Universidade da Beira Interior
Departamento de Informática

Inteligência Artificial – Relatório do Projeto Prático

Elaborado por:

Rúben Camilo dos Santos nº 41308

Rui Pedro de Almeida Barata nº 41872

Docente:

Professor Luís Filipe Barbosa de Almeida Alexandre

Índice

1. Índice de figuras

2. Introdução

O tema deste projeto é a criação e desenvolvimento da inteligência de um robô.

Consiste no robô ter a capacidade de responder a diversas questões para as quais necessita de se movimentar num mundo virtual, que corresponde a um piso de um hospital, identificar diversos objetos e diferenciar diferentes tipos de salas, médicos, enfermeiros e doentes.

Ao longo do tempo e das diversidades que o robô encontra, o agente vai aprendendo com as experiências e adaptando-se ao ambiente podendo responder às questões com mais certeza.

O desenvolvimento do código é em *Python*, que é uma linguagem de programação de alto nível, funcional, de tipagem dinâmica e forte.

3. Constituição do mundo

O mundo é constituído por quatro corredores que interligam as diversas salas que estão espalhadas pelo mundo. Nestas, podemos encontrar diferentes objetos desde camas, cadeiras, mesas e livros e também encontramos pelo mundo doentes, que são representados pela letra “D”, enfermeiros “E” e por médicos “M”.

O robô tem uma bateria que se vai descarregando ao longo do tempo, para a recarregar o robô pode aceder a um dos carregadores situados no mundo.

Existem diferentes tipos de divisões, que se distinguem pelos objetos que cada uma contêm. Um quarto tem no mínimo uma cama, uma sala de enfermeiros já não possui camas, mas sim cadeiras e mesas, e por último, uma sala de espera não tem mesas nem camas, mas tem mais de duas cadeiras.

4. Questões

4.1. Questão 1

"Qual foi a penúltima pessoa que viste?"

Para começar a resolver esta questão, analisamos os possíveis casos que o robô pode encontrar. Começamos por ver se o robô viu apenas uma ou nenhuma pessoa, pois neste caso seria impossível devolver a informação da penúltima pessoa que viu.

Caso já tenha visto no mínimo duas pessoas, o robô irá responder recorrendo à posição 1 da lista global "*people*" (*people[1]*).

Uma das dificuldades que nos deparámos ao longo da resolução desta questão foi quando o robo ainda não conhecer pessoas suficientes para a análise da situação. Outra dificuldade era quando o robô encontrava a mesma pessoa, este iria sempre adicioná-la à lista, por isso, quando uma pessoa se repete na lista, removemos a primeira vez que entrou na lista e adicionamo-la ao topo da mesma.

4.2. Questão 2

"Em que tipo de sala estás agora?"

Para a resolução desta questão, criámos um dicionário "*rooms*" que contêm os objetos presentes e outras informações de cada divisão, como o tipo de divisão, onde "*0*" significa que é um corredor, "*1*" que é um quarto, "*2*" uma sala de enfermeiros, "*3*" uma sala de espera e por último "*-1*" que é quando ainda não existe informação para concluir o tipo da mesma.

```
rooms = {   # dicionario que contem os objetos presentes em cada quarto e outras informacoes sobre cada sala
    1: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': 0},
    2: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': 0},
    3: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': 0},
    4: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': 0},
    5: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    6: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    7: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    8: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    9: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    10: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    11: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    12: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    13: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
    14: {'doente': [], 'medico': [], 'enfermeiro': [], 'cama': [], 'livro': [], 'cadeira': [], 'mesa': [], 'type': (-1)},
} # tipo de divisão: 0 -> corredor || 1 -> quarto || 2 -> sala de enfermeiros || 3 -> sala de espera || -1 -> unknown
```

Figura I - dicionário de todas as divisões

Através do tipo (parâmetro do dicionário) conseguimos analisar se a divisão é uma sala ou um corredor. Caso seja um corredor o robô responde que a divisão atual é um corredor. Caso não seja, iremos analisar os objetos que se encontram na divisão atual e consoante isso o robô consegue responder o tipo da divisão em que se situa. Caso o robô não tenha informação suficiente para responder em que sala está, ele responde dizendo que ainda não tem informação suficiente.

A função "*update_list*" é executada em todos as iterações da função "*work*", esta junta os objetos às listas corretas, de forma a que o robô possa ter mais informação para responder à questão.

A dificuldade principal foi saber como armazenar os objetos que existiam em cada divisão, para isso recorremos ao dicionário anterior (Figura I).

