Simplificação da análise forense de logs utilizando Grandes Modelos de Linguagem com a técnica RAG

Carlos G. L. Barros[1], João P. A. Lima[2], Alexandre Arruda[1], Rubens Abraão da Silva Sousa[3], Alan Portela Bandeira[4]

[1]Universidade Federal do Ceará – Campus de Russas (UFC)
CEP: 62900-420 – Russas – CE – Brasil

[2]Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP) – São Paulo, SP - Brasil

[3]Instituto Federal do Ceará (IFCE) – Fortaleza – CE – Brasil

[4]Universidade Estadual do Ceará (UECE) – Fortaleza – CE – Brasil

carlosgabrieldev@alu.ufc.br, pedro.lima@ime.usp.br, alexandre.arruda@ufc.br

rubens.silva@darmlabs.ifce.edu.br, alan.portela@uece.br

Abstract. *In the digital age, the growing complexity of computer systems and the sophistication of cyberattacks significantly increase the volume of logs generated, posing challenges to cybersecurity professionals. Detecting and interpreting attacks or issues in these records are crucial for a swift response to security incidents. In this context, Large Language Models (LLMs) emerge as fundamental tools for understanding and generating natural language. This study presents an approach to analyze system and network logs, aiming to detect, correlate, and interpret anomalies using the Retrieval-Augmented Generation (RAG) technique with LLMs and interactions through targeted questions. The results demonstrate the effectiveness of the proposed approach in generating relevant insights and simplifying forensic analysis for professionals in the field.*

Resumo. *Na era digital, a crescente complexidade dos sistemas informáticos e a sofisticação dos ataques cibernéticos aumentam significativamente o volume de logs gerados, desafiando os profissionais de cibersegurança. A detecção e interpretação de ataques ou problemas nesses registros são essenciais para uma resposta rápida a incidentes de segurança. Nesse contexto, os Grandes Modelos de Linguagem (LLMs) destacam-se como ferramentas fundamentais na compreensão e geração de linguagem natural. Este estudo apresenta uma abordagem para analisar logs de sistemas e redes, visando detectar, correlacionar e interpretar anomalias, utilizando a técnica de* Retrieval-Augmented Generation *(RAG) com LLMs e interações por meio de perguntas direcionadas. Os resultados comprovam a eficácia da abordagem proposta em gerar informações relevantes e simplificar a análise forense para profissionais da área.*

Na era da informação, em que a Internet é utilizada por cerca de 67% da população mundial [International Telecommunication Union (ITU) 2023], a segurança da

informação torna-se uma preocupação cada vez mais relevante. Em um cenário de aumento contínuo na complexidade e frequência dos ataques cibernéticos, os métodos tradicionais de detecção de ataques, embora eficazes em diversos contextos, têm demonstrado limitações em ambientes altamente dinâmicos e com grandes volumes de dados [Ahmad et al. 2023].

A detecção de intrusões, se torna essencial para o reconhecimento e a preparação contra ataques futuros, é um elemento central na segurança de redes [Kwon et al. 2019]. Nesse contexto, a análise de logs constitui uma fonte crucial para detectar e mitigar falhas de segurança, possibilitando a compreensão do estado atual do ambiente, a identificação de anomalias de configuração e comportamento, e a atuação dos profissionais de segurança na resposta a incidentes [Silva and Avanço 2024].

A crescente variedade e complexidade dos sistemas, bem como dos dados por eles gerados, impõem desafios significativos aos profissionais de segurança e forense digital nas organizações [Padilha et al. 2021]. Um único log pode atender a múltiplos propósitos, enquanto outros podem ser mal definidos ou pouco estruturados. Além disso, conjuntos de logs frequentemente contêm eventos irrelevantes ou informações redundantes, o que dificulta a análise e prejudica a identificação de incidentes relevantes [Oliner et al. 2012].

Consequentemente, a complexidade e o volume dos dados digitais, aliados à diversidade de fontes e formatos, tornam a resposta rápida e eficaz a incidentes como vazamentos de dados, uso indevido de software, falhas e outras ocorrências críticas uma tarefa cada vez mais desafiadora [Nayerifard et al. 2023]. Tal cenário é especialmente problemático diante do aumento dos riscos cibernéticos e das exigências legais, como a Lei Geral de Proteção de Dados (LGPD), que demandam das organizações a capacidade de detectar, responder e se recuperar de ataques de forma eficiente [Vazquez 2024].

