Houser, identificação do usuário por meio de padrões vocais

Eduardo C. Jaureguy[1], **Sylvio A. Garcia Vieira**[1]

[1]Ciência da Computação – Universidade Franciscana (UFN)
CEP 97.010-032 – Santa Maria – RS – Brazil

cerutti.business@gmail.com, sylviovieira@gmail.com

Abstract. *In the last few decades people have been searching for comfort, practicality and acesscibility at the home or professional daily life. Therefore new applications and tools has been developed to bring even more to our reality in shared places those three factors. This project have the objective of develop a smart behaviour system that can recognize some human voice commands and the human voice to control a house. To achieve this we use Python with a few libraries like SciKit-Learn and Google Speech Recognition. In the end we're able to capture the voice frequency of each user and simulate a house control, therefore we found some computational barriers tied of each person recognition.*

Resumo. *Nas últimas décadas busca-se por comodidade, praticidade e acessibilidade no cotidiano doméstico ou profissional. Diante disso vem-se desenvolvendo aplicativos e ferramentas, para que esses fatores se tornem cada vez mais uma realidade nos ambientes compartilhados. Este projeto tem o objetivo de desenvolver um sistema de comportamento inteligente, que seja capaz de reconhecer comandos vocais e o usuário com quem se comunica para controle residencial. Para isto utilizou-se a linguagem Python com algumas bibliotecas, como a SciKit-Learn e a Google Speech Recognition. Ao término conseguiu-se capturar a frequência vocal dos usuários e fazer a simulação do controle da residência, porém houve algumas barreiras computacionais quanto ao reconhecimento de cada indivíduo.*

1. Introdução

O ser humano, nas últimas décadas, vem buscando cada vez mais comodidade, praticidade e acessibilidade, seja no trabalho, ao fazer compras ou até mesmo em sua residência. Diante disso vem-se desenvolvendo aplicativos e ferramentas, para que esses fatores se tornem cada vez mais uma realidade no cotidiano.

De acordo com [Krishna and Lavanya 2017], o avanço das pesquisas em novas tecnologias, desperta nas pessoas o desejo de ter mais conforto em suas vidas. Nessa nova era de coisas automáticas, tais como carros automáticos, lava-roupas automáticas, *bots* automáticos surgem então as casas automáticas, onde pessoas tem a possibilidade de fazer as coisas com o mínimo de esforço possível, como acender lâmpadas, ligar televisores, escolher a música a tocar, etc.

Atualmente a procura pelas *smart homes* ou casas inteligentes, tem crescido devido à comodidade em que elas trazem à vida das pessoas, porém existem alguns problemas, um deles é a falta de reconhecimento do usuário com quem se comunica.

Sistemas já disponíveis no mercado, controlam alguns objetos como lâmpadas, ares-condicionados, televisores, etc. dentro de residências. Porém estes ainda não são capazes de aprender com o cotidiano do usuário e nem de reconhecer o usuário com quem está interagindo, assim gerando algumas falhas, por exemplo, se o usuário X pedir para colocar uma música e o usuário Y pedir para trocar a música, este sistema vai obedecer os comandos de ambos os usuários podendo assim gerar um desconforto para o usuário X. Outra situação pode ser o caso de o usuário X estar fazendo uma torrada e sem querer, sai de casa deixando a torradeira ligada. O sistema poderia identificar esta situação de risco e desligar a torradeira de forma a prevenir um mal maior.

Assim, este trabalho tem como objetivo geral o desenvolvimento de um sistema de comportamento inteligente chamado Houser [1], que seja capaz de, reconhecer comandos vocais para controle de uma residência e reconhecer por meio de padrões vocais o usuário, com quem se comunica.

Desta maneira, pode-se apresentar os seguintes objetivos específicos:

- Estudar a linguagem de programação Python;
- Estudar bibliotecas para reconhecimento da fala como a Speech Recognition;
- Estudar bibliotecas para a captura das frequências vocais como a PyAudio;
- Estudar bibliotecas de aprendizado de máquina (*machine learning*) como a Scikit-Lean;
- Desenvolver um algoritmo de reconhecimento vocal que, além da identificação das frequências da voz humana, execute os comandos executados por este usuário.

2. Referencial Teórico

Nesta seção, serão apresentadas e explicadas todas as ferramentas e conceitos utilizados neste trabalho final de graduação, para auxiliar no melhor entendimento do mesmo.

2.1. Internet das coisas

A Internet das coisas (*Internet of things*, IoT) é uma rede de objetos inteligentes ou controlada por sistemas inteligentes, capazes de interagirem com o ser humano das mais variadas formas possíveis [Holler et al. 2014].

