AskDB: Sistema de Conversação com Bancos de Dados Baseado em Agentes LLM sem Fine-tuning

1st Caio Rocha
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
caio.rocha@dcc.ufmg.br

2nd Gisele L. Pappa
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
glpappa@dcc.ufmg.br

***Resumo*—A tradução de linguagem natural para SQL (NL-to-SQL) representa um desafio fundamental para democratizar o acesso a bancos de dados relacionais, pois a sintaxe SQL exige conhecimento técnico que limita usuários não-especialistas. Este trabalho apresenta o AskDB, uma solução com Large Language Models (LLMs) sem fine-tuning baseada em uma arquitetura multi-agente modular, na qual agentes especializados colaboram em um grafo de execução condicional para roteamento de perguntas, decomposição semântica, geração automática de metadados e correção iterativa de consultas SQL. Metadados são extraídos automaticamente por LLMs, fornecendo contexto estruturado que melhora significativamente a precisão das consultas geradas. Em experimentos com o banco de dados AACT (mais de 50 tabelas de ensaios clínicos reais), o sistema alcançou 68% de acurácia média em 30 perguntas testadas. As principais contribuições incluem: (1) uma arquitetura multi-agente reutilizável; (2) sistema automático de metadados estruturados; (3) pipeline iterativo de correção de SQL; e (4) um framework extensível e de fácil manutenção. Os resultados demonstram a viabilidade de soluções NL-to-SQL acessíveis e adaptáveis, reduzindo custos e ampliando o acesso de profissionais não-técnicos.**

***Index Terms*—NL-to-SQL, agentes LLM, metadados estruturados, correção iterativa, bancos de dados relacionais, processamento de linguagem natural, arquitetura multi-agente**

I. Introdução

O crescimento cada vez maior no uso de dados relacionais em domínios especializados como saúde, finanças e comercio expande em paralelo a uma barreira significativa para usuários não-técnicos. A linguagem SQL, embora poderosa e amplamente utilizada, requer conhecimento técnico específico que limita o acesso democratizado aos dados. Este trabalho aborda o problema fundamental: como permitir que pesquisadores, médicos, analistas de dados e outros profissionais acessem informações complexas de bancos de dados através de perguntas em linguagem natural?

O problema de tradução de linguagem natural para SQL (NL-to-SQL) tem sido objeto de intensa pesquisa na comunidade de processamento de linguagem natural e banco de dados. Sistemas tradicionais baseados em regras e templates apresentam limitações significativas em cenários complexos [1], especialmente quando lidam com esquemas multi-tabela e relacionamentos complexos. O banco de dados AACT (Aggregate Analysis of ClinicalTrials.gov) [18], muito utilizada durante o desenvolvimento deste projeto, exemplifica esses desafios, possuindo mais de 50 tabelas relacionadas com dependências complexas entre estudos, intervenções, condições e resultados.

Modelos de linguagem como GPT-4, Claude e Llama demonstram capacidades impressionantes em compreensão de linguagem natural, mas enfrentam limitações específicas em dados abulares e tarefas NL-to-SQL [2]: compreensão de esquemas de banco de dados complexos, geração de SQL sintaticamente correto, e correção iterativa de erros. Abordagens baseadas em fine-tuning, embora eficazes, apresentam custos elevados de desenvolvimento e baixa flexibilidade para diferentes domínios.

A decisão de não utilizar fine-tuning é fundamentada em múltiplos fatores práticos e econômicos. Primeiro, o custo de desenvolvimento de datasets específicos para fine-tuning pode exceder as centenas de milhares de dólares para domínios complexos, incluindo anotação manual, validação e iterações. Segundo, a flexibilidade para adaptação a novos esquemas de banco de dados requer retreinamento completo, resultando em custos operacionais contínuos. Por fim, a dependência de dados de treinamento específicos limita a aplicabilidade em domínios com dados sensíveis ou restritos e a manutenção de modelos fine-tuned requer expertise especializada e infraestrutura computacional dedicada.

