FELIPE SINYEE TSAI
JIAHAO ZHAO

MODELO DE MEMORIA EPISÓDICA PARA ASSISTENTES ROBÓTICOS

São Paulo
2021

FELIPE SINYEE TSAI
JIAHAO ZHAO

MODELO DE MEMORIA EPISÓDICA PARA ASSISTENTES ROBÓTICOS

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

Orientador:

Professor Dr. Marcos Ribeiro Pereira Barretto

São Paulo
2021

Catalogação-na-publicação

Tsai, Felipe
Modelo de memoria episódica para assistentes robóticos / F. Tsai, J. Zhao - São Paulo, 2021.
55 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1.Inteligência Artificial 2.Memória 3.Robótica I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II.t. III.Zhao, Jiahao

AGRADECIMENTOS

Ao Professor Dr. Marcos Ribeiro Pereira Barretto, por todos os conselhos, pela paciência e pela inspiração com a qual guiaram o nosso aprendizado.

Às nossas famílias, que estiveram sempre ao nossos lados e nos apoiaram nesses tempos difíceis de pandemia.

A Ângelo, Augusto, Erick, Rodrigo e Vinícius pela amizade e companheirismo durante estes anos de graduação e pelos que ainda virão.

Ao Departamento de Engenharia Mecatrônica e à Escola Politécnica pela nossa formação acadêmica.

RESUMO

Assistentes virtuais robóticos estão presentes cada vez mais para ajudar as pessoas, sejam para aqueles que possuem algum tipo de necessidade ou até aqueles que visam facilitar seu dia-a-dia. Para que consiga atender a todos os usuários, é necessário que esses assistentes sejam capazes de se adaptar as situações, ou seja, possuir um tipo de memória que o permita aprender com os eventos presenciados. Nos seres humanos, a memória pode ser separada em várias categorias, como a memória semântica e a memória episódica. Neste trabalho o foco é na memória episódica, mais especificamente, em como criar um módulo capaz de representá-la em um assistente robótico virtual. O objetivo deste trabalho é criar um módulo capaz de aprender, a partir de visitas anteriores, as preferências de restaurantes de um indivíduo. Assim, o assistente deve ser capaz de capturar, armazenar, e requisitar essas memórias quando solicitadas. O modelo pode ser analisado em duas partes, a primeira responsável por capturar os dados e inserir em um banco de dados orientado a grafos, e a segunda responsável por realizar a requisição destes. O armazenamento das memórias é realizada por meio de átimos, que representam uma unidade de memória episódica. A implementação do modelo é realizada em módulos programados em *Python* e é utilizado o *RabbitMQ* como seu serviço de mensageria. Com os resultados obtidos foram possíveis verificar melhorias para o modelo, tanto para o processamento quanto otimização do módulo.

Palavras-Chave – Memória Episódica, Assistentes Virtuais, Banco de Dados orientado a Grafos, Robôs Sociáveis.

ABSTRACT

Robotic virtual assistants are increasingly present to help people, whether for those who have some kind of special needs or even those who aim to make their everyday life easier. In order to be able to serve all users, these assistants need to be able to adapt to situations, that is, to have a type of memory that allows them to learn from the witnessed events. In human beings, memory can be separated into several categories, such as semantic memory and episodic memory. In this work, the focus is on episodic memory, more specifically, on how to create a module capable of representing it in a virtual robotic assistant. The objective of this work is to create a module capable of learning, from previous visits, an individual's restaurant preferences. Thus, the assistant must be able to capture, store, and recall these memories when requested. The model can be analyzed in two parts, the first responsible for capturing the data and inserting it into a graph-oriented database, and the second responsible for carrying out their request. The representation of memories is performed by átimos, which represent an episodic memory unit. The model implementation is performed in modules programmed in *Python* and *RabbitMQ* is used as its messaging service. With the obtained results, it was possible to verify improvements to the model, both for the processing and optimization of the module.

Keywords – Episodic Memory, Virtual Assistants, Graph-oriented Database, Sociable Robots.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

1.1 Contextualização

Com uma rotina cada vez mais corrida e ocupada, as pessoas estão gradativamente contando com os assistentes robóticos virtuais para planejar e realizar atividades de forma rápida e eficiente. Atualmente, os assistentes virtuais são capazes de realizar vários tipos de funções, como a realização de chamadas telefônicas, o envio de mensagens de texto, a criação de lembretes e compromissos, a busca de perguntas e locais, formas de entretenimento e muitas outras. Estão presentes não só na maioria dos *Smartphones* mas também estão aparecendo gradualmente em *Smart Homes*, como a *Siri* (*Apple*), *Google Assistant* e *Google Home* (*Google*), *Cortana* (*Windows*), *Bixby* (*Samsung*), *Alexa* (*Amazon*), etc.

As aplicações para esses tipos de assistentes são imensas. Além das atividades citadas anteriormente, estes robôs podem ser direcionados a um público mais específico. Como um assistente pessoal para ajudar os idosos. Com um aumento da idade, um indivíduo sofre por várias mudanças biológicas, sendo uma delas a mudança nas suas capacidades do processo de armazenamento de memórias [1]. Desse modo, assistentes robóticos conseguem ajudar no suporte e na independência desta população mais idosa. Um exemplo de um assistente é a *Ellli.Q* [2], um robô assistente capaz de fazer várias atividades como medições diárias de saúde, lembretes de remédios e exercícios, jogos cognitivos, fatos interessantes e pequenas conversas para manter o idoso saudável e ativo durante o dia. Foi realizado uma entrevista [3] com os usuários testes deste robô, e as respostas obtidas foram em geral positivas. Alguns usuários afirmam que ficam solitários em casa, ficando mais propensos a desenvolver problemas de saúde, principalmente os problemas mentais, assim a *Elli.Q* ajudou bastante nessa parte. Um dos idosos confessou que apesar da *Elli.Q* ter a funcionalidade de conversação, são apenas diálogos curtos, um ponto que poderia melhorar.

Outra aplicação seria a utilização de um assistente robótico para ajudar indivíduos com o transtorno do espectro do autismo (TEA). Porém, como cada caso é diferente, seria

necessário a exclusividade de um assistente para cada pessoa, o que não está presente nos assistentes virtuais atuais.

Algo que estes assistentes robóticos citados anteriormente têm em comum, é que são todos padronizados, isso quer dizer que independentemente do usuário, o assistente virtual possui as mesmas funções e os mesmos conhecimentos pré-programados. Esses assistentes não são únicos, não se adaptam as diferentes necessidades de cada indivíduo, possuem apenas os conhecimentos que são inseridos nestes. Então o que é necessário para que um robô consiga atender melhor as necessidades de cada usuário? Como fazer com que este aprenda os hábitos, as preferências e os costumes de cada um? Para entender isso é necessário discutir sobre os tipos de memória.

Uma das separações formalizadas [4] é da memória implícita e da memória explicita. Esse trabalho terá foco na memória explicita, que por sua vez, pode ser distinguida entre memória semântica e episódica.

A memória semântica é o tipo de memória que armazena linguagens, símbolos, habilidades, fatos etc. Um exemplo é do estudo sobre um indivíduo chamado K.C. [5]. Este sofreu um acidente de moto acarretando lesões em várias regiões do seu cérebro, resultando em um tipo de amnésia. A maior parte das suas capacidades cerebrais estavam intactas, conseguia falar, mexer, raciocinar como qualquer outra pessoa. A parte do cérebro afetada foi a sua memória, enquanto K.C. conseguia lembrar de fatos como o seu aniversário, o endereço da sua casa de infância, a cor do carro que já teve e os ensinamentos que aprendeu na escola, não conseguia lembrar de nenhum acontecimento ou evento da sua vida pessoal. Esta memória que se mantém intacta no K.C. é a memória semântica, a mesma que os robôs assistentes atuais possuem. Estes não possuem a memória do processo de aprendizado, apenas a linguagem, habilidade e aprendizado que foi obtido.

A memória episódica é um contraste a memória semântica, apesar de ter surgido desta [6], vai além de guardar somente os fatos [7]. A memória episódica é o único sistema de memória que permite que o indivíduo consiga, conscientemente, relembrar do passado e poder reviver e reexperimentar estes acontecimentos [5], permite lembrar das sensações como as emoções, os cheiros e os sons que foram presenciados no momento descrito. Retomando ao exemplo citado anteriormente, essa memória é justamente a que o K.C. perdeu devido ao acidente, resultando na falta da capacidade de gravar e lembrar eventos, ocasiões e situações que passou em algum momento da sua própria vida.

Portanto, para ter um assistente customizável, que se adapte e aprenda com acontecimentos tornando-se único para cada usuário, é necessário a construção de um assistente

robótico com memória episódica. Para que não só tenha funções pré-programadas, mas que consiga também aprender com os eventos e as situações do dia-a-dia, implementando esse aprendizado para atender de forma mais adequada as vontades de um indivíduo.

Entretanto, criar um sistema de memória episódica artificial é um tema vasto e novo. Desse modo, este trabalho tem como objetivo estudar uma parte deste tema, para contribuir no desenvolvimento de conhecimento para a criação de uma nova geração de assistentes robóticos pessoais. Assim será desenvolvido neste trabalho um modelo conceitual para representar uma memória episódica.

1.2 Objetivos

Este trabalho tem como objetivo central desenvolver um sistema de representação da memória episódica. E a partir desse módulo, demonstrar o uso desta representação em um sistema robótico de recomendação de restaurantes.

Este objetivo pode ser analisado em duas partes. A primeira é desenvolver um modelo conceitual que represente uma memória episódica: criar um modelo de como armazenar e associar as informações em um banco de dados. A segunda é utilizar esse modelo de memória episódica num contexto de um robô sociável: propor uma maneira para retribuir as informações desejadas do banco de dados de uma forma eficiente.

1.3 Estrutura do Trabalho

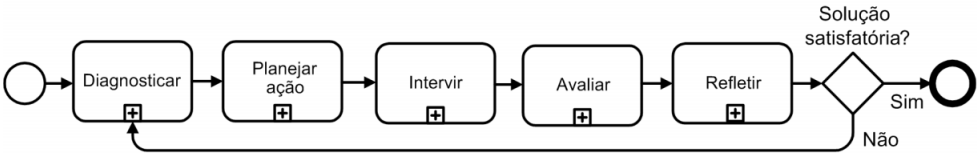
Após a definição do objetivo, será descrito neste trabalho um capítulo para a "Revisão Bibliográfica", que conterá os estudos, no estado da arte, e as teorias necessárias, como os tipos de sistemas de memória, para a modelagem de uma solução. Em seguida, será detalhado no capítulo de "Técnicas e Tecnologias" as ferramentas e os *softwares* que serão utilizados. Posteriormente, no capítulo de "Solução Proposta" será descrito com detalhes a solução elaborada, com os requisitos e as arquiteturas desenvolvidas para cada tipo de memória. Por fim, terá o capítulo de "Resultados", com os resultados e o capítulo "Conclusões", com as conclusões do modelo conceitual de representação de um sistema de memória episódica elaborado neste trabalho.

1.4 Metodologia

A metodologia adotada para este estudo, é a da “Pesquisa-ação”. A “Pesquisa-ação” é um método de pesquisa que tem como propósito ampliar o conhecimento obtido a partir de ações para solucionar um problema. Este propósito pode ser separado em duas partes. A primeira é pesquisar com o intuito de ampliar o seu conhecimento a um determinado assunto, que representa a parte teórica de um estudo. A segunda é realizar uma ação com o objetivo de melhorar o local em que a pesquisa está sendo realizada, representando a parte prática da pesquisa [8].

A “pesquisa-ação” pode ser representada por um ciclo, mostrado na figura 1.

Figura 1: Etapas do ciclo da pesquisa-ação



Fonte: Davison et al., 2004 [9]

A primeira etapa de “diagnosticar” é feita a análise do problema, a definição das motivações para a realização deste estudo. Caso não seja no primeiro ciclo, é realizada nessa etapa o diagnóstico dos resultados obtidos no ciclo anterior.

A próxima etapa, de “planejar ação”, é realizado as definições dos objetivos e os estudos sobre o assunto. A próxima etapa, “intervir”, marca a realização de ações e experimentos em relação a pesquisa.

“Avaliar” é a etapa em que os resultados obtidos anteriormente são avaliados, identificando os problemas e se foram ou não totalmente resolvidos.

Por fim, temos a etapa de “refletir” que é onde ocorre a reflexão das atividades realizadas no ciclo e a determinação se há ou não necessidade de realizar mais ciclos.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo serão explicados as teorias e os estudos necessários para a construção do modelo capaz de representar uma memória episódica.

2.1 Sistemas de memória

2.1.1 Memórias

Memória é um sistema capaz de realizar a aquisição, o armazenamento e a retribuição de dados. Estes dados podendo ser informações, situações, emoções ou sensações adquiridas em um dado tempo. O sistema de memória é um sistema imenso e complexo, podendo ser dividido em diversas categorias. Primeiramente pode ser separado entre memória de curto prazo e memória de longo prazo. Como o nome já diz, a memória de curto prazo são as informações que o indivíduo consegue reter apenas temporariamente, e após um certo período essas informações ou são esquecidas ou se transformam na memória de longo prazo.

Dentre as memórias de longo prazo, há uma separação em dois tipos [4], a memória implícita ou não declarativa e a memória explicita ou declarativa. A memória implícita são os acontecimentos que não são possíveis de relembrar, porém tem um efeito sobre a vida de cada um, por exemplo, as habilidades motoras adquiridas por um indivíduo. De maneira oposta, a memória explicita são os acontecimentos que podem ser relembrados.

A memória explicita ou memória declarativa, por sua parte, pode ser dividida entre duas categorias, a memória semântica: representa os fatos e os significados das coisas, e a memória episódica: representa os acontecimentos e eventos presenciados.

Este trabalho terá foco na memória explicita. Dentre suas subdivisões, será estudado um pouco a memória semântica, mas principalmente a memória episódica.

