UNIVERSIDADE FEDERAL DE MINAS GERAIS
Departamento de Ciência da Computação

Selene Melo Andrade

AGENTE CONVERSACIONAL DE BUSCA COM LLM

Roteamento entre parques e pontos de interesse de BH

Documentação do trabalho prático I da disciplina DCC – Introdução a Inteligência Artificial
Curso: Sistemas de Informação
Professora: Gisele L. Pappa

Belo Horizonte

2025

1) Descrição do Problema

Este trabalho prático, desenvolvido para a disciplina “Introdução à Inteligência Artificial”, tem como objetivo implementar um agente conversacional capaz de planejar rotas urbanas em Belo Horizonte, com foco nos parques da cidade. O sistema responde a perguntas em linguagem natural, identificando rotas entre bairros, locais e parques, além de destacar os parques presentes ao longo do trajeto. Dessa forma, o sistema permite que um usuário formule perguntas como “qual o caminho do bairro União ao bairro Buritis?”, “qual o parque mais próximo da UFMG” e receba como resposta tanto um percurso textual detalhado quanto uma visualização gráfica da rota. O sistema também destaca os parques presentes ao longo do trajeto, integrando-os à resposta final.

A implementação utilizou a biblioteca smolagents, integrando-a a modelos de linguagem grande (LLMs) locais via Ollama, como o Qwen2:7b, e posteriormente com testes no Qwen3:8b. Para a manipulação e aquisição dos dados geográficos da cidade, foi empregada a biblioteca OSMnx, que permitiu a construção de um grafo representando as vias urbanas de Belo Horizonte classificadas como primárias, secundárias e terciárias. Esse grafo resultou em uma estrutura com aproximadamente 15 mil nós e 40 mil arestas, além de 117 parques mapeados e indexados. O principal desafio foi integrar linguagem natural, algoritmos de busca e visualização gráfica de forma eficiente e funcional, o que foi superado com sucesso, resultando em uma ferramenta prática e aderente às exigências do projeto.

2) Dados do grafo

Para representar a malha urbana de Belo Horizonte, foi utilizado um grafo extraído do OpenStreetMap por meio da biblioteca OSMnx. Visando eficiência, foram mantidas apenas as vias classificadas como *primary*, *secondary* e *tertiary*, resultando em um grafo enxuto com cerca de 3.800 nós e 11.600 arestas. Essa filtragem garante boa cobertura da cidade, com foco em caminhos realistas para deslocamentos a pé.

O grafo é não direcionado, permitindo percursos em ambos os sentidos, e foi enriquecido com 117 parques identificados e vinculados a nós próximos. Um processo de filtragem adicional removeu locais que não se enquadravam no escopo de "parques urbanos", como praças, quadras e áreas técnicas. Essa estrutura fornece a base para o cálculo das rotas e identificação de pontos de interesse ao longo dos trajetos.

3) Algoritmos implementados

O sistema implementa dois algoritmos distintos de busca para o cálculo de rotas urbanas no grafo de Belo Horizonte: Dijkstra e A***.** Essa escolha visa atender ao requisito de utilizar pelo menos um algoritmo de busca sem informação (Dijkstra) e outro de busca com informação (A*), também conhecida como busca heurística.

O algoritmo de Dijkstra foi escolhido por sua robustez e confiabilidade ao calcular o caminho mais curto em grafos ponderados, baseando-se exclusivamente nos pesos das arestas (no caso, a distância real das ruas). Ele é um algoritmo clássico de busca cega, ou seja, não utiliza qualquer estimativa de onde está o objetivo — apenas expande os caminhos mais curtos conhecidos até encontrar o destino.

Já o A* foi implementado como uma alternativa informada, que tenta reduzir o número de nós explorados ao estimar a distância restante até o objetivo. Para isso, utilizamos como heurística **a** distância haversine, uma fórmula que calcula a distância geográfica entre dois pontos na superfície da Terra com base em suas coordenadas de latitude e longitude. Ela assume

um modelo esférico do planeta e é muito comum em aplicações de geolocalização por fornecer uma boa aproximação do "caminho em linha reta" entre dois pontos — ou seja, a menor distância possível entre eles ignorando obstáculos. Essa heurística é considerada admissível (nunca superestima a distância real) e consistente (respeita a desigualdade triangular), o que garante que o A* encontrará a solução ótima — a mesma que Dijkstra encontraria — mas, em muitos casos, com menos esforço computacional.

