Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Introdução à Inteligência Artificial
1o semestre de 2025
Profa. Gisele L. Pappa

TRABALHO PRÁTICO 1
AGENTES CONVERSACIONAIS DE BUSCA

Data de entrega: 18/05

Este trabalho é individual.

Este projeto tem como objetivo implementar um agente conversacional utilizando a biblioteca *smolagents*. O agente irá interagir com o mundo usando um grande modelo de linguagem que, a partir de um conjunto de instruções, deve ser capaz de responder ao usuário e, quando necessário, realizar o planejamento de uma tarefa utilizando um algoritmo de busca.

https://huggingface.co/learn/agents-course/unit0/introduction

Os algoritmos de busca podem ser usados em diversas tarefas, desde jogos até aplicações de redes. Neste trabalho, o agente será especializado em sugerir rotas dentro da cidade de Belo Horizonte em **um contexto específico**. Por exemplo, você pode traçar rotas para restaurantes, bares, pontos turísticos, ou outros estabelecimentos da sua escolha. Uma lista de possíveis tipos de estabelecimentos disponíveis pode ser encontrada aqui.[1]

A proposta é usar algoritmos de busca para criar rotas entre destinos solicitados pelo usuário, via LLM. Caso o destino não esteja no seu mapa, o modelo deve informar que a localização está fora de alcance.

Para gerar o grafo onde a busca ocorrerá, utilize a biblioteca `osmnx`[2]. Como vamos trabalhar com grafos não-direcionados, utilize um grafo para caminhar, e não para dirigir, conforme exemplo abaixo. Para informações mais detalhadas, veja o link[3]:

```
import osmnx as ox
G = ox.graph.graph_from_place("belo horizonte - MG", network_type="walk")
```

Para trabalhar com um tipo específico de local, por exemplo, **estacionamentos (*parking*)**, você precisa identificar o nó no grafo mais próximo do estabelecimento seguindo o código a seguir, detalhado neste link[4]:

---

[1] https://wiki.openstreetmap.org/wiki/Map_features#Building
[2] https://osmnx.readthedocs.io/en/stable/getting-started.html
[3] https://github.com/gboeing/osmnx-examples/blob/main/notebooks/03-graph-place-queries.ipynb
[4] https://github.com/gboeing/osmnx-examples/blob/main/notebooks/16-download-osm-geospatial-features.ipynb

```
features = ox.features.features_from_place(place, {"amenity": "parking"})
feature_points = features.representative_point()
nn = ox.distance.nearest_nodes(G, feature_points.x, feature_points.y)
useful_tags = ["access", "parking", "surface", "capacity", "fee"]
for node, feature in zip(nn,
features[useful_tags].to_dict(orient="records")):
    feature = {k: v for k, v in feature.items() if pd.notna(v)}
    G.nodes[node].update({"parking": feature})
```

Portanto, se o usuário solicitar uma rota entre um ponto A e B, primeiramente verifique se ambos os pontos existem, e depois gere a rota entre os nós A e B.

Tendo o grafo, você deverá encontrar caminhos usando pelo menos 2 algoritmos: um de busca sem informação e um de busca com informação ou busca local.

Figura 1: Grafo das ruas de Belo Horizonte.

Para facilitar a visualização, apenas as vias primárias, secundárias e terciárias foram plotadas. Para gerar o mesmo gráfico, basta usar o filtro custom_filter='["highway"~"primary|secundary|tertiary"]'.

Em resumo, sugiro que você siga da seguinte forma:

1. Escolher com que tipo de locais irá trabalhar;
2. Criar o grafo usando as diretivas das páginas anteriores;
3. Usar pelo menos 2 algoritmos para traçar as rotas: um de busca sem informação e um de busca com informação ou busca local.
4. Implementar o agente, que deverá chamar seu algoritmo de rotas.

O que deve ser entregue:

1. Código fonte: O agente precisa ser implementado em python, e a parte de busca pode ser implementada em Python, c, c++, Java. Se você conseguir rodar o agente online, o link (Não obrigatório).
2. Documentação: uma documentação simplificada, com: (de 3-5 páginas)
    a. Descrição do problema que você irá trabalhar, incluindo o número de nós, arestas e destinos desejados, e lista dos algoritmos implementados. (Por exempo: problema de rotas para pontos turísticos, grafo com 2000 nós e 8000 arestas, 134 pontos turístico, testado com algoritmos X e Y);
    b. Desenho da arquitetura que você utilizou para integrar agente/busca;
    c. Um subconjunto de prompts testados e seus resultados;
    d. Análise dos resultados do teste com o prompt usando os 2 algoritmos implementados.