2.2. Casas Inteligentes

Uma casa inteligente é uma residência em que se usa aparelhos interconectados, para monitorar e controlar aplicações e sistemas a distância, como as luzes ou o ar-condicionado. As tecnologias utilizadas em uma casa inteligente oferecem aos moradores da mesma, segurança, conforto, conveniência e eficiência energética a partir do momento em que permite que seus usuários a controlem por dispositivos como smartfones, notebooks ou qualquer outro dispositivo que esteja conectado na mesma rede. Sistemas de gerenciamento de casas inteligentes geralmente são sistemas baseados em arquiteturas de internet das coisas, possibilitando assim com que o usuário tenha completo controle sobre os objetos suportados pelo sistema e algumas vezes até automatizando as decisões tomadas pelo mesmo [Rouse 2018].

[1] Um trocadilho legal que mistura as palavras *house*, em português casa, e *worker* que significa trabalhador

2.3. Reconhecimento da fala e da voz humana

Tecnologias de reconhecimento da fala permitem que computadores equipados com microfones reconheçam a fala humana transcrevendo-a em texto que pode ser revertido em comandos. Por outro lado, o reconhecimento de voz é um problema distinto no qual o objetivo é identificar de maneira automática o falante em uma conversa. Em outras palavras, enquanto o reconhecimento de fala se preocupa em transcrever o áudio de uma fala para que um sistema computacional possa compreender a informação que está sendo falada, o reconhecimento de voz se preocupa em descobrir quem é o falante. As técnicas de reconhecimento de voz podem ter objetivo forense ou objetivarem a aplicação do sistema computacional a um ambiente com múltiplos falantes, como uma conversa entre seres humanos [Kincaid 2015].

2.4. Aprendizado de Máquina

O aprendizado de máquina pode ser do tipo indutivo ou do tipo não indutivo, conhecido por dedutivo. O do tipo indutivo emprega o princípio de inferência chamado indução. Desta forma é possível obter conclusões genéricas partindo de um conjunto de exemplos. Neste aprendizado, chamado indutivo, pode-se dividir em dois grupos principais, o aprendizado supervisionado e o aprendizado não supervisionado [Vieira 2011].

No aprendizado supervisionado, possuímos a figura de um professor externo, ou também chamado de indutor, que apresenta o conhecimento do ambiente através de conjuntos de exemplos da seguinte forma: entrada e saída desejada [Haykin 1999].

O algoritmo de aprendizado de máquina deve extrair a representação do conhecimento a partir destes conjuntos de exemplos, cujo objetivo é que a representação gerada possa ser capaz de produzir saídas corretas para quaisquer entradas que possam ser apresentadas [Haykin 1999].

O aprendizado não supervisionado não possui esta presença do professor, logo não existem exemplos rotulados. Neste caso o algoritmo de aprendizado de máquina deve aprender a representar ou agrupar as entradas, que foram submetidas através de uma medida de qualidade. Utilizam-se estas técnicas principalmente quando se tem por objetivo, encontrar padrões ou tendências que auxiliem no entendimento dos dados [De Souto et al. 2003].

2.5. Mineração de dados

A mineração de dados é o processo de descoberta de novas correlações significativas, padrões e tendências de peneirar grandes quantidades de dados armazenados em repositórios, usando tecnologias de reconhecimento de padrões, bem como técnicas de estatística e matemática [Larose 2005].

2.6. Python e bibliotecas

Python trata-se de uma linguagem de programação interpretada, que permite ser utilizada em várias áreas, não somente por ser de alto nível como por possuir bibliotecas que a permitem interagir com as mais variadas tecnologias. Para o desenvolvimento deste trabalho foram utilizadas as seguintes bibliotecas:

- PyAudio: fornece a ligação entre a linguagem Python e a API PortAudio para que seja possível efetuar gravações ou reproduções de áudio [PyAudio 2019];

- SpeechRecogntion: é uma biblioteca que tem suporte para as versões 2.6, 2.7, 3.3 ou superior da linguagem Python e visa adicionar um meio mais acessível de interação com o usuário nas aplicações feitas nesta linguagem. É capaz de reconhecer palavras que o ser humano fala e transforma-las em texto para melhor compreensão do sistema [Python 2019c];
- NumPy: esta biblioteca contém funções para análise de dados com álgebra linear, transmissão de dados (*data streaming*) e manipulação de dados em matrizes multidimensionais [NumPy 2019];
- Google Text-To-Speech (gtts): é uma biblioteca e uma ferramenta de interface de linha de comando (*command line interface*, CLI), que utiliza a API Google Translate's text-to-speech para criar áudios .mp3 a partir de textos previamente digitados ou salvos [Python 2019b];
- PyQtGraph: biblioteca desenvolvida para a construção em tempo real de gráficos matemáticos [Python 2019a];
- Scikit-learn: tem simples e eficientes ferramentas para a mineração de dados e análise dos mesmos [Python 2019a];
- Pymongo: é uma biblioteca desenvolvida pelos criadores do sistema gerenciador de banco de dados MongoDB, que permite a linguagem python salvar e utilizar informações de bases de dados criadas neste sistema[MongoDB 2008]

2.7. Model-View-Control (MVC)

O MVC é é uma metodologia utilizada em projetos devido à sua arquitetura, que permite dividir o projeto em camadas distintas. Cada camada, a Model, a Controller e a View, executa apenas o que lhe for definido. A sua utilização permite isolar as regras de negócios da lógica de apresentação e da interface com o usuário. Desta forma possibilita a existência de várias interfaces, de acordo com a necessidade, com o usuário. Estas podem ser modificadas sem que haja a necessidade da alteração das regras de negócios, o que permite flexibilidade e dá chances de reutilizar as classes [Dooley 2011].