Durante os testes, verificamos que tanto Dijkstra quanto A* retornaram rotas idênticas na maioria das consultas. Isso pode ser atribuído ao fato de que o grafo é relativamente bem conectado e não muito profundo, e que a heurística haversine, embora eficiente, não chega a "guiar" a busca por rotas significativamente diferentes. Mesmo assim, manter as duas abordagens implementadas cumpre o requisito do trabalho e ilustra claramente a diferença conceitual entre estratégias de busca informada e não informada.

4) Arquitetura do sistema e fluxo

O sistema é construído sobre a biblioteca *smolagents*, que permite a integração entre modelos de linguagem natural e ferramentas necessária para a extração e cálculo de rotas. A principal ferramenta implementada é a RouteTool, responsável por todo o processo de resolução da tarefa: ela realiza a geocodificação dos locais de origem e destino, carrega o grafo da cidade, calcula as rotas utilizando os algoritmos de busca, extrai os nomes das ruas do trajeto e identifica os parques que estão próximos ao caminho.

O grafo de Belo Horizonte é carregado apenas uma vez e reutilizado em todas as chamadas graças à estrutura *singleton* BHGraphManager, que garante eficiência de tempo e memória. O modelo de linguagem Qwen2:7b é executado localmente via Ollama e configurado para gerar chamadas de função (*tool calls*) com base nos prompts recebidos. A resposta retornada pela ferramenta já é formatada como resposta final, ou seja, o modelo não precisa interpretar, reprocessar ou modificar o conteúdo retornado.

Durante os testes, também foi realizada uma tentativa de executar o modelo Qwen3:8b, visando obter melhores capacidades de compreensão e generalização. No entanto, o ambiente local disponível contava com apenas 8 GB de memória RAM e 7 GB de placa de vídeo, o que se mostrou insuficiente para executar esse modelo de forma eficiente. Mesmo em prompts simples, a execução ultrapassava 600 segundos, tornando a experiência inviável para uso prático. Assim, optou-se por manter o Qwen2:7b como modelo padrão, dado seu desempenho mais estável e compatível com os recursos computacionais disponíveis.

Além da resposta textual estruturada, o sistema também gera dinamicamente uma imagem contendo o mapa da cidade com a rota traçada e, quando houver, destaca visualmente os parques próximos ao trajeto, oferecendo uma representação gráfica complementar para o usuário.

Adicionalmente, foi desenvolvida uma interface gráfica com o uso da biblioteca *Tkinter*, que encapsula toda a interação com o agente de forma mais acessível e amigável. A janela permite que o usuário insira sua pergunta em linguagem natural, clique em um botão para enviar a solicitação e visualize a resposta formatada, sem exposição dos logs internos de execução. Isso melhora a experiência de uso ao esconder informações técnicas como execução de código, tokens e etapas do modelo, apresentando apenas o diálogo final. A interface simula uma conversa simples e direta entre humano e agente, tornando o sistema mais aplicável a usuários finais.

O fluxo completo do sistema pode ser descrito como:

Usuário envia uma solicitação em linguagem natural → o modelo de linguagem interpreta o prompt e gera a chamada da ferramenta → a ferramenta processa a rota e formata o resultado → o modelo apenas exibe a resposta final ao usuário por texto e imagem → opcionalmente, essa resposta é apresentada em uma janela Tkinter com entrada e saída estilizadas para uso interativo.

5) Instruções de execução

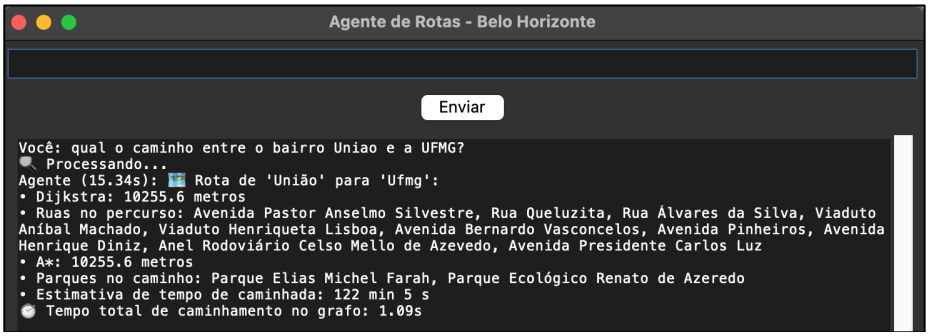
No diretório raíz onde o programa foi instalado:

- Certifique-se que as biblioteca smolagents e osmnx estão corretamente instaladas