O AskDB propõe uma abordagem alternativa que supera essas limitações através de uma arquitetura multi-agente especializada sem necessidade de fine-tuning. O sistema utiliza engenharia de prompts estruturada, geração automática de metadados estruturados, e um grafo de execução condicional que permite fluxo adaptativo baseado no tipo de pergunta e resultados intermediários. A integração multi-API (Groq, Claude, AwanLLM) garante robustez e performance, enquanto a arquitetura modular permite extensibilidade e manutenibilidade.

As principais contribuições deste trabalho incluem: (1) uma arquitetura multi-agente reutilizável para NL-to-SQL sem fine-tuning; (2) sistema de geração automática de metadados e anotações de bancos de dados; (3) pipeline de correção iterativa de erros SQL; (4) software modular e facilmente expansível.

O restante deste artigo está organizado da seguinte forma: a Seção II apresenta o referencial teórico e trabalhos relaciona-

dos. A Seção III descreve a abordagem proposta, detalhando a arquitetura multi-agente e o sistema de metadados. A Seção IV apresenta os detalhes da implementação, incluindo o stack tecnológico e a arquitetura de agentes. A Seção V descreve a avaliação experimental. A Seção VI apresenta uma discussão crítica dos resultados, identificando limitações e oportunidades de melhoria. Finalmente, a Seção VII conclui o trabalho e apresenta direções futuras.

II. Trabalhos Relacionados

A tarefa de tradução de linguagem natural para SQL (NL-to-SQL) tem evoluído ao longo das últimas décadas, acompanhando os avanços em processamento de linguagem natural e aprendizado de máquina. Inicialmente dominada por sistemas baseados em regras e templates, a área avançou com o uso de modelos neurais especializados e, mais recentemente, com a adoção de Large Language Models (LLMs), que ampliaram a capacidade de generalização frente à crescente complexidade de esquemas de bancos de dados [1].

A. Abordagens Tradicionais e com Fine-Tuning

Métodos tradicionais baseados em regras apresentavam baixo custo computacional e previsibilidade, mas falhavam em lidar com ambiguidades linguísticas e estruturas relacionais complexas. Com o advento dos modelos pré-treinados (PLMs), como BERT e RoBERTa, surgiram abordagens com fine-tuning supervisionado que obtiveram bons resultados em benchmarks como o Spider [11]. No entanto, essas abordagens exigem datasets anotados, sofrem com generalização entre domínios e impõem altos custos operacionais.

A literatura recente apresenta dois paradigmas principais para LLMs aplicados ao NL-to-SQL: o *fine-tuning*, geralmente usado com modelos open-source, e o *in-context learning* (ICL), empregado em LLMs proprietários como GPT-4 e Claude. Enquanto o primeiro oferece maior controle e adaptação a domínios específicos, o segundo permite generalização sem ajuste de parâmetros.

B. Sistemas Baseados em Agentes LLM

O trabalho de Liang et al. [4] apresenta uma categorização dos sistemas de agentes LLM, dividindo-os em agentes digitais, físicos e híbridos. Esses sistemas utilizam LLMs como núcleo de raciocínio e planejamento, auxiliados por módulos de memória, ferramentas externas (APIs) e mecanismos de verificação. Embora o foco seja amplo, a estrutura modular discutida fundamenta-se em princípios similares aos adotados no AskDB, como a orquestração de agentes especializados para tarefas específicas.

C. Avanços Recentes com LLMs em NL-to-SQL

A pesquisa de Hong et al. fornece uma boa visão do o uso de LLMs em NL-to-SQL, abordando tanto estratégias com fine-tuning quanto com ICL [1]. São destacados métodos como DIN-SQL [7], que aplica decomposição da consulta e autocorreção iterativa; DAIL-SQL, que utiliza descrições semânticas para auxiliar a execução; MAC-SQL, que divide a tarefa entre múltiplos agentes especializados; e SuperSQL, que combina dados brutos e metadados com raciocínio passo a passo.

Essas abordagens demonstram que o uso de raciocínio iterativo, decomposição e agentes auxiliares melhora significativamente a precisão e a robustez. No entanto, muitas delas dependem de acesso a APIs proprietárias, múltiplas chamadas custosas ou conhecimento especializado para engenharia de prompts complexos — fatores que podem dificultar sua adoção em ambientes restritos.