2.1.2 Memória Semântica

A memória semântica é responsável pelo conhecimento geral das coisas, sejam os significados das palavras, os conceitos e fatos existentes. Todo conhecimento que não está associado as experiências passadas está armazenada nesta memória, são informações que não precisam saber a origem que surgiu, mas o conteúdo em si [10].

Este tipo de memória é um subcomponente da memória que é responsável pela aquisição, representação e processamento das informações conceituais [11]. A memória semântica permite as pessoas assimilarem e interpretarem os significados em palavras e frases, além de reconhecer objetos e recapitular informações aprendidas anteriormente.

Essas memórias ainda podem ser classificadas em dois tipos: as sensórias e as não sensórias. As sensórias representam os conhecimentos obtidos através dos sentidos, como o formato dos objetos, a sua textura e a sua cor. Já as memórias não sensórias, são as informações que não são adquiridas diretamente pelos sentidos, são os conhecimentos enciclopédicos, obtidos através de livros ou estudos. E por tratar de memórias obtidas de fontes diferentes, estas podem estar armazenadas em seções diferentes da memória semântica [10].

O que difere a memória semântica da memória episódica é que a semântica representa o conceito, por exemplo, o que é um martelo? Martelo é um acessório ou ferramenta que pode ser de madeira ou metal; qual sua utilidade? O martelo pode ser utilizado para golpear objetos. Enquanto a memória episódica representa as experiências, por exemplo, quando utilizou o martelo pela última vez ou onde que utilizou.

2.1.3 Memória Episódica

A memória episódica é um sistema de memória que possui propriedades únicas que a diferenciam dos outros tipos de sistema. Tem foco em “o que”, “onde” e “quando” os acontecimentos ocorreram [12]. O sistema de memória episódica é capaz de permitir ao indivíduo lembrar de um momento passado de sua vida, assim revivendo esse evento junto com as emoções e as sensações que já passaram. Ou como disse Tulving, “É um sistema que torna possível a viagem no tempo mental através do tempo subjetivo” [5].

Há uma teoria que dita que o sistema de memória episódica surgiu a partir da memória semântica [6] e “evoluiu” desta. Que os dois sistemas de memória são separados por possuírem funções diferentes [13]. Essa ideia foi assunto de muitas contradições, de um lado aceitavam essa separação, e do outro negavam esta possibilidade [5].

Atualmente, há muitos estudos que fortalecem a separação desses dois sistemas de memória, muitos destes realizados em pacientes que sofrem de amnésia. Como os estudos realizados por Nielsen, que acredita que existem dois tipos de amnésia. A primeira, é quando há perda de memória episódica, o indivíduo é incapaz de guardar as experiências passadas. A segunda é quando há perda da memória semântica, o indivíduo perde o conhecimento dos fatos e conceitos aprendidos [14]. Outro estudo é o realizado no indivíduo K.C. [5], mencionado anteriormente.

Entretanto apesar dessa separação, ambos tipos de memória não são completamente independentes. Ainda é um assunto de debate, mas há discussões que a memória semântica depende de certa forma da episódica, uma vez que para obter o conhecimento semântico foi necessário um evento na qual foi aprendido [10]. Por outro lado, a memória episódica depende da memória semântica, pois a episódica guarda apenas as experiências e acontecimentos, enquanto a semântica complementa com os conhecimentos. Por exemplo, "um indivíduo comeu uma maçã de manhã", "comer uma maçã de manhã" representa a memória episódica, já o significado de que "maçã" é uma fruta vermelha representa a memória semântica.

2.2 Estado da Arte

Atualmente já existem estudos sobre memória episódica, desde métodos de recolhimento de dados e arquitetura de armazenamento até o sistema de requisição da memória. Cada um destes utiliza métodos e modelos diferentes. Para a finalidade desse trabalho será analisado principalmente o modo de armazenamento das memórias, os dados a serem armazenados e o tipo de banco.

O primeiro estudo analisado descreve o desenvolvimento de um robô humanizado chamado de *Eva* [15], este era capaz de memorizar os nomes dos usuários e as conversas ocorridas em eventos anteriores integrando-as nas conversas do presente. A *Eva* recolhia os dados através de uma câmera e um microfone, armazenando-as no SQLite, um banco relacional. Em uma futura conversa, o robô através do reconhecimento por voz ou face, solicita uma requisição no banco de dados sobre seus eventos anteriores relacionados a este usuário. Neste projeto, o processo de criação da memória episódica é composto por 3 etapas, processamento, armazenamento e requisição.

Na primeira etapa, o sistema decide se um certo episódio deve ser armazenado na memória de longo prazo. Para isso todo evento que ocorre é primeiramente armazenado

na memória de curto prazo, após o evento encerrar, é feito uma análise do evento, apenas as situações importantes e/ou eventos com impactos emocionais são armazenados na memória de longo prazo, a memória de curto prazo é então deletada. Neste projeto é feito o armazenamento apenas de parâmetros importantes de cada evento, como tempo do acontecimento, nome do usuário, o objetivo do robô no evento, o resultado após o evento, status do objetivo (objetivo atingido ou não), estado emocional e a intensidade da emoção.

Na segunda etapa, é feito o armazenamento das memórias em um banco relacional, neste caso foi utilizado banco SQLite. Em eventos onde a emoção é importante, é aplicado uma função de decaimento, para que emoções antigas sobre um certo indivíduo possam ser esquecidas, assim como funciona a memória humana, possibilitando que as memórias mais recentes tenham mais relevância.

No terceiro e último passo, a requisição de uma memória pode ocorrer de duas maneiras, espontaneamente ou voluntariamente. Requisições espontâneas de memórias ocorre quando há ocorrências de eventos que tenha emoções relacionadas a de um outro episódio ocorrido anteriormente, como rever um usuário que em um evento anterior teve impactos emocionais fortes. A requisição voluntaria ocorre quando o usuário solicita um acontecimento para realizar uma certa tarefa.

Neste projeto as memórias consistem no acontecimento, nome do usuário, o objetivo do robô no evento, o resultado após o evento, status do objetivo, estado emocional e a intensidade da emoção. Estes foram armazenadas em um banco de dados relacional (SQLite), e a requisição ocorre ou pela emoção predominante no momento ou através da solicitação do usuário.

O próximo estudo analisado mostra o desenvolvimento de um software de um robô, chamado *Rity* [16], usando memória episódica. Este possui dois sistemas de memória, uma memória procedural, que armazena os vários tipos de comportamento aprendidos e a memória episódica, que armazena *snapshots*. A ativação do armazenamento desses *snapshops* dependem da provocação de um agente interno ou externo, a partir disso, o *snapshot* é criado combinando os dados espaciais, temporais, perceptíveis, de estado interno e comportamentais do momento em que foi acionado, mostrado na figura 2.

Figura 2: Modelo de um *snapshot* da memória episódica

Fonte: Kuppuswamy et al., 2006 [16]

Para a seleção da memória mais adequada utilizam primeiramente o *Reactive Agent* para selecionar o comportamento mais adequado da memória procedural a partir do estado interno. Em seguida utiliza um algoritmo (*Introspetive Agent*) que procura os dados mais similares entre a percepção e o tempo atual nos *snapshots*. Faz se a comparação entre o *snapshot* encontrado e o comportamento escolhido pelo *Reactive Agent* verificando se são correspondentes, caso sim, procura a memória que teve o menor tempo de execução dos algoritmos.

Outro estudo se baseia no desenvolvimento de um robô que sugere os pratos de café da manhã de acordo com cada usuário [17]. Esse aprendizado é realizado a partir de repetições, em um primeiro instante é selecionado aleatoriamente uma bandeja com pratos variados. Após o consumo da pessoa é feito a leitura dos alimentos que sobraram na bandeja e não foram consumidos. Em uma próxima ocasião, o robô ajusta os pratos e o tamanho da porção de acordo com o histórico de cada refeição de cada indivíduo, além disso, recebe também *inputs* como a saúde para determinar o tipo de comida a ser servido, como café ou chá. Repetindo por algumas iterações o robô chegará em uma combinação de pratos ideal para cada usuário.

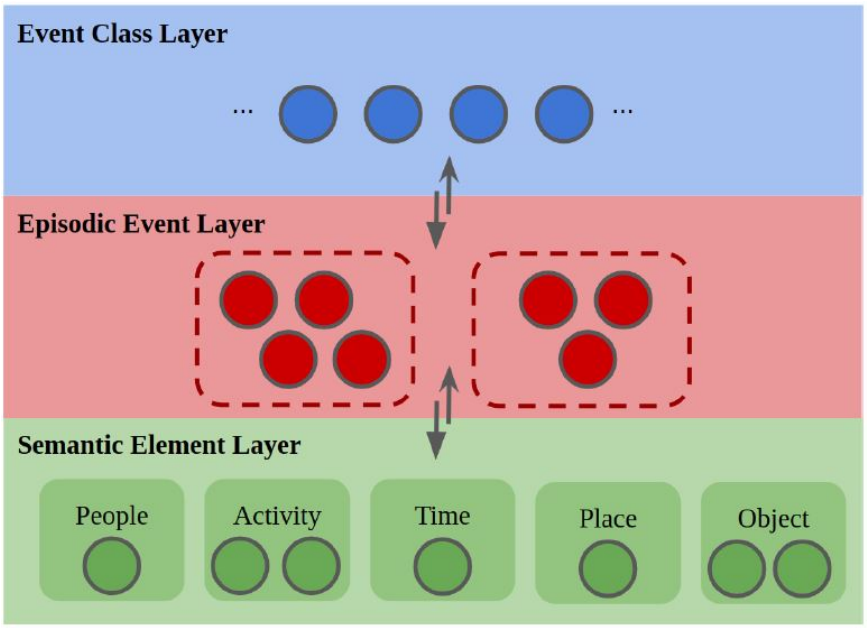
Neste projeto cada episódio da memória é categorizado implicitamente em entidades, começando do mais genérico para mais específico, como por exemplo, café da manhã, preparação do café e moer café. Formando assim uma arvore hierárquica como pode ser visto na figura 3. Cada uma das entidades possui uma importância, quando menor é a sua importância mais rápido é removido da memória, enquanto entidades que ocorreram em novos episódios podem ser relembrados e ganham importância.

Figura 3: Arvore hierárquica de um episódio da memória

Fonte: Sigalas et al., 2017 [17]

Neste outro trabalho, foi implementado um modelo de memória episódica em um assistente robótico. Foi desenvolvido um modelo e implantado em um robô assistente chamado *"Pepper"* [18]. Este robô foi utilizado em uma série de testes com idosos com o intuito de ajudá-los a lembrar de acontecimentos passados. Foi proposto um modelo composto de três camadas, a camada de evento, a camada episódica e a camada semântica, como representado pela figura 4.

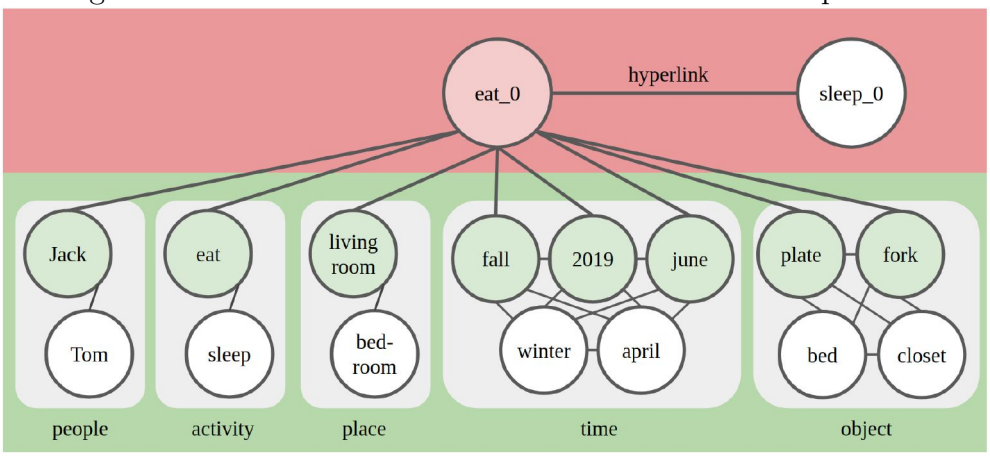
Figura 4: Diferentes camadas do modelo proposto



Fonte: Yang et al., 2021 [18]

A camada semântica possui os dados e elementos semânticos de eventos passados, como nomes e significados. A camada episódica não possui dados em si, ela faz a conexão da camada de eventos com os elementos semânticos correspondentes. Por fim, a camada de eventos é onde guardam os nódulos que representam um evento. Cada um dos nódulos é representado por cinco características semânticas: o nome da pessoa, a atividade que está exercendo, o momento em que o evento ocorre e um objeto que está relacionado. Um exemplo é a figura 5, que mostra o evento *"eat_0"* ligado as suas cinco características na memória semântica.

Figura 5: Armazenamento de um módulo de memória episódica



Fonte: Yang et al., 2021 [18]

Para armazenar esses dados foi utilizado um banco de dados orientado a grafos.

Em um outro estudo, foi criado um modelo de simulação de uma memória humana chamado *MINERVA 2* [19]. Um modelo que representa uma tentativa de juntar a memória episódica e a memória genérica das atividades em um único sistema. Neste é definido dois sistemas, *Primary Memory* que representa a memória de curto prazo e a *Secondary Memory* que representa a memória de longo prazo.

O modelo propõe que cada episódio, unidade de memória episódica, é armazenado em um vetor de alta dimensão em um banco de dados relacional. A retribuição desses dados é através de uma reconstrução, na qual o robô procura o episódio que mais se aproxima dos parâmetros selecionados.

O próximo trabalho tem como tema a utilização da teoria de Ressonância Adaptativa Profunda para aprender memória episódica [20]. É proposto um modelo de rede, *Deep ART*, que possui uma estrutura maior para a codificação das características.

