

Protótipo de ChatBot Open Source com RAG e Dados Estruturados para Apoio Académico

Pedro Almeida¹, Vítor Rocio²

¹ Universidade Aberta de Portugal, Lisboa, Portugal,
2200036@uab.pt

² INESC TEC e Universidade Aberta de Portugal, Lisboa, Portugal,
vitor.rocio@uab.pt

Resumo

Este artigo descreve o desenvolvimento de um protótipo de *chatbot* educativo open source para apoio a estudantes da Universidade Aberta. Recorrendo a grandes modelos de linguagem natural (LLM) com RAG (*Retrieval-Augmented Generation*) e integração com bases de dados estruturadas, o sistema fornece respostas a perguntas sobre o percurso académico, a oferta formativa, estrutura curricular e procedimentos administrativos. A arquitetura modular, os modelos de classificação, recuperação e geração, e os testes realizados confirmam a viabilidade técnica da solução.

Palavras-chave: *chatbot*, RAG, LLM, *open source*, base de dados relacional

Title: Open Source ChatBot Prototype Using RAG and Structured Data for Academic Support

Abstract: This article presents the development of an open-source educational chatbot prototype for student support at Universidade Aberta. Combining large language models (LLM) with RAG (Retrieval-Augmented Generation) and integration with structured databases, the system responds to academic queries on courses, curriculum and administrative processes. Its modular architecture, classification and generation models, and test results demonstrate the technical feasibility of the solution.

Keywords: chatbot, RAG, LLM, open source, relational database

1. Introdução

A navegação nos sites web e documentação online disponibilizados pelas instituições é muitas vezes uma tarefa difícil e morosa para quem procura informação específica, ou simplesmente para saber por onde começar. A atual abundância de informação e o fácil acesso cria também a dificuldade de distinguir o que é relevante do que é acessório e, pior que isso, a informação relevante e acessória varia de utilizador para utilizador. No contexto da Universidade Aberta (UAb), que depende quase exclusivamente da interação online, este problema é ainda mais acentuado, sendo comum haver picos de questões recebidas em alturas chave da vida académica (inscrições, avaliações, etc.), a que se torna difícil responder atempadamente.

Tradicionalmente, para não sobrecarregar os canais de resposta nessas alturas, são elaboradas listas de questões frequentes que podem ser consultadas antes da comunicação mais direta, mas que atualmente se revelam inadequadas: para além de serem pouco convenientes para o utilizador, não refletem as prioridades de cada um. Nesse sentido, as recentes tecnologias de IA generativa permitem a criação de agentes conversacionais interativos que mais diretamente possam responder e ajudar os utilizadores, neste caso, os alunos e candidatos à UAb, nos seus processos académicos.

O presente projeto tem como objetivo o desenvolvimento de um protótipo de *chatbot open source*, orientado para prestar apoio administrativo a estudantes da UAb. Este apoio incide sobretudo na orientação quanto ao percurso académico, incluindo informações sobre prazos, regulamentos, inscrições, planos de estudo, entre outros aspetos administrativos.

O *chatbot* será alimentado por documentação pública da instituição e, sempre que possível, por dados estruturados (ex: guia de curso). A arquitetura adotada recorre a um modelo de linguagem de código aberto, complementado por RAG (*Retrieval Augmented Generation*), permitindo assim fornecer respostas mais precisas e fundamentadas, com referências às fontes consultadas.

Este artigo começa por formalizar, na secção seguinte, o âmbito e os requisitos do projeto, e apresentando a arquitetura e desenho da solução na secção 3. É descrita a implementação na secção 4, sendo relatados os resultados dos testes efetuados na secção 5. O artigo termina, na secção 6, com conclusões e considerações finais sobre a solução proposta e perspectivas de desenvolvimento futuro.

2. Âmbito e requisitos do Projeto

2.1. Funcionalidades principais

O *chatbot* desenvolvido fornece apoio automatizado a estudantes, especificamente no que respeita a questões de natureza administrativa relacionadas com o percurso académico. Entre as principais funcionalidades estão a capacidade de responder a perguntas frequentes sobre processos como inscrição, matrícula, cancelamento de disciplinas e cumprimento de prazos, bem como o fornecimento de informações sobre o calendário académico. O sistema deverá também direcionar os estudantes para documentos ou páginas institucionais relevantes, e auxiliá-los na navegação pelos regulamentos académicos. A recuperação de informação será baseada numa base de conhecimento indexada, recorrendo a técnicas RAG e à utilização de uma base de dados vetorial codificando as relações semânticas entre termos relevantes. O *chatbot* deverá aceitar perguntas em linguagem natural, permitindo também pesquisas por palavras-chave.

2.2. Exclusões do escopo do projeto

Apesar do seu carácter funcional, existem limitações claras ao escopo deste protótipo. O *chatbot* não poderá fornecer atendimento personalizado nem aceder a dados académicos individuais dos estudantes. Do mesmo modo, não terá como função a explicação de conteúdos curriculares nem poderá processar solicitações que envolvam autenticação ou identificação do utilizador.

2.3. Requisitos Funcionais

O sistema deverá ser capaz de interpretar perguntas formuladas em linguagem natural e devolver respostas fundamentadas em documentos previamente indexados. Sempre que possível, as respostas deverão conter referência às fontes utilizadas. A interação com o utilizador deverá ocorrer através de uma interface web simples ou, se tal não for viável dentro do tempo disponível, através de uma interface de linha de comandos. A arquitetura deverá prever também a possibilidade de integração futura com plataformas institucionais como o Moodle ou o portal académico.

2.4. Requisitos Não Funcionais

No que toca aos requisitos não funcionais, espera-se que o sistema responda em menos de três segundos nas situações comuns. A estrutura deverá ser escalável, permitindo a adição de novos documentos à base de conhecimento de forma eficiente. Em termos de segurança, o *chatbot* deverá operar exclusivamente com documentação pública, sem qualquer acesso a dados sensíveis. A usabilidade é também uma preocupação central, sendo importante garantir uma experiência intuitiva mesmo para utilizadores com pouca familiaridade com este tipo de tecnologia.

2.5. Tecnologias Necessárias

Relativamente às tecnologias adotadas, o sistema será baseado num modelo LLM *open source*, disponibilizado pela plataforma Hugging Face, por forma a poder ser executado localmente, sem depender de ambientes de *cloud* externos. A indexação e recuperação de conteúdos será assegurada pela Chroma DB, um motor de base vetorial moderno e eficaz, compatível com a utilização de *embeddings* semânticos. O *frontend* poderá inicialmente ser apenas uma consola interativa, enquanto o *backend* será responsável pelo processamento das interações e pela integração com os modelos e bases de dados.

2.5.1. Chroma DB vs Milvus

No contexto do desenvolvimento de um protótipo individual e local, a escolha do Chroma DB como sistema de base de dados vetorial, em detrimento de alternativas como o Milvus, deve-se a uma combinação de fatores práticos e técnicos.

