*Copyright © 2021 Autoria*. Todo o conteúdo da Revista (incluindo-se instruções, política editorial e modelos) está sob uma licença Creative Commons Atribuição 4.0 Internacional. Ao serem publicados por esta Revista, os artigos são de livre uso em ambientes educacionais, de pesquisa e não comerciais, com atribuição de autoria obrigatória. Mais informações em <http://revistas.ufpr.br/atoz/about/submissions#copyrightNotice>.

**Título do *short* *paper*: Consultando informações em documentos com o uso de inteligência artificial: um estudo de caso integrando o serviço do *Gemini Pro* a uma aplicação WEB**

**Title in English: Retrieving information from documents using artificial intelligence: a case study integrating the Gemini Pro service into a WEB application**

**Resumo**

**Introdução:** O artigo explora a viabilidade de construir um chatbot que responda perguntas sobre um documento pré-definido usando a técnica RAG e ferramentas gratuitas. **Método:** o estudo de caso implementa um chatbot usando o modelo Gemini Pro da Google e o framework LangChain e avalia as respostas obtidas confrontando as informações com o texto original. O chatbot processa perguntas sobre o Projeto Pedagógico do Curso de Graduação em Ciência da Computação de uma universidade brasileira. **Resultados:** os testes indicam que o chatbot é capaz de responder perguntas com um grau de precisão satisfatório. Observou-se que tanto a escolha dos parâmetros influencia na precisão das respostas, quanto a qualidade da pergunta. Uma vez que, com perguntas mais detalhadas e coerentes com o texto original, obteve-se respostas mais precisas. **Conclusão:** A hipótese de que é possível construir um chatbot eficaz com ferramentas gratuitas foi validada a partir da construção da aplicação e testes efetuados. O chatbot se mostra útil para responder perguntas sobre o documento, especialmente quando as perguntas são bem formuladas.

**Palavras-chave:** Chatbot. RAG. Gemini. LangChain. Aplicação web.

***Abstract***

***Introduction:*** *the article explores the feasibility of building a chatbot that answers questions about a pre-defined document using the RAG technique and free tools.* ***Method:*** *the case study implements a chatbot using Google's Gemini Pro model and the LangChain framework. The answers obtained are compared with the information in the original text to assess their accuracy. The chatbot processes questions about the Pedagogical Project of the Undergraduate Course in Computer Science at a Brazilian university.* ***Results:*** *The tests indicate that the chatbot is able to answer questions with a satisfactory degree of accuracy. It was observed that both the choice of parameters and the quality of the question influence the accuracy of the answers. More detailed and coherent questions with the original text obtained more accurate answers.* ***Conclusions:*** *Based on the application construction and tests carried out, the hypothesis that it is possible to build an effective chatbot with free tools was validated. The chatbot proves useful for answering questions about the document, especially when the questions are well-formulated.*

***Keywords:*** *Chatbot. RAG. Gemini. LangChain. Web application.*

**INTRODUÇÃO**

Atualmente, o uso de inteligência artificial em atividades do dia-a-dia está cada vez mais comum. Consequentemente, a cada dia que passa, novas tecnologias, ferramentas e aplicações destinadas a este propósito são disponibilizadas na internet.

Dentre as mais diversas tecnologias que envolvem o uso de inteligência artificial, sem dúvidas, os *Chatbots* são os mais populares no momento. Estes que são construídos em cima de *Large Language Models* (LLM), que são modelos pré-treinados com uma grande quantidade de dados. No entanto, o uso de LLMs muitas vezes pode restringir a capacidade de resposta do *Chatbot*, pois, o tanto de informação que ele “sabe” está relacionado com os dados que foram usados para treinar o modelo. Ou seja, se foi usada uma base de dados pequena ou antiga, o *Chatbot* pode apresentar dificuldades em responder algumas perguntas específicas.