Embora técnicas de aprendizado de máquina tenham se mostrado promissoras na detecção de anomalias em ambientes forenses digitais, sua aplicação prática enfrenta desafios relevantes, sobretudo devido à necessidade de pipelines complexos de pré-processamento e treinamento. Em contextos heterogêneos e com restrições de recursos, esses requisitos impactam diretamente a escalabilidade, a adaptabilidade e o custo computacional das soluções [Nayerifard et al. 2023].

Nesse cenário, os Grandes Modelos de Linguagem (LLMs), em conjunto com a técnica de *Retrieval-Augmented Generation* (RAG), apresentam uma alternativa avançada para a compreensão e geração de linguagem natural [Yang et al. 2024], permitindo a recuperação e contextualização de informações a partir de bases extensas e não estruturadas. Essas capacidades vêm sendo exploradas em aplicações que exigem elevado grau de interpretação e síntese textual, como auditoria, suporte técnico e análise de sistemas [Maryamah et al. 2024].

Este trabalho propõe uma abordagem de análise forense de logs baseada em LLMs pré-treinados utilizando a técnica RAG, com foco no processamento de registros e na identificação de evidências a partir de perguntas direcionadas. A proposta destaca-se pelo foco na execução local, respeitando restrições de custo computacional, o que a diferencia de outras soluções baseadas em serviços em nuvem. A arquitetura desenvolvida utiliza o

framework LangChain[1] e o modelo Llama 3.2[2], com 3 bilhões de parâmetros, executado localmente.

A estrutura deste trabalho está organizada da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 explora os Grandes Modelos de Linguagem; a Seção 4 introduz o conceito de *embeddings*; a Seção 5 aprofunda a técnica RAG; a Seção 6 descreve em detalhes a abordagem proposta; a Seção 7 discute os resultados obtidos; e, por fim, a Seção 8 apresenta as conclusões.

1. Trabalhos relacionados

No trabalho de Kwon et al. [Kwon et al. 2019], foi investigada a aplicação de *deep learning* por meio de um modelo de Rede Totalmente Conectada (FCN), demonstrando sua eficácia na identificação de tráfego malicioso. Os autores destacaram que os métodos tradicionais de detecção de intrusões apresentam limitações, incluindo baixa flexibilidade, dificuldades de escalabilidade e alto custo associado ao uso de hardware especializado, como arranjos de portas programáveis em campo (FPGA). Diferentemente dessa abordagem, a solução proposta neste trabalho não requer treinamento supervisionado nem estrutura prévia de classificação, utilizando modelos de linguagem natural combinados com recuperação semântica para realizar análises orientadas por perguntas em tempo real.

Iniciativas como o ChatTCU[3], implementado pelo Tribunal de Contas da União, demonstram a aplicação de LLMs como o GPT-4 na estruturação de relatórios de incidentes, conferindo clareza e agilidade ao processo de auditoria [Da Silva et al. 2024]. Esses casos evidenciam o uso de modelos de linguagem em ambientes que exigem precisão e rastreabilidade, demonstrando a necessidade de adaptação contínua das técnicas de análise a novas configurações, padrões de tráfego e sistemas. Esse cenário impõe desafios a modelos que dependem de pipelines extensivos de pré-processamento e treinamento, dificultando sua adaptação a diferentes ambientes e versões de sistemas [Nayerifard et al. 2023].

Liu et al. [Liu et al. 2024] aplicaram LLMs *open-source* (Llama 3.2) com técnicas de *Self-RAG* para otimizar chatbots automotivos, enfrentando desafios semelhantes aos da análise de logs, como privacidade, limitação computacional e complexidade dos dados. A solução obteve precisão contextual superior a 0,86, mesmo em ambiente local e restrito. O uso de LLMs pré-treinados em conjunto com a técnica RAG tem se mostrado viável para alcançar resultados satisfatórios sem incorrer em altos custos de treinamento [Melz 2023].