Antes de mais, o Chroma DB é uma solução mais leve, simples de configurar e totalmente local, o que o torna particularmente adequado para ambientes de desenvolvimento individuais ou projetos académicos (Chroma, 2023). Ao contrário do Milvus, que exige a instalação de serviços adicionais como Docker, servidores GRPC ou bases de dados subjacentes (ex: PostgreSQL), o Chroma pode ser executado imediatamente com uma simples instalação via pip, sem necessidade de infraestrutura externa (Zilliz, 2023).

Além disso, o Chroma DB integra-se facilmente com bibliotecas modernas de NLP como LangChain, Hugging Face e Sentence Transformers, sendo possível criar coleções, adicionar documentos e efetuar buscas sem grande complexidade ou dependências (LangChain, 2023; Reimers & Gurevych, 2019). Para este projeto, que utiliza uma abordagem de RAG, a simplicidade e rapidez na indexação e consulta vetorial foram aspetos decisivos.

Por outro lado, embora o Milvus ofereça maior desempenho e escalabilidade em ambientes empresariais (Zilliz, 2023), essas características não se justificam na fase inicial do projeto, onde o volume de documentos e o número de utilizadores é reduzido.

O Chroma DB oferece performance suficiente para prototipagem e testes com o volume de documentos, mantendo-se leve e eficiente.

No entanto, num cenário de evolução para um ambiente de produção com necessidade de escalabilidade horizontal, maior robustez ou integração com sistemas distribuídos, o projeto contempla a necessária modularidade por forma a que a base de dados vetorial seja facilmente substituída por tecnologia Milvus.

2.5.2. Agentização no Desenvolvimento de um *chatbot* com LLM e RAG

A agentização consiste na organização do sistema em torno de múltiplos agentes especializados, cada um com responsabilidades bem definidas dentro do fluxo de funcionamento do *chatbot*. No contexto do desenvolvimento de um *chatbot* baseado em LLM com RAG, esta abordagem permite construir uma arquitetura modular, escalável e altamente adaptável, onde diferentes componentes colaboram para fornecer respostas mais relevantes, contextuais e úteis (Schick et al., 2023; LangChain, 2023).

Um agente pode ser encarado como uma entidade autónoma responsável por uma tarefa específica, como por exemplo: identificar a intenção da pergunta do utilizador, procurar informação relevante numa base de dados semântica, resumir documentos ou gerar a resposta final com base nos dados recuperados (Brown et al., 2020). Ao dividir o sistema em agentes especializados, torna-se possível ajustar, expandir ou substituir funcionalidades sem comprometer o funcionamento global, promovendo assim uma maior flexibilidade e manutenção do sistema.

No caso de um *chatbot* educativo com RAG, podem ser definidos diversos agentes com papéis distintos. Por exemplo, um agente classificador de intenções pode analisar a entrada do utilizador e determinar se se trata de uma pergunta factual, um pedido de resumo, uma explicação ou uma dúvida sobre o percurso académico (Kirk et al., 2023). Um agente orquestrador decide então que ferramentas ou agentes devem ser ativados para lidar com essa intenção. Outros agentes podem ser responsáveis pela recuperação semântica de informação em bases vetoriais como Chroma DB ou Milvus, pela extração de texto relevante de fontes como PDF ou CSV, pela geração da resposta através do modelo de linguagem, ou mesmo pela invocação de ferramentas externas, como calculadoras simbólicas ou APIs académicas (Mialon et al., 2023).

Esta abordagem permite ainda a integração com *frameworks* modernas como LangChain, que facilitam a criação e coordenação de agentes com capacidades específicas. Estas ferramentas fornecem mecanismos para encadear tarefas, atribuir papéis, planear passos intermédios e interagir com o utilizador de forma mais proativa (LangChain, 2023). Em particular, o uso de LangChain permite definir pipelines dinâmicos com ferramentas específicas associadas a agentes, tornando o processo de desenvolvimento mais eficiente e reutilizável.

Em suma, a agentização no desenvolvimento de um *chatbot* com LLM e RAG representa uma evolução importante em direção a sistemas mais inteligentes, interpretativos e adaptáveis. Ao estruturar a aplicação com base em agentes cooperativos, é possível atingir um maior grau de personalização, reutilização de componentes e adequação ao contexto específico do utilizador, o que é particularmente relevante em cenários educativos e de apoio académico (Mialon et al., 2023).

2.5.3. Janela de Contexto (Context Window) em Modelos de Linguagem

A janela de contexto, é um conceito fundamental no funcionamento de modelos de linguagem de grande escala (LLMs). Refere-se ao número máximo de *tokens* (unidades mínimas de texto, como palavras, partes de palavras ou pontuação) que o modelo consegue processar e considerar de forma simultânea ao gerar uma resposta (OpenAI, 2023a; Brown et al., 2020). Em termos práticos, a janela de contexto define o limite de memória de curto prazo do modelo, influenciando diretamente a sua capacidade de compreender e responder com base em entradas mais longas e complexas.

Num *chatbot* baseado em LLM, esta janela representa a quantidade total de informação que pode ser incluída na entrada para que o modelo a tenha em consideração no momento de gerar a resposta (Mialon et al., 2023). Esta informação pode incluir a pergunta do utilizador, partes de documentos recuperados por um sistema RAG, instruções adicionais, histórico de conversação e anotações contextuais. Tudo isto tem de caber dentro da janela de contexto, sob pena de partes importantes da informação serem descartadas, levando a respostas incompletas ou incorretas.

O tamanho da janela de contexto varia consoante o modelo utilizado. Por exemplo, modelos mais antigos como o GPT-2 tinham janelas relativamente curtas (cerca de 512 a 1024 *tokens*), enquanto modelos mais recentes como o GPT-4, Claude 2 ou variantes do LLaMA podem lidar com 8.000, 32.000 ou até mais de 100.000 *tokens* (Touvron et al., 2023; Anthropic, 2023). No entanto, janelas maiores implicam maior custo computacional e, em muitos casos, exigem uma gestão mais cuidadosa da informação incluída.

No desenvolvimento de sistemas com RAG, a gestão eficiente da janela de contexto torna-se crucial. Como nem toda a informação pode ser incluída simultaneamente, é necessário aplicar técnicas como *chunking* (divisão de documentos em partes menores), ranking por relevância semântica, e *context filtering* para selecionar apenas os trechos mais relevantes a serem apresentados ao modelo (Lewis et al., 2020). Estas estratégias permitem maximizar a utilidade do espaço disponível, garantindo que o modelo tem acesso à informação mais pertinente para gerar respostas precisas e contextualizadas.

Em suma, a janela de contexto constitui um dos principais limites técnicos dos modelos de linguagem, influenciando a profundidade e coerência das respostas geradas. A sua gestão eficaz, especialmente em sistemas com recuperação de informação como RAG, é essencial para garantir a qualidade das interações entre o *chatbot* e o utilizador — sobretudo em domínios como a educação, onde o rigor e a contextualização são fundamentais.

2.6. Desafios

Finalmente, entre os principais desafios identificados destacam-se a necessidade de garantir a precisão das respostas, a manutenção e atualização contínua da base de conhecimento, o tratamento adequado de perguntas ambíguas ou mal formuladas, e uma eventual integração com os sistemas académicos da instituição.