É neste contexto que surgiu a técnica de *Retrieval-Augmented Generation* (RAG), em português “Geração Aumentada de Recuperação”. O RAG “aprimora os LLMs recuperando fragmentos de documentos relevantes de uma base de conhecimento externa por meio de cálculos de similaridade semântica.”(Gao, 2023, p. 1)

Quando se fala sobre LLMs, duas empresas se destacam por conta das suas soluções, a primeira delas é a *Google*, com a família de modelos ***Gemini*** e a segunda é a *OpenAI* com o modelo ***GPT***. E em relação ao desenvolvimento de aplicações que integram LLMs, uma das soluções que vêm ganhando destaque é o *framework* ***LangChain***, que visa simplificar o processo de criação destas aplicações desde o desenvolvimento até o *deploy*. Para conduzir o estudo de caso, optou-se pelo uso do modelo ***Gemini Pro*** junto com o ***LangChain***.

A hipótese levantada durante as pesquisas foi: “é realmente possível construir uma aplicação web com estas ferramentas de forma gratuita e obter resultados satisfatórios?”. Sendo assim, a partir desta hipótese, decidiu-se elaborar o estudo de caso e relatá-lo neste *paper*. Para nortear o estudo de caso, definiu-se como objetivo geral a validação desta hipótese por meio do desenvolvimento de uma aplicação WEB que, por meio de um *Chatbot*, consiga responder perguntas relacionadas a um documento previamente escolhido que, neste caso, foi o **Projeto Pedagógico do Curso de Graduação em Ciência da Computação** de uma universidade brasileira. Para isso, a *API Gemini Pro* da *Google* será integrada à aplicação com o auxílio do *LangChain*. Desta forma, podendo então relatar como se dá o processo de desenvolvimento de uma aplicação deste tipo, desde a sua concepção até a análise dos resultados obtidos. E consequentemente, responder a hipótese levantada anteriormente.

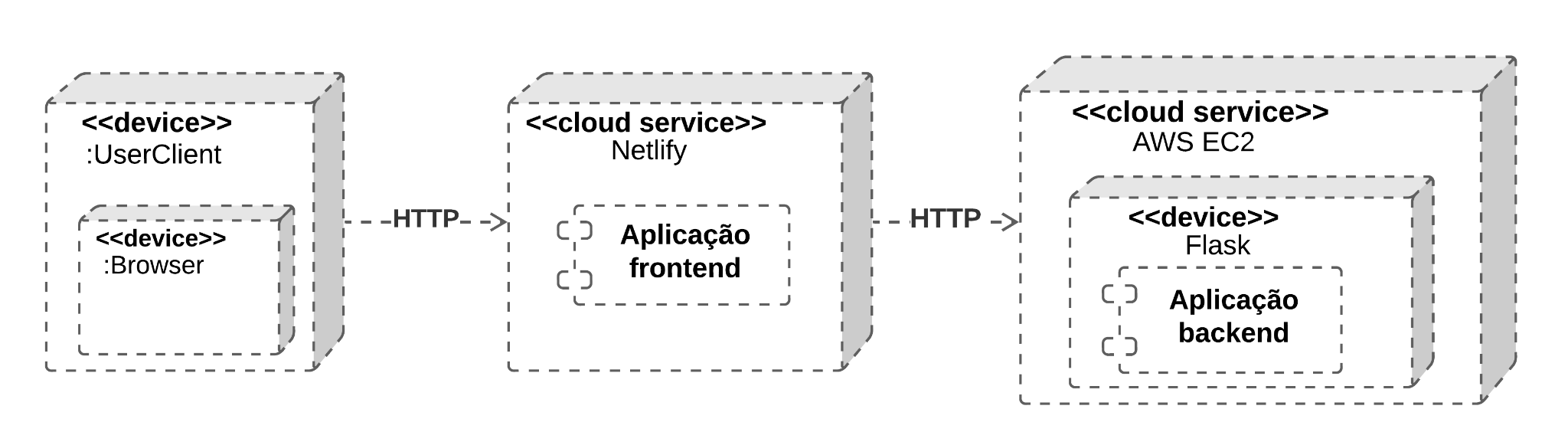
**PROCEDIMENTOS METODOLÓGICOS E RESULTADOS**

Inicialmente, além da infraestrutura, fora estudado também quais tecnologias iriam compor a *tech stack* (conjunto de ferramentas tecnológicas) do estudo de caso. Por conta da grande popularidade no contexto de Inteligência Artificial e vastidão de conteúdo na internet, optou-se por utilizar a linguagem Python para construir o *backend* da aplicação, com o auxílio do *framework* ***Flask*** para criar a rota responsável por receber a pergunta do usuário e executar a lógica de processamento do *Question Answering RAG* (QA-RAG).