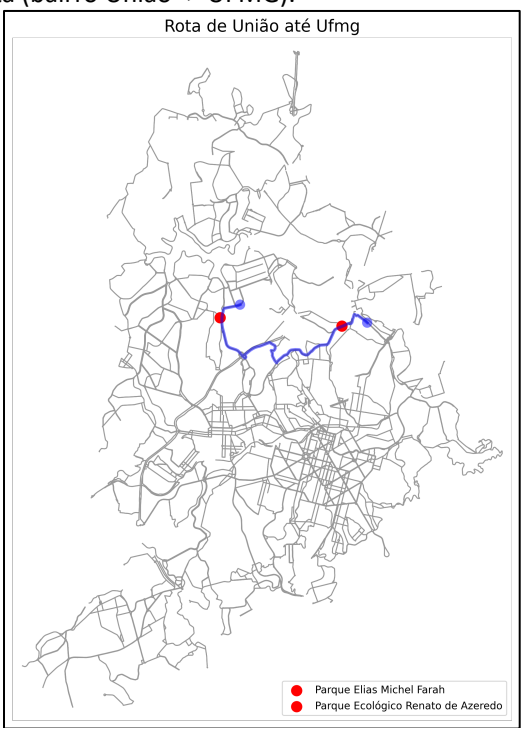
```
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt (dentro do ambiente virtual)
pip install 'smolagents[litellm]' (dentro do ambiente
virtual)
```

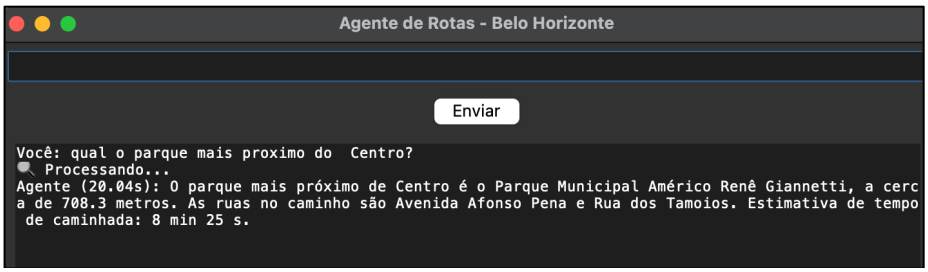
6) Subconjunto de prompts testados e resultados

- Testes realizados no modelo Qwen2-7b

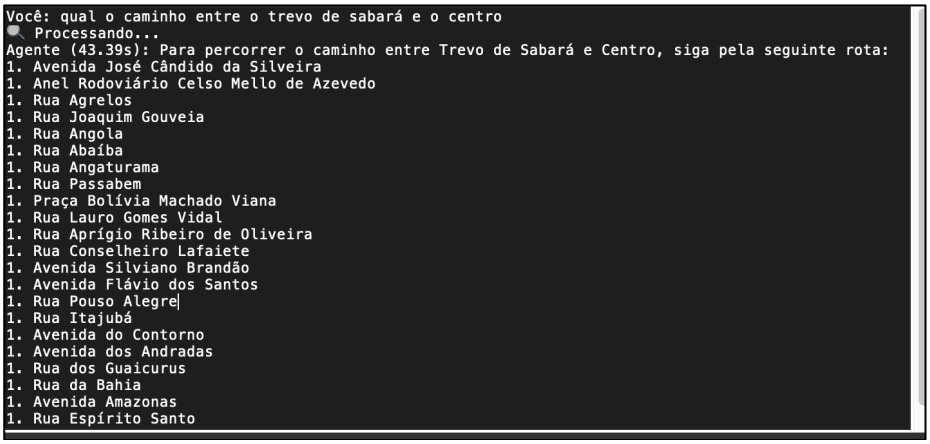


- Visualização da rota (bairro União -> UFMG):