Apesar dos avanços, dois desafios persistem: (1) como operar em domínios especializados sem datasets rotulados para fine-tuning; e (2) como lidar com esquemas complexos de bancos de dados usando apenas LLMs genéricos. O AskDB busca suprir essa lacuna ao propor uma arquitetura modular baseada em agentes especializados, combinando o uso de LLMs com engenharia de prompts e metadados automaticamente extraídos do banco. Essa abordagem elimina a necessidade de fine-tuning, reduz custos e permite maior adaptabilidade a diferentes domínios.

III. Abordagem Proposta

A. O Sistema Proposto

O princípio central do AskDB é a decomposição do problema NL-to-SQL em agentes especializados que colaboram através de um grafo de execução. A arquitetura de agentes é implementada através de classes especializadas que encapsulam lógica específica para cada tipo de agente. Agentes LLM utilizam prompts estruturados com templates dinâmicos, enquanto agentes funcionais implementam lógica determinística para tarefas específicas. O grafo de execução é implementado através de classes que gerenciam transições condicionais entre agentes, permitindo assim um fluxo adaptativo que se basea nos resultados intermediários de cada agente. O sistema de estado compartilhado que foi implementado permite que informações sejam passadas entre agentes de forma prática.

A arquitetura cognitiva dos agentes segue o padrão estabelecido na literatura de agentes LLM, composta por três componentes essenciais: o modelo de linguagem como tomador de decisões central, as ferramentas que conectam o agente ao mundo externo, e a camada de orquestração que governa o processamento de informações e raciocínio.

Cada agente implementa um 'ciclo cognitivo' que inclui observação do estado atual, raciocínio baseado em prompts especializados, e ação através de ferramentas específicas. O Question Router Agent, por exemplo, observa a pergunta do usuário, utiliza prompt estruturado para classificação, e retorna a categoria identificada. Este padrão se repete em todos os agentes, com variações específicas para cada tipo de tarefa. Essa camada de orquestração implementa as ideias por trás de frameworks de raciocínio como ReAct (Reasoning and Acting) [10], permitindo que os agentes decomponham problemas complexos em passos menores e tomem decisões baseadas no encontrado.

A escolha de JSON como formato de comunicação entre agentes é fundamentada em múltiplos fatores. JSON oferece