4.3. Questão 3

"Qual o caminho para a sala de enfermeiros mais próxima?"

Para responder a esta questão, criámos um grafo com 23 nodos, onde cada divisão é representada por um nodo e os corredores estão divididos em vários nodos, como demonstra a imagem seguinte (Figura II).

```
nodeBorders = { # bordas dos nodos
    "0": [(30,30), (245, 89)],
    "1_1": [(86,90), (245, 135)],
    "1_2": [(246,30), (395, 135)],
    "1_3": [(396,30), (564, 135)],
    "2_1": [(30,90), (85, 170)],
    "2_2": [(30,171), (85, 329)],
    "3_1": [(565,30), (635, 85)],
    "3_2": [(565,86), (635, 329)],
    "4_1": [(30,330), (85, 410)],
    "4_2": [(86,330), (245, 410)],
    "4_3": [(246,330), (395, 410)],
    "4_4": [(396,330), (580, 410)],
    "4_5": [(581,330), (770, 410)],
    "5": [(86,136), (245,329)],
    "6": [(246,136), (395,329)],
    "7": [(396,136), (564,329)],
    "8": [(636,30), (770,85)],
    "9": [(636,86), (770,185)],
    "10": [(636,186), (770,329)],
    "11": [(30,411), (245,570)],
    "12": [(246,411), (395,570)],
    "13": [(396,411), (580,570)],
    "14": [(581,411), (770,570)]
}
```

Figura II - dicionário com as bordas de cada nodo

Este é incializado com as ligações entre os corredores já criadas, pois estas são sempre constantes (Figura III).

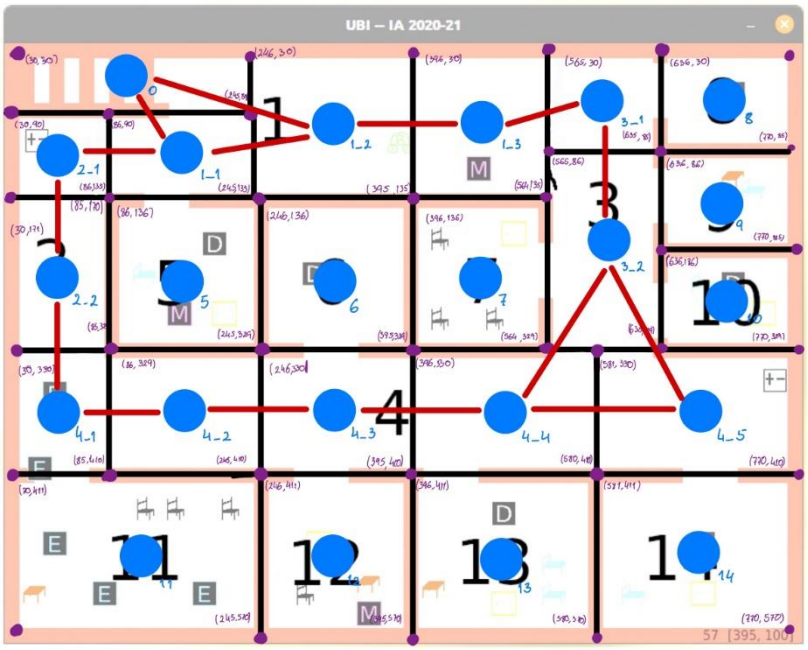

Figura III - grafo inicializado com as ligações entre corredores

Há medida que andamos de sala em sala vamos adicionando ligações ao grafo, desde o nodo origem, até ao nodo destino, fruto do agente aprender que pode entrar no nodo destino a partir do nodo origem.

Começamos por ver se o robô já se situa numa sala de enfermeiros, caso não se encontre em nenhuma, o robô vai comparar todos os caminhos para todas as salas de enfermeiros que conhece e escolhe o mais curto.

Em seguida caso a variável "*minRoom*" se encontre a "-1", significa que não encontrou ou ainda não conhece nenhuma sala de enfermeiros, caso contrário o robô imprime o melhor caminho para a mesma.

4.4. Questão 4

"Qual a distância até ao médico mais próximo?"

Criámos um dicionário que contêm para cada nodo um dicionário, com o nome e as coordenadas de todos os médicos da última vez que foram vistos dentro desse nodo.

Depois iremos percorrer este dicionário de modo a encontrar o caminho para o nodo mais próximo que contêm médicos presentes.