2.8. Bancos de dados não relacionais

Os bancos de dados não relacionais (*Not Only SQL*, NoSQL) utilizam um sistema de armazenamento otimizado, buscando sempre o melhor desempenho na busca de dados em tempo real e o menor consumo de memória no momento de armazenar informações [Grolinger et al. 2013]. Neste trabalho será utilizado o banco MongoDB que é um banco de dados não relacional, orientado a documentos, de código aberto, multiplataforma e que foi desenvolvido na linguagem C++.

3. Trabalhos Correlatos

Nesta seção serão apresentados os artigos os quais contribuíram para o desenvolvimento desta pesquisa.

3.1. A Smart Home Appliance Control System for Physically Disabled People

Em [Mtshali and Khubisa 2019], fala-se sobre atuais sistemas de controle residencial, os quais normalmente controlam casas utilizando smartphones, notebooks, entre outros dispositivos que são capazes de se conectar a uma rede *Wi-Fi* (*wireless fidelity*, em português, fidelidade sem fio) com facilidade, porém, como o objetivo deste trabalho é desenvolver

um sistema capaz de contemplar deficientes físicos e idosos, os quais tem muitas dificuldades para realizar certos afazeres dentro de suas residências, o principal meio de interação com o sistema foi adaptado para ser o meio vocal, ou seja, a utilização da fala para comandar alguns objetos dentro da residência, assim permitindo maior acessibilidade.

Como resultado, obteve-se um sistema baseado nas arquiteturas e ferramentas utilizadas nos projetos Amazon Alexa e Google Home, criando também uma ferramenta nomeada pelo autor de *smart plug* (plugue inteligente) para conseguir controlar alguns objetos os quais não possuem conexão a redes sem fio.

3.2. Intelligent Home Automation System using BitVoicer

Em [Krishna and Lavanya 2017] foi criado um sistema básico de controle residencial utilizando Arduino juntamente com um modulo de Ethernet e um software chamado BitVoicer, o qual é extremamente leve e eficiente. O resultado desta combinação foi um sistema capaz de controlar alguns objetos conectados na mesma rede ou por infravermelho de um jeito muito eficaz, barato e prático, para trazer mais conforto a todos os moradores de uma residência.

3.3. Pre-processing Voice Signals for Voice Recognition Systems

Em [Berdibaeva et al. 2017] foi desenvolvida uma inteligência artificial capaz de criar padrões para reconhecer a voz humana, este sistema captura a fala do usuário, analisa as frequências e agrupa as mesmas em várias partes similares, uma espécie de agrupamento por tons de voz, conseguindo assim agrupar ruídos e ignora-los, após isto, busca no banco de dados informações para cada um dos tons encontrados e calcula a compatibilidade da voz capturada com as vozes presentes no banco de dados.

O autor comenta que optou por fazer um sistema mais simples separando os áudios em várias partes, pois acredita que este seja o modo o mais rápido possível para fazer uma análise vocal em uma arquitetura local. Ele também fala que optou por uma arquitetura local ao invés de uma baseada em nuvem (como grandes empresas vem fazendo) foi a opção mais viável, caso contrário teria que levar em consideração outros fatores como latência e alta velocidade na conexão com a internet.

4. Metodologia

Foi utilizado para o desenvolvimento do projeto Python 3.6 e o modelo MVC (*Model View Controller*), junto com o padrão de design *services* para melhor organização do código. Neste projeto cada arquivo contém somente uma classe que podem ser observadas na figura 1, os arquivos setup e start são os arquivos de configuração e inicialização do sistema respectivamente, todos os arquivos nomeados *__init__.py* e as pastas *__pycache__* são gerados automaticamente pela linguagem para que seja possível realizar a orientação a objetos.