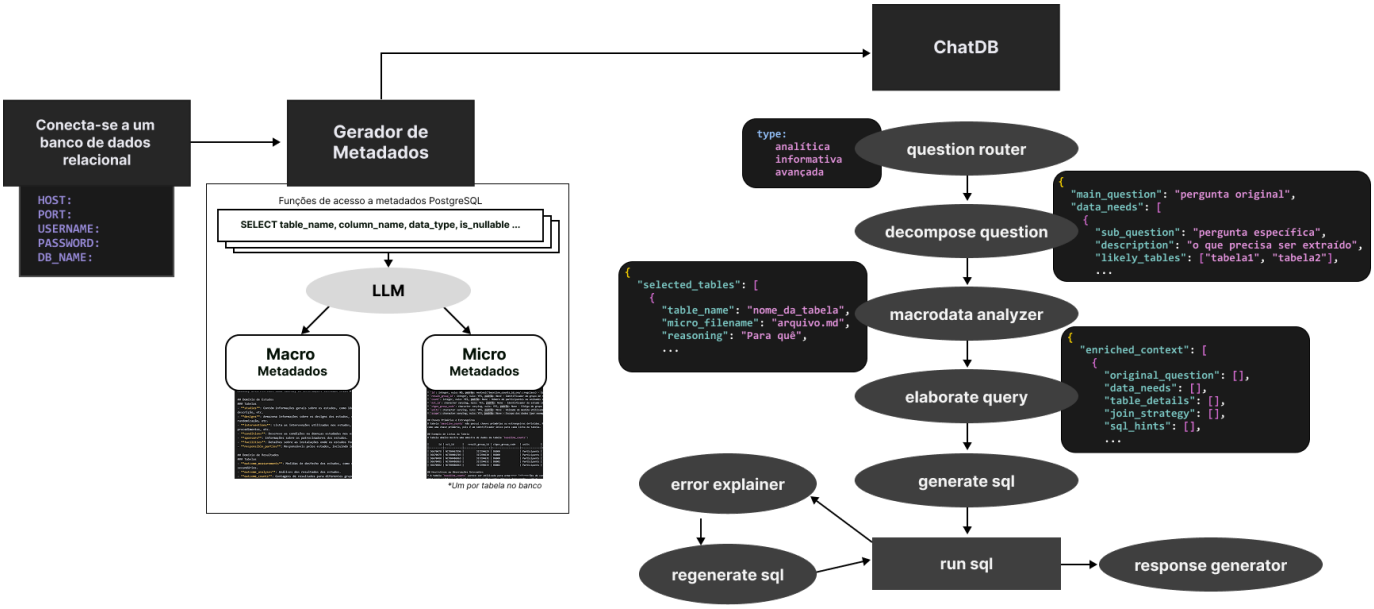

Figura 1. Arquitetura do AskDB

estrutura hierárquica que permite representação complexa de dados sem perda de informação. Além disso, a compatibilidade nativa com LLMs [2] facilita a geração e interpretação de respostas estruturadas. E por último, a legibilidade humana facilita debug e desenvolvimento.

A implementação do passo-a-passo (step-by-step) é motivada pela necessidade de decompor problemas complexos em sub-problemas gerenciáveis. Esta abordagem segue o framework Chain-of-Thought [9], que demonstra superioridade em tarefas que requerem raciocínio estruturado. Cada agente recebe contexto específico e retorna resultados estruturados que alimentam o próximo agente, criando um pipeline de processamento que maximiza a precisão e permite correção iterativa.

B. Sistema de Metadados

O sistema de metadados é composto por dois níveis complementares:

Metadados Macro: Fornece visão geral do banco, categorização de tabelas, e relacionamentos principais. É gerado automaticamente via LLM com análise de esquema, armazenado em formato estruturado e Markdown descritivo. Inclui informações sobre domínios de dados, padrões de relacionamento, e heurísticas de uso.

Metadados Micro: Oferece descrição detalhada de cada tabela, colunas, tipos, e amostras de dados. É gerado também via LLM, para cada tabela, armazenado em formato estruturado Markdown. Aqui se inclui estatísticas de distribuição, padrões de valores, e exemplos de uso ou exemplares dedados.

C. Prompts Especializados

Os prompts são projetados para maximizar a eficácia de cada agente através de engenharia cuidadosa de prompts. Question Router Prompt classifica perguntas com exemplos específicos, Decompose Question Prompt realiza análise semântica detalhada, Generate SQL Prompt implementa estratégias de otimização, e Error Explainer Prompt fornece análise detalhada de erros.

IV. Detalhes da Implementação

O sistema utiliza Python 3.10+ como linguagem principal, FastAPI como framework web, PostgreSQL com asyncpg para banco de dados, integração com múltiplas APIs de LLM (Groq, Claude, AwanLLM), processamento assíncrono, e sistema de logging centralizado customizado. A escolha de tecnologias assíncronas é motivada pela necessidade de processamento paralelo de múltiplas consultas e integração eficiente com APIs externas.

A integração multi-API é implementada através de classes especializadas que abstraem diferenças entre provedores de LLM. APIs suportadas incluem Groq para velocidade otimizada, Claude para qualidade superior, e AwanLLM como fallback. Hiperparâmetros são configuráveis também, permitindo otimizações específica para cada tipo de tarefa e agente. O sistema implementa retry automático e fallback entre APIs para garantir robustez.

A. Sistema de Geração Automática de Metadados

O sistema de geração de metadados representa uma inovação significativa na abordagem NL-to-SQL, criando automaticamente descrições estruturadas do banco de dados que servem como contexto para os agentes LLM. Este processo é dividido em duas fases principais: geração de metadados macro e micro.