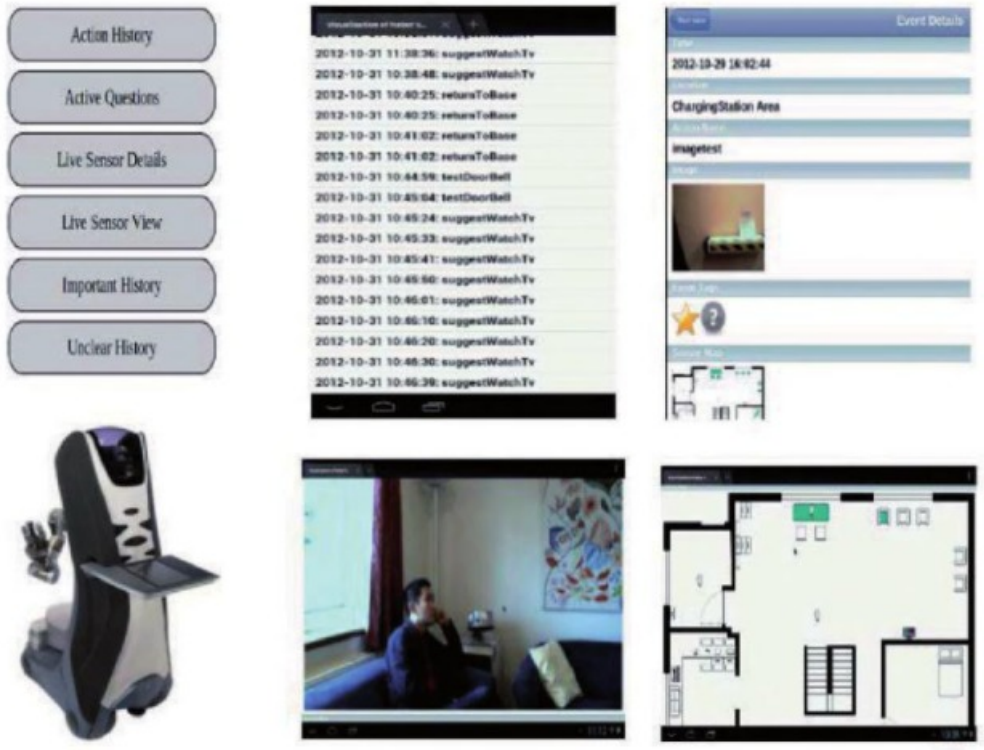
Este trabalho é baseado em uma rede neural não supervisionado que categoriza os

eventos automaticamente. A rede neural transforma o episódio em atributos semânticos, estes são guardados e em casos de retribuição da memória, são buscadas as memórias de acordo com os atributos semânticos.

O processo de aprendizado episódica consiste em duas etapas, a primeira é a codificação de uma sequência de eventos, e a segunda é a codificação de um episódio usando a sequência de eventos.

Neste estudo foi proposto um robô assistivo que ajuda a salvar memórias das ações de idosos com problemas de perda de memória, esses idosos por sofrer de perda de memória se tornam menos independentes e com maiores problemas de saúde e segurança, como por exemplo esquecer de tomar remédio ou esquecer o que estava fazendo a momentos atrás.

Figura 6: Interface de visualização das memórias episódicas do robô



Fonte: Ho et al., 2013 [21]

Este robô é equipado com câmeras onde captura memórias episódicas, e pode ser visualizada por uma tela. Essas memórias além de ajudar o idoso a relembrar sobre os acontecimentos, também é uma ferramenta de monitoramento para os familiares e médicos. Estes dados são captados pelas câmeras e processadas, o valor resultante é posteriormente armazenada em um banco relacional. Essas memórias são organizadas em pequenos eventos e em uma lista de ordem cronológica como mostra a figura 6, onde o usuário pode selecionar o evento em que se deseja visualizar.

Estes exemplos de representação da memória episódica não só utilizam métodos e ferramentas diferentes, como também adotam visões diferentes em relação ao que é guardado na memória. Por exemplo, se o robô irá esquecendo detalhes ao longo do tempo e consequentemente perdendo a memória toda ao invés de guardá-las para sempre, e se o robô deveria salvar 100% dos dados adquiridos ou guardar apenas as informações importantes. Isso depende dos fatores que estão sendo levados em conta, podendo ser o tamanho do espaço que ocupará a memória, ou alguma questão de similaridade com a memória humana. Neste trabalho, por se tratar de um estudo da maneira de como armazenar e retribuir os dados, será adotado que o assistente robótico guardará 100% das informações e não se esquecerá destas, para aproveitar o máximo de dados.

	Tipo de dados a serem armazenados	**Tipo de banco de dados**	**Captura dos dados**	**Breve descrição**
KASAP et al., 2010 [15]	Acontecimentos, nome do usuário, objetivo do robô, resultado, status, estado emocional e a intensidade da emoção	Banco de dados relacional (SQLite)	Câmera e microfone	Memórias são armazenados primeiramente no sistema de curto prazo, após o termino do evento, os parâmetros importantes são encaminhados para o sistema de longo prazo.
KUPPUSWAMY et al., 2006 [16]	Posição, tempo, percepção, estado interno, comportamento, fsm target	Banco de dados relacional	Não especificado	São divididos em memória procedural, onde armazena os comportamentos aprendidos, e memória episódica, onde armazena os snapshots (os eventos).
SIGALAS et al., 2017 [17]	Não especificado, apenas que são entidades	Não especificado, apenas que são guardadas em um formato de árvore hierárquica	Não especificado	As memórias são armazenadas em uma estrutura de arvore hierárquica com entidades, onde cada entidade possui a sua importância, os eventos mais recentes representam mais importância e os eventos com menor importância são esquecidos.
YANG et al., 2021 [18]	Nome, atividade, momento, relações com o objeto	Banco de dados orientado a grafos	Não especificado	O sistema de memória é dividido em 3 camadas, semântica, episódica e evento. Onde a semântica representa a base de dados dos significados, o evento armazena as informações temporais e situacionais de cada acontecimento e a episódica é a camadas que realiza a conexão entre as camadas de evento e semântica.
HINTZMAN, 1984 [19]	Não especificado, apenas que são vetores de alta dimensão	Banco de dados relacional	Não especificado	O sistema de memória é dividido em memória de curto prazo e longo prazo, uma vez que o evento é encerrado o evento é inserido na memória de longo prazo e removido no curto prazo.
PARK et al., 2016 [20]	Não especificado	Não especificado	Não especificado	O sistema transforma por meio de uma rede neural os eventos episódicos em atributos semânticos, e estes atributos são armazenados no banco.
Ho et al., 2013 [21]	Não especificado, apenas que são pequenos eventos	Banco de dados relacional	Câmeras	As memórias gravadas são organizadas em pequenos eventos e em uma lista de ordem cronológica.
Pretendido neste trabalho	Classe, valor, tempo, relações	Banco de dados orientado a grafos	Câmera	As memórias são processadas e transformadas em unidades de evento, são armazenadas na memória de curto prazo até que o evento se encerre, após isso, a unidade é inserida no banco de dados de longo prazo.

Tabela 1: Tabela de análise dos trabalhos estudados.

Na tabela 1 foi feito uma comparação entre os estudos. Neste é possível analisar pontos como o tipo de dados inseridos no banco, o tipo de banco utilizado e a forma de captação das informações, além de uma breve descrição do fluxo de memória nesses diferentes projetos. No final desta foi adicionado uma linha com as características pretendidas neste projeto.

3 TÉCNICAS E TECNOLOGIAS

Nos capítulos anteriores foi introduzido sobre o tema e os objetivos deste trabalho, neste capítulo será introduzido as ferramentas e *softwares* que serão utilizados para a realização de testes e protótipos do modelo que será desenvolvido no projeto.

3.1 *Resource Description Framework*

Resource Description Framework ou RDF [22] foi formulado na década de 90 como uma forma de representar dados, de descrever coisas, utilizado para representar informações em *Web* na forma de *triples* ou triplas em português, podendo ser armazenados em formatos como XML, RDF, Turtle etc.

A ideia do conceito RDF é formular triplas que consiste em sujeito, predicado e objeto, cada tripla expressa um fato específico sobre o sujeito, por exemplo, "Maria - gosta de - João" ou simplesmente "Maria gosta do João". O predicado é uma propriedade do sujeito e o objeto é o valor deste. Portanto, neste caso "Maria" possui um valor "João" na sua propriedade "gosta de".

RDF também é um vocabulário, oferecendo termos pré-definidos para facilitar o uso e padronizar o jeito que usuários descrevem um dado. Este possui termos que descrevem os dados em um nível básico, como *rdf:type* e *rdf:property*, que são instâncias de classes. Existe também uma extensão do vocabulário de RDF, chamado de RDF *Schema*, este possui termos que ajudam a descrever classes e a relação entre elas, como *rdfs:subClassOf* que descreve uma classe A como subclasse de B. No exemplo abaixo temos duas triplas com termos RDF e RDFS.

```
 @prefix : <http://www.example.org/> .
:joão    rdf:type           :homem .
:homem    rdfs:subClassOf    :humano .
```

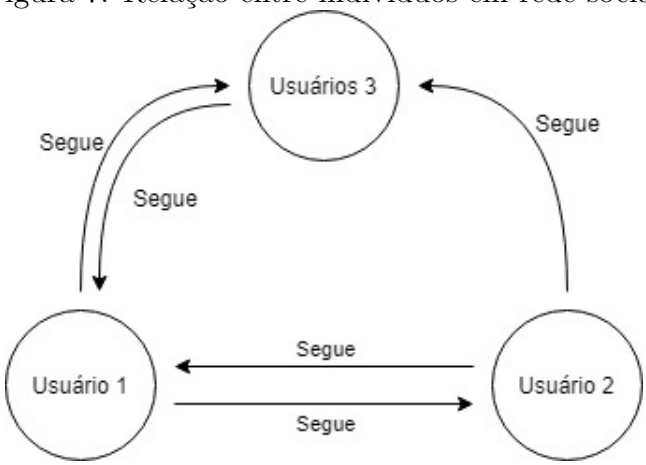
RDF e RDFS são atrelados um ao outro e oferecem termos que conseguem descre-

ver apenas dados básicos, no caso de dados com descrições mais complexas e precisas é necessário utilizar termos do vocabulário OWL.

3.2 Banco de dados orientado a grafos

Bancos de dado orientado a grafos [23] é um modelo de armazenamento de dados composto por nós e relações, na qual cada nó representa uma entidade, como pessoa, lugar ou comida, enquanto a relação faz a associação de dois nós. Por exemplo, "maça" e "fruta", ambos representam nós, e a relação entre os dois seria "tipo de" apontando de maça para fruta, ou seja, a maça é um tipo de fruta. Este tipo de banco é muito utilizado em empresas que tratam de informações que necessitam de relacionamentos complexos, como as plataformas de redes sociais. Seguindo esse exemplo de redes sociais, a figura 7 mostra um relacionamento entre três indivíduos, na qual cada usuário é um nó e a relação entre eles é de quem "segue" quem. Vale ressaltar que este exemplo possui um número pequeno de nós e relações, uma vez que cada indivíduo que é aumentado, há um aumento exponencial das relações.

Figura 7: Relação entre indivíduos em rede social



Fonte: Autoria própria, 2021

Nas próximas subseções será tratado sobre a linguagem utilizada neste tipo de banco e o software estudado para este trabalho.

3.2.1 *SPARQL*

O SPARQL é um acrônimo que significa *SPARQL Protocol and RDF Query Language*. É uma linguagem de consulta, utilizado para pesquisar e manipular dados no formato

RDF. Foi desenvolvido em 2004 pelo antigo *RDF Data Access Working Group*, atual *SPARQL Working Group* [24], e apenas em 2008, que foi recomendado oficialmente pela W3C (Organização de padronização da *World Wide Web*) [25].

As *querys* no SPARQL consistem basicamente em duas estruturas, "SELECT" e "WHERE", na qual o primeiro identifica as variáveis que desejam como resultado e o segundo utiliza essas variáveis como parâmetros das triplas. Observa-se que antes de cada termo da tripla existe um prefixo, este é necessário para identificar a ontologia que o termo pertencerá. No exemplo a seguir, a *query* tem como variável "person", representada pelo símbolo "?" prefixado no início da palavra. A tripla presente nesta chamada pode ser separada em sujeito, predicado e objeto, na qual o sujeito é a variável a ser procurada, enquanto o predicado e o objeto são valores fixos pré-definidos. A *query* realizará uma busca no banco de dados de todos os sujeitos que possuem a propriedade predicado com o valor objeto, mais especificamente neste caso, as pessoas que gostam do restaurante 1. Percebe-se que os prefixos "db1" e "db2" são abreviações dos endereços URI de ontologias, declarados no início da *query*, para que em chamadas mais longas não seja necessário reescrevê-las.

```
PREFIX db1: <http://example.com/exampleOntology1>
PREFIX db2: <http://example.com/exampleOntology2>
SELECT ?person
WHERE {
    ?person db1:like db2:restaurant1
}
```

3.3 GraphDB e Protégé

GraphDB [26] é um banco grafo que suporta tanto RDF quanto SPARQL. Este banco utiliza RDF4J como biblioteca, além de suas APIs para realizar *querys* de retribuição e armazenamento de dados. RDF4J é um *framework* modular em java que trabalha com dados RDF, permitindo armazenar, analisar e consultá-los.

Protégé [27] é um *framework* capaz de criar e editar ontologias. Um software desenvolvido pela *Stanford Center for Biomedical Informatics Research (BMIR)*. O Protégé oferece uma interface gráfica para a criação e edição das ontologias, incluindo também classificadores capazes de validar a consistência do modelo.