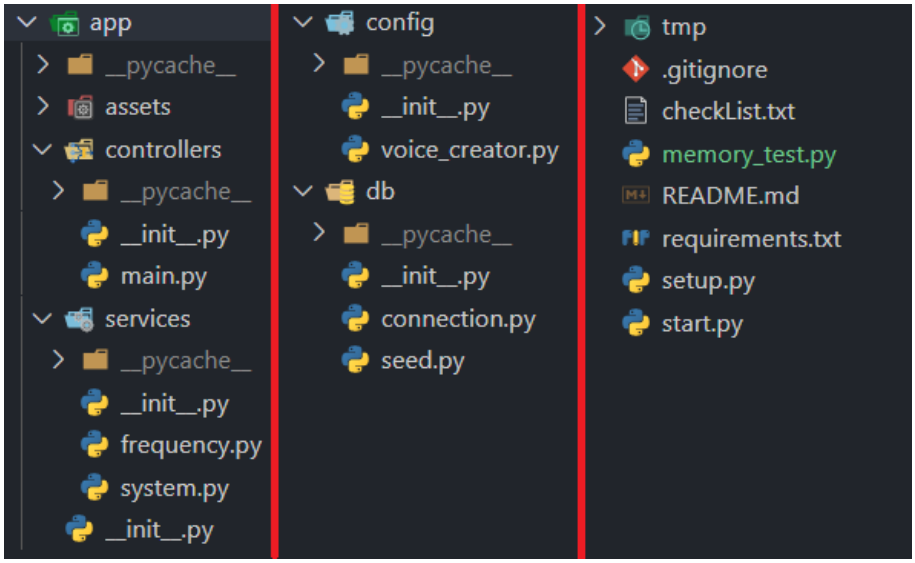

Figura 1. Estrutura de pastas do projeto

Para armazenagem dos dados foi utilizada a ferramenta MongoDB, que por suas características não relacionais, facilita a otimização do sistema e o processo de manipulação dos dados. A estrutura do banco pode ser observada nas figuras 2 e 3.

```
_id: ObjectId("5ed9598180b3a8955aa9cfc9")
command: "hello Houser"
answer: "how can I help you?"

_id: ObjectId("5ed9598180b3a8955aa9cfca")
command: "can you turn on the bedroom light"
answer: "alright, badroom light is on"

_id: ObjectId("5ed9598180b3a8955aa9cfcb")
command: "can you turn on the bathroom light"
answer: "alright, bathroom light is on"
```

Figura 2. Coleção de comandos nomeada Dialog

```
_id: ObjectId("5ee6c87b80aa843171e5d1a4")
sample: 0
name: "Edward"
> frequency: Array

_id: ObjectId("5ee6c88080aa843171e5d1a5")
sample: 1
name: "Edward"
> frequency: Array

_id: ObjectId("5ee6c88480aa843171e5d1a6")
sample: 2
name: "Edward"
> frequency: Array
```


Figura 3. Coleção de amostra das frequências da voz dos usuários nomeada Person

A figura 4 contém o diagrama de casos de uso do sistema para melhor compreensão das funcionalidades do mesmo.

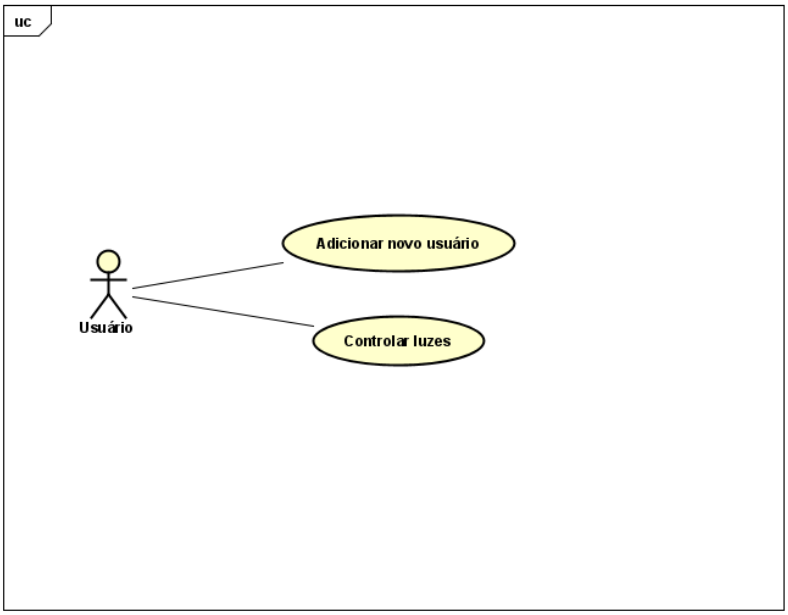

Figura 4. Diagrama de casos de uso do sistema

Para facilitar a compreensão do desenvolvimento do projeto, dividiu-se o mesmo em quatro seções as quais são, configuração do projeto, controle residencial, adição de um novo usuário e reconhecimento da voz. Primeiramente, para configuração do projeto, o usuário deve acessar a pasta do projeto pelo terminal do sistema operacional (Linux ou Windows) e utilizar o comando *python setup.py*, desta maneira Houser chama os arquivos *connection.py* e *seed.py* que fazem o uso da biblioteca *PyMongo* para efetuar criação do banco de dados e da coleção *Dialog* (figura 2), assim são inseridos todos os comandos que o usuário pode dizer e todas as respostas do sistema para o usuário, logo após a criação desta coleção, Houser faz a chamada do arquivo *voice_creator.py* que busca todos os registros na coleção *Dialog* e utiliza estes juntamente com a biblioteca *Google text-to-speech* para fazer a criação de todos os áudios que não estiverem presentes na pasta do projeto, estes áudios são as respostas do sistema para cada comando reconhecido que o usuário pode falar.