A geração de metadados macro inicia com a extração automática do esquema do banco de dados através de consultas SQL nos metadados do schema do banco. O sistema coleta

informações sobre tabelas, colunas, tipos de dados, chaves primárias e estrangeiras, e relacionamentos. Estas informações são então processadas por um LLM que gera uma descrição textual da visão geral do banco, categorização de tabelas por domínio, e mapeamento de relacionamentos principais. A geração de metadados micro é realizada tabela por tabela, criando descrições detalhadas que incluem propósito de cada coluna, padrões de dados, estatísticas de distribuição, e exemplos de uso. O sistema também gera amostras de dados representativas que ajudam os agentes a compreender o contexto e formato dos dados.

O processo utiliza prompts bem cosntruídos que guiam o LLM na análise do esquema, garantindo que as descrições sejam consistentes, úteis e adequadas para o contexto NL-to-SQL. Os metadados gerados são armazenados em formato estruturado (JSON) e descritivo (Markdown), permitindo fácil acesso e interpretação pelos agentes tanto em nível de código (o JSON) quanto para input nas prompts dos agentes (o Markdown).

B. Sistema de Banco de Dados

A interface de banco de dados implementa operações assíncronas com sanitização automática de queries, resultados estruturados, e tratamento robusto de erros. A sanitização inclui correções automáticas de sintaxe comum e validações de segurança. O sistema suporta múltiplos tipos de banco de dados através de adaptadores especializados, permitindo portabilidade entre diferentes sistemas de gerenciamento de banco de dados.

V. Avaliação Experimental

O ambiente de testes utilizou o banco de dados AACT (Aggregate Analysis of ClinicalTrials) com mais de 50 tabelas relacionadas, dados de ensaios clínicos reais, e esquema complexo com múltiplos

O banco AACT representa um cenário real e desafiador, com dados de ensaios clínicos de múltiplos países, diferentes tipos de intervenções, e relacionamentos complexos entre estudos, participantes, e resultados.

Alguns outros testes foram intensamente conduzidos no banco de dados do sistema de gestão integrado utilizado na prefeitura de Cascavel, mas devido à natureza privada dos dados os mesmos não foram utilizados na experimentação final, apenas servindo como apoio e feedback de desenvolvimento.

A. Resultados

Os resultados foram analisados de forma qualitativa, dentre as 30 perguntas criadas para testes e repetidas 20 vezes cada a taxa de acerto final foi de 68% (20,4). Mais experimentos e testes, bem como a aplicação de benchmarks são necessários, porém as limitações financeiras de uso das APIs de LLM introduziram uma limitação na condução dos experimentos. Um estudo subsequente focado nos testes pode expandir a consolidar os resultados deste sistema.

VI. Discussão e Análise

A taxa de sucesso de 68% indica que há espaço significativo para melhoria, três padrões principais foram identificados inicialmente: (1) dificuldade com joins complexos envolvendo múltiplas tabelas, (2) interpretação incorreta de perguntas ambíguas, e (3) geração de SQL sintaticamente incorreto ou com erros pequenos com repetitivos. A eficácia da abordagem baseada em agentes, observada nos entendimentos intermediários das perguntas, apresenta vantagens significativas mas também desafios como latência acumulativa e complexidade de debug. O impacto dos metadados estruturados foi significativo e mostra que metadados melhoram a seleção de tabelas e relacionamentos.

As principais **limitações** incluem: (1) dependência da qualidade dos prompts, que pode variar significativamente entre diferentes LLMs, (2) dificuldade em lidar com esquemas de banco de dados altamente normalizados, (3) falta de compreensão contextual profunda para perguntas que requerem conhecimento de domínio específico, e (4) sensibilidade a variações na formulação de perguntas. Os principais **pontos fortes** incluem flexibilidade para adaptação a diferentes esquemas sem fine-tuning, robustez através do sistema de correção iterativa eficaz, escalabilidade através da arquitetura modular que permite extensões, e manutenibilidade através de agentes independentes e reutilizáveis.

VII. Conclusão

O objetivo principal de desenvolver um sistema NL-to-SQL-to-NL baseado em agentes sem fine-tuning foi alcançado com sucesso. Resultados concretos incluem o framework modular e reutilizável implementado, um sistema de geração de anotações e metadados automático funcional, pipeline iterativo com auto-correção na geração de SQL, e avaliação experimental em cenário real, embora limitada. O sistema funciona e responde questões ao usuário.