Em relação ao *frontend* (interface) da aplicação, utilizou-se o *framework* ***NextJS*** devido a familiaridade da equipe. Sendo assim, foi possível criar uma interface amigável para o usuário conseguir realizar as suas consultas ao documento escolhido.

Para poder implantar a solução proposta, foram usados dois serviços em nuvem:

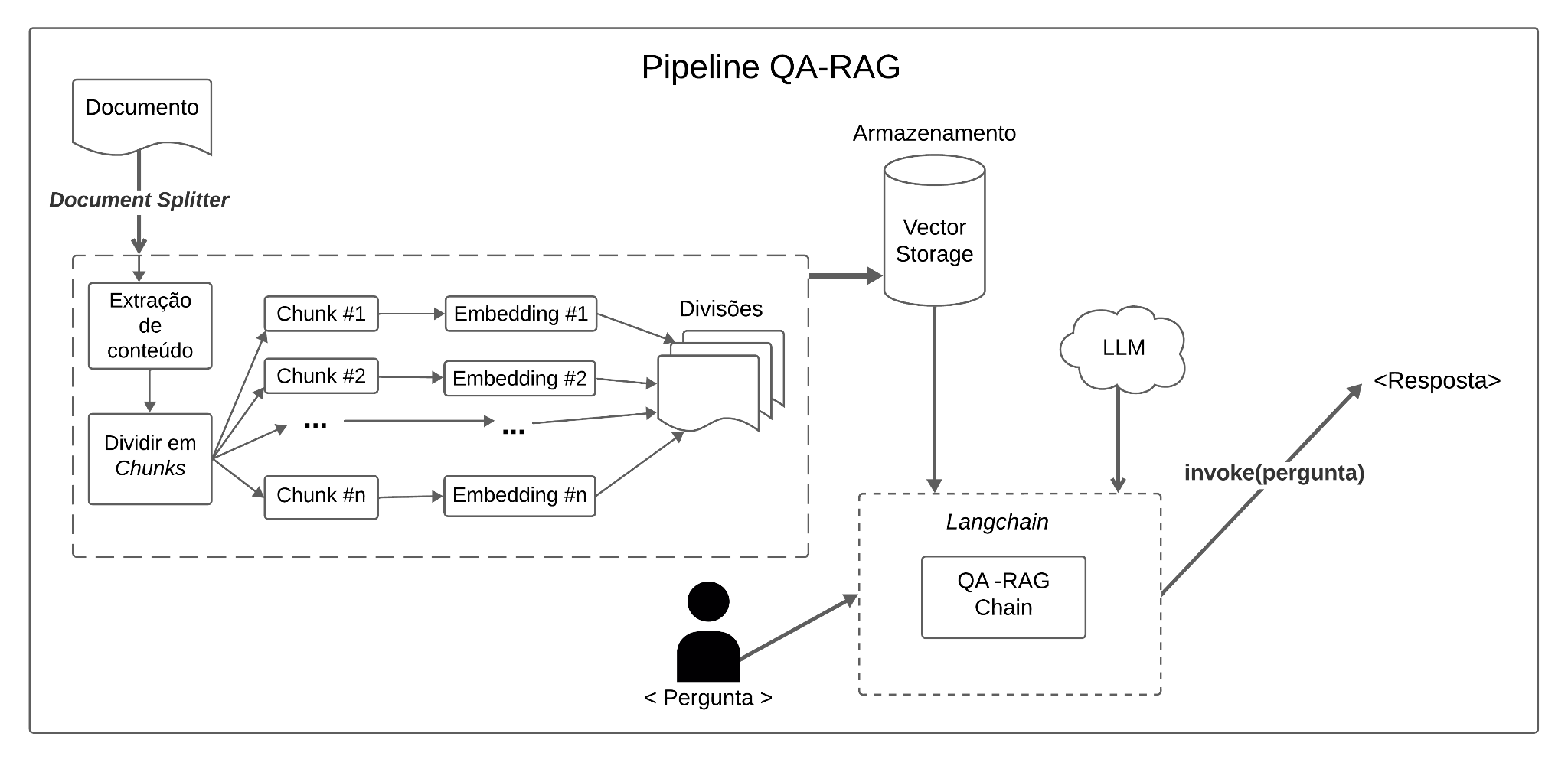
1. *Netlify*: para poder hospedar o *frontend* da aplicação;
2. *Amazon Web Services* (AWS): mais especificamente, o serviço *Amazon Elastic Compute Cloud* (EC2), para hospedar o *backend* da aplicação.