3. Desenho

3.1. Arquitetura Geral

A arquitetura geral do sistema assenta numa estrutura modular e extensível, que separa claramente a interação do utilizador da lógica de processamento e das fontes de dados. A comunicação inicia-se no *frontend* (consola interativa ou interface web), que envia a pergunta ao *backend*. Este é responsável por decidir entre consultar diretamente a base de dados relacional (para perguntas estruturadas) ou ativar o pipeline de recuperação aumentada (RAG), que envolve a consulta à base vetorial ChromaDB e a geração de resposta através do modelo de linguagem *flan-t5-large*. Um classificador de intenções complementa este fluxo, assegurando que apenas perguntas académicas são processadas.

Para clarificar o fluxo, apresenta-se de seguida um diagrama de sequência simplificado da interação com o sistema, bem como um diagrama de componentes da arquitetura técnica.

3.1.1. Diagrama de Sequência

O diagrama de sequência apresentado na fig. 1 ilustra o fluxo de comunicação entre os principais componentes do sistema, no momento em que o utilizador interage com o *chatbot*. O processo tem início com o utilizador, que escreve a sua pergunta através da interface, seja ela uma consola ou uma aplicação web. Esta entrada é recebida pelo módulo denominado Frontend / Classificador, que tem como função inicial identificar a natureza da pergunta e determinar se se trata de uma questão académica relevante.

Caso a intenção da pergunta seja válida, esta é encaminhada para a base de dados relacional (BD), onde é avaliado se existem dados estruturados capazes de responder diretamente ao que foi questionado. Este processo abrange, por exemplo, pedidos sobre coordenadores de curso, ECTS de unidades curriculares, ou cursos associados a um determinado departamento.

Se a base de dados não contiver a resposta adequada, o sistema ativa o componente RAG. Este módulo procede à consulta da documentação institucional previamente indexada, procurando por blocos de texto semanticamente relevantes para a pergunta em questão. Os trechos mais relevantes são então reunidos e encaminhados, juntamente com a pergunta original, para o modelo de linguagem (LLM), que tem a responsabilidade de gerar uma resposta textual final, contextualizada e fundamentada.

Por fim, a resposta gerada pelo LLM é enviada de volta ao Frontend e apresentada ao utilizador. Este diagrama evidencia de forma clara a separação entre as fases de classificação da pergunta, consulta estruturada, recuperação semântica e geração de linguagem, refletindo a arquitetura modular e extensível adotada no desenvolvimento do sistema.

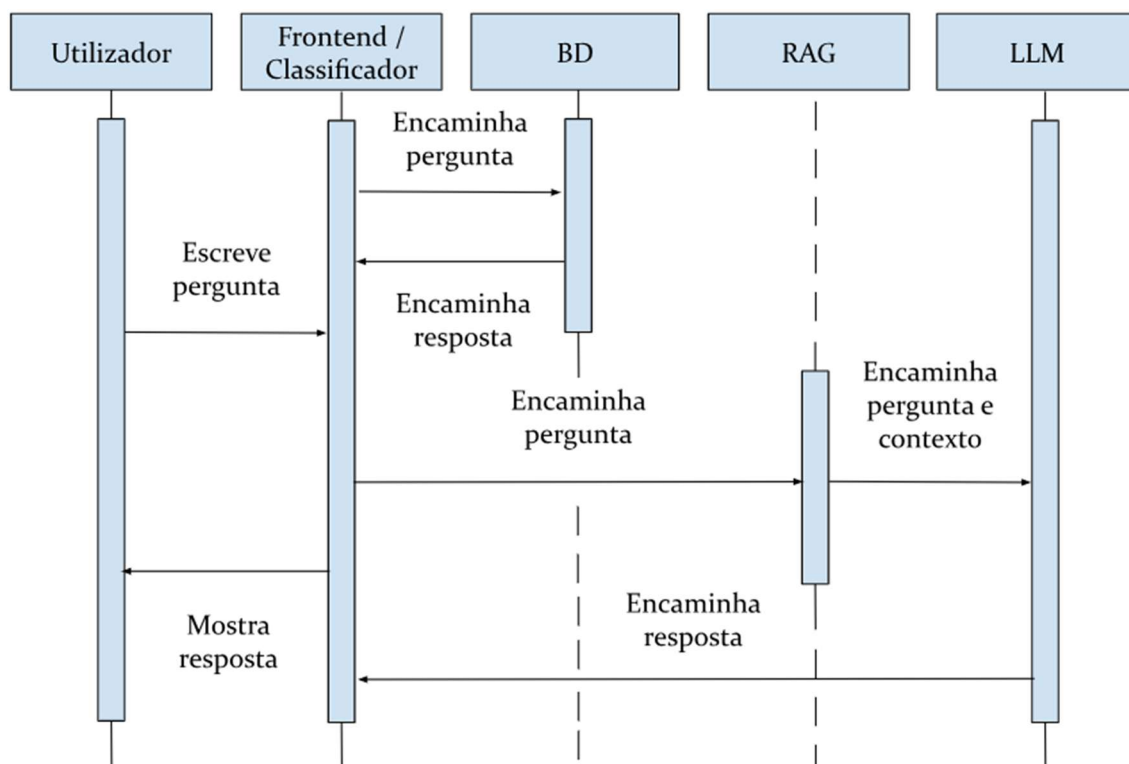


Figura 1. Diagrama de sequência para o caso de uso de resposta à pergunta do utilizador

3.1.2. Diagrama de componentes

Complementarmente, o diagrama de componentes (fig. 2) descreve a organização estrutural dos principais módulos da aplicação e as suas interações durante a execução do processo de resposta. O Frontend / Classificador atua como ponto de entrada, recebendo a pergunta do utilizador e decidindo, com base em palavras-chave e *embeddings* semânticos, qual a via mais apropriada para a obtenção da resposta.

A base de dados relacional é consultada sempre que a pergunta possa ser resolvida com dados estruturados, como é o caso de informações académicas codificadas. Caso contrário, a pergunta é reencaminhada para o módulo RAG, que procura o contexto mais adequado nos documentos previamente indexados (como regulamentos, guias de curso ou calendários académicos). Este contexto, juntamente com a pergunta, é então enviado para o LLM, que devolve uma resposta formulada em linguagem natural. A resposta é então transmitida para o utilizador através do *frontend*.

Esta representação ajuda a compreender os limites e as responsabilidades de cada componente, realçando a importância da integração entre fontes de dados estruturadas e não estruturadas para alcançar respostas fiáveis e contextualizadas.

3.2. Modelos *open source* selecionados da Hugging Face

A implementação do *chatbot* educativo proposto assenta na utilização de modelos de linguagem e ferramentas de processamento de linguagem natural (PLN) de código aberto, com o objetivo de fornecer respostas relevantes e contextualizadas a dúvidas de natureza administrativa colocadas pelos estudantes da Universidade Aberta. A arquitetura do sistema foi desenhada para tirar partido de técnicas modernas de RAG, integrando diferentes componentes especializadas na geração, recuperação e classificação de informação.