O sistema AskDB demonstra que é possível construir soluções NL-to-SQL funcionais sem recorrer ao fine-tuning, utilizando uma arquitetura baseada em agentes especializados e metadados estruturados. Os resultados experimentais, embora modestos com 68% de taxa de sucesso, validam a viabilidade da abordagem proposta e abrem caminho para aplicações práticas em diversos domínios.

A combinação de engenharia de prompts estruturada, sistema de metadados automático e correção iterativa representa uma contribuição ao campo de processamento de linguagem natural aplicado a bancos de dados. No entanto, é importante reconhecer que esta abordagem apresenta limitações significativas em termos de performance comparada a sistemas fine-tuned, e que a escolha entre as duas abordagens deve ser baseada nas prioridades específicas de cada aplicação.

O AskDB representa um passo importante na direção de sistemas NL-to-SQL mais flexíveis e acessíveis, oferecendo uma alternativa viável ao fine-tuning em cenários onde a adaptabilidade e o custo são prioritários. Os resultados demonstram tanto o potencial quanto as limitações desta abordagem.

REFERÊNCIAS

[1] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, "Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL," arXiv preprint arXiv:2406.08426v5, 2025.

[2] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, and C. Faloutsos, "Large Language Models (LLMs) on Tabular Data: Prediction, Generation, and Understanding—A Survey," Transactions on Machine Learning Research, Jul. 2024. arXiv preprint arXiv:2402.17944v4 [cs.CL].

[3] F. Lei, J. Chen, Y. Ye, R. Cao, D. Shin, H. Su, Z. Suo, H. Gao, W. Hu, P. Yin, V. Zhong, C. Xiong, R. Sun, Q. Liu, S. I. Wang, and T. Yu, "SPIDER 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows," arXiv preprint arXiv:2411.07763v2, 2025.

[4] G. Liang, "LLM-Powered AI Agent Systems and Their Applications in Industry," arXiv preprint arXiv:2505.16120v1, 2025.

[5] Z. Qin, C. Luo, Z. Wang, H. Jiang, and Y. Sun, "Relational Database Augmented Large Language Model," arXiv preprint arXiv:2407.15071v1, 2024.

[6] Z. Qiu, Y. Peng, G. He, B. Yuan, and C. Wang, "TQA-Bench: Evaluating LLMs for Multi-Table Question Answering with Scalable Context and Symbolic Extension," arXiv preprint arXiv:2411.19504v1, 2024.

[7] T. Scholak, N. Schucher, and D. Bahdanau, "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction,"in Advances in Neural Information Processing Systems, 2023.

[8] M. Pourreza and D. Rafiei, "MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL,"in Proceedings of ACL, 2023.

[9] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models,"in Advances in Neural Information Processing Systems, 2022.

[10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models,"in International Conference on Learning Representations, 2023.

[11] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task,"in Proceedings of EMNLP, 2018.

[12] A. Fokoue, S. Jayaraman, E. Khabiri, J. O. Kephart, Y. Li, D. Shah, Y. Drissi, F. F. Heath III, A. Bhamidipaty, F. A. Tipu, and R. J. Baseman, "A System and Benchmark for LLM-based QA on Heterogeneous Data,"arXiv preprint arXiv:2409.05735, 2024.

[13] S. Patil, T. Zhang, X. Wang, and J. E. Gonzalez, "Gorilla: Large Language Model Connected with Massive APIs,"arXiv preprint arXiv:2305.15334, 2023.

[14] OpenAI, "GPT-4 Technical Report,"arXiv preprint arXiv:2303.08774, 2023.

[15] Anthropic, "Claude 3.5 Sonnet,"Technical Report, 2024.

[16] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and Efficient Foundation Language Models,"arXiv preprint arXiv:2302.13971, 2023.

[17] LangChain, "LangChain Documentation,"2024.

[18] ClinicalTrials.gov, "Aggregate Analysis of ClinicalTrials.gov (AACT),"2024.

[19] PostgreSQL Global Development Group, "PostgreSQL Documentation,"2024.

[20] FastAPI, "FastAPI Documentation,"2024.

[21] asyncpg, "An async PostgreSQL driver for Python,"2024.

[22] Groq, "Groq API Documentation,"2024.