Analisamos se o nodo em que o robô está atualmente é o mesmo que o nodo anterior, caso seja, então o robô vai ver a distância de todos os médicos presentes no nodo até ele mesmo, escolhe o mais próximo e mostra a distância entre eles. Caso contrário, o robô vai calcular a soma das distâncias entre nodos, desde o nodo atual até ao penúltimo nodo no caminho e ainda a distância do penúltimo nodo até ao médico mais próximo deste.

A grande dificuldade na resolução desta questão foi a proximidade dos médicos ao centro do último nodo daí a nossa resolução descrita anteriormente.

4.5. Questão 5

"Quanto tempo achas que demoras a ir de onde estás até às escadas?"

Inicialmente calculamos a velocidade do robô no mundo mal ele efetue um movimento (em qualquer eixo), através da seguinte fórmula "$v = \frac{d}{\Delta t}$". A distância é calculada desde o nodo atual do robô até ao nodo das escadas. Utilizando a fórmula anterior, em função do intervalo de tempo, conseguimos assim calcular uma estimativa do tempo que o robô demora a ir da sua posição atual até às escadas.

Encontrar a velocidade aproximada do robô foi a principal dificuldade para esta questão.

4.6. Questão 6

"Quanto tempo achas que falta até ficares sem bateria?"

Para esta questão recorremos à função "*curve_fit*" do módulo "*scipy.optimize*" que encontra os melhores valores para podermos ajustar uma função exponencial ($y = e^{(a + b \times x)}$, em que x é uma dada percentagem de bateria e y é o tempo que o robô demorou a gastar a bateria desde os 100% até um dado x, logo $\{x \in \mathbb{N};\ x \geq 2 \text{ e } x \leq 100\}$) tendo em conta os dados que o robô recolhe da sua bateria ao longo da execução do programa.

Uma das dificuldades que tivemos ao resolver esta questão foi o facto de a bateria não descarregar de forma linear, pois quanto menos bateria tem, mais devagar descarrega. Para resolvermos esta questão decidimos tentar aproximar a reta que descreve o "gasto" da bateria.

Outra dificuldade foi o facto de a bateria descarregar mais rápido ou mais devagar quando o robô está a andar ou parado respetivamente. Para termos em conta os movimentos que o robô faz ao longo do programa, decidimos executar testes ao longo da execução programa, assim o robô vai aprendendo como funciona a sua bateria tendo em conta o seu movimento e podendo aproximar cada vez melhor o tempo restante que tem. Desda Figura IV à Figura VII mostramos como o robô aprende com os dados que recolhe e vai aproximando a estimativa ao valor real logo no primeiro ciclo da bateria. A 1% de bateria o robô foi carregado, podemos ver então a diferença quando o robô estimou o tempo restante que tinha a 90% (51026948.92s na Figura IV) apenas com 10 pontos de dados e quando já tinha 101 pontos de dados, a 100% (117.77s na Figura VII).

É normal não ser “perfeito” logo de inicio, pois o robô ainda não tem conhecimeento suficiente sobre a sua bateria.

