Projeto Mineração de Dados

Catarina Costa, Marta Aguiar, Rita Dantas

Mestrado em Engenharia Informática, Universidade do Minho.

Contributing authors: pg526762@alunos.uminho.pt;
pg52694@alunos.uminho.pt; pg51605@alunos.uminho.pt;

Abstract

Este artigo introduz o **Champi**, um chatbot desenvolvido para auxiliar utilizadores iniciantes em fitness a criar rotinas de exercícios personalizadas. Este projeto dividiu-se em três etapas: recolha das fontes de dados, treino do modelo, testes e avaliações.

Palavras-chave: Chatbot, exercício físico, Fitness, IA, Ollama, Llama3, Structure Chat Agent, RAG, Pinecone.

1 Introdução

No âmbito da UC de **Mineração de Dados**, foi-nos proposto a elaboração de um projeto, cujo o objetivo consistiu no desenvolvimento de uma aplicação com a integração de um Large Language Model (**LLM**). A partir disto, decidimos desenvolver o "*Champi*".

Champi é um *chatbot* assistente de treino que cria planos de treino personalizados para iniciantes. Também sugere exercícios e desafios de acordo com idade, peso, objetivos do utilizador, etc.

Os objetivos deste *chatbot* são a poupança de tempo por parte do utilizador, uma vez que este não terá de que criar as routinas manualmente, a precisão nos treinos que são gerados, que serão avaliados através de métricas e por fim, a liberdade do utilizador puder mudar o plano de acordo com as suas preferências de exercícios e atributos físicos.

2 Estado da Arte

Nesta seção, são apresentados alguns projetos já existentes que abordam o mesmo tópico que o nosso, explorando o estado e as soluções disponíveis atualmente.

2.1 FitnessAI

FitnessAI é uma app desenhada para iPhone que usa inteligência artificial para gerar planos de treino personalizados. Esta app tem em base 5,9 milhões de treinos que a IA usa para otimizar séries, repetições e pesos para cada exercício sempre que o utilizador treina.

A app permite guardar todos os planos de treino do utilizador e também faz uso de um algoritmo que indica os tempos de repouso entre os treinos e ainda ajustar a dificuldade dos exercícios conforme desejado.

2.2 taskade

Taskade é um gerador de planos de treino personalizados que utilizada uma IA como ferramenta para criar os planos. Considera fatores como os objetivos de fitness, preferências de exercício e atributos físicos do utilizador, para fornecer uma rotina de treino personalizada que se adaptada às suas necessidades.

Em termos de objetivos, o gerador ressoa com os do nosso projeto na medida em que permite poupar tempo ao utilizador, que não precisa de criar rotinas manualmente. Além disso, a precisão do gerador de treinos, que utiliza algoritmos para analisar objetivos de fitness, preferências de exercício e atributos físicos, permite criar rotinas de treino precisas e relevantes. Por fim, oferece ainda variedade na quantidade de exercícios.

3 Métodos

Após a escolha do tema do projeto, procedemos à pesquisa e recolha de fontes de dados. Estas fontes de dados foram utilizadas, no projeto, para fornecer dados de conhecimento sobre o tema ao LLM.

Realizamos uma pesquisa sobre o desenvolvimento de *chatbots* para identificar as ferramentas necessárias para a nossa aplicação. Com base nesta análise, concluímos que precisaríamos de escolher os seguintes componentes:

- **Fontes de dados**
- **Modelo de Linguagem**
- **Base de Dados Vetorial**

3.1 Ferramentas

Para o desenvolvimento e implementação do nosso *chatbot* selecionamos um conjunto específico de ferramentas essenciais para garantir a eficácia e funcionalidade do assistente de treino.

Cada ferramenta desempenha um papel crucial, desde a recolha de dados até a geração de respostas pelo modelo de linguagem.

3.1.1 Fontes de Dados

Uma das partes mais importantes deste projeto foi a recolha de dados, devido ao facto de o modelo necessitar de ser posteriormente treinado com os dados fornecidos. Por essa razão, a escolha dos dados foi extremamente crucial. Desta forma, dedicamos um tempo significativo para selecionar dados com conteúdo relevante e necessário.

Para esta seleção tivemos em consideração os fatores importantes para o nosso modelo ser treinado, como o tipo de exercício físico (cardio, musculação, etc.), o nível de treino ser para iniciantes, a duração de toda a rotina, a lista dos diferentes exercícios, entre outros.

Desta forma, foram selecionados dois datasets **CSV** obtidos do *kaggle* com exercícios de fitness e ginástica e um conjunto de **PDFs**, obtidos através do site MS, com exemplos planos de treino somente para iniciantes, organizados por dias ou por semanas, dependendo da duração do plano.