Neste projeto será utilizado o banco GraphDB para realizar os experimentos e reali-

zar os testes iniciais do modelo. O Protégé será utilizado para a criação das ontologias necessárias para a representação da memória semântica e posteriormente importado para o banco de dados orientado a grafos.

3.4 OpenCV

Open Source Computer Vision Library (OpenCV) [28] é uma biblioteca de visão computacional e aprendizado de máquina *Open Source*, foi desenvolvido inicialmente pela Intel. Essa biblioteca possui mais de 2500 algoritmos otimizados, incluindo algoritmos de reconhecimento de face, identificação de objetos, classificação de ações humanas em vídeo, reconhecimento de sorriso, encontrar imagens similares e muitos outros.

A ferramenta de reconhecimento de rosto funciona em tempo real, ela possui vários modelos treinados para reconhecer rostos, rostos de perfil, sorrisos, óculos, olho esquerdo ou direito e reconhece o corpo inteiro ou só uma porção dela. A biblioteca faz reconhecimento do rosto em tons de cinza, ou seja, é necessário converter a imagem em branco e preto para passar pelo processamento. O *output* da ferramenta é um *bounding box* no formato de 4 valores inteiros *(x, y, w, h)* representando um quadrado, os valores *(x, y)* formam um ponto, e os valores *(w, h)* representam o tamanho dos lados desse quadrado. Este *output* é muito importante para o reconhecimento de emoção e de pessoa, nos próximos parágrafos serão explicadas como funciona o processo de reconhecimento de emoção e pessoa.

O reconhecimento de pessoa precisa ser primeiramente treinado, portanto é necessário ter um banco com imagens dos indivíduos que se deseja reconhecer, a partir dessas imagens é treinado um algoritmo para reconhecimento de novas imagens. O *output* da ferramenta de reconhecimento de rosto é utilizado neste passo para realizar o reconhecimento de pessoa, comparando com os rostos já treinados e devolvendo um id do indivíduo.

Assim como o módulo de reconhecimento de pessoa, o reconhecimento de emoção também se utiliza do *output* do módulo de reconhecimento de rosto como *input* para realizar o processamento e reconhecimento da emoção. Neste caso será utilizada a biblioteca *DeepFace* para reconhecimento da emoção, essa biblioteca consegue reconhecer as seguintes emoções: raiva, nojo, medo, felicidade, tédio, surpresa ou neutro, além do gênero, idade e raça. Porém neste trabalho não será tratado sobre essas outras características.

No projeto a *OpenCV* será de grande importância, será utilizado para reconhecimento de rosto e seu resultado será utilizada para reconhecimento de usuário e emoções.

3.5 RabbitMQ

RabbitMQ é um software open source de mensageria, é amplamente utilizado tanto em pequenos startups quanto em grandes empresas. Este software fornece comunicação assíncrona de dados entre diferentes processos, aplicações ou servidores, pode ser utilizado com facilidade e suporta múltiplos protocolos de mensageria incluindo o AMQP.

No projeto, o RabbitMQ será responsável por transferir dados de uma aplicação para outra, integrando assim todas as ferramentas por uma mensageria. Algumas aplicações realizarão postagens em uma fila, enquanto outras aplicações irão consumi-las, esses processos ocorrem de forma assíncrona, evitando a perda de informação no caminho.

4 SOLUÇÃO PROPOSTA

Como definido anteriormente, o objetivo pode ser separado em duas partes. A primeira será o desenvolvimento do modelo conceitual que represente uma memória episódica, e a segunda será a utilização do modelo proposto num contexto de um robô assistente.

Para a primeira parte, será desenvolvido um método de representar a informação obtida em um formato RDF e um modo de inseri-los em um banco de dados orientado a grafos a partir da linguagem SPARQL, relacionando aos dados já existentes.

A segunda parte do trabalho será feito a implementação de um robô assistente, para que seja isso possível, será realizado testes e validações em um ambiente controlado. Utilizando a linguagem SPARQL, será desenvolvido através de *querys* um método de aquisição de dados específicos a partir das relações propostas pelo modelo desenvolvido pela primeira parte. Serão analisados os testes, e discutidos os devidos ajustes.

4.1 Requisitos

Este estudo é focado no desenvolvimento de um módulo de armazenamento de memórias episódicas, portanto não será aprofundado muito em temas que não estejam no escopo do projeto, como a estrutura para realizar a captação das memórias, o mecanismo de interação social com o usuário e entre outros. Portanto, é necessário definirmos requisitos e restrições, para garantir que o projeto tem uma complexidade adequada para o escopo do trabalho.

Em diversos estudos realizados por outros autores mencionados na seção 2.2, as memórias eram armazenadas no banco e eram atribuídas funções de decaimento sobre elas, simulando assim a memória humana, podendo perder memórias ao longo do tempo. No nosso projeto, por estar tratando de um estudo sobre o armazenamento e recuperação de dados e o esquecimento da memória não está no nosso campo de interesse, o robô será capaz de guardar 100% das informações obtidas sem as funções de decaimentos, ou seja,

sem perdas de memória.

Para se tornar um assistente pessoal adaptável a qualquer usuário, o robô precisaria ter acesso a todas as informações do seu usuário, assim como um assistente humano, com poder auditivo, visual e sensorial. No caso deste projeto, para não tornar este estudo muito complexo, foi considerado apenas imagem como forma de *input*, desconsiderando qualquer outro tipo de entrada. Foi definido que será tirado uma foto a cada segundo, e que todos os participantes e objetos a serem analisados tem que aparecer dentro do espaço de uma mesma foto.

No processo de criação das memórias, as memórias episódicas são relacionadas com os termos semânticos, portanto é necessário levar em consideração que o robô possui todos os conhecimentos semânticos necessários, ou seja, a inserção de dados no banco será apenas de dados episódicos.

Serão analisados e armazenados os dados em relação aos indivíduos, às emoções e aos restaurantes.

Desse modo, os requisitos e as restrições desse projeto são:

- Armazenamento de todas as informações, sem o esquecimento de memórias;
- *Input* apenas no formato de imagens;
- Todas as pessoas a serem analisadas tem que aparecer dentro do espaço de uma foto;
- Tratamento apenas de rostos nas imagens;
- O processo de reconhecimento dos restaurantes não será feito neste estudo;
- Todos os dados semânticos a serem utilizados já estarão no robô;
- O armazenamento e análise será apenas de pessoas, emoções e restaurantes;
- A conversa com o usuário (*chatbox*) será realizada através de outro programa externo que não será tratado no projeto;

4.2 Visão Geral

A estrutura do módulo conceitual é composta por ambos os tipos de memória citados anteriormente. A parte semântica representará os significados e os fatos dos tópicos

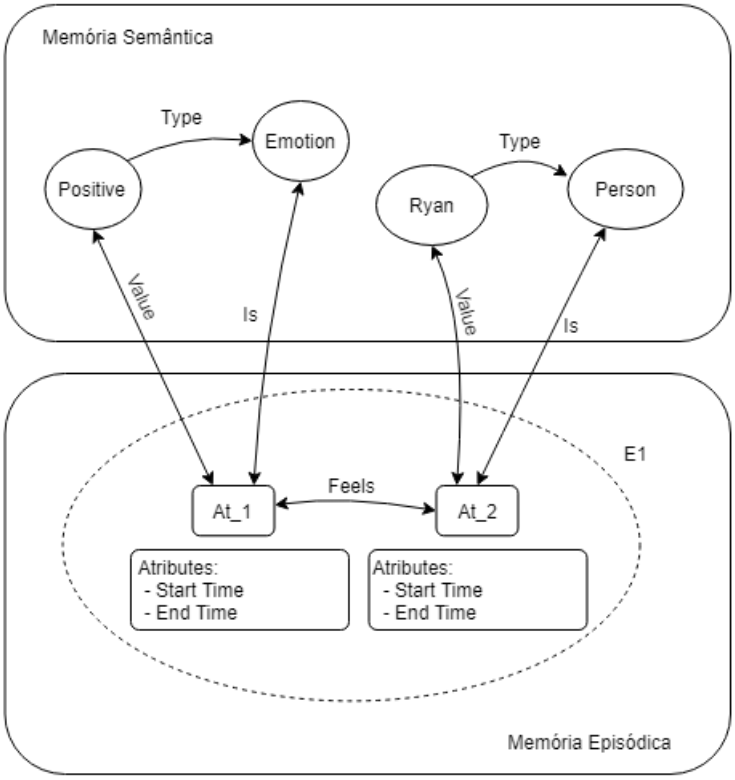
analisados, por exemplo as informações de um indivíduo, seja nome, idade ou gênero, e as suas relações. A parte episódica representará os eventos e os acontecimentos ocorridos, as interações entre os dados semânticos.

Sem a definição providenciada pela memória semântica, não seria possível identificar a pessoa que está se referindo, a emoção que está sentindo ou com que está conversando. Desta forma as duas partes funcionam juntas, uma complementando a outra.

O armazenamento da memória episódica se dá através de átimos, recipientes que representam uma unidade de memória. Cada átimo possui uma identificação única (At_id), e não só está ligado a sua definição semântica como também pode estar ligado a outros átimos.

O modelo pode ser representado pela figura 8. Um exemplo com apenas dois átimos, um representando uma emoção e outro uma pessoa, cada um com seus atributos dos momentos de início e fim, ligados entre si e em suas respectivas memórias semânticas.

Figura 8: Modelo de arquitetura de cada memória



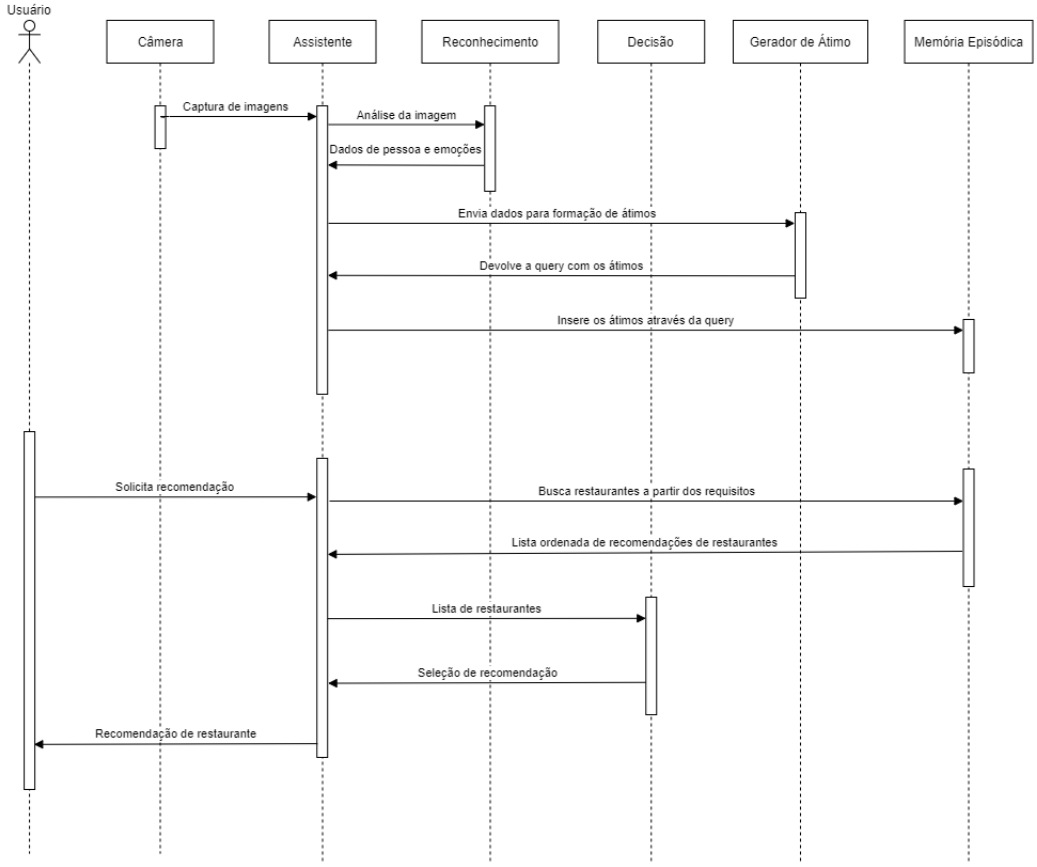
Fonte: Autoria própria, 2021

Esses átimos são armazenados através de triplas no banco de dados. A qualquer momento o usuário pode pedir sugestões de restaurantes. Assim, o assistente robótico processa essas informações e a transforma em uma *query* usando a linguagem *SPARQL*.

A partir desta *query* faz a requisição dos dados no *GraphDB*.

A figura 9 mostra um diagrama de sequência que representa como se obtém, guarda e recupera as memórias episódicas.

Figura 9: Diagrama de sequência do módulo de memória episódica.



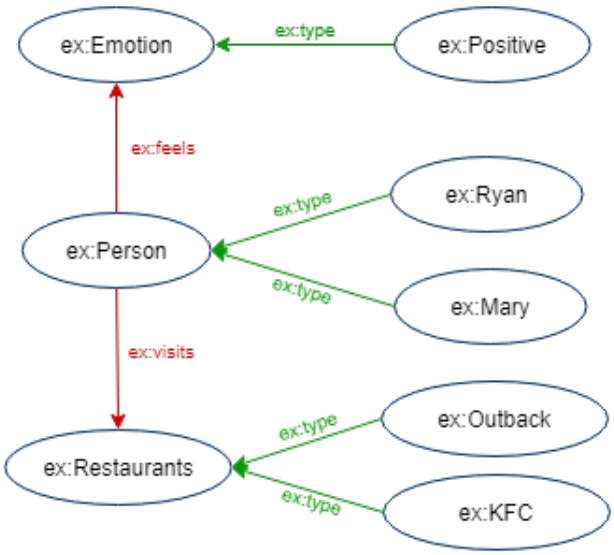
Fonte: Autoria própria, 2021

Nas próximas seções serão detalhadas as arquiteturas destas partes, e a estrutura de armazenamento e requisição das memórias.