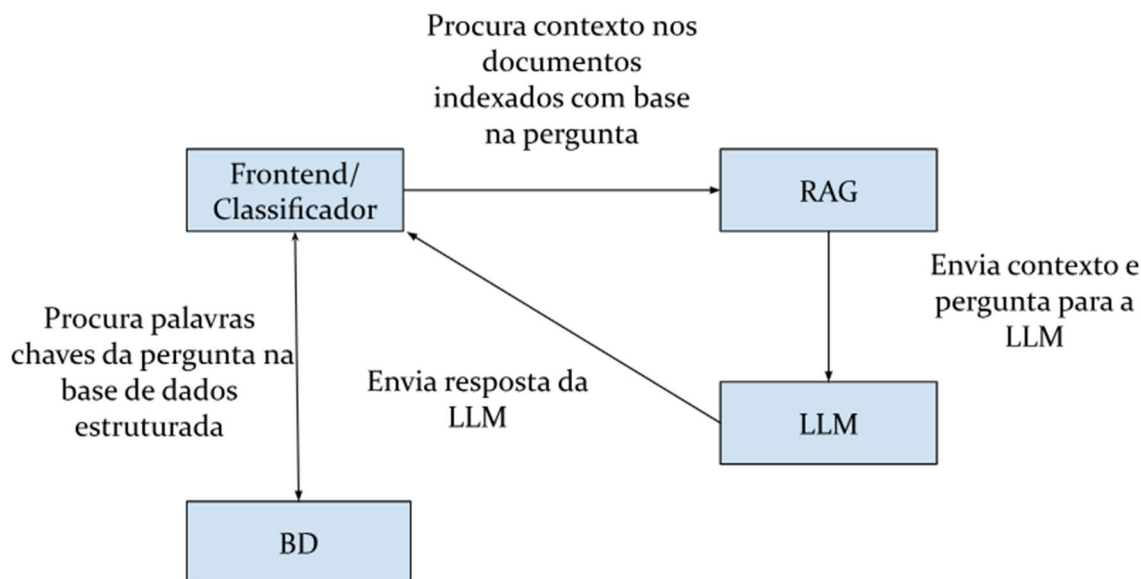


Figura 2. Diagrama de componentes

3.2.1. Modelo de geração de texto (google/flan-t5-large)

Para a geração de respostas no chatbot acadêmico, foi adotado o modelo google/flan-t5-large, desenvolvido pela Google. Este modelo pertence à família T5 (*Text-to-Text Transfer Transformer*), e destaca-se por oferecer um desempenho significativamente superior ao do flan-t5-large, especialmente em tarefas complexas de compreensão e geração de linguagem natural.

O flan-t5-large é especialmente eficaz na interpretação de *prompts* contextuais, sendo capaz de produzir respostas mais detalhadas, coerentes e ajustadas ao tom pretendido. No âmbito deste projeto, é utilizado para duas funções principais: (1) gerar respostas contextualizadas com base na informação extraída por RAG, e (2) produzir resumos e reformulações de blocos informativos provenientes de documentos acadêmicos oficiais, facilitando a leitura e consulta por parte do estudante.

A escolha do modelo flan-t5-large prende-se com a sua elevada capacidade de generalização, que o torna ideal para situações em que o contexto fornecido é extenso ou ambíguo. Embora apresente requisitos computacionais superiores aos da versão base, é compatível com execução em CPU, sendo ainda assim viável para ambientes locais com otimização adequada. A integração é realizada via Hugging Face Transformers, permitindo uma configuração simples e eficaz. Este modelo representa um compromisso entre qualidade da resposta e versatilidade, sendo especialmente adequado para sistemas que exigem clareza, naturalidade e cobertura linguística sólida em português.

3.2.2. Modelo de *embeddings* (sentence-transformers / paraphrase-multilingual-MiniLM-L12-v2)

A componente de recuperação de informação semântica baseia-se no modelo paraphrase-multilingual-MiniLM-L12-v2, da biblioteca Sentence Transformers. Este modelo foi utilizado para gerar representações vetoriais de perguntas e trechos de texto, permitindo calcular a semelhança entre o que o utilizador pergunta e os conteúdos disponíveis na base de dados. Através deste mecanismo é possível identificar os trechos mais relevantes e usá-los como contexto para gerar respostas mais precisas. A integração com o sistema vetorial Chroma DB garante a rapidez e escalabilidade desta recuperação. Este modelo

foi escolhido por ser multilíngue, eficaz em português e leve o suficiente para ambientes de computação modestos.

2.2.3. Modelo de classificação (joeddav / xlm-roberta-large-xnli)

Para além da geração e recuperação, o sistema integra ainda uma componente de classificação, baseada no modelo joeddav/xlm-roberta-large-xnli. Trata-se de um classificador do tipo zero-shot, capaz de inferir a intenção do utilizador mesmo sem treino adicional. Este classificador é usado para distinguir perguntas reais de comentários ou interações irrelevantes, bem como para adaptar o tom da resposta consoante a atitude detetada (por exemplo, uma reclamação ou um agradecimento). Em casos específicos, pode também gerar resumos das interações anteriores, promovendo continuidade na conversa. Esta funcionalidade contribui para tornar o *chatbot* mais versátil e “humano”, sem perder o foco no apoio académico.

4. Implementação

A implementação do protótipo do *chatbot* académico da Universidade Aberta foi desenvolvida integralmente em Python, com uma arquitetura modular e escalável, centrada em quatro componentes principais: interface, motor de recuperação de contexto (RAG), modelo de linguagem (LLM) e integração com base de dados estruturada. Esta modularização permitiu separar responsabilidades, facilitar testes e possibilitar futuras expansões sem comprometer a lógica já implementada.

4.1. Preparação do ambiente

4.1.1. load_db.py

Este módulo tem como função inicializar e povoar a base de dados relacional uab.db, a partir de ficheiros .json estruturados com dados simulados (ou exportados de sistemas institucionais). Para garantir integridade e normalização, o ficheiro define o esquema relacional da base de dados, que inclui tabelas como Cursos, UnidadesCurriculares, Professores, UC_Cursos, UC_Professores, ViceCoordenadores_Cursos, e departamentos. O carregamento é realizado com a biblioteca sqlite3 em conjunto com o pandas, que permite ler os ficheiros JSON e fazer inserções em massa na base de dados. Este módulo é tipicamente executado apenas uma vez, no processo de preparação do ambiente, garantindo que a base de dados local contém a informação académica necessária para a componente de consulta estruturada do *chatbot*.

4.1.2. index_pdf.py

A indexação de documentos institucionais, como o guia académico ou regulamentos, é tratada em index_pdf.py. Este script percorre documentos PDF com a ajuda da biblioteca PyMuPDF (fitz), extraíndo texto de forma limpa e estruturada. Para maximizar a utilidade semântica das consultas, o texto extraído é dividido em blocos coesos (com base em parágrafos ou frases compostas), os quais são convertidos em *embeddings* vetoriais utilizando o modelo paraphrase-multilingual-MiniLM-L12-v2 da biblioteca Sentence Transformers. Cada bloco é então armazenado numa coleção persistente da base vetorial ChromaDB, com metadados que incluem a fonte do documento original. A indexação é crucial para o funcionamento eficaz do módulo RAG, assegurando que os conteúdos mais relevantes são recuperáveis durante uma interação com o *chatbot*.