Para além dessas informações, estes PDFs também contém informações acerca do(s) músculo(s) trabalhados em cada exercício, a duração total do treino, o tempo por cada sessão e o equipamento necessário.

A partir dos ficheiros PDFs foram extraídas informações textuais para serem utilizadas como *metadata*, o que contribuiu para enriquecer as respostas geradas pelo modelo. A utilização de *metadata* no modelo de linguagem serviu para invocar uma resposta do chat, o que melhorou significativamente a precisão e relevância das respostas fornecidas.

A *metadata* incluiu informações contextuais, como os dias para cada exercício da rotina completa, a semana a que pertence cada exercício, a lista de exercício, a parte do corpo exercitada e o equipamento utilizado.

Modelo de Linguagem (LLM)

O modelo escolhido para ser aplicado no nosso projeto foi o modelo "*llama3*" (Large Language Model Meta AI) através da ferramenta *Ollama*.

Escolhemos este modelo devido ao seu design "decoder-only" e ao facto de ele conseguir suportar 128 mil tokens o que torna muito eficiente na codificação da linguagem. Também este modelo foi integrado cerca de 8 bilhões e 70 bilhões de parâmetros treinado, o faz com que tenha um processamento bastante focado e ágil.

Com este modelo foi possível gerar os embeddings com a função *OllamaEmbeddings*, que mais tarde são inseridos na base de dados vetorial (este processo é explicado mais à frente), e a função *ChatOllama* que serve para invocar uma resposta do *chatbot*.

3.1.2 Base de Dados Vetorial

Quanto à base de dados vetorial, utilizamos a ***Pinecone***, que é uma solução de armazenamento e pesquisa vetorial rápida e escalável. Pinecone possui a capacidade de armazenar e pesquisar vetores de alta dimensionalidade, o que foi essencial para lidar com os embeddings gerados pelo modelo de linguagem.

Utilizamos *embeddings* gerados pelo modelo *llama3* para representar semanticamente os conteúdos dos documentos. Esses embeddings foram, então, indexados na base de dados vetorial Pinecone, permitindo buscas rápidas e precisas baseadas na similaridade do conteúdo fornecido pelo utilizador.

3.1.3 Chat

Para a implementação do chat, utilizamos a biblioteca *LangChain*, que facilita a criação de aplicações que integram modelos de linguagem de grande porte. LangChain fornece ferramentas para manipulação de texto, integração de modelos e construção de fluxos de trabalho complexos que utilizam modelos de linguagem.

3.1.4 *Structured Chat Agent*

Adicionamos ainda um *agent* de chat estruturado ao nosso projeto com o objetivo de dar continuidade ao *chatbot* de acordo com mensagens anteriores, usando um mecanismo de histórico. Este *agent* utiliza um modelo de linguagem para manter o contexto da conversa, garantindo que as respostas do chatbot sejam coerentes com as interações passadas.

Na figura 1 encontra-se um exemplo de uma segunda resposta do *chatbot* após o utilizador ter solicitado uma rotina de fitness e a aplicação ter retornado uma resposta. O utilizador volta a enviar outra mensagem a fazer uma referência à mensagem anterior do chat e é enviado o histórico de mensagens anteriores para que seja possível para o *agent* ter conhecimento do contexto necessário para fornecer uma resposta ao utilizador. Neste exemplo, o utilizador pergunta o que tem que fazer no dia 3 (relativamente ao plano de treino) e o *agent* fornece a resposta de forma correta.

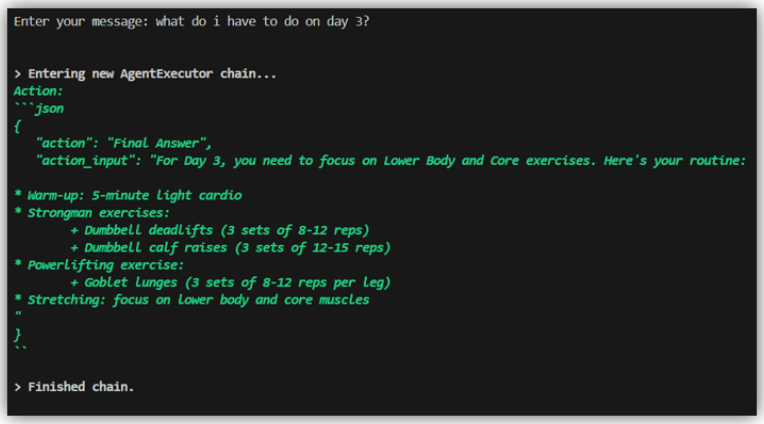

Fig. 1: Exemplo de resposta do *agent*.