É importante mencionar que para a criação de todos os áudios utiliza-se o comando que o usuário pode dizer ao sistema, ou seja o campo *command* da coleção *Dialog*, como título do arquivo, porque não se pode colocar caracteres, como acentos e pontuações da linguagem escrita, também pode-se notar que alguns registros nesta mesma coleção tem seu título com a palavra *command* no início, como apresentado na figura 5, pois estes são respostas e requisições que Houser pode fazer sem que o usuário faça uma pergunta ou dê um comando para o sistema.

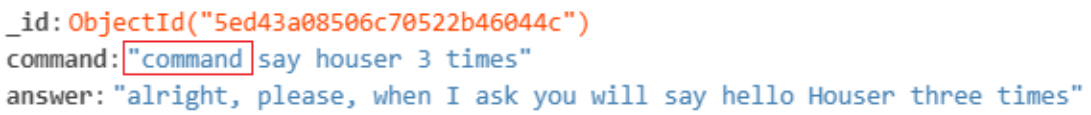

Figura 5. Exemplo de um comando do sistema para o usuário

Após esta etapa do projeto, Houser já está pronto para ser utilizado, para iniciar sua execução utiliza-se o comando *python start.py* pelo terminal na pasta do projeto, desta forma, Houser faz a chamada do arquivo *main.py* e este chama o arquivo *system.py* que inicia a gravação de tudo o que é falado utilizando a biblioteca *Speech Recognition*, esta biblioteca converte o áudio em texto para que o sistema possa comparar o mesmo com os comandos presentes na coleção *Dialog*.

Durante o contato vocal com o sistema, o usuário pode simular o controle de uma residência, pedindo para o sistema ligar ou desligar as luzes de um determinado cômodo dizendo comandos como *can you turn on the bedroom light*, a figura 6 permite observar com detalhes o fluxo do controle residencial.