4.3 Arquitetura para Memória Semântica

A arquitetura da memória semântica é representada pelos RDFs (*Resource Description Framework*), por triplas compostas de "Sujeito", "Predicado" e "Objeto". Como mostra o diagrama da figura 10. Na qual "ex:Positive" representa "Sujeito", "ex:type" o "Predicado" e "ex:Emotion" o Objeto.

Figura 10: Exemplo de RDF



Fonte: Autoria própria, 2021

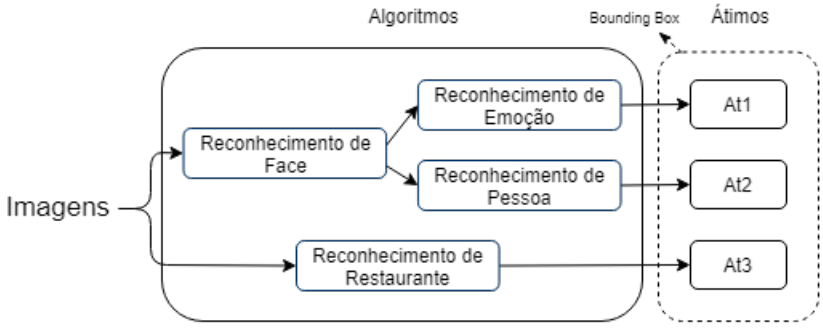
Para este trabalho, devido a sua pequena amplitude de situações e testes, não há uma necessidade de importar imensos bancos de dados. Portanto foi utilizado a ferramenta *Protégé* para a criação de um pequeno banco de dados de memória semântica, possibilitando assim a realização de testes.

4.4 Arquitetura para Memória Episódica

Para explicar melhor o que é um átimo, será utilizado como exemplo a figura 11. Nesta, está ilustrada o fluxo de processamento e tradução de um *input* na forma de imagem. Por exemplo, uma imagem tirada do ambiente passa por um algoritmo ($A1$) do *OpenCV* de reconhecimento de face, o resultado será um *Bounding Box* de onde o rosto está localizado na imagem, caso exista, este resultado será postado na fila de mensageria RabbitMQ. Em seguida essas mensagens são consumidas pelas aplicações de reconhecimento de pessoa ($A11$) e de reconhecimento de emoção ($A12$). A aplicação de reconhecimento de emoção após receber o *Bounding Box* de onde o rosto está localizado, utiliza algorítimos da *OpenCV* para realizar a identificação da emoção, o resultado desse processamento será um id da emoção e postado em uma outra fila de mensageria que será consumido pela aplicação principal. A aplicação de reconhecimento de pessoa realizará comparação através de algorítimos da *OpenCV* do rosto detectado pela primeira aplicação com os rostos presentes em seu banco de dados para identificar a pessoa que está na imagem, o resultado desse processo será um id da pessoa correspondente e também postado

na mesma fila de mensageria da aplicação de reconhecimento de emoção.

Figura 11: Modelo de processamento de um *input* na forma de imagem

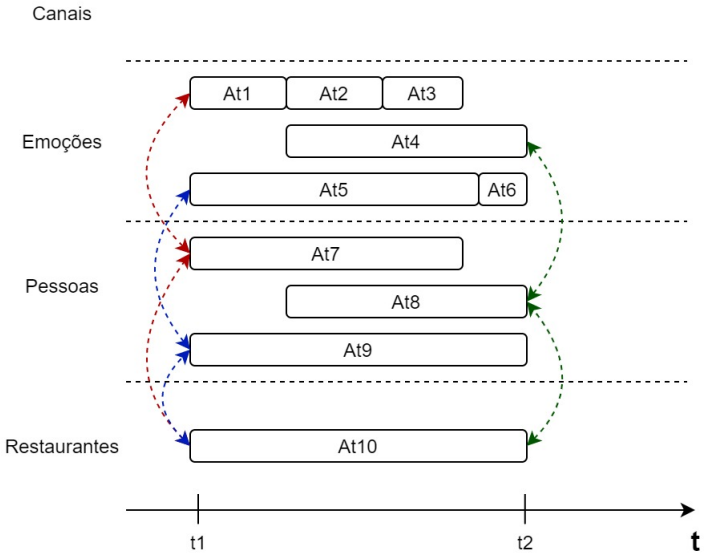


Fonte: Autoria própria, 2021

Após processado todas as informações e postado na fila, a aplicação principal consumirá os dados e o transforma em dois átimos, um de pessoa e outro de emoção. O átimo é um recipiente que conterá uma classificação, um valor que estarão representados na memória semântica, e uma duração. Desse modo, o átimo At_1 terá uma classificação de "Person", um valor como "Ryan" e uma duração que seria o momento em que apareceu nas imagens até quando saiu destas. Reforçando, o átimo At_2 teria uma classificação de "Emotion", um valor como "Positive" e uma duração de quanto tempo essa emoção predominar no indivíduo. Simultaneamente, a mesma imagem passa por outros algoritmos, por exemplo $A2$ de reconhecimento de restaurante, gerando um átimo At_3 com classificação "Restaurant", valor "Outback" e a sua duração. Como esses átimos surgem da mesma fonte, estes são agrupados por um *Bounding Box* que representa a ligação entre esses átimos.

Com a geração dos átimos, é possível representá-los em uma linha do tempo, para uma melhor visualização. A figura 12 mostra uma situação de uma refeição em um restaurante, na qual os átimos estão separados por canais que representam sua classificação e estão ligadas por setas de cores diferentes que representam seus *Bounding Box*. Neste exemplo temos uma situação de refeição entre três indivíduos, representados pelos átimos At_7, At_8 e At_9, em um restaurante representado por At_10 e as diversas emoções representadas pelos átimos de At_1 a At_6. Com a linha do tempo é possível observar com mais clareza o início e fim de cada átimo e as suas ligações, como o indivíduo representado por At_7, que saiu mais cedo da refeição e demonstrou as emoções At_1, At_2 e At_3 enquanto estava lá.

Figura 12: Modelo de arquitetura de uma memória episódica



Fonte: Autoria própria, 2021

Este é apenas um exemplo pequeno da arquitetura de uma memória episódica, a cada classificação nova de um átimo, haverá um novo canal na linha do tempo. Este número pode aumentar indefinitivamente sem prejudicar as memórias (átimos) pré-existentes, dependendo apenas de sua definição semântica.

4.5 Armazenamento das memórias

O armazenamento tanto da memória semântica quanto da memória episódica é feito a partir de triplas e inserido em um banco de dados orientado a grafos. Após a obtenção dos átimos, é necessário entender como é feito a representação dos dados em triplas. No exemplo citado na seção anterior, o At_1 representa uma pessoa com nome "Ryan" e está relacionada ao átimo At_2, que representa uma emoção com o valor "Positive". Desse modo a representação em triplas dessas informações é:

```
At_1    class       "Person"
At_1    value       "Ryan"
At_1    related     At_2
At_2    class       "Emotion"
At_2    value       "Positive"
```

Embora a memória semântica e a memória episódica estejam representadas em arquiteturas diferentes, ambas serão representadas por triplas e armazenadas em um mesmo

banco de dados.

Para a memória episódica, cada átimo terá no mínimo quatro triplas, uma tripla para sua classificação, uma para o seu valor e duas para o momento em que ocorreu, além do número de ligações com outros átimos. Para a memória semântica as triplas serão obtidas através da importação da ontologia. Neste exemplo, as triplas relacionadas existentes no banco serão as seguintes.

```
At_1        class       "Person"
At_1        value       "Ryan"
At_1        start       1628164856
At_1        end         1628172056
At_1        related     At_2
At_2        class       "Emotion"
At_2        value       "Positive"
At_2        related     At_1
At_2        start       1628164856
At_2        end         1628172056
"Positive"  type        "Emotion"
"Ryan"      type        "Person"
```

Na qual o horário é representado no formato unix timestamp.

4.6 Utilização das memórias

Nesta seção explicaremos como é feito a requisição dos dados do banco em uma situação de recomendação de restaurantes, o processo de filtragem dos possíveis restaurantes recomendados.

A requisição das memórias ocorrerá no momento em que o usuário solicitar uma recomendação de restaurante através de um *Chatbot*, o robô realizará no primeiro momento filtragem dos restaurantes de acordo com os pré-requisitos do usuário, por exemplo um restaurante que alguém gosta. Após obter todos os requisitos, o robô realizará uma *query* no banco de dados onde está armazenado tanto as memórias episódicas quanto a memória semântica, retornando uma recomendação de restaurante. Caso o usuário não goste da recomendação, poderá solicitar outra.

O processo de filtragem de restaurante será feito de acordo com os pré-requisitos do

usuário. Para isso, o robô fará uma série de perguntas, com as respostas será possível analisar e filtrar os restaurantes com maior chance de aceitação. Abaixo está um exemplo de interação do robô com o usuário.

```
User: Pode me recomendar um restaurante para janta por favor?
Robô: Claro! Estará acompanhado de alguém?
User: Do Ryan.
Robô: Tem preferência de experimentar algum restaurante novo?
User: Não, pode ser os já conhecidos.
Robô: O que acha do restaurante A que faz tempo que não vai?
```

Através da conversa, o robô consegue captar informações importantes para realizar o afunilamento. No exemplo acima, temos informação dos indivíduos que estarão presentes no restaurante com o usuário, portanto é importante que recomende um local onde todos os indivíduos gostem ou ao menos sejam neutros, para isso o robô analisa as emoções que cada um teve nas experiências anteriores caso exista. Em seguida é solicitada a preferência por algum tipo de comida, o usuário pode responder por sim ou não, caso sim, será selecionado apenas restaurantes de um dado tipo de comida. Por fim, o usuário pode escolher por um restaurante novo ou um restaurante já conhecido por ele anteriormente.

No processo de filtragem será feito elaborado uma *query* na linguagem *SPARQL* com todos os requisitos definidos, após isso é realizado a requisição dos dados no *GraphDB*.

Com o resultado da *query* o robô terá uma lista de restaurantes, dentro dessa lista escolherá uma opção e retorna para o usuário, que caso não fique contente com as opções poderá solicitar novas recomendações.

Como definido na seção 4.1, o *Chatbox* não será implementado nesse projeto, portanto a utilização das memórias será realizada com requisitos predeterminados.

Neste projeto será utilizado as memórias apenas para a sugestão de restaurantes. Entretanto, essas memórias podem ser utilizadas para outras funcionalidades. Para relembrar momentos, como com onde ou com quem estava acompanhado em um determinado tempo, ou qual foi o último restaurante que visitou com determinada pessoa.

5 RESULTADOS

5.1 Detalhamento do modelo criado

Para o detalhamento do modelo, foram revisto os requisitos e a solução proposta de acordo com o desenvolvimento da solução. A partir disso, foram feitos os devidos ajustes. Com o modelo detalhado, foi possível dividir o modelo em pequenos módulos para o desenvolvimento da aplicação em *Python*.

Os módulos serão:

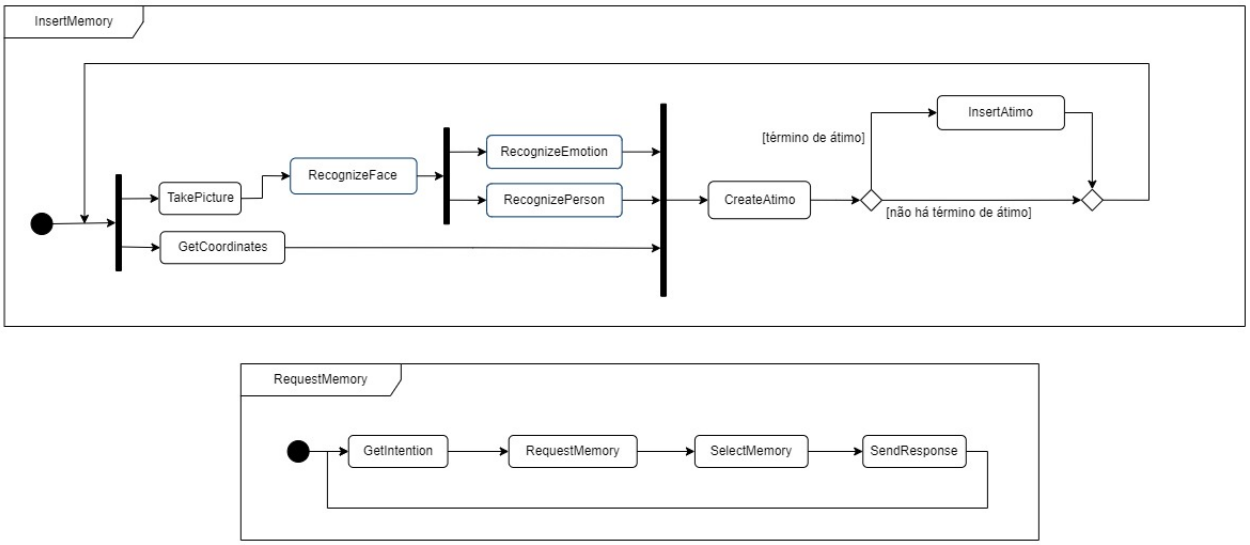
- *Assistant* (Representa o módulo principal, responsável por chamar os outros módulos).
- *TakePicture* (Que captura as imagens a serem armazenadas);
- *RecognizeFace* (Representado pelo *OpenCV*, reconhecimento de rosto nas imagens);
- *RecognizePerson* (Reconhecimento da pessoa a partir do rosto obtido do módulo de Reconhecimento de Face);
- *RecognizeEmotion* (Reconhecimento da emoção a partir do rosto obtido do módulo de Reconhecimento de Face);
- *GetCoordinates* (Simular o local onde a foto foi tirada, para obter o restaurante frequentado);
- *CreateAtimo* (Transformação dos dados recebidos pelos módulos de Reconhecimento e GPS em átimos);
- *InsertAtimo* (Transforma o átimo em uma *query* e a insere no banco orientado a grafos);
- *GetIntention* (Simula a comunicação entre o usuário, transformando o pedido em *querys*);

- *RequestMemory* (Busca os restaurantes recomendados a partir da *query* obtida no módulo Teclado);
- *SelectMemory* (Escolha da sugestão de restaurante a partir da lista obtida no módulo de Leitura);
- *SendResponse* (Simula a conversa com o usuário, apresentando o restaurante sugerido);

Cada um dos módulos é representado por uma aplicação que é executado em paralelo, a integração entre estes é feito por mensageria do tipo *Pubsub* utilizando a biblioteca *RabbitMQ*.