3.2 Arquitetura

A arquitetura do nosso sistema é composta por vários componentes integrados para fornecer uma experiência de chat robusta e eficiente. A seguir, descrevemos os principais componentes:

- **Extração de Documentos e Metadata**: Carregamento e divisão dos documentos PDF e CSV em chunks menores através das ferramentas *PyPDFLoader* e *RecursiveCharacterTextSplitter* da biblioteca LangChain; análise do conteúdo dos documentos para extrair *metadata* relevante, como dias da semana, semanas, tipos de exercícios, partes do corpo e equipamentos.
- **Base de Dados Vetorial**: Transformação dos documentos em embeddings e armazenamento dos embeddings dos documentos para a base de dados Pinecone.
- **Extração de contexto**: Realizada uma pesquisa de similaridade da query do utilizador na base de dados para fornecer ao modelo de linguagem os documentos mais dentro do contexto.
- **Modelo de Linguagem**: Utilização do modelo *llama3* através da ferramenta *Ollama* para gerar respostas contextualmente relevantes através da implementação de RAG (Retrieval-Augmented Generation), que é explicado de seguida.
- ***Chatbot* e *Structure Chat Agent***: Utilização da biblioteca LangChain para gerir as interações de chat e manter o contexto da conversa.

Na figura 2 está representada a arquitetura do nosso projeto de acordo com as descrições descritas anteriormente. Cada processo está identificado com uma cor.

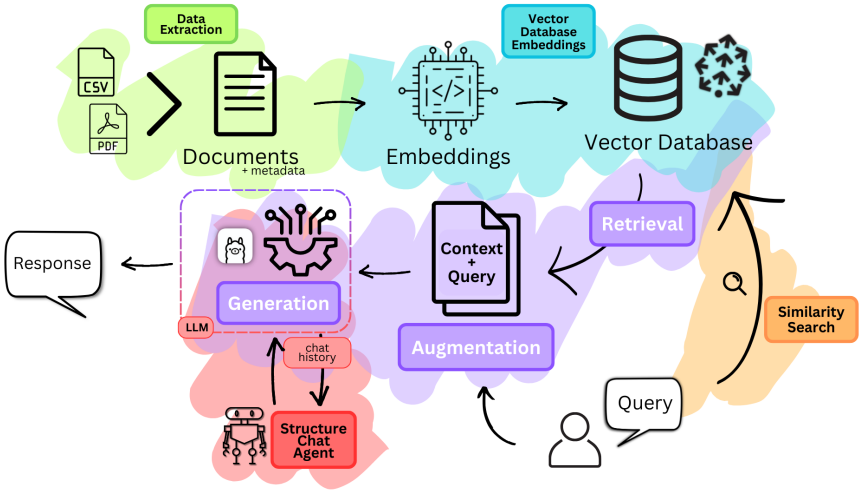

Fig. 2: Arquitetura do projeto.

4 Resultados e Discussão

Nesta secção são ilustrados e abordados os resultados provenientes do *chatbot*.

A figura 3 é um exemplo de mensagens trocadas entre o *chatbot* e o utilizador, em que este envia um pedido de um plano de treino para uma mulher de 25 anos iniciante em fitness e o chat devolve uma resposta. Após isso, o utilizador volta a solicitar uma pergunta relativamente ao plano e o chatbot, com ajuda do structure chat agent, referido anteriormente, consegue responder a esta mensagem de forma correta.

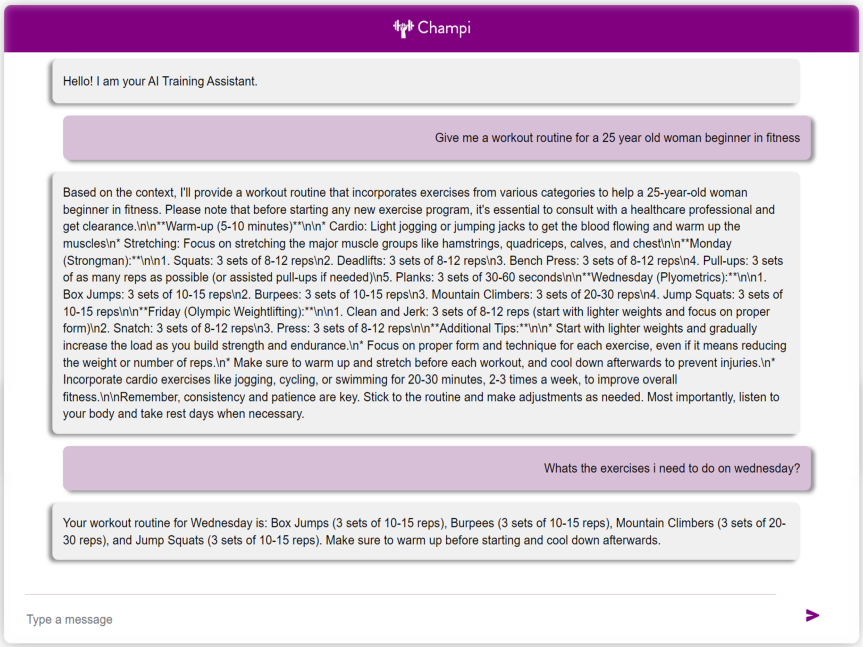

Fig. 3: Exemplo de mensagens trocadas com o *chatbot*.