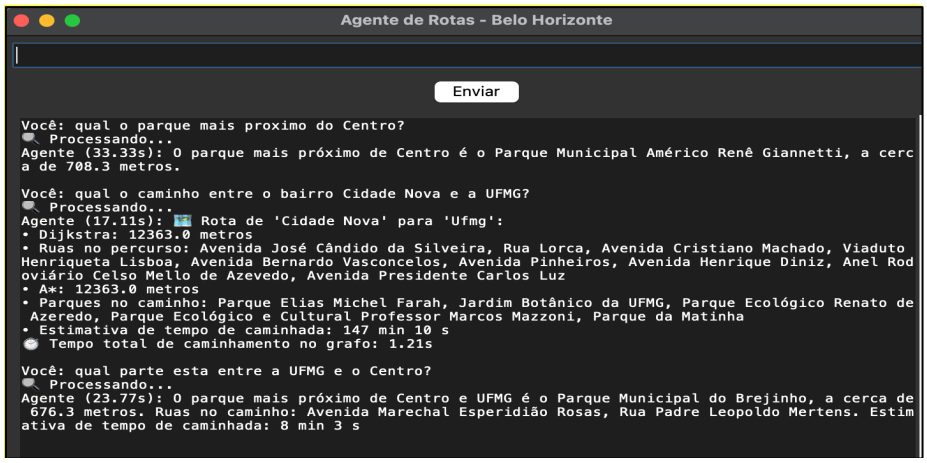




- Caso na condição de contorno: Localidade fora do grafo mapeado de Belo Horizonte (Encontra o nó mais próximo no grafo)



- Caso na condição de contorno: Usuário digitou algo errado (Modelo converge para o esperado)



7) Eficiência dos algoritmos

Ambos os algoritmos implementados — Dijkstra e A* — apresentaram bom desempenho no contexto específico do grafo filtrado de Belo Horizonte. Como o grafo foi reduzido para apenas 3834 nós e 11668 arestas, as buscas são executadas em menos de 3 segundos mesmo para trajetos longos entre bairros distantes.

O algoritmo de Dijkstra, apesar de ser mais custoso por expandir uniformemente todos os caminhos de menor custo, mostra-se eficiente nesse cenário graças à limitação do tamanho do grafo. Já o algoritmo A*, ao utilizar a heurística de distância haversine, explora menos nós em

trajetos longos e, por isso, geralmente tem desempenho ligeiramente melhor, embora ambos retornem rotas de mesmo custo e extensão na maioria dos casos.

8) Desempenho e limitações

O desempenho do sistema foi otimizado ao máximo com os recursos disponíveis. O tempo de resposta médio para consultas comuns fica entre 1 e 4 segundos, com picos de até 10 segundos para consultas que exigem geocodificação complexa ou visualização de mapas. O uso de cache e estrutura singleton evita o recarregamento do grafo, reduzindo drasticamente o tempo de inicialização..

Entretanto, o sistema apresenta algumas limitações:

- **Geocodificação imprecisa**: alguns nomes populares de locais, como "Praça Sete de Setembro" e avenidas grandes (localização pontual imprecisa), podem não ser reconhecidos de imediato, exigindo tentativas alternativas de localização. Funciona melhor para bairros na cidade.
- **Limite semântico**: o modelo Qwen2:7b não compreende totalmente a semântica de expressões mais ambíguas, podendo ignorar determinados comandos mesmo com a ferramenta corretamente implementada.
- **Respostas reprocessadas**: o agente às vezes tenta reanalisar ou filtrar as respostas retornadas pela ferramenta, o que gera erros como AttributeError ao tentar acessar .get() em strings formatadas.
- **Dependência de internet na primeira execução**: o carregamento do grafo e dos parques depende do acesso ao OpenStreetMap, embora o cache local evite múltiplas requisições.
- **Capacidade computacional**: o sistema foi projetado para rodar localmente, mas modelos maiores como o Qwen3:8b não puderam ser utilizados devido à limitação de memória (8 GB de RAM e 7 GB de VRAM), o que restringe a capacidade de compreensão semântica em prompts mais complexos.

9) Conclusão

O projeto atendeu com sucesso ao objetivo proposto: desenvolver um agente conversacional capaz de interpretar linguagem natural e fornecer rotas urbanas detalhadas na cidade de Belo Horizonte, com foco temático em parques. A solução é funcional, eficiente e apresenta tanto uma interface textual estruturada quanto uma visualização gráfica que destaca os elementos mais relevantes do trajeto.

Além da funcionalidade básica de rotas, o sistema consegue identificar o parque mais próximo de um ponto qualquer da cidade, listar os parques no caminho entre dois locais e exibir tempo estimado de caminhada, ruas percorridas e imagem gerada com o mapa da rota. A implementação com a biblioteca smolagents permitiu integrar facilmente a lógica do agente com os modelos de linguagem via Ollama, enquanto o uso do Tkinter tornou a experiência mais acessível para usuários finais.

As limitações observadas são pontuais e, em sua maioria, podem ser superadas com maior capacidade computacional ou ajustes finos no parsing de linguagem natural. O projeto demonstra claramente o potencial da integração entre geoprocessamento, inteligência artificial e interação humano-máquina para aplicações urbanas reais.