4.2. Código Funcional

4.2.1. main.py

O ponto de entrada do sistema é o ficheiro `main.py`, que disponibiliza uma interface de linha de comandos simples e intuitiva. Ao receber uma pergunta do utilizador, é avaliada a intenção da pergunta do avaliador em cinco categorias: "pergunta académica", "agradecimento", "comentário irrelevante", "reclamação", "cumprimento", e só caso seja identificada a intenção como "pergunta académica" a função principal (`call_chatbot`) é chamada em que tenta primeiro responder com base na base de dados relacional (SQLite). Caso a pergunta não se enquadre nas categorias reconhecidas como estruturadas (ex: identificação de coordenadores, vice-coordenadores, ECTS, cursos por departamento, entre outras), o sistema recorre ao pipeline RAG para gerar uma resposta com base em documentos previamente indexados.

4.2.2. intention_utils.py

Para promover uma experiência mais humana e adaptativa, o módulo `intencao_utils.py` introduz uma camada de análise de intenção baseada em classificação zero-shot. Utilizando o modelo `joeddav/xlm-roberta-large-xnli` da Hugging Face, o módulo consegue classificar qualquer entrada textual do utilizador em categorias como: "pergunta académica", "agradecimento", "reclamação", "comentário irrelevante", e "cumprimento". Esta classificação não requer qualquer fase de treino e permite, por exemplo, distinguir perguntas legítimas de interjeições irrelevantes, adaptar o tom da resposta em função da atitude do utilizador, ou até ignorar entradas sem valor informacional. Esta abordagem, embora simples, contribui substancialmente para uma conversa mais natural e robusta, e pode ser facilmente expandida para incorporar novas intenções à medida que o sistema evolui.

4.2.3. retriever.py

A componente de recuperação aumentada por geração (RAG), implementada em `retriever.py`, começa pela obtenção de representações vetoriais da pergunta do utilizador através do modelo `paraphrase-multilingual-MiniLM-L12-v2` da biblioteca `Sentence Transformers`. Estas *embeddings* são comparadas com os vetores armazenados na base vetorial persistente `ChromaDB`, para determinar os blocos de texto mais semanticamente relevantes. Até três blocos são considerados, sendo selecionados os que melhor se enquadram dentro da janela de contexto do modelo (limitada a cerca de 1024 *tokens*, de acordo com o *tokenizer* do modelo `flan-t5-large`). Esta lógica garante que o conteúdo passado ao LLM é informativo, mas não excede os limites do modelo.

4.2.4. llm.py

A seguir, o texto recuperado é formatado juntamente com a pergunta original e alimentado ao modelo de linguagem propriamente dito, cuja configuração está em `llm.py`. O modelo escolhido foi o `google/flan-t5-large`, um modelo leve e orientado para tarefas de instrução, capaz de compreender português. A integração com a biblioteca `transformers` e a *framework* `LangChain` permite encapsular o modelo num *pipeline* reutilizável, suportado por `PromptTemplate`, que define uma estrutura de *prompt* clara com secções para "Contexto" e "Pergunta".

4.2.5. db_utils.py

Do ponto de vista da persistência e lógica de negócio baseada em dados estruturados, o módulo `db_utils.py` é responsável por todas as operações com a base de dados SQLite localizada em `db/uab.db`. Este módulo carrega listas de cursos, unidades curriculares e departamentos à partida, utilizando `pandas` e `spacy` para pré-processamento linguístico. Utiliza também o mesmo modelo MiniLM para calcular similaridade entre a pergunta do utilizador e os nomes reais das entidades académicas, permitindo extrair automaticamente o curso, unidade curricular ou departamento mais provável referido na pergunta. A partir disso, são executadas consultas SQL específicas que retornam os resultados formatados — por exemplo, professores associados a uma unidade, número de ECTS, ou coordenadores de curso.

4.3. Conclusão

Esta abordagem híbrida — combinando RAG com consulta estruturada — confere ao *chatbot* maior robustez e flexibilidade, sendo capaz de responder tanto com base em documentação institucional como com dados tabulados mais recentes. A modularidade da arquitetura permite ainda a fácil substituição ou expansão de componentes, como a troca do modelo de *embeddings*, migração para uma base vetorial como Milvus, ou a criação de uma interface web sobre o *backend* já funcional.

Em suma, a fase de implementação concretizou com sucesso a infraestrutura técnica do sistema, validando a viabilidade da criação de um *chatbot* educativo independente de APIs proprietárias e capaz de utilizar múltiplas fontes de informação para responder eficazmente a questões académicas.

5. Resultados e Testes

A fase de testes foi conduzida com o objetivo de validar o comportamento do *chatbot* nas suas duas componentes principais: o mecanismo de recuperação e geração (RAG) e a camada de consulta estruturada à base de dados académica. Estes testes foram realizados num ambiente local controlado, utilizando dados reais fornecidos pela instituição e documentação académica em formato PDF.

Começou-se por realizar testes unitários aos módulos individualizados, nomeadamente à função de recuperação semântica com ChromaDB, à geração de respostas com o modelo `flan-t5-large` e às consultas SQL encapsuladas em `db_utils.py`. Cada função foi verificada isoladamente para garantir que devolvia resultados corretos quando alimentada com inputs específicos, por exemplo, perguntas como “Quem é o coordenador da licenciatura em Engenharia Informática?” ou “Quantos ECTS tem a unidade de Bases de Dados?”.

Seguidamente, foram efetuados testes de integração, simulando o fluxo completo da interação com o utilizador. Nestes testes, o sistema recebeu perguntas variadas — algumas interpretáveis como estruturadas (por exemplo, perguntas sobre coordenadores ou ECTS), outras mais abertas e dependentes do motor RAG (como “Como funciona o regime de avaliação?”). O *chatbot* demonstrou uma boa capacidade de distinguir entre estes dois tipos de perguntas e de encaminhar cada uma para o mecanismo mais adequado.

Foram formuladas, ao todo, mais de 30 perguntas de teste, das quais cerca de 20 envolveram a base de dados relacional e 12 visaram testar a eficácia da recuperação em

documentos PDF através da pipeline RAG. A título de exemplo, perguntas como “Quando decorre o módulo de ambientação online?” ou “O que é necessário para participar na avaliação online?” foram corretamente respondidas com base nos documentos institucionais. Já perguntas como “Quais são os cursos oferecidos pelo departamento de Ciências e Tecnologia?” ou “O professor Manuel Silva leciona que UCs?” resultaram em respostas geradas por SQL com elevado grau de precisão.