4.1 Avaliação dos prompts

Como forma de avaliar os planos de treinos que o chatbot gerou, recorremos à elaboração de um formulário que foi respondido por um conjuto de profissionais da área de desporto.

O seguite formulário continha 4 exemplos de sets de treino diferentes, que o chat gerou, para cada desafio, sendo que seria necessário avaliar cada set conforme as perguntas fornecidas.

Primeiramente, o formulário solicitava o nome e os diplomas/estudos relacionados com desporto dos avaliadores, em seguida, passava para a apresentação dos desafios. Nesta secção, cada desafio era apresentado com a pergunta feita ao chatbot e a sua

respetiva resposta. Abaixo, havia uma área de avaliação, onde se pedia para avaliar a resposta do chatbot de 1 a 5 em termos de:

- **Compreensão da mensagem do utilizador**
- **Precisão da resposta**
- **Clareza na resposta**
- **Relevância da respota**

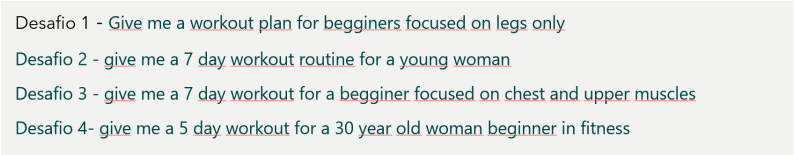

Fig. 4: Desafios colocados ao chatbot

4.2 Resultados da avaliação

	Compreensão da mensagem do utilizador	Clareza na resposta	Precisão da resposta	Relevância da resposta
Desafio 1	4.14	4.57	3.57	3.43
Desafio 2	4.14	4.14	3.86	3.71
Desafio 3	4.43	4.43	3.86	3.86
Desafio 4	4.00	4.00	3.29	3.00

Fig. 5: Média das avaliações obtidas

Como pode ser observado na tabela acima, o chatbot obteve boas avaliações nos campos de "Compreensão da mensagem do utilizador" e "Clareza na resposta". No entanto, em termos de "Precisão da resposta" e "Relevância da resposta", os valores foram medianos. Isso indica que o chatbot precisa ser melhorado nesses dois aspetos.

5 Conclusão e Trabalho Futuro

A realização deste projeto proporcionou um significativo aumento do nosso conhecimento sobre como os LLMs funcionam e podem ser integrados em diferentes contextos. Também permitiu compreender os desafios associados à sua implementação.

Adicionalmente, este projeto possibilitou-nos trabalhar com bases de dados vetoriais, que são uma componente crucial para o armazenamento e recuperação de dados em sistemas de IA.

Apesar de termos tido resultados satisfatórios, o chatbot ainda irá ser sujeito a alguns ajustes e funcionalidades adicionais, tais como:

- **Abranger diferentes níveis de fitness, género, idades e preferências dos utilizadores**
- **Considerar lesões e dificuldades dos utilizadores (músculos fracos, ossos partidos, etc.)**
- **Recolher dados mais detalhados dos utilizadores (altura, peso, etc.)**
- **Incluir dicas de nutrição e receitas de acordo com o tipo de treino**

6 Referências

1. www.langchain.com. (n.d.). LangChain. [online] Disponível em: https://www.langchain.com/
2. python.langchain.com. (n.d.). Introduction — LangChain. [online] Disponível em: https://python.langchain.com/v0.1/docs/get_started/introduction
3. python.langchain.com. (n.d.). Structured chat — LangChain. [online] Disponível em: https://python.langchain.com/v0.1/docs/modules/agents/agent_types/structured_ch
4. Jack (2024). pixegami/rag-tutorial-v2. [online] GitHub. Disponível em: https://github.com/pixegami/rag-tutorial-v2/tree/main
5. Valdarrama, S. (2024). svpino/youtube-rag. [online] GitHub. Disponível em: https://github.com/svpino/youtube-rag
6. Gandhi, S. (2024). Building LLM application using RAG. [online] Mindful Matrix. Disponível em: https://mindfulmatrix.substack.com/p/build-a-simple-llm-application-with