A figura 13 mostra um diagrama de atividades do modelo desenvolvido com os módulos citados anteriormente. Apenas o módulo *Assistant* não está representado no diagrama. Este módulo, é responsável pela comunicação entre os outros módulos, assim o *Assistant* está presente em todas as ligações entre os módulos. Para manter o diagrama menos poluído e mais fácil de analisar, foi optado em ocultar este módulo do visual.

Figura 13: Diagrama de atividades do conjunto de módulos



Fonte: Autoria própria, 2021

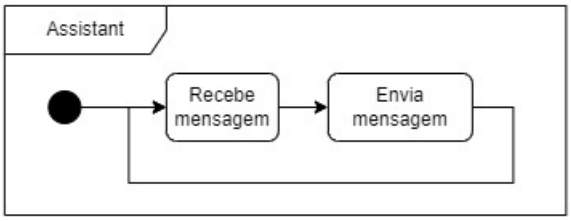
5.1.1 Módulos

Nessa subseção vamos explicar com mais detalhes o funcionamento de cada um dos módulos criados.

5.1.1.1 *Assistant*

O módulo *Assistant* representa o módulo principal do programa. Responsável por realizar a comunicação entre os outros módulos através do serviço de mensageria. Os módulos encaminham mensagens para o assistente e este encaminha a mensagem para uma fila onde um outro módulo realizará o consumo dessa mensagem, assim todas as mensagens passam obrigatoriamente pelo assistente. É possível utilizar o log deste módulo para acompanhar o funcionamento de todos os demais módulos, facilitando a análise em caso de erro.

Figura 14: Diagrama de atividades do módulo *Assistant*



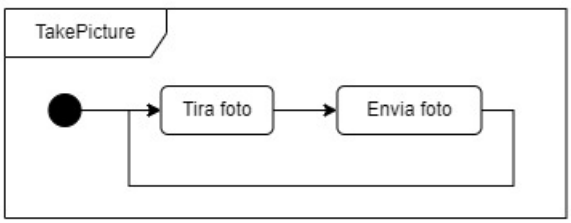
Fonte: Autoria própria, 2021

5.1.1.2 *TakePicture*

O módulo de *TakePicture* é responsável pela captura dos dados através de fotos periódicas, e o envio destas ao módulo *Assistant* que posteriormente é encaminhado para os módulos de reconhecimento de imagem.

Utilizando a biblioteca *OpenCV*, foi possível obter a captura de dados utilizando uma câmera conectada ao computador.

Figura 15: Diagrama de atividades do módulo *TakePicture*



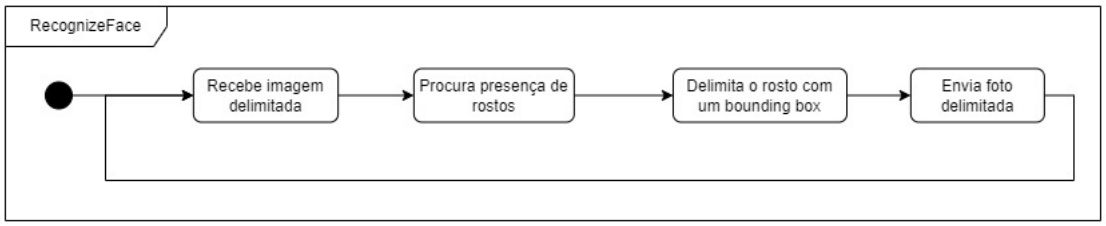
Fonte: Autoria própria, 2021

5.1.1.3 *RecognizeFace*

Esse módulo é responsável por localizar a presença de um rosto na imagem recebida. Caso encontre um rosto, a área é demarcada por um retângulo vermelho e enviada de volta

ao *Assistant* para as próximas etapas de reconhecimento, o reconhecimento de pessoa e o de emoção. Caso não for detectado rosto na imagem, o fluxo de dados é encerrado.

Figura 16: Diagrama de atividades do módulo *RecognizeFace*

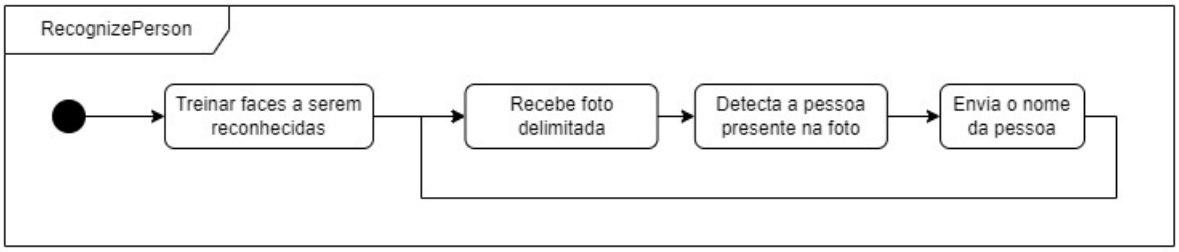


Fonte: Autoria própria, 2021

5.1.1.4 *RecognizePerson*

A partir da imagem com rosto demarcado, ocorre o reconhecimento da pessoa. Para isso, é necessário o treinamento deste modelo identificador. Foram utilizadas algumas fotos próprias para esse treino.

Figura 17: Diagrama de atividades do módulo *RecognizePerson*



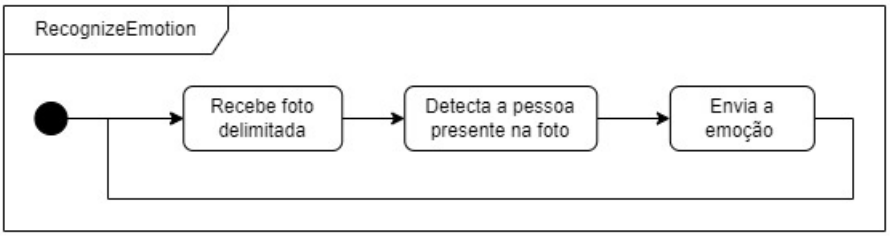
Fonte: Autoria própria, 2021

Após treinar os rostos das pessoas, o módulo espera o recebimento de uma foto delimitada pelo módulo anterior. A partir dessa foto, é detectado a pessoa presente e enviado o seu nome de volta para o *Assistant* e posteriormente ao módulo *CreateAtimo*.

5.1.1.5 *RecognizeEmotion*

O módulo *RecognizeEmotion* recebe a mesma imagem delimitada que é enviada ao módulo *RecognizePerson*. E a partir dessa imagem, a aplicação realiza identificação da emoção demonstrada pelo indivíduo, devolvendo a emoção dominante na face do indivíduo. Essa informação é encaminhado para o *Assistant* para ser posteriormente encaminhado para o módulo *CreateAtimo*.

Figura 18: Diagrama de atividades do módulo *RecognizeEmotion*

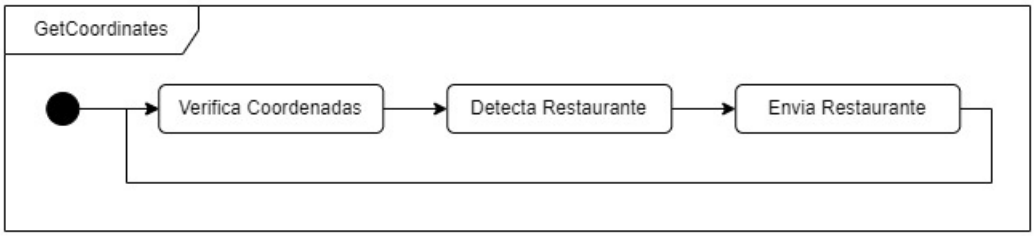


Fonte: Autoria própria, 2021

5.1.1.6 *GetCoordinates*

O módulo *GetCoordinates* é responsável pelo reconhecimento de restaurante. A partir das coordenadas do local em que as fotos estão sendo tiradas é possível obter o restaurante frequentado. O nome do restaurante é enviado ao *Assistant* e depois ao *CreateAtimo*.

Figura 19: Diagrama de atividades do módulo *GetCoordinates*



Fonte: Autoria própria, 2021

Entretanto, como definido nos requisitos, esse módulo não foi estudado em detalhes. Desse modo, para esse trabalho foi feito apenas uma simulação deste, onde ao chamar esse módulo, é devolvido um restaurante predeterminado.

5.1.1.7 *CreateAtimo*

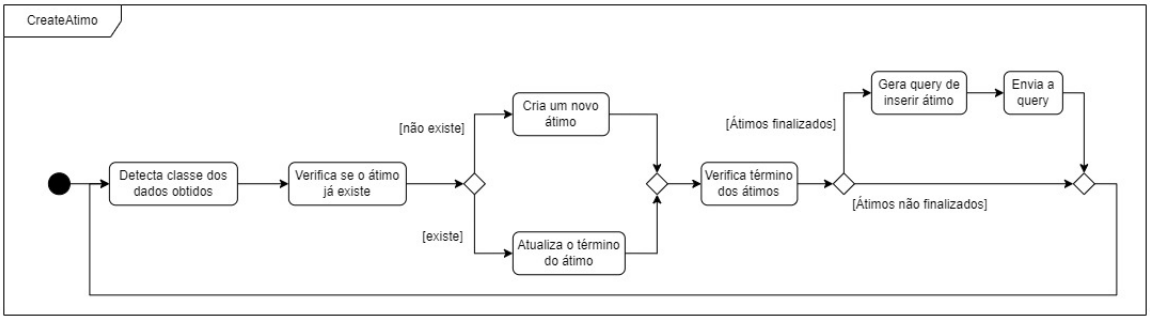
O módulo de *CreateAtimo* é responsável por receber os dados obtidos pelos módulos *RecognizePerson*, *RecognizeEmotion* e *GetCoordinates*. Gerando os átimos e fazendo as relações adequadas entre estes. Estes serão armazenados na memória de curto prazo até que o evento se encerre, após isso, o átimo será salvo na memória de longo prazo, o banco de dados orientado a grafo.

O módulo é separado em dois passos. O primeiro começa quando recebe-se um dado do *Assistant*, é detectado o tipo de dado, se é uma pessoa, emoção ou um restaurante. Em seguida, verifica se a existência de um átimo com esse valor na memória de curto

prazo. Caso encontre, é atualizado o tempo de término do átimo, caso não encontre, é criado um átimo novo e inserido na memória de curto prazo.

O próximo passo é verificar os átimos existentes na memória de curto prazo. Foi definido que caso um átimo não tenha sido atualizado em dois minutos (diferença entre o tempo de término e o tempo atual), o átimo é encerrado. Com estes átimos é gerado uma *query* de inserção e enviado para o próximo módulo, para que o átimo seja inserido na memória de longo prazo.

Figura 20: Diagrama de atividades do módulo *CreateAtimo*

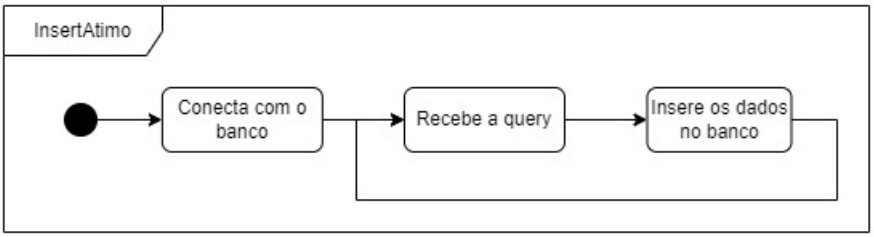


Fonte: Autoria própria, 2021

5.1.1.8 *InsertAtimo*

Esse módulo recebe uma *query* pronta, criada pelo módulo *CreateAtimo* e encaminhado através do *Assistant*. No módulo é estabelecido a conexão com o banco de dados orientado a grafo e envia a *query* para inserir os dados do Átimo.

Figura 21: Diagrama de atividades do módulo *InsertAtimo*

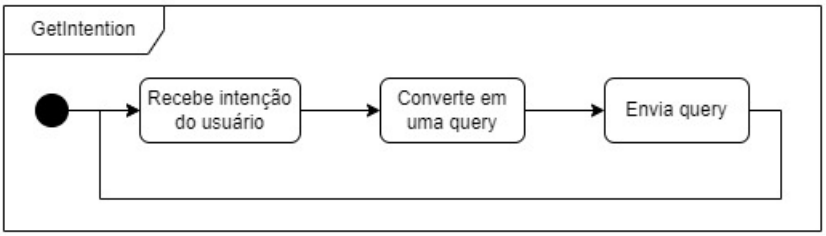


Fonte: Autoria própria, 2021

5.1.1.9 *GetIntention*

A função desse módulo é obter a intenção do usuário e transformá-lo em uma *query*. Neste o assistente robótico fara perguntas ao usuário, para restringir melhor a busca. Também é um módulo não foi estudado nesse trabalho e, portanto, feito uma simulação.