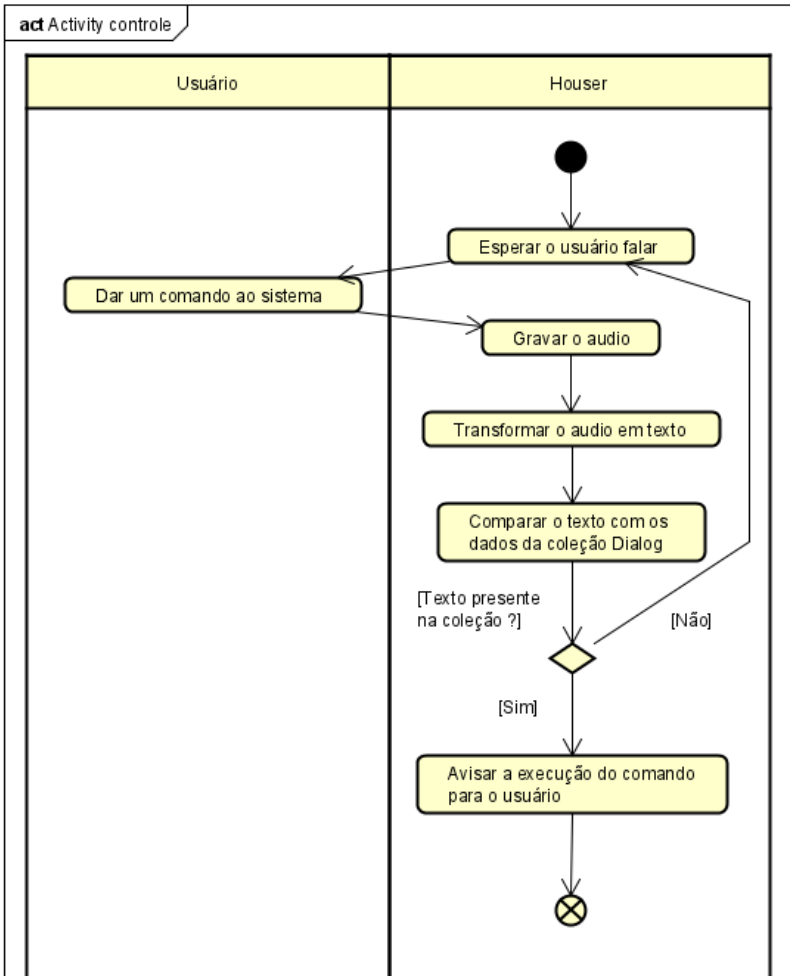

Figura 6. Diagrama de atividades do controle residencial

Para adicionar uma nova pessoa ao sistema de reconhecimento o usuário já cadastrado deve falar o comando *add another person*, neste momento o sistema questiona o nome do novo usuário que será inserido e após isso pede para o mesmo falar *Hello Houser* três vezes para efetuar a gravação desta fala, porém ao invés de ser utilizada a biblioteca *Speech Recognition* para a gravação do áudio, utiliza-se a biblioteca *PyAudio*, pois esta transforma a fala da pessoa em frequências. Para cada segundo é capturado um número X de amostras da voz, onde X é definido por 2 X a variável CHUNK (uma das variáveis de configuração da *stream* da biblioteca *PyAudio*), esta configuração está presente na função *listen frequency* do arquivo *system.py* (figura 7). Então Houser faz a criação da coleção *Person* e insere o nome do novo usuário, junto com as três amostras da sua frequência vocal (figura 3).

```python
def listen_frequency(timeout):
    CHUNK = 2048

    houser = pyaudio.PyAudio()

    stream = houser.open(
        format = pyaudio.paInt16,
        channels = 1,
        rate = 44100,
        input = True,
        output = True,
        frames_per_buffer = CHUNK
    )

    record_timeout = time.time() + timeout
    frequency_list = []

    print('----------Start----------')

    while (time.time() < record_timeout):
        frequency = stream.read(CHUNK)
        frequency_list.extend(struct.unpack(str(2 * CHUNK) + 'B', frequency))

    print('----recording is over----')
```

Figura 7. Parte da função de captura da frequência vocal

A figura 8 apresenta o diagrama de atividades do método de inserção do sistema para que fique mais clara a explicação do mesmo, onde o usuário 0 representa a pessoa que foi previamente inserida no sistema.