Apesar dos avanços, muitas abordagens ainda dependem de estruturas rígidas de treinamento, com menor flexibilidade para execução local e custos computacionais elevados. Observa-se também que a integração entre compreensão contextual, recuperação semântica e uso local de LLMs voltados à análise forense de logs ainda é pouco abordada de forma unificada na literatura.

Neste contexto, este trabalho propõe uma abordagem que utiliza a técnica RAG localmente, com o uso do LangChain e do Llama 3.2, voltada à análise de logs por meio de interação orientada por perguntas. A proposta elimina a necessidade de pré-

[1]https://www.langchain.com/

[2]https://github.com/meta-llama/llama-models/tree/main/models/llama3_2

[3]https://chat-tcu.apps.tcu.gov.br/

processamento supervisionado e prioriza a eficiência computacional, destacando-se por operar em ambiente local sem dependência de infraestrutura em nuvem.

2. Grandes Modelos de Linguagem

Os Grandes Modelos de Linguagem (LLMs) são modelos de inteligência artificial que utilizam técnicas de aprendizado profundo, especialmente arquiteturas baseadas em transformadores (do inglês, *transformers*), capazes de compreender e gerar linguagem humana de forma coerente [Hadi et al. 2023]. Uma característica essencial dos LLMs é sua capacidade de processar grandes quantidades de dados, incluindo textos não estruturados, e capturar relações semânticas entre palavras e frases [Adnan and Akbar 2019].

O processo de treinamento dos LLMs envolve o pré-treinamento em corpora massivos, o qual fornece ao modelo uma base de conhecimento genérico. O ajuste fino (fine-tuning) é a especialização do modelo para tarefas específicas quando necessário. Essa estrutura permite que o modelo reconheça padrões linguísticos, raciocine sobre conceitos e gere respostas contextualizadas, mesmo em domínios técnicos [Hadi et al. 2023].

No entanto, esses modelos enfrentam desafios importantes, como a dificuldade em expandir ou revisar facilmente sua memória [Lewis et al. 2020] e sua tendência a gerar "alucinações", produzindo informações imprecisas ou factualmente incorretas. Isso é particularmente problemático em contextos como a análise forense de logs, nos quais eventos e padrões podem evoluir rapidamente. Isso ocorre porque os modelos não conseguem obter informações atualizadas em tempo real, o que pode levar a respostas desatualizadas ou imprecisas [Naveed et al. 2024, Lewis et al. 2020].

3. Embeddings

Embeddings são representações matemáticas de documentos de texto em um espaço multidimensional contínuo, construídas de forma que a distância entre vetores reflita a similaridade semântica entre os conteúdos representados. Essas representações permitem que documentos com significados semelhantes fiquem próximos no espaço vetorial, facilitando a comparação e análise. Em comparação com abordagens tradicionais como TF-IDF, que consideram as relações de forma complexa, os embeddings lidam com o contexto e a dependência gramatical e semântica por meio dos modelos treinados [Petukhova et al. 2025].

Além disso, com o avanço dos modelos de linguagem, especialmente os baseados em redes neurais profundas, os *embeddings* passaram a capturar não apenas relações semânticas entre palavras, mas também estruturas sintáticas e dependências contextuais mais complexas [Petukhova et al. 2025]. Essa capacidade é essencial para a estrutura RAG adotada, permitindo que o modelo LLM acesse diretamente os conteúdos relevantes para a construção da sua resposta, minimizando os riscos de "alucinações"ou perda do contexto estabelecido.

4. RAG (Retrieval-Augmented Generation)

A técnica de Geração Aumentada por Recuperação (RAG) visa atender às limitações de trabalhar com tarefas específicas e lidar com a geração de conteúdo incorreto, como consultas que vão além de seus dados de treinamento, recuperando fragmentos relevantes

de documentos a partir de uma base de conhecimento externa, por meio de um cálculo de similaridade semântica [Gao et al. 2024].

Diferentemente da geração, a recuperação visa localizar objetos relevantes em um vasto repositório de recursos. O conhecimento recuperado atua como memória não-paramétrica, que é facilmente atualizável. Além disso, a recuperação pode reduzir os custos e eliminar certas etapas de geração [Zhao et al. 2024].

Um processo típico de RAG ocorre da seguinte forma: a partir de uma consulta de entrada, o recuperador identifica fontes de dados relevantes e utiliza as informações recuperadas para interagir com o gerador, melhorando o processo de geração [Zhao et al. 2024].

