia

Inštalácia daného balíčka sa realizuje pomocou príkazu:

sudo pip install pyttsx (pre Linux a OSX)

pip install pyttsx (pre Windows)

Privolať modul **pyttsx** je možne prostredníctvom príkazu

import **pyttsx**

7.2.2 Príklady použitia

Listing 7.1: Syntéza bežnej reči

```
import pyttsx
engine = pyttsx.init()
engine.say('Sally sells seashells by the seashore.')
engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

Na obrázku 7.1 sa vykonáva syntéza bežnej reči spúšťacim príkazom **engine.runAndWait()**.

Listing 7.2: Syntéza reči so zmenou hlasu

```
import pyttsx
engine = pyttsx.init()
voices = engine.getProperty('voices')
for voice in voices:
    engine.setProperty('voice', voice.id)
    engine.say('The quick brown fox jumped over the lazy dog.')
engine.runAndWait()
```

V tomto prípade premenná **voices** predstavuje celú databázu prízvukov rôznych jazykov. Takže výstupom programu je syntéza vety vo všetkých prípadoch.

7.2.3 Experimentálna validácia

Ako aj pri rozpoznávaní reči, tak aj pri jej syntéze je dostačujúcim použiť základný zrojový kód s meňšími úpravami a zapísať to forme funkcie, vstupom do ktorej bude veta, ktorá sa má syntetizovať.

Listing 7.3: Funkcia na syntézu vstupnej vety

```
import pyttsx
def synt(sentence):
    engine = pyttsx.init()
    engine.say(sentence)
    engine.runAndWait()
synt('hello world')
```

Okrem iného, ako už bolo spomenuté, že sa daný modul využíva na spätnú komunikáciu s užívateľom. Tou sa program uberá potom, ako usúdi, že získanej informácie nebolo dosť, alebo nebola identifikovateľná.

Listing 7.4: Neúspešné odhadnutie teplotnej zmeny

```
import pyttsx
def recognition():        #speech input
    global r,audio
    r = st.Recognizer()
    with sr.Microphone() as source:
        audio = r.listen(source)
    try:
            print(r.recognize(audio))
        except LookupError:
            print("Sorry, could you repeat, please?")
            recognition()
def synt(sentence):
        engine = pyttsx.init()
```

```
        engine.setProperty('rate',120)
        engine.say(sentence)
        engine.runAndWait()
def posch():                #pozitivna zmena
        global b
        b = c
        execution()
def execution():                    #okamzita zmena teploty
        global ntemp
        ntemp = temp + b
        print('Temperature was changed to', ntemp)
for tag in a.tags:
        for s in wn.synsets('cold'):
                if tag[0] in s.lemma_names():
                        try:
                                posch()
                        except UndefinedChange:
                                synt('It is not possible to define
... temperature change. Please specify the change.')
                                recognition()
a = 'I am getting cold'
```

Vyššie uvedený kód je typickým príkladom toho, ako sa môže uberať program. Vstupná veta (*I am getting cold*) je príliš nekonkrétna na to, aby sa po jej analýze priradili príslušné premenné. Potom, ako program zistí, že mu chýba premenná **c**, vyvolá funkciu **synt(sentence)** na spätnú komunikácu a oznamuje užívateľovi o nedostatku údajov, následne vyvoláva funkciu **recognition()** a opätovne čaká na hlasový vstup.

Kapitola 8

Záver

V práci sme vývijali užívateľské rozhranie pre inteligentný termostat, ktoré sa opieralo o hlasovú komunikáciu. V cieľoch práce sme rozdelili príslušný program na 5 do seba zapadajúcich častí, ktoré nemohli fungovať jeden bez druhého. Každej jednotlivej časti sa venovala kapitola, ktorá predstavovala teoretický úvod do problematiky, historický vývoj daných vedeckých disciplín a taktiež spôsoby interpretácie daných odvetví v programovacom jazyku python. Pre jednotlivé spôsoby interpretácie som opísal požiadavky na použitie daných modulov a taktiež ich inštaláciu a implementáciu. Kapitoly sa ukončovali experimentálnou validáciou, ktorú sprevádzal vybraný zdrojový kód programu, predstavujúci vhodné priklady na opis príslušných častí.

Na spracovanie vstupných hlasových údajov a ich prepísanie do digitálnej formy sa v programe použil modul **Google Speech Recognition API**. Vďaka vhodnej kombinácii modulov na analýzu textu **TextBlob** a **NLTK** sa popri vývoji programu vytvorila databázu slov, ktoré pokývajú vyše 50 možných vyjadrení o teplotnom stave užívateľa. Analýza požiadavky užívateľa na zmenu teploty, resp. budúcu obsadenosť miestnosti sa opierala o skúmanie vstupu na obe možnosti a neskôršom priradení na základe porovnánia, koľko informácie sa prepísalo do jednotlivej možnosti. Po priradení k jednej z možnosti požiadavky užívateľa nasledovala výstupná časť programu, ktorá pozostávala z údaja o okamžitej zmene teploty výpisom na obrazovku, alebo výpisom teplotného kalendára pre príslušný deň. Okrem toho výstupnú časť programu predstavoval aj hlasový výstup, ktorý sa v programe realizoval prostredníctvom použitia modulu na syntézu reči **Pyttsx**.

Vytvorené rozhranie je uľahčené v porovnaní s použitím manuálnych termostatov, pretože, ako vyplýva z práce spôsob interakcie užívateľa s termostatom pomocou hlasovej komunikácie je nielen možný, ale aj funkčný, čo zaručuje perspektívu vo vývoji daného zariadenia. Negatívnym dôsledkom daného rozhrania je však občasná nepresnosť v spracovaní vstupov, resp. nepostačujúca databáza pri analýze textov. Pri lepšom doladení daných nedostatkov po aplikácii na extérne použitie by toto rozhranie mohlo byť porovnateľné so súčasnými inteligentnými termostatmi.

Literatúra

[1] McCarthy, J. 2007. What Is Artificial Intelligence?.

[2] Russell, S.J. - Norvig, P. 2003. Artificial Intelligence: A Modern Approach.

[3] McCorduck, P. 2004. Machines Who Think.

[4] Asimov, I. 1950. I, Robot.

[5] Turing, A. 1950. Computing machinery and intelligence.

[6] Captcha: Carnegie Mellon University. 2000-2010. Telling Humans and Computers Apart Automatically.

[7] Searle, J. 1980. Minds, Brains, and Programs. Behavioral and Brain Sciences 3

[8] Kvasnička, V. - Pospíchal, J. - Kozák, Š. - Návrat, P. - Paroulek, P. 2009. Umelá inteligencia a kognitívna veda I

[9] Al-Naymat, G. - Chawla, S. - Taheri, J. 2012. Sparse DTW: A novel approach to speed up Dynamic Time Warping.

[10] Python Software Foundation. 1990-2015.

[11] Hearst, M. A. 1999. Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics.

[12] NLTK 3.0 documentation. 2015

[13] TextBlob: Simplified Text Processing. 2014

Príloha

CD nosič so súborom main.py, obsahujúcim zdrojový kód programu.