Figura 22: Diagrama de atividades do módulo *GetIntention*



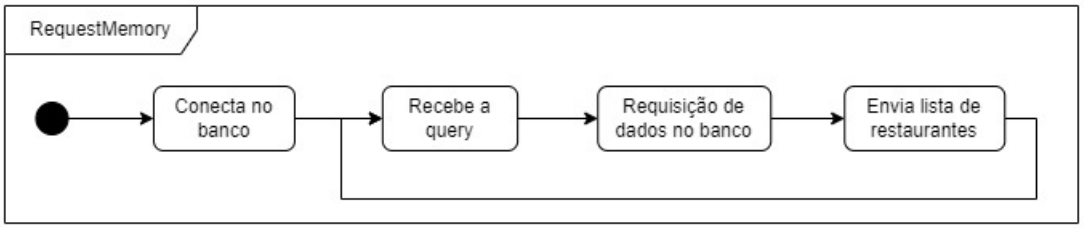
Fonte: Autoria própria, 2021

Para simulação, foram gerados *querys* prontas que representam a intenção de obter uma sugestão de restaurantes.

5.1.1.10 *RequestMemory*

O módulo realiza a conexão com o banco de dados e envia a query gerada pelo módulo anterior, obtendo uma lista de restaurantes e um peso atribuído a estes.

Figura 23: Diagrama de atividades do módulo *RequestMemory*



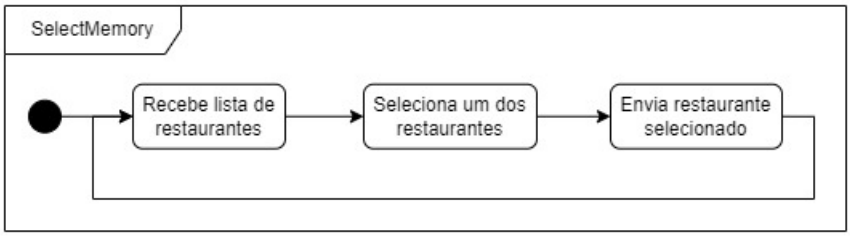
Fonte: Autoria própria, 2021

Essa lista de restaurantes e pesos é enviada ao *Assistant* e posteriormente ao módulo *SelectMemory*.

5.1.1.11 *SelectMemory*

A partir da lista de restaurantes e seus pesos, seleciona o restaurante mais adequado ao interesse do usuário. Enviando ao próximo módulo.

Figura 24: Diagrama de atividades do módulo *SelectMemory*

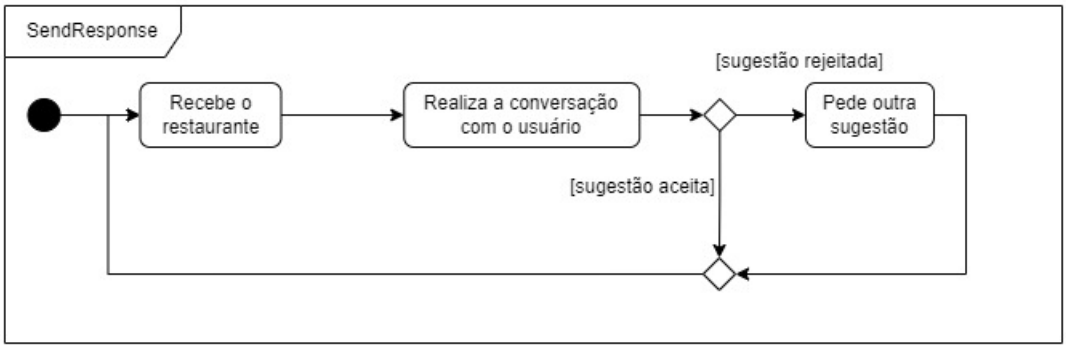


Fonte: Autoria própria, 2021

5.1.1.12 *SendResponse*

O módulo que realizará a conversa com o usuário. Este receberá o restaurante sugerido pelo módulo de *SelectMemory* e apresentará ao usuário. Caso a sugestão seja rejeitada, é solicitado outro restaurante para o módulo *SelectMemory*.

Figura 25: Diagrama de atividades do módulo *SendResponse*



Fonte: Autoria própria, 2021

5.2 Funcionamento do módulo

Para validar o funcionamento do módulo, foram realizados diversos testes com imagens de 2 pessoas em um restaurante, a aplicação deverá ser capaz de identificar os usuários e a emoção deste e inserir no banco com o tempo de início e fim dos eventos. Posteriormente podemos realizar a requisição de uma recomendação de restaurante

No início do processo, é enviado um comando para realizar a captura de uma imagem pela câmera, a imagem é salva e o diretório da imagem é encaminhada para o módulo *Assistant* como pode ser visto na figura 26, em seguida o módulo de assistente encaminha a mensagem para o módulo *RecognizeFace*.

Figura 26: Console do Módulo *TakePicture*

```
PS C:\Users\Haul\Documents\TCC> py
 .\TakePicture.py
Images/frame_0.png written!
 [x] Sending 'Images/frame_0.png'
```

Fonte: Autoria própria,

Na figura 27 podemos ver a imagem tirada pela câmera, onde possui duas faces com emoções diferentes.

Figura 27: Imagem tirada pela câmera

Fonte: Autoria própria,

O módulo *RecognizeFace* recebe a mensagem do assistente e inicia o processo de reconhecimento de face, é possível observar no terminal deste módulo, figura 28, que a imagem "Images/frame_0.png" foi recebida, e a partir desta imagem é feito o reconhecimento das faces existentes. Em seguida, o módulo as encaminha para o assistente e posteriormente é recebido pelos módulos de reconhecimento de emoção e de pessoa.

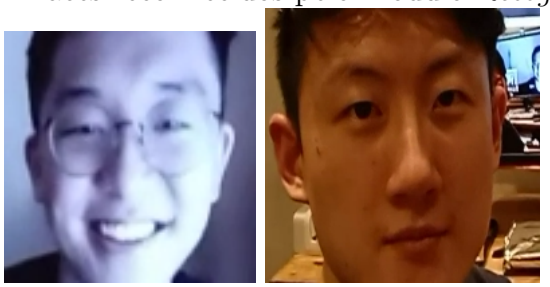
Figura 28: Faces reconhecidos pelo Módulo *RecognizeFace*

```
PS C:\Users\Haul\Documents\TCC> py .\Re
cognizeFace.py
 [*] Waiting for Images. To exit press
CTRL+C
 [x] 'Images/frame_0.png'
 Face Detected 'faces/face0_0.png'
 Face Detected 'faces/face0_1.png'
```

Fonte: Autoria própria,

Na figura 29 mostra as duas figuras geradas pelo módulo e enviados para reconhecimento de emoção e pessoa.

Figura 29: Faces reconhecidos pelo Módulo *RecognizeFace*



Fonte: Autoria própria,

No módulo *RecognizePerson* inicia-se com a codificação das imagens salvas dos usuários como mostra a figura 30, em seguida o módulo estará pronto para receber imagens de face. Depois do recebimento das imagens o módulo realiza o reconhecimento de pessoa e encaminha o nome dos usuários reconhecidos ao módulo gerador de átimos.

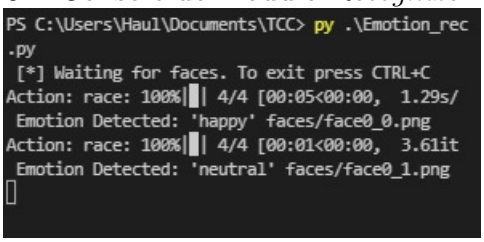
Figura 30: Console do módulo *RecognizePerson*

```
PS C:\Users\Haul\Documents\TCC> py .\RecognizePerson.py
2 encoding images found.
Encoding images loaded
 [*] Waiting for Images. To exit press CTRL+C
 Person Detected 'Jiahao_Zhao' faces/face0_0.png
 Person Detected 'Felipe_Tsai' faces/face0_1.png
```

Fonte: Autoria própria,

Assim como o módulo de reconhecimento de pessoa, o módulo *RecognizeEmotion* recebe as imagens das faces já reconhecidos pelo *RecognizeFace*. Após o recebimento da imagem de uma face, inicia-se o processo de reconhecimento de emoção como pode ser visto na figura 31, o resultado do processo é encaminhado para o assistente e posteriormente para o modulo gerador de átimos.

Figura 31: Console do módulo *RecognizeEmotion*



Fonte: Autoria própria,

No módulo *CreateAtimo* é realizado a criação dos átimos, na qual ocorre duas verificações. A primeira para verificar se já existe um átimo aberto com esse valor, caso sim, o tempo de término é atualizado, caso não, um átimo é criado e adicionado a memória de curto prazo. E a segunda para verificar se possuem átimos na lista que encerraram e podem ser inseridos no banco de dados.

Depois que os átimos forem encerrados, estes serão adicionados ao banco, como mostra a figura 32.

Figura 32: Dados inseridos no banco de dados orientado a grafos

	subject	predicate	object
1	db1:at1	db1:class	"person"
2	db1:at1	db1:end	"1638289800"^^xsd:integer
3	db1:at1	db1:related	db1:at2
4	db1:at1	db1:related	db1:at3
5	db1:at1	db1:start	"1638288000"^^xsd:integer
6	db1:at1	db1:value	"User_1"
7	db1:at2	db1:class	"emotion"
8	db1:at2	db1:end	"1638289800"^^xsd:integer
9	db1:at2	db1:start	"1638288000"^^xsd:integer
10	db1:at2	db1:value	"Happy"

Fonte: Autoria própria,

No processo de requisição da memória, o banco é populado com dados artificiais para realizar testes. Foram inseridas dez refeições, o usuário a ser analisado está presente em todos os eventos, e em cada um destes está acompanhado de uma a quatro pessoas diferentes em cinco restaurantes diferentes.

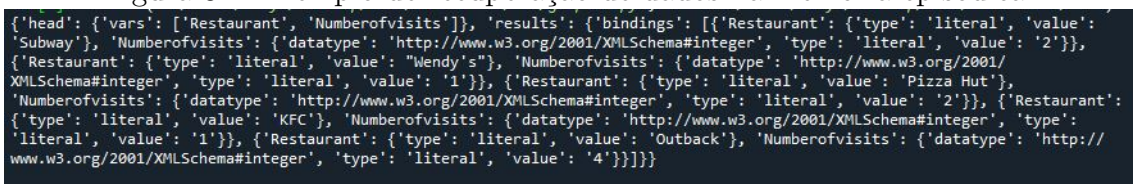
A figura 33 mostra os eventos inseridos no banco de dados, onde cada linha da tabela representa uma tripla. As seis primeiras linhas representam um átimo, com sua classe, valor, começo, fim e as relação com outros átimos, no caso os átimos at2 e at3. Para inserir esses dez eventos foram inseridos no total 58 átimos e 280 triplas. É possível observar que com o aumento de eventos, o número de triplas necessárias aumenta drasticamente

ocupando bastante espaço. Entretanto, uma das vantagens de utilizar um banco orientado a grafos ao invés de um banco relacional é a possibilidade de acrescentar elementos distintos no futuro e relacioná-los sem a necessidade de modificar o banco. Já um banco relacional, caso deseje acrescentar um novo elemento a ser analisado é necessário mudar o banco e adicionar uma coluna específica para esse novo tipo de dado.

Para o teste foi gerado pelo módulo *GetIntention* uma *query* para recuperar os restaurantes e o número de vezes visitados por um indivíduo.

A figura 34 mostra os dados obtidos após a requisição dos restaurantes realizado pelo módulo *RequestMemory*. Em *head* → *vars* tem os nomes das colunas criadas "Restaurant" e "Numberofvisits". E em *head* → *results* → *bindings* tem-se a lista que representa as linhas da tabela, que contém o nome do restaurante seguido do número de vezes visitados.

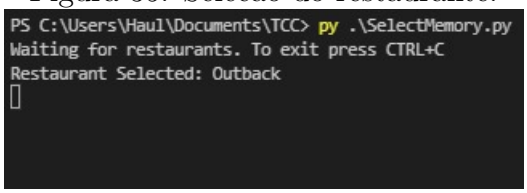
Figura 34: Exemplo de recuperação de dados na memória episódica.



Fonte: Autoria própria, 2021

Em seguida, o módulo *SelectMemory* seleciona um dos restaurantes (o mais visitado) e envia ao *Assistant*, figura 35.

Figura 35: Selecão do restaurante.