**Figura 1.** Diagrama de implantação da solução

Como pode ser visto na **Figura 1**, a infraestrutura proposta para a solução foi representada em um diagrama de implantação. O diagrama mostra que o usuário, a partir do seu navegador de internet (*browser*), acessa o *frontend* da aplicação e, este que por sua vez, realiza as requisições para o *backend* da aplicação por meio do protocolo de comunicação HTTP.

No que se diz respeito ao QA-RAG implementado no *backend*, construiu-se um *pipeline* baseado no *Langchain* e na sua biblioteca de integração com o *Google Generative AI*, pois a LLM escolhida foi o ***Gemini Pro***, que se trata de um produto da Google. Neste *pipeline*, ocorrem uma série de processamentos desde o carregamento do PDF do documento até a geração da resposta baseada na pergunta enviada pelo usuário.



**Figura 2.** Diagrama do *pipeline* QA-RAG

Observa-se na **Figura 2** que, de forma resumida, o *pipeline* desenvolvido pode ser entendido como um processo de etapas, onde cada uma desempenha um papel importante no processo como um todo. São as etapas do *pipeline*:

1. carregamento do documento;
2. extração do conteúdo;
3. divisão do conteúdo em *chunks*;
4. criação dos *embeddings* a partir dos *chunks*;
5. armazenamento das divisões geradas a partir dos *embeddings* em um *vector storage*;
6. criação da QA-RAG *Chain* a partir do *vector storage* e da LLM (*Gemini Pro*);
7. execução da QA-RAG *Chain* através do método ***invoke()*** passando como parâmetro a pergunta enviada pelo usuário.

Desta forma, a partir das etapas supracitadas, a aplicação é capaz de processar a pergunta do usuário e retornar a resposta baseada no documento.

Vale ressaltar que, por estar sendo usada uma LLM previamente treinada e que seu serviço de processamento é fornecido pela própria Google, o servidor responsável por hospedar o *backend* da aplicação não é sobrecarregado, pois o serviço em nuvem da Google fica responsável pela maior carga de processamento. No entanto, o *backend* fica responsável por lidar com as etapas **a), b), c), d)** e **e)**, onde fica toda a lógica de processamento do documento. Logo, caso o documento escolhido seja muito grande, a aplicação pode sofrer com lentidão a depender do poder de processamento da máquina responsável por hospedar a aplicação.

Quanto aos resultados obtidos, foram feitos alguns testes no intuito de avaliar a capacidade da aplicação em retornar respostas válidas, ou seja, de acordo com as informações obtidas no documento original.

Antes de detalhar os testes, é importante definir quais são os parâmetros utilizados pelo aplicativo para manipular o desempenho do QA-RAG. São eles:

1. ***CHUNK\_SIZE***: define o tamanho das seções do PDF que serão processadas pelo aplicativo;
2. ***CHUNK\_OVERLAP***: define a quantidade de texto que se repete entre seções consecutivas do PDF;
3. ***TEMPERATURE***: permite controlar o equilíbrio entre precisão e criatividade na geração de texto, onde um valor mais baixo fornece saídas mais conservadoras á um valor mais alto retorna maior aleatoriedade, gerando textos mais criativos e inesperados.