No caso das perguntas dependentes do RAG, os resultados variaram consoante a formulação. Perguntas bem estruturadas e específicas levaram a respostas concisas e corretas, enquanto perguntas ambíguas ou vagas nem sempre produziram respostas úteis. Esta limitação está associada à ausência de contexto conversacional persistente e à natureza zero-shot do classificador de intenção.

Também se realizaram testes à robustez do sistema perante dados imperfeitos, como nomes de unidades ou cursos com erros ortográficos. A combinação de *embeddings* semânticos (com paraphrase-multilingual-MiniLM-L12-v2) com *fuzzy matching* (rapidfuzz) demonstrou ser eficaz na recuperação de entidades corretas, mesmo com variações na grafia.

Com base nas observações recolhidas durante os testes, foram implementadas diversas melhorias incrementais no sistema. Entre estas, destaca-se a correção automática das respostas geradas em português, através da integração da ferramenta `language_tool_python`, que assegura uma redação mais cuidada. A lógica de extração de entidades — como cursos, unidades curriculares, professores e anos — foi igualmente expandida, combinando técnicas de *embeddings* semânticos com processamento de linguagem natural (NLP), o que permitiu aumentar a precisão na identificação dos elementos relevantes nas perguntas dos utilizadores.

Adicionalmente, foi melhorada a estrutura das respostas da base de dados, que passaram a ser apresentadas em formato de texto natural e contínuo, abandonando a apresentação em forma de tabela. Passou também a ser incluída informação sobre a origem da resposta, indicando se esta foi obtida a partir da base de dados relacional ou através de recuperação semântica a partir de documentos PDF. Por fim, foi aplicada uma gestão mais rigorosa da janela de contexto do modelo de linguagem, otimizando a seleção dos blocos de texto mais relevantes a serem incluídos na geração da resposta, garantindo assim maior coerência e adequação informativa.

Foi também além das funcionalidades principais de consulta estruturada e recuperação semântica, testado o componente responsável pela caracterização da intenção do utilizador. Esta funcionalidade, baseada num classificador zero-shot multilingue, permite distinguir entre perguntas relevantes e interações conversacionais, como cumprimentos, agradecimentos ou reclamações. Esta camada não só melhora a experiência de utilização, como contribui para modular o tom das respostas e redirecionar interações não informativas.

Foram realizados testes específicos para avaliar a capacidade do classificador em identificar corretamente diferentes tipos de intenção. Por exemplo, para perguntas como “Olá!” ou “Bom dia, tudo bem?”, a intenção de saudação foi corretamente atribuída com elevada confiança (0.93 e 0.84, respetivamente), conduzindo a respostas adequadas do tipo: “Estou aqui para ajudar com questões académicas. Como posso ser útil?”.

Também se verificou a correta detecção de agradecimentos, como em “Obrigado, era só isso” (intenção: agradecimento, 0.68) e “Obrigado, muito útil!” (0.92), com o *chatbot* a responder de forma empática: “Fico feliz em ajudar! Tem mais alguma dúvida?”.

No entanto, alguns casos revelaram classificações incorretas ou ambíguas. Frases como “Excelente resposta!” foram classificadas como reclamação (0.41), o que levou a uma resposta inadequada do tipo: “Lamento pela experiência. Pode indicar o que gostaria de ver melhorado?”. Isto destaca a necessidade de refinar a capacidade do classificador para distinguir elogios de queixas, sobretudo em contextos positivos.

Adicionalmente, frases como “Essa resposta não faz sentido” ou “Isso está errado” foram corretamente interpretadas como reclamações (0.54 e 0.58), levando o sistema a propor melhorias com mensagens como: “Lamento pela experiência. Pode indicar o que gostaria de ver melhorado?”.

Estes testes demonstram que o classificador de intenção é, na maioria dos casos, capaz de detetar o tom e a finalidade da interação, permitindo uma gestão mais flexível do diálogo. Esta camada de interpretação revela-se particularmente útil para modular o comportamento do *chatbot* em cenários não factuais, bem como para identificar interações que exigem reformulação da pergunta por parte do utilizador.

No futuro, será importante refinar ainda mais este componente com exemplos adicionais e, idealmente, recorrer a modelos alinhados com contextos educativos multilíngues. A sua integração contribui decisivamente para uma experiência de interação mais natural, educada e adaptável, aproximando o sistema de um assistente académico verdadeiramente conversacional.

Por fim, foi planeada uma fase de testes com utilizadores reais (estudantes da instituição), recolhendo feedback informal através da interface de linha de comandos. Esta validação empírica permitirá aferir não só a precisão das respostas, mas também a utilidade prática do sistema. Contudo, uma validação mais alargada fará mais sentido numa versão futura com interface gráfica, suporte para histórico conversacional e uma gestão mais dinâmica do diálogo.

5.1. Testes de Caracterização

Foram realizados 9 testes de caracterização, com o objetivo de avaliar a capacidade do classificador zero-shot (joeddav/xlm-roberta-large-xnli) em identificar corretamente a intenção comunicativa do utilizador. Estes exemplos incluíam cumprimentos, agradecimentos, elogios, reclamações e comentários ambíguos, como se elencam na tabela 1.

Tabela 1. Tabela resumo dos testes de caracterização

Tipo de Intenção	Nº Exemplos	Precisão Observada	Observações
Saudação	3	Alta (> 0.8)	Detetadas corretamente com respostas adequadas e consistentes.
Agradecimento	2	Moderada ($\sim 0.7\text{--}0.9$)	Classificação adequada, resposta empática do <i>chatbot</i> .
Reclamação	3	Baixa a média	Um elogio foi incorretamente classificado como reclamação (0.41).
Comentário irrelevante	1	Média (0.54)	Tratado corretamente com proposta de reformulação da pergunta.

O classificador demonstra boa performance em intenções explícitas (como saudações), mas apresenta vulnerabilidade em ambiguidade semântica, confundindo elogios com reclamações, como ilustrado na tabela 2. Isto evidencia a necessidade de refinamento com dados reais ou ajustamento de *thresholds*.

Tabela 2. Tabela com os testes de caracterização

Utilizador	Intenção	Confiança	Resposta do <i>chatbot</i>
Olá!	saudação	0.93	Estou aqui para ajudar com questões académicas. Como posso ser útil?
Bom dia, tudo bem?	saudação	0.84	Estou aqui para ajudar com questões académicas. Como posso ser útil?
Obrigado, era só isso.	agradecimento	0.68	Fico feliz em ajudar! Tem mais alguma dúvida?
Até à próxima!	saudação	0.73	Estou aqui para ajudar com questões académicas. Como posso ser útil?
Excelente resposta!	reclamação	0.41	Lamento pela experiência. Pode indicar o que gostaria de ver melhorado?
Obrigado, muito útil!	agradecimento	0.92	Fico feliz em ajudar! Tem mais alguma dúvida?
Essa resposta não faz sentido.	comentário irrelevante	0.54	Poderia reformular a sua pergunta para que eu possa ajudar?
Isso está errado.	reclamação	0.58	Lamento pela experiência. Pode indicar o que gostaria de ver melhorado?
Não encontrei o que procurava.	reclamação	0.37	Lamento pela experiência. Pode indicar o que gostaria de ver melhorado?