5. Abordagem proposta

A abordagem proposta neste artigo tem como objetivo desenvolver uma arquitetura que aprimora a acessibilidade e a obtenção de informações presentes em logs de sistemas sem a necessidade de treinamento de modelos de IA. Para isso, a pesquisa fundamenta-se no framework LangChain e no modelo Llama 3.2, adaptando-os para atender às necessidades específicas da análise e do gerenciamento desses logs.

Frameworks de serviço de Grandes Modelos de Linguagem de código aberto, como Llama, têm ganhado atenção por sua capacidade de implantar rapidamente LLMs em ambientes de baixo desempenho [Yin et al. 2024]. Por sua vez, os modelos Llama têm se destacado ao obter um desempenho competitivo em relação aos melhores LLMs existentes. Por exemplo, o Llama-13B supera o GPT-3 na maioria dos *benchmarks*, apesar de ser 10 vezes menor [Touvron et al. 2023]. Esses avanços na implantação local de LLMs são relevantes para a análise forense de logs no que se refere a recursos operacionais, pois permitem a utilização desses modelos sem a necessidade de confiar dados sensíveis a serviços de terceiros.

O LangChain é um framework de código aberto projetado para simplificar o desenvolvimento de aplicações que utilizam LLMs. Servindo como uma interface universal para um grande conjunto de modelos, o LangChain fornece um ambiente de desenvolvimento centralizado que facilita a integração de LLMs com fontes de dados externas e fluxos de trabalho de software [IBM 2023]. A construção do LangChain abrange 3 componentes principais[Wang et al. 2024]:

- O primeiro é o LLM, que atua como o componente central do processo de geração.
- O segundo componente é o pesquisador (*searcher*), responsável por consultar a base de conhecimento com base na entrada do usuário.
- O terceiro componente envolve a construção de prompts, onde as informações recuperadas são combinadas com a consulta original do usuário para criar prompts detalhados que orientam os modelos de linguagem a gerar respostas.

Com base no framework LangChain, a abordagem desenvolvida implementa uma estrutura RAG local do Llama versão 3.2. Neste sentido, conforme ilustrado na Figura 1, a estrutura desenvolvida é composta pelos seguintes componentes:

- **Ingestão de logs**: Criação de documentos baseados nos logs atribuindo a cada documento os respectivos metadados, potencializando a recuperação de informações.