Para o primeiro teste, foi avaliado como a aplicação se comporta quando é necessário recuperar informações extensas, ou seja, dispostas ao longo de textos compridos ou até mesmo dispostas em várias páginas. Logo, o teste consiste em variar os parâmetros ***CHUNK\_SIZE*** e ***CHUNK\_OVERLAP***, começando com um ***CHUNK\_SIZE*** de 10000 e um ***CHUNK\_OVERLAP*** correspondente a 10% deste valor, ou seja, 1000. O parâmetro ***TEMPERATURE*** foi fixado em 0.5975, para chegar a um meio termo entre criatividade e conservadorismo.

Tabela 1

**Relação dos parâmetros *CHUNK\_SIZE e CHUNK\_OVERLAP* com a precisão da resposta para a pergunta: “Quais são as disciplinas optativas ofertadas pelo departamento de computação?”**

| CHUNK\_SIZE | CHUNK\_OVERLAP | Respostas corretas | Respostas incorretas |
| --- | --- | --- | --- |
| 10000 | 1000 | 60 | 15 |
| 11000 | 1100 | 24 | 0 |
| 12000 | 1200 | 0 | 0 |
| 13000 | 1300 | 24 | 6 |
| 14000 | 1400 | 4 | 7 |
| 15000 | 1500 | 3 | 0 |
| 16000 | 1600 | 12 | 0 |
| 17000 | 1700 | 38 | 0 |
| 18000 | 1800 | 53 | 0 |
| 19000 | 1900 | 12 | 0 |
| 20000 | 2000 | 32 | 0 |

Analisando a Tabela 1, percebe-se que o ***CHUNK\_SIZE*** e ***CHUNK\_OVERLAP*** não são diretamente proporcionais ao número de respostas corretas. E o resultado que obteve o maior número de respostas corretas e não conteve nenhuma incorreta, foi utilizando um ***CHUNK\_SIZE*** de 18000 e ***CHUNK\_OVERLAP*** de 1800.

Portanto, os testes subsequentes adotaram estes parâmetros para dar continuidade ao estudo de caso.

Tabela 2

**Relação entre o detalhamento das perguntas e a precisão da resposta**

| Pergunta | Resposta | Está correta |
| --- | --- | --- |
| A | 210 | Não |
| B | 2250 | Não |
| C | 2670 | Sim |

Em relação ao segundo teste, verificou-se a capacidade de compreensão da pergunta e recuperação da informação relacionada a ela por parte do QA-RAG.

Os parâmetros usados foram os mesmos do teste anterior, como citado anteriormente. E, no que se refere às perguntas utilizadas, foram usadas três perguntas com níveis de detalhamento diferentes. Da menos detalhada para a mais detalhada, são elas:

1. **A**: “Quantas horas eu preciso cumprir de carga horária obrigatória?”;
2. **B**: “Quantas horas eu preciso cumprir de carga horária de componentes obrigatórios?”;
3. **C**: “Para concluir o curso de Ciência da Computação, é necessário cursar o equivalente a quantas horas de componentes curriculares obrigatórios?”.

Logo, analisando a Tabela 2, percebe-se que o QA-RAG se mostrou mais competente quando a pergunta foi devidamente detalhada, estando mais de acordo com o texto original. Desta forma, percebe-se que, para obter melhores resultados, o usuário deve se preocupar em escrever uma boa pergunta, assim as chances de sucesso serão maiores.

Foram conduzidos testes com múltiplos documentos, incluindo o Projeto Pedagógico do Curso de Graduação em Ciência da Computação e o Projeto Pedagógico do Curso de Graduação em Sistema de Informação. Esses testes utilizaram um tamanho de *chunk* de 10000 e um *overlap* de 1000, escolhidos devido a um problema relacionado ao tempo de resposta. Os resultados desses testes estão apresentados nas Tabelas 3 e 4. A Tabela 3 exibe a correlação entre o tempo de resposta e a precisão das respostas para uma determinada pergunta, considerando dois documentos distintos. Por sua vez, a Tabela 4 apresenta esses mesmos dados, entretanto, para apenas um documento.

As perguntas utilizadas foram:

1. “O curso de Graduação em Ciência da Computação tem como objetivos?”;
2. “Os profissionais egressos do curso de Graduação em Ciência da Computação deverão:”;
3. “Qual a carga horária do curso de Graduação em Ciência da Computação?”.