4.2. Testes de Dados Estruturados

Foram realizados 17 testes com perguntas que envolvem dados tabulados da UAb, como cursos, coordenadores, ECTS, professores e unidades curriculares, cujo resumo se observa na tabela 3.

Tabela 3. Tabela resumo dos testes de dados estruturados

Tipo de Consulta Estruturada	Nº Exemplos	Precisão	Observações
Listagem de cursos por departamento	3	Elevada	Recuperações completas e corretas.
Consulta sobre coordenadores e docentes	6	Elevada	Correta identificação de professores e funções associadas.
ECTS e sinopses de unidades curriculares	3	Elevada	Valores e descrições extraídos corretamente.
Extração de UCs por curso e semestre	3	Elevada	Boa cobertura e estrutura da resposta.
Correlações cruzadas (ex: docente + função)	2	Elevada	Mapeamento correto, mesmo em perguntas compostas.

A extração de informação estruturada teve desempenho exemplar, refletindo a eficácia do módulo `db_utils.py` com *embeddings*, *fuzzy matching* e consultas SQL otimizadas. A resposta em linguagem natural foi clara, e os testes confirmam a viabilidade da integração com bases de dados institucionais. A tabela 4 ilustra os resultados obtidos.

Tabela 4. Tabela com os testes de dados estruturados

Utilizador	Confiança	Resposta do <i>chatbot</i>
Quais são os cursos oferecidos pelo departamento de Ciências e Tecnologia?	0.97	Os cursos do departamento DCEt são: Licenciatura em Engenharia Informática, Licenciatura em Matemática e Aplicações, Licenciatura em Matemática Aplicada à Gestão, Licenciatura em Ciências do Ambiente, Mestrado em Informação e Sistemas Empresariais, Mestrado em Estatística, Matemática e Computação, Mestrado em Engenharia Informática e Tecnologia Web, Mestrado em Bioestatística e Biometria, Mestrado em Ciências do Consumo Alimentar, Mestrado em Cidadania Ambiental e Participação, Doutoramento em Matemática Aplicada e Modelação, Doutoramento em Didática das Línguas, Doutoramento em Estudos Portugueses, Doutoramento em Sustentabilidade e Desenvolvimento, Doutoramento em Ciência e Tecnologia Web, Doutoramento em Média-Arte Digital.
Qual é o ciclo do curso de Estudos Portugueses Multidisciplinares?	0.96	O curso Mestrado em Estudos Portugueses Multidisciplinares pertence ao 2.º ciclo de estudos.
Que cursos existem no departamento DEED?	0.95	Os cursos do departamento DEED são: Licenciatura em Educação, Mestrado em Gestão da Informação e Bibliotecas Escolares, Mestrado em Pedagogia do E-learning, Mestrado em Administração e Gestão Educacional, Mestrado em Supervisão Pedagógica, Doutoramento em Educação a Distância e eLearning.

Quem é o coordenador do curso de História?	0.92	Os coordenadores do curso Licenciatura em História são: <Coordenador1>.
Quem são os professores da unidade curricular de Estatística?	0.95	Os professores para a unidade curricular 'Tópicos de Estatística Matemática' do curso 'Doutoramento em Matemática Aplicada e Modelação' são: <Professor1>.
Qual é a sinopse da UC Fundamentos de Matemática?	0.96	Nesta unidade curricular são introduzidos os conceitos e técnicas básicas sobre combinatória enumerativa, teoria elementar de números e recursões lineares. Este curso termina com uma breve introdução à teoria de grafos e algumas das suas aplicações.
Quantos ECTS tem a unidade curricular de Bioestatística?	0.95	A unidade curricular Elementos de Bioestatística tem uma ponderação de 6 créditos.
Qual é o nome do departamento do curso de Estudos Europeus?	0.97	O departamento do curso Mestrado em Estudos sobre a Europa é o DH.
Quais são os professores que lecionam a UC de Programação I?	0.9	Os professores para várias UCs ligadas à Programação I incluem: <Professor1>, <Professor2>, <Professor3>, <Professor4>, <Professor5>, <Professor6>, <Professor7>, <Professor8>, <Professor9>, <Professor10>.
O professor <Professor1> leciona que unidades curriculares?	0.99	O professor <Professor1> leciona várias unidades, incluindo: Introdução à Programação, Inteligência Artificial, Gestão de Projetos, entre outras.
Quem são os vice-coordenadores da Licenciatura em Educação?	0.95	<Vice-Coodenador1>, <Vice-Coodenador2>.
Quais são as unidades curriculares opcionais do curso de Ciências Sociais no 2.º ano?	0.98	Matemática Preparatória, Antropologia das Religiões, Sociedade Portuguesa Contemporânea, Igualdade, Exclusão Social e Cidadania, entre outras.
Quais UCs pertencem ao curso de Engenharia Informática no primeiro ano?	0.91	Álgebra Linear I, Arquitectura de Computadores, Introdução à Programação, Sistemas Computacionais, Análise Infinitesimal, entre outras.
Quais são os professores que lecionam no curso de Gestão?	0.9	<Professor1>, <Professor2>.
Quais unidades curriculares do curso de Matemática Aplicada à Gestão são dadas no segundo semestre?	0.98	Elementos de Análise Infinitesimal II, Investigação Operacional, Macroeconomia, Estatística Multivariada, entre outras.
Existe algum professor que seja coordenador e também leciona?	0.92	<Professor1>, <Professor2>, <Professor3>.

4.3. Testes de RAG

Foram conduzidos 12 testes RAG, com perguntas respondidas com base em documentos PDF da UAb previamente indexados com ChromaDB. A tabela 5 apresenta um resumo dos resultados destes testes.

Tabela 5. Tabela resumo dos testes de RAG

Tipo de Informação Recuperada	Nº Exemplos	Sucesso	Observações
Datas e prazos académicos	5	Elevado	Ex: datas de módulos, creditações, limite de ECTS, tudo corretamente extraído.
Procedimentos administrativos	4	Bom	Respostas claras sobre matrícula, avaliação, perfil de utilizador.
Natureza ambígua ou mal formulada	2	Parcial	Requerem melhoria na seleção de contexto.
Respostas sem intenção explícita	1	OK	Ex: cálculo de propinas — resposta extraída mesmo sem valor de intenção.

O mecanismo RAG mostrou elevada eficácia na recuperação de blocos relevantes. A gestão da janela de contexto e uso de *embeddings* garantiram que os textos mais informativos fossem enviados ao modelo. No entanto, perguntas mal estruturadas ainda podem gerar respostas vagas, o que poderá ser mitigado com melhoria nos *prompts* ou histórico de diálogo. O tipo de respostas obtidas é ilustrado na tabela 6.