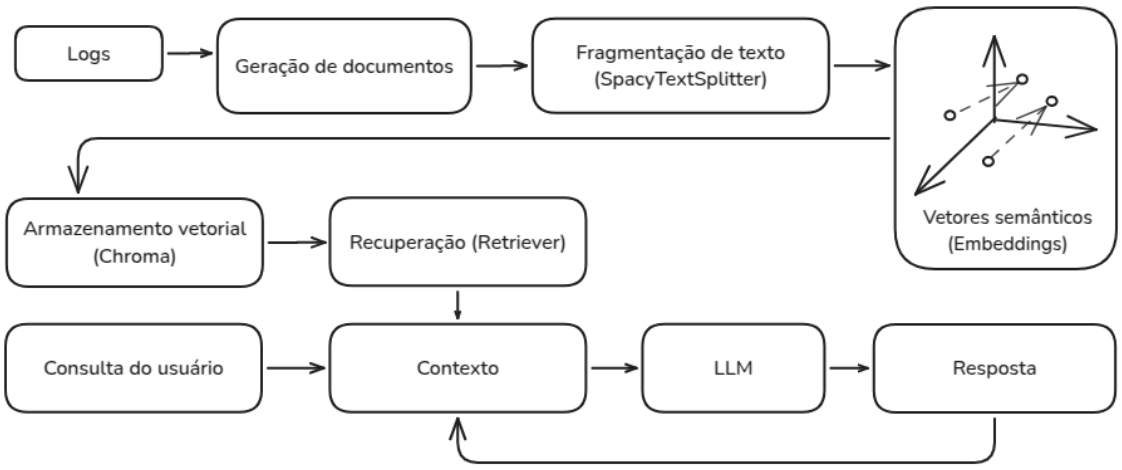

Figura 1. Estrutura RAG para análise forense de logs

- **Fragmentação de texto (*Chunking*)**: Implementação de um modelo de tokenização para dividir os documentos em pequenos fragmentos de texto, otimizados para a posterior recuperação.
- **Armazenamento**: Codificação dos fragmentos de texto em vetores semânticos por meio de um modelo de *embedding* e armazenamento dos vetores no banco de dados vetorial Chroma, criando um mecanismo de recuperação de texto indexado adaptado à análise de logs.
- **Recuperação**: Busca dos documentos mais relevantes para responder às consultas, utilizando algoritmos de recuperação e técnicas para a redução de textos irrelevantes ou redundantes.
- **Geração de diálogo**: Uso de prompts e dos textos recuperados para construir um contexto que, combinado às perguntas do usuário, é processado pelo modelo Llama hospedado localmente, gerando as respostas.

5.1. Ingestão de logs

Para este estudo, foi utilizado o conjunto de dados *Windows Event Log* [4], disponibilizado na plataforma Kaggle[5]. A escolha desse conjunto justifica-se por sua relevância para a pesquisa, uma vez que contém um amplo volume de informações um total de 158 mil logs de sistema abrangendo uma diversidade significativa de tipos e categorias de eventos.

A biblioteca Pandas[6] foi empregada para extrair os logs do conjunto de dados e, utilizando a classe "*Document*"do Langchain [7], foram criados documentos individuais para cada log, adicionando metadados com as informações presentes em cada log.

Com o objetivo de preservar a simplicidade estrutural e assegurar a unicidade de cada registro, optou-se pela criação de um documento individual para cada log. Essa abordagem visa manter a integridade e a granularidade das informações, evitando a introdução de vieses que poderiam surgir a partir da agregação de múltiplos registros

[4]https://www.kaggle.com/datasets/mehulkatara/windows-event-log
[5]https://www.kaggle.com/
[6]https://pandas.pydata.org/
[7]https://www.langchain.com/

em um único documento. Essa escolha também facilita eventuais análises forenses posteriores, nas quais a rastreabilidade e a individualização dos eventos são fundamentais para a reconstituição precisa de incidentes.

Foram realizados experimentos empíricos para a definição dos campos mais relevantes a serem mantidos nos metadados. Inicialmente, todos os dados de cada log foram considerados, e, em seguida, diferentes combinações possíveis de campos foram testadas, avaliando-se a incidência de logs relevantes recuperados na etapa de busca. A análise dos resultados concentrou-se em maximizar o recall durante a recuperação, levando à identificação da combinação mais eficaz de metadados. Como resultado, os campos "MachineName", "EntryType", "Category" e "TimeGenerated" foram selecionados como os mais impactantes para o desempenho do sistema. Ressalta-se que a extração de metadados diretamente das mensagens dos logs não foi contemplada nesta etapa e permanece como uma oportunidade para investigações futuras.

5.2. Fragmentação de texto e armazenamento

A otimização da qualidade da recuperação de informações inicia com o pré-processamento do documento. A segmentação dos documentos em parágrafos menores e a sobreposição dos segmentos (*tokens* de sentença) podem potencialmente afetar o processo de recuperação, uma vez que os documentos incorporados no espaço vetorial são selecionados como candidatos a contexto pelo algoritmo de busca [Şakar and Emekci 2025].

O tamanho dos segmentos de texto recuperados e processados é um fator crítico que influencia o desempenho dos sistemas RAG [Juvekar and Purwar 2024]. Segmentos entre 512 e 1024 tokens equilibram a oferta de contexto e a filtragem de informações irrelevantes, proporcionando respostas mais precisas. [Juvekar and Purwar 2024].

Neste estudo, foi utilizada a classe *SpacyTextSplitter* do LangChain, baseada na biblioteca de código aberto para processamento avançado de linguagem natural SpaCy[8], para processar a fragmentação dos documentos em pedaços de 1024 *tokens* com 256 *tokens* de sobreposição.

Com base nos fragmentos de texto incorporados, o banco de dados vetorial é construído utilizando o Chroma[9]. Índices vetoriais independentes, como o Chroma, melhoram a busca eficiente por *embeddings* vetoriais e os cálculos de similaridade, além de complementar facilmente outras ferramentas