Tabela 3

**Relação entre tempo de resposta e taxa de acerto com dois documentos**

| Pergunta | Tempo de resposta | Taxa de acerto |
| --- | --- | --- |
| A | 13.91 s | 0% |
| B | 13.55 s | 100% |
| C | 11.96 s | 0% |

Tabela 4

**Relação entre tempo de resposta e taxa de acerto com um documento**

| Pergunta | Tempo de resposta | Taxa de acerto |
| --- | --- | --- |
| A | 14.94 s | 100% |
| B | 12.56 s | 100% |
| C | 9.23 s | 100% |

Como resultado, observamos que ao utilizar mais de um documento, a taxa de acerto diminuiu drasticamente, com uma média de 33% das perguntas respondidas corretamente, e o tempo de resposta também aumentou ligeiramente. Por outro lado, ao utilizar apenas um documento, obtivemos uma taxa de acerto de 100% nas respostas. Com base nesses resultados, optamos por utilizar a abordagem de apenas um documento para os testes subsequentes.

**CONCLUSÕES**

O estudo de caso teve como objetivo desenvolver uma aplicação web que, por meio de um chatbot, respondesse perguntas sobre um documento pré-definido utilizando a técnica RAG e as ferramentas Gemini Pro da Google e LangChain. A hipótese central era que tal aplicação poderia ser desenvolvida de forma gratuita e com resultados satisfatórios.

O estudo implementou um pipeline QA-RAG baseado no LangChain e na integração com o Google Generative AI. O documento utilizado foi o Projeto Pedagógico do Curso de Graduação em Ciência da Computação de uma universidade brasileira.

Os resultados indicaram que a aplicação foi capaz de responder perguntas sobre o documento com um grau de precisão satisfatório. A escolha dos parâmetros *CHUNK\_SIZE* e *CHUNK\_OVERLAP* influenciou na qualidade das respostas que necessitavam de uma busca extensa sobre o documento, sendo que o valor de 18000 para *CHUNK\_SIZE* e o valor de 1800 para o *CHUNK\_OVERLAP* resultaram em melhor desempenho. A qualidade da pergunta também foi um fator importante, com perguntas mais detalhadas e coerentes com o texto original, obtendo respostas mais precisas. No entanto, testes com múltiplos documentos mostraram uma queda na taxa de acerto e um aumento no tempo de resposta.

Uma dificuldade observada durante o desenvolvimento do estudo de caso foi em relação ao acesso a estes serviços de API de LLMs de forma gratuita. Pois, com o aprimoramento destas tecnologias e aumento nos investimentos sobre elas, boa parte dos serviços disponibilizados pela internet deixaram de ser experimentais, consequentemente, deixaram de ser gratuitos. No entanto, a depender da complexidade do projeto e do uso da API, a contratação destes serviços pode ser viável, cabe ao desenvolvedor ou equipe avaliar.

A hipótese foi validada, comprovando a viabilidade de construir uma aplicação web gratuita com estas ferramentas e obter resultados satisfatórios. A aplicação se mostrou útil para responder perguntas sobre o documento, especialmente quando as perguntas eram bem formuladas.

A técnica QA-RAG mostrou-se promissora para o contexto apresentado. Ademais, a aplicação desenvolvida no estudo de caso demonstra o potencial dessa técnica e abre caminho para novas pesquisas e aplicações, como por exemplo: suporte ao cliente, educação, treinamento, pesquisa, entre outros.

**REFERÊNCIAS**

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems, 33, 9459-9474.

Elgedawy, R., Srinivasan, S., & Danciu, I. (2024). Dynamic Q&A of Clinical Documents with Large Language Models. arXiv preprint arXiv:2401.10733.

Medeiros, T., Medeiros, M., Azevedo, M., Silva, M., Silva, I., & Costa, D. G. (2023). Analysis of language-model-powered chatbots for query resolution in pdf-based automotive manuals. Vehicles, 5(4), 1384-1399.

**AGRADECIMENTOS**

Agradecer ao *Gemini* por auxiliar na realização dos testes com a API e dado os textos deste artigo nos gerou uma base para conclusão e resumo.