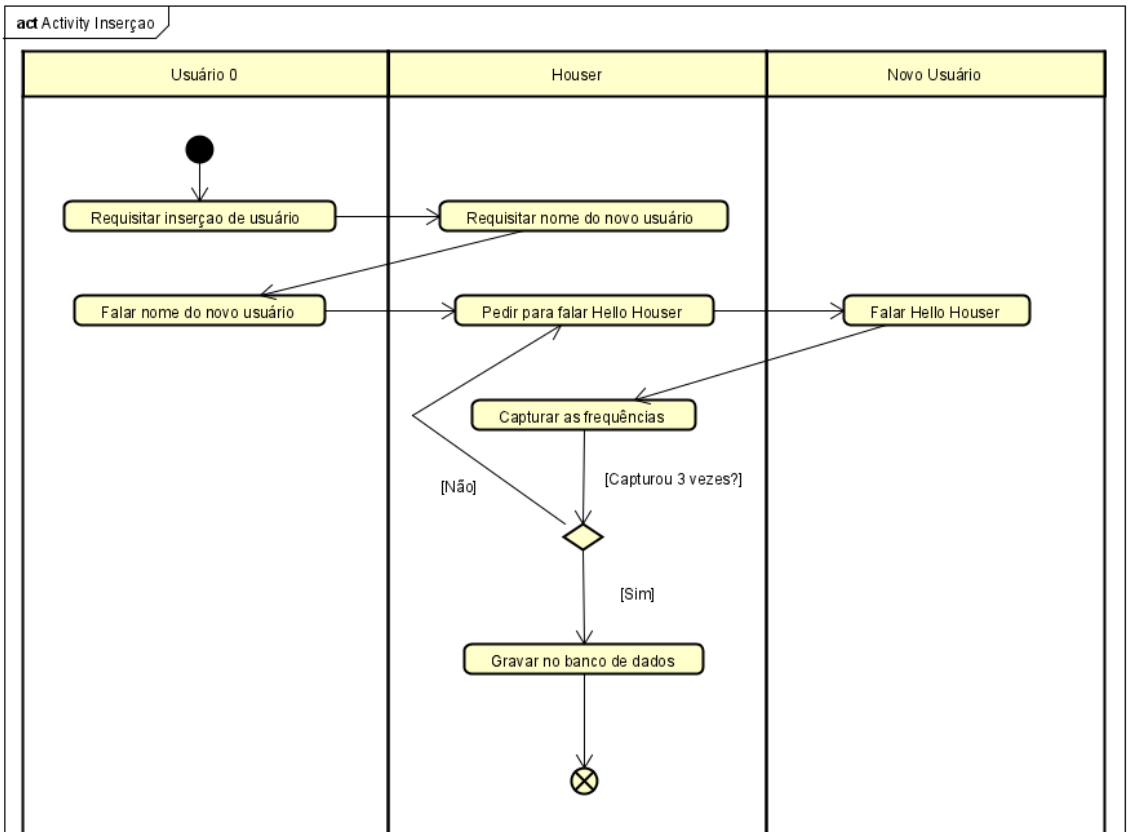

Figura 8. Diagrama de atividades da inserção de um novo usuário

Para reconhecer o usuário com quem está falando, é necessário que o mesmo diga o comando *Recognition*, desta forma Houser através do arquivo *main.py* chama o arquivo *frequency.py* que busca todas as amostras das vozes dos usuários presentes na coleção *Person*, e pede para que o utilizador diga mais uma vez o comando *Hello Houser*, gravando o mesmo com a biblioteca *PyAudio*, após este processo Houser cria duas matrizes bidimensionais utilizando a biblioteca *NumPy*, uma destas matrizes será criada com o vetor de frequências recém gravado e outra com o vetor de frequências trazido da tabela *Person*. Para a criação de pontos em um plano cartesiano cada uma destas matrizes tem o seu Y como o vetor de frequências do usuário e o seu X como um vetor de números criado que vai de 0 até o tamanho máximo do vetor de frequências (entre 260 mil e 280 mil elementos), desta forma ele começa a comparar as duas matrizes utilizando a biblioteca *Scikit-Learn*, desta biblioteca foi utilizado o modelo *Linear Regression*, que utiliza uma técnica de mineração de dados para agilizar a criação de pontos em um plano cartesiano, logo apos a biblioteca traça uma linha entre os pontos criados pelas duas matrizes (exemplo representado na figura 9), assim gerando uma porcentagem que se refere ao quanto uma matriz se assemelha da outra, quanto maior a porcentagem, maior a certeza de que Houser reconheceu o usuário. O sistema de reconhecimento não foi colocado inicialmente quando o usuário chama Houser pois ele necessita de mais aprimoramento como está sendo relatado nas secções seguintes.

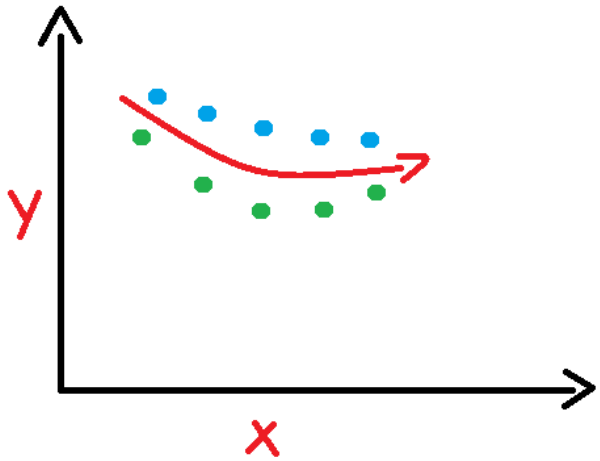

Figura 9. Exemplo da comparação de duas matrizes utilizando Linear Regression

5. Resultados

Como resultados obtidos pode-se considerar todo o estudo e conhecimento adquirido sobre a linguagem Python em si e também algumas de suas bibliotecas as quais foram utilizadas para o desenvolvimento deste trabalho. O desenvolvimento do sistema que consegue traduzir para texto os comandos que cada usuário fornece e executá-los, também foi possível capturar as frequências vocais de cada pessoa a qual interage com Houser, porém quando se trata do algoritmo de reconhecimento do usuário o mesmo consegue comparar matrizes de até 10 mil elementos, mas isso ainda não é o necessário para conseguirmos trabalhar com todos 3 segundos de áudio gravado (tempo necessário para falar *Hello Houser*), o sistema captura entre 260 mil e 280 mil valores da frequência da voz do usuário por fala, variando conforme o valor da variável CHUNK, o que resulta em enormes matrizes que serão comparados até três vezes para cada amostra na tabela *Person*, gerando assim um erro de falta de memória como mostra a imagem a seguir (figura 10).

```
C:\Users\Cerutti\Desktop\FACUL\TFG\Houser>python memory_test.py
Database Connected!!
Traceback (most recent call last):
  File "memory_test.py", line 18, in <module>
    linear.fit(frequency_array_0, frequency_array_1)
  File "C:\Users\Cerutti\AppData\Local\Programs\Python\Python36\lib\site-packages\sklearn\linear_model\_base.py", line 5
47, in fit
    linalg.lstsq(X, y)
  File "C:\Users\Cerutti\AppData\Local\Programs\Python\Python36\lib\site-packages\scipy\linalg\basic.py", line 1192, in
lstsq
    b2 = np.zeros((n, nrhs), dtype=lapack_func.dtype)
MemoryError: Unable to allocate 528. GiB for an array with shape (266240, 266240) and data type float64
```

Figura 10. Erro de memória insuficiente

6. Conclusão

A automação residencial é um grande passo para trazer mais conforto ao dia a dia de todos, e isso se torna ainda mais importante para aqueles que necessitam de ajuda para

certos afazeres dentro de casa, é uma área em que tem-se muito a explorar e desenvolver, principalmente quando o assunto é segurança e interconexão dos objetos a serem controlados. O estudo da linguagem Python permitiu ampliar os conhecimentos da graduação e sua aplicação tornou este projeto possível.