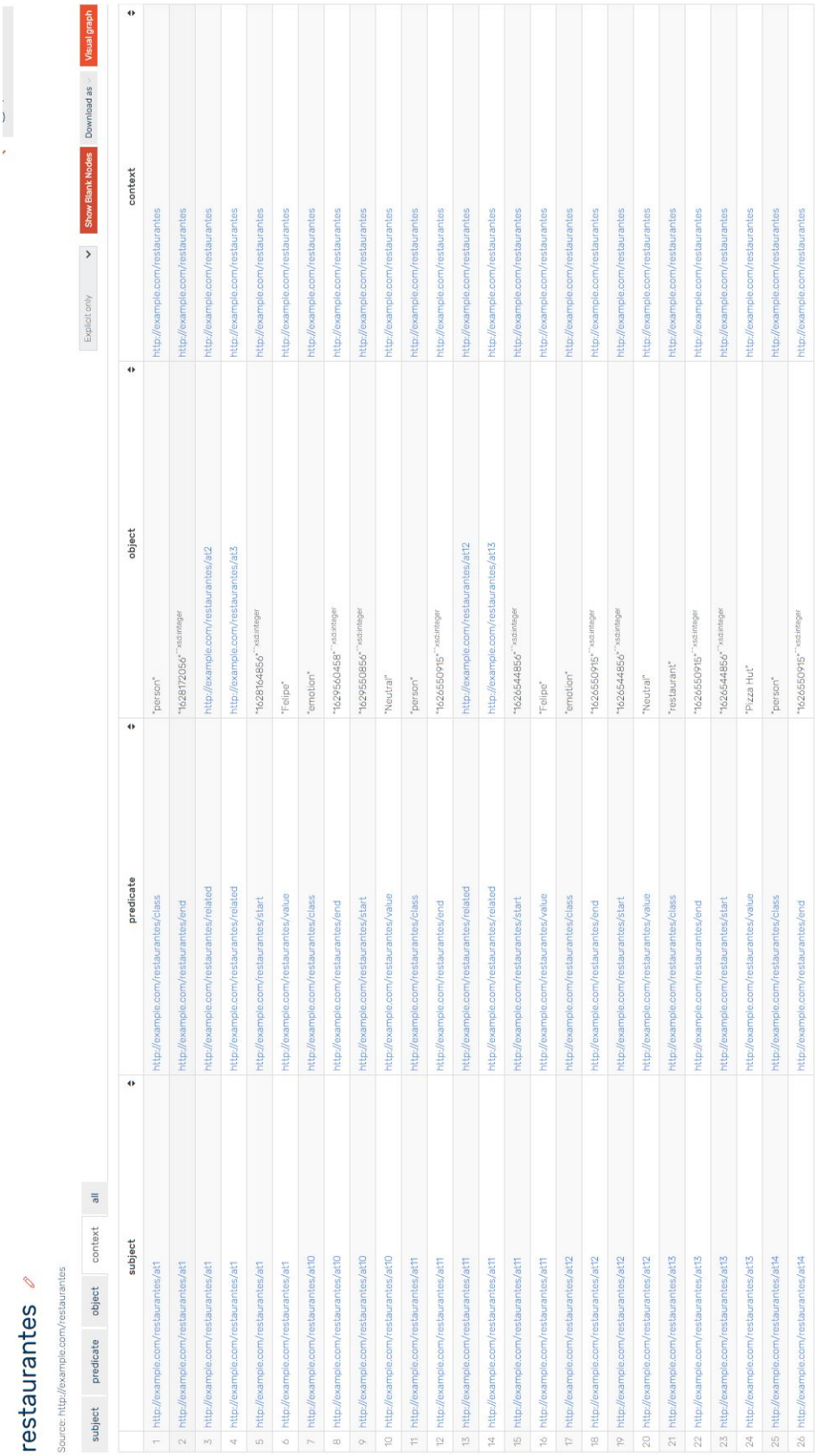
Fonte: Autoria própria, 2021

O módulo *SendResponse* recebe o restaurante e apresenta ao usuário, caso a sugestão seja rejeitada, seja requisitado uma nova ao módulo *SelectMemory*.

5.3 Link do código no GitHub

https://github.com/felipetsai/TCC-Memoria-Episodica

Figura 33: Eventos inseridos no GraphDB.



Fonte: Autoria própria, 2021

6 CONCLUSÕES

Neste trabalho foi feito um estudo e desenvolvimento de uma arquitetura de memória para os assistentes robóticos virtuais atuais, para torná-los mais adequados a atenderem as necessidades de indivíduos específicos. O foco deste estudo foi na memória do robô, para desenvolver um modo em que os assistentes robóticos sejam capazes de aprender com os acontecimentos presenciados. Ou seja, um modo de implementar a memória episódica nos assistentes virtuais. Nesse projeto, foi implementado um módulo de memória episódica para um assistente robótico capaz de aprender as preferências de restaurantes a partir das experiências do usuário, e a partir desse aprendizado ser capaz de sugerir opções de restaurantes.

O módulo desenvolvido se baseia nos átimos, recipientes que representam uma unidade de memória episódica. Cada átimo possui cinco características, o ID (número identificador), sua classe, seu valor, sua relação com outros átimos, e seu tempo de início e de término. Estes são armazenados em um banco de dados orientado a grafos, que em contraste a um banco de dados relacional, permite uma maior flexibilidade para uma adição futura de uma classe nova de átimos.

Para a implementação deste modelo foram separados em diversos módulos: *TakePicture*, *RecognizeFace*, *RecognizePerson*, *RecognizeEmotion*, *GetCoordinates*, *CreateAtimo*, *InsertAtimo*, *GetIntention*, *RequestMemory*, *SelectMemory*, *SendResponse*, *Assistant*. Os sete primeiros responsáveis para capturar as informações, transformando em átimos e inserindo no banco de dados. Já os quatro seguintes responsáveis receber a intenção do usuário, fazer a requisição dessa memória e devolvê-la a ele. Por último, tem o módulo *Assistant*, responsável por realizar a comunicação entre o os outros módulos.

Com os resultados obtidos a partir dos testes realizados, foi possível observar o modelo proposto em prática. A arquitetura proposta para a memória episódica se mostrou adequada para o trabalho realizado. Entretanto, com a implementação do módulo foi possível verificar pontos que poderiam ser melhorados e otimizados. Um dos pontos é o número de terminais utilizados, com o grande número de módulos presentes no modelo,

foi necessário muitos terminais abertos, diminuindo a performance do modelo. Umas das possibilidades para sua melhoria seria utilização de uma outra plataforma de mensageria, alguma que otimizaria esse processo. Outro ponto é que o tempo de execução dos módulos está muito acima do proposto nos requisitos do trabalho (um segundo), devido ao tempo de processamento de alguns módulos específicos, como *RecognizeEmotion* que pode chegar a mais de três segundos. Uma das possíveis soluções seria a utilização de processadores mais potentes.

Após o desenvolvimento do modelo, é possível observar que, em relação aos trabalhos definidos na tabela 1, é um modelo que pode ser mais versátil para armazenar os diferentes tipos de dados na memória episódica. A maioria destes estudos fez uso de um banco de dados relacional para o armazenamento, desse modo é necessário definir com antecedência os tipos de dados que serão armazenados. Para evitar isso, neste projeto foi utilizado um banco de dados orientado a grafos e inseridos recipientes (átimos) que podem ter diferentes tipos de dados. Por exemplo, para acrescentar um novo tipo de dado neste estudo, como tipo de comida, basta criar um átimo com essa classe. Já em um banco relacional, seria necessário criar uma coluna nova, alterando a estrutura da tabela.

Para os próximos trabalhos o objetivo é implementar os módulos que foram simulados neste trabalho, os módulos *GetCoordinates*, *GetIntention* e *SendResponse*. Além de aumentar a dimensão do estudo, utilizado alguma base de significados já existentes para a memória semântica, implementando o banco de dados na nuvem e otimizando os processos e o fluxo de mensageria. Após essas melhorias, o próximo passo seria desenvolver e montar um modelo robótico móvel capaz de capturar esses dados, inserir na nuvem, e retribuir os dados do banco de dados.

REFERÊNCIAS

[1] GOLD, P. E.; MCGAUGH, J. L. Changes in Learning and Memory During Aging. In: ORDY, J. M.; BRIZZEE, K. R. (Ed.). *Neurobiology of Aging: An Interdisciplinary Life-Span Approach*. Boston, MA: Springer US, 1975, (Advances in Behavioral Biology). p. 145–158. ISBN 978-1-4684-0925-3. Disponível em: <https://doi.org/10.1007/978-1-4684-0925-3_6>.

[2] INTUITION ROBOTICS LTD. *ElliQ, the sidekick for happier aging*. 2019. (Acesso em: 04/06/2021). Disponível em: <https://https://elliq.com/>.

[3] BLOOMBERG QUICKTAKE. *The Companion Robot Designer*. 2018. (Acesso em: 07/06/2021). Disponível em: <https://www.youtube.com/watch?v=eGySFLW0qDs>.

[4] TULVING, E. Episodic and semantic memory. In: *Organization of memory*. Oxford, England: Academic Press, 1972. p. xiii, 423–xiii, 423. Disponível em: <http://alumni.media.mit.edu/ jorkin/generals/papers/Tulving_memory.pdf>.

[5] TULVING, E. Episodic Memory: From Mind to Brain. *Annual Review of Psychology*, v. 53, n. 1, p. 1–25, 2002. _eprint: https://doi.org/10.1146/annurev.psych.53.100901.135114. Disponível em: <https://doi.org/10.1146/annurev.psych.53.100901.135114>.

[6] TULVING, E. Relations among components and processes of memory. *Behavioral and Brain Sciences*, v. 7, n. 2, p. 257–268, jun. 1984. ISSN 1469-1825, 0140-525X. Publisher: Cambridge University Press. Disponível em: <https://www.cambridge.org/core/journals/behavioral-and-brain-sciences/article/abs/relations-among-components-and-processes-of-memory/07FE861647F1D03C3D5E2EDC84D984A3>.

[7] TULVING, E.; MARKOWITSCH, H. J. Episodic and declarative memory: Role of the hippocampus. *Hippocampus*, v. 8, n. 3, p. 198–204, 1998. ISSN 1098-1063. https://onlinelibrary.wiley.com/doi/pdf/10.1002/%28SICI%291098-1063%281998%298%3A3%3C198%3A%3AAID-HIPO2%3E3.0.CO%3B2-G. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291098-1063%281998%298%3A3%3C198%3A%3AAID-HIPO2%3E3.0.CO%3B2-G>.

[8] Denise Filippo. *Pesquisa-ação em sistemas colaborativos*. 2011. (Acesso em: 07/06/2021). Disponível em: <http://sistemascolaborativos.uniriotec.br/wp-content/uploads/sites/18/2019/06/SC-cap26-pesquisaacao.pdf>.

[9] DAVISON, R.; MARTINSONS, M. G.; KOCK, N. Principles of canonical action research. *Information Systems Journal*, v. 14, n. 1, p. 65–86, 2004. ISSN 1365-2575. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2575.2004.00162.x. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2575.2004.00162.x>.

[10] YEE, E.; CHRYSIKOU, E. G.; THOMPSON-SCHILL, S. L. Semantic memory. In: *The Oxford handbook of cognitive neuroscience, Vol. 1: Core topics*. New York, NY, US: Oxford University Press, 2014, (Oxford library of psychology). p. 353–374. ISBN 978-0-19-998869-3.

[11] SAUMIER, P. D.; CHERTKOW, M. H. Semantic memory. *Current Neurology and Neuroscience Reports*, springer, v. 2, p. 512–516, 2002.

[12] NYBERG, L. et al. General and specific brain regions involved in encoding and retrieval of events: what, where, and when. *Proceedings of the National Academy of Sciences*, v. 93, n. 20, p. 11280–11285, out. 1996. ISSN 0027-8424, 1091-6490. Publisher: National Academy of Sciences Section: Research Article. Disponível em: <https://www.pnas.org/content/93/20/11280>.

[13] TULVING, E. *Elements of Episodic Memory*. [S.l.]: Oxford University Press, 1983.

[14] NIELSEN, J. *Memory and Amnesia by J.M. Nielsen: Very Good Hardcover (1958) | Casanova Books*. Disponível em: <https://www.abebooks.com/Memory-Amnesia-J.M-Nielsen-San-Lucas/1123961192/bd>.

[15] KASAP, Z.; MAGNENAT-THALMANN, N. Towards episodic memory-based long-term affective interaction with a human-like robot. In: *19th International Symposium in Robot and Human Interactive Communication*. [S.l.: s.n.], 2010. p. 452–457. ISSN: 1944-9437.

[16] KUPPUSWAMY, N. S.; CHO, S.-h.; KIM, J.-h. A Cognitive Control Architecture for an Artificial Creature using Episodic Memory. In: *2006 SICE-ICASE International Joint Conference*. [S.l.: s.n.], 2006. p. 3104–3110.

[17] SIGALAS, M.; MANIADAKIS, M.; TRAHANIAS, P. Episodic memory formulation and its application in long-term HRI. In: *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. [S.l.: s.n.], 2017. p. 599–606. ISSN: 1944-9437.

[18] YANG, C.-Y. et al. A Brain-Inspired, Self-Organizing Episodic Memory Model for a Memory Assistance Robot. *IEEE Transactions on Cognitive and Developmental Systems*, p. 1–1, 2021. ISSN 2379-8939. Conference Name: IEEE Transactions on Cognitive and Developmental Systems.

[19] HINTZMAN, D. L. 15 - MINERVA 2: A simulation model of human memory. *Behavior Research Methods, Instruments, &amp Computers*, v. 16, n. 2, p. 96–101, mar. 1984. ISSN 0743-3808, 1532-5970. Disponível em: <http://link.springer.com/10.3758/BF03202365>.

[20] PARK, G.-M.; KIM, J.-H. Deep Adaptive Resonance Theory for learning biologically inspired episodic memory. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2016. p. 5174–5180. ISSN: 2161-4407.

[21] ENCARNAÇÃO, P. et al. Episodic memory visualization in robot companions providing a memory prosthesis for elderly users. *Assistive Technology: From Research to Practice: AAATE 2013*, IOS Press, v. 33, p. 120, 2013.

[22] WORLD WIDE WEB CONSORTIUM. *Resource Description Framework (RDF)*. 2014. (Acesso em: 04/06/2021). Disponível em: <https://www.w3.org/2001/sw/wiki/Main$_{P}age$¿.

[23] AWS. *Getting started with graph databases*. 2021. (Acesso em: 07/06/2021). Disponível em: <https://docs.aws.amazon.com/neptune/latest/userguide/graph-get-started.html>.

[24] WORLD WIDE WEB CONSORTIUM. *SPARQL Working Group*. 2008. (Acesso em: 04/06/2021). Disponível em: <https://www.w3.org/2001/sw/DataAccess/homepage-20080115>.

[25] WORLD WIDE WEB CONSORTIUM. *SPARQL Query Language for RDF*. 2008. (Acesso em: 04/06/2021). Disponível em: <https://www.w3.org/TR/rdf-sparql-query/>.

[26] ONTOTEXT. *GraphDB*. 2021. (Acesso em: 04/06/2021). Disponível em: <https://graphdb.ontotext.com/documentation/free/about-graphdb.html>.

[27] Stanford Center for Biomedical Informatics. *Protégé*. 2016. (Acesso em: 18/07/2021). Disponível em: <https://protege.stanford.edu/>.

[28] Intel. *Open Source Computer Vision Library*. 2000. (Acesso em: 18/07/2021). Disponível em: <https://opencv.org/>.