Tabela 6. Tabela com os testes de RAG

Utilizador	Confiança	Resposta do <i>chatbot</i>
Quando decorre o módulo de ambientação online no 1.º semestre?	0.47	16 a 27 de setembro de 2024
O módulo de ambientação online é obrigatório?	0.36	O Módulo de Ambientado 'Online' (gratuito)
Qual é o objetivo do módulo de ambientação online?	0.67	A melhor integrar os estudantes no Modelo Pedagógico Virtual da UA e decorre na plataforma de 'e-learning'
Onde é feita a matrícula na Universidade Aberta?	0.89	Portal Académico
Qual é o limite máximo de ECTS por ano letivo?	0.93	84 ECTS
Quantas unidades curriculares é obrigatório inscrever por ano?	0.99	Pelo menos uma UC por ano letivo
O que acontece se for um estudante antigo com mais do que um curso na UAb?	0.94	Se é ou já foi estudante de outro curso na UA deve alterar o perfil no Portal Académico para visualizar a informação dos diferentes cursos
Como é calculado o valor da propina?		Em função da(s) UC/ECTS em que se inscreve
É preciso pagar algo antes da primeira matrícula?	0.49	Antes de proceder à inscrição em UC
Quando devo contactar os serviços académicos sobre situações pendentes na inscrição?	0.95	1º semestre: até 30 de novembro de 2024 no 2º semestre: até 30 de abril de 2025

Quando posso submeter um pedido de creditação de competências?		1 a 15 de setembro ou 1 a 15 de fevereiro
O que é necessário para participar na avaliação online?	0.46	Só deve contactar os serviços se esta situação não alterar para “definitiva” até ao dia 10 de setembro ou até 10 de fevereiro.

Tabela 7. Tabela com resumo geral dos testes

Tipo de Teste	Nº de Exemplos	Precisão Geral	Pontos Fortes	Pontos a Melhorar
Caracterização	9	Média-Alta	Respostas empáticas, gestão básica de intenção	Deteção de elogios vs reclamações
Base de Dados	17	Elevada	Consultas estruturadas precisas e linguagem natural	Escalonamento futuro com mais atributos
RAG	12	Elevada	Recuperação eficiente, contexto relevante	Respostas a perguntas ambíguas

Em resumo, a precisão das respostas revelada pelos testes efetuados (tabela 7) é geralmente alta, tendo sido identificados alguns exemplos que apresentam oportunidades de melhoria.

6. Conclusões

O desenvolvimento deste protótipo de *chatbot* demonstrou a viabilidade de construir uma solução funcional, baseada exclusivamente em tecnologias *open source*, capaz de responder de forma autónoma a perguntas de natureza administrativa colocadas por estudantes da Universidade Aberta. Através da combinação de modelos de linguagem, recuperação semântica com RAG e integração com uma base de dados relacional, foi possível criar um sistema híbrido e eficaz, que oferece respostas contextualizadas com base em múltiplas fontes.

O principal ponto forte do sistema reside na sua modularidade: cada componente (RAG, LLM, BD estruturada, *embeddings*) foi implementado de forma desacoplada, o que facilita a manutenção, a escalabilidade e futuras atualizações. Além disso, a decisão de recorrer a ChromaDB como base vetorial local revelou-se acertada para esta fase de prototipagem, permitindo rapidez na indexação e consultas, sem dependências externas complexas.

A integração com dados estruturados aumentou significativamente a utilidade do *chatbot*, tornando-o capaz de responder a perguntas específicas com base em dados reais — como o nome de coordenadores, número de ECTS de unidades curriculares ou cursos oferecidos por departamento. Esta camada estruturada permitiu colmatar algumas limitações do modelo de linguagem, especialmente em perguntas muito factuais.

Apesar dos resultados positivos, foram também identificadas limitações importantes, que se traduzem em oportunidades de melhoria, nomeadamente a sensibilidade à formulação da pergunta e a ausência de um mecanismo de diálogo contínuo. A interface atual, baseada em consola, cumpre a sua função, mas poderá ser substituída no futuro por uma interface web responsiva, mais adequada ao público-alvo académico. Do ponto de vista técnico, seria também vantajoso reforçar o classificador de intenções com um modelo personalizado ou com base em dados reais recolhidos na instituição.

Em conclusão, o sistema atingiu os objetivos definidos na proposta inicial, oferecendo uma base sólida para o desenvolvimento de um *chatbot* educativo institucional. A continuidade do projeto poderá passar pela migração para ambiente *cloud*, integração com plataformas como o Moodle, e eventual disponibilização pública aos estudantes, após uma fase adicional de testes e validação institucional.

Agradecimentos: Agradecemos à coordenação do curso, pelo encaminhamento do projeto no âmbito da licenciatura em Engenharia Informática na UAb, e aos membros do júri que validaram este trabalho.

REFERÊNCIAS

Chroma. (2023). Chroma: The AI-native open-source embedding database. <https://www.trychroma.com/>

LangChain. (2023). LangChain Documentation. <https://docs.langchain.com/>

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint arXiv:1908.10084. <https://arxiv.org/abs/1908.10084>

Zilliz. (2023). Milvus: Vector Database Built for Scalable AI Applications. <https://milvus.io/>

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. arXiv preprint arXiv:2005.14165. <https://arxiv.org/abs/2005.14165>

Kirk, M., Patel, D., & Liu, M. (2023). Intent Classification in Conversational Agents. arXiv preprint arXiv:2303.11356.

Mialon, G., Scialom, T., Gallé, M., & Staerman, G. (2023). Augmented Language Models: a Survey. arXiv preprint arXiv:2302.07842. <https://arxiv.org/abs/2302.07842>

Schick, T., Dwivedi-Yu, J., Schütze, H., & Sridhar, D. (2023). Toolformer: Language Models Can Teach Themselves to Use Tools. arXiv preprint arXiv:2302.04761. <https://arxiv.org/abs/2302.04761>

Anthropic. (2023). Claude 2. <https://www.anthropic.com/index/introducing-claude>

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. arXiv preprint arXiv:2005.11401. <https://arxiv.org/abs/2005.11401>

OpenAI. (2023a). GPT-4 Technical Report. <https://openai.com/research/gpt-4>

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Scialom, T. (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. Meta AI. <https://ai.meta.com/llama/>



Pedro Almeida é um entusiasta da ciência de dados e do desenvolvimento de software, com foco crescente em processamento de linguagem natural, aprendizagem máquina e sistemas inteligentes. Atualmente, está a concluir a licenciatura em Engenharia Informática, onde o seu projeto final envolveu o design e a implementação de um chatbot conversacional baseado em Python, integrando geração aumentada por recuperação de informação e técnicas de análise estatística. O seu desenvolvimento académico e profissional também abrange IA aplicada, sistemas distribuídos e tecnologias educacionais. Possui experiência adicional em ambientes militares e de segurança, tutoria técnica e desenvolvimento de soluções orientadas a dados.



Vitor Rocio, Professor Associado com Agregação da Universidade Aberta. Doutorado em Informática pela Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (2002), e licenciado em Engenharia Informática pela FCT-UNL (1993). É diretor do Departamento de Ciências e Tecnologia da UAb. Os seus principais interesses são as tecnologias das linguagens humanas, o processamento automático de línguas naturais, os sistemas de análise sintática evolutivos, e as tecnologias de elearning.