A biblioteca *Speech Recognition* permitiu o reconhecimento de sons emitidos pelos usuários, possibilitando que a biblioteca de reconhecimento de fala *PyAudio* pudesse efetuar a tarefa de captura das frequências moduladas pela voz humana e consequentemente armazena-las na base de dados.

A biblioteca *Scikit-learn* embora extremamente eficaz em reconhecimento de matrizes e mineração de dados exige um alto poder computacional que não estava disponível neste projeto, fazendo com que os resultados não pudessem ser apresentados da forma como foram planejados e esperados.

O algoritmo desenvolvido neste projeto permitiu a interligação das bibliotecas propostas promovendo execuções de alta complexidade no reconhecimento vocal e exigindo igualmente alta capacidade computacional, demonstrando que este projeto é possível, mas que necessita ser executado em estruturas de alto desempenho.

No decorrer deste trabalho final de graduação buscou-se desenvolver um sistema de controle residencial por meio de comandos vocais que fosse capaz de reconhecer o usuário com que fala, nota-se que o algoritmo de reconhecimento não consegue processar todos os dados necessários por falta de poder computacional e por este motivo alguns trabalhos futuros podem ser desenvolvidos para complementar ainda mais este trabalho como por exemplo.

- Algoritmo de compressão e ou normalização de vetores e matrizes;
- Implementação de APIs, como a do Spotify;
- Algoritmo de predição de ações utilizando rotinas;
- Implementação de um controlador e respectivos sensores para controle real de uma residência.

Referências

Berdibaeva, G. K., Bodin, O. N., Kozlov, V. V., Nefed'ev, D. I., Ozhikenov, K. A., and Pizhonkov, Y. A. (2017). Pre-processing voice signals for voice recognition systems. In *2017 18th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*, pages 242–245. IEEE.

De Souto, M., Lorena, A., Delbem, A., and de Carvalho, A. (2003). Técnicas de aprendizado de máquina para problemas de biologia molecular. *Sociedade Brasileira de Computaçao*.

Dooley, J. (2011). *Software Development and Professional Practice*. Apress.

Grolinger, K., Higashino, W. A., Tiwari1, A., and Capretz, M. A. (2013). Data management in cloud environments: Nosql and newsql data stores. Journal of Cloud Computer.

Haykin, S. (1999). *Neural Networks - A Compreensive Foundation*. Prentice Hall PTR Upper Saddle River, NJ, USA.

Holler, J., Tsiatsis, V., Mulligan, C., Karnouskos, S., Avesand, S., and Boyle, D. (2014). *From Machine-To-Machine to the Internet of Things*. Acedemic Press.

Kincaid, J. (2015). The power of voice: A conversation with the head of google's speech technology. Tech Crunch.

Krishna, I. and Lavanya, K. (2017). Intelligent home automation system using bitvoicer. In *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, pages 14–20. IEEE.

Larose, D. T. (2005). Discovering knowledge in data: An introduction to data mining. John Wiley and Sons, New Jersey.

MongoDB (2008). Tutorial pymongo 3.9.0. `https://api.mongodb.com/python/current/tutorial.html`. Consultado em 10 de Fevereiro de 2020.

Mtshali, P. and Khubisa, F. (2019). A smart home appliance control system for physically disabled people. In *2019 Conference on Information Communications Technology and Society (ICTAS)*, pages 1–5. IEEE.

NumPy, D. (2019). Numpy. `https://numpy.org`. Consultado em 20 de Outubro de 2019.

PyAudio (2019). Pyaudio documentation. site oficial do python software foundation. `https://people.csail.mit.edu/hubert/pyaudio/docs/`. Consultado em 11 de Novembro de 2019.

Python, D. (2019a). Pyqtgraph scientific graphics and gui library for python. `http://www.pyqtgraph.org`. Consultado em 11 de Novembro de 2019.

Python, S. F. (2019b). gtts 2.0.4. `https://pypi.org/project/gTTS/`. Consultado em 05 de Outubro de 2019.

Python, S. F. (2019c). Speechrecognition 3.8.1. `https://pypi.org/project/SpeechRecognition/`. Consultado em 09 de Novembro de 2019.

Rouse, M. (2018). Smart home or building (home automation or domotics). https://internetofthingsagenda.techtarget.com/definition/smart-home-or-building. Consultado em 12 de Outubro de 2019.

Vieira, S. A. G. (2011). Aplicação de máquinas de vetores de suporte na investigação da atividade gênica de câncer de colo de intestino. Universidade Franciscana.