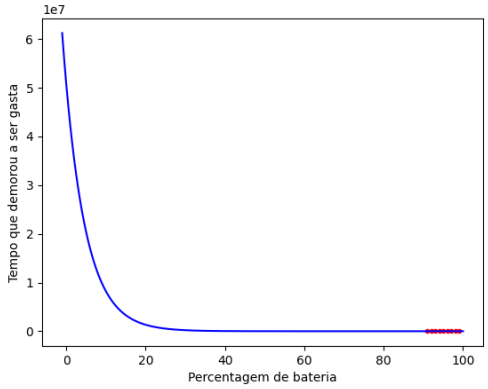

Figura IV - 90% de bateria restante

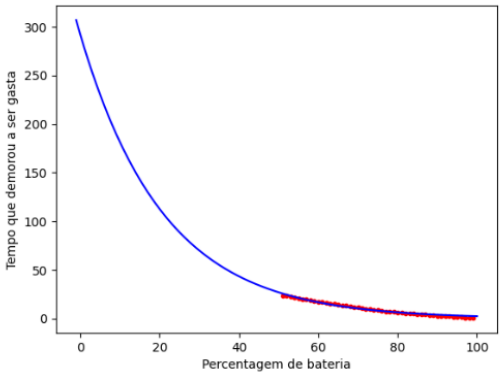

Figura V - 50% de bateria restante

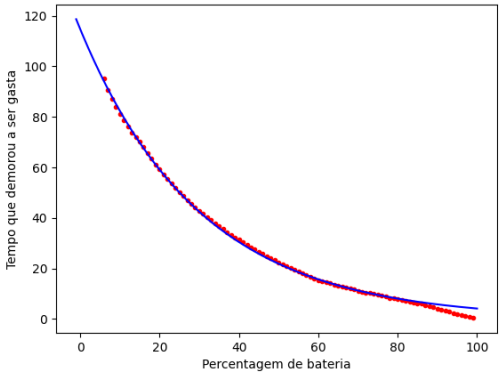

Figura VI - 5% de bateria restante

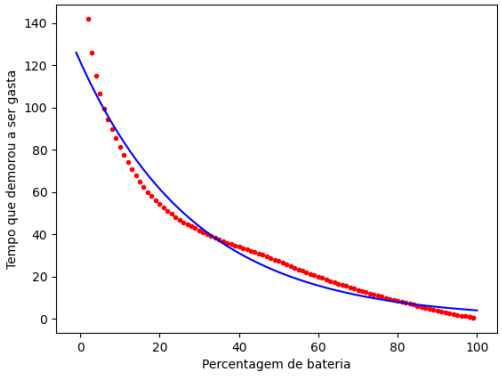

Figura VII - 100% de bateria restante

4.7. Questão 7

“Qual a probabilidade de encontrar um livro numa divisão, se já encontraste uma cadeira?”

Interpretamos esta questão como uma probabilidade condicionada, onde vamos calcular a probabilidade de encontrar um livro numa divisão sabendo que já encontramos uma cadeira.

Começamos por percorrer todas as divisões que não são corredores e ir ver em quais delas contêm uma cadeira e um livro em simultâneo, em caso afirmativo aumentamos os casos favoráveis de encontrar um livro e uma cadeira (“*cfLC*”). Também vamos analisar os casos favoráveis de encontrar cadeiras numa sala (“*cfC*”).

Por último vamos calcular a probabilidade dividindo os casos favoráveis de ter livros e cadeiras pelos casos favoráveis de ter cadeiras, dando assim a probabilidade condicionada.

4.8. Questão 8

"Se encontrares um enfermeiro numa divisão, qual é a probabilidade de estar lá um doente?"

Interpretamos também esta questão como uma probabilidade condicionada, onde vamos calcular a probabilidade de encontrar um doente numa divisão sabendo que já encontramos um enfermeiro.

Aqui também vamos ver em quantas das divisões encontramos um doente e um enfermeiro, ou seja, calcular a probabilidade da interseção de ambos ($P(Doente \cap Enfermeiro)$) e dividir esta pela probabilidade de encontrar enfermeiro/s, resultando a condicionada referida anteriormente.

5. Divisão das tarefas

Para a realização deste trabalho como eram pedidas a resolução de oito questões, decidimos que cada elemento do grupo realizava quatro tarefas, sendo o trabalho dividido da seguinte forma:

Questão [1], Rúben Santos;

Questão [2] , Rui Barata;

Questão [3], Rui Barata;

Questão [4], Rúben Santos;

Questão [5], Rui Barata;

Questão [6], Rui Barata;

Questão [7], Rúben Santos;

Questão [8], Rúben Santos.

6. Conclusão

O presente projeto teve como objetivo principal desenvolver a inteligência de um robô de modo a conseguir responder de forma eficaz e correta a questões que lhe são propostas.

O robô ao longo das suas execuções e das suas experiências vai adquirindo novas competências e conhecimentos de forma a adaptar-se ao ambiente em questão (piso de um hospital) com o objetivo de aprender e estar apto a resolver situações que inicialmente não conseguiria.

De uma forma geral, é possível considerar que o robô está funcional para responder a todas as questões propostas correspondendo ao que é pedido.

7. Bibliografia

NetworkX Developers. (27 de 12 de 2020). *Tutorial*. Obtido de NetworkX - Network Analysis in Python: https://networkx.org/documentation/stable/tutorial.html

Okada, S. (05 de 01 de 2021). *Modeling Functions - From linear to logistic regression*. Obtido de Towards Data Science: https://towardsdatascience.com/modeling-functions-78704936477a#7b2b

Python Software Foundation. (03 de 01 de 2021). *Data Structures*. Obtido de Python: https://docs.python.org/3/tutorial/datastructures.html

Sceenivasan, S. (14 de 12 de 2020). *How to implement a switch-case statement in Python*. Obtido de jaxenter: https://jaxenter.com/implement-switch-case-statement-python-138315.html

tsznxyz. (03 de 01 de 2021). *How to fit math function to the dataset in Python*. Obtido de stack overflow: https://stackoverflow.com/questions/35779869/how-to-fit-math-function-to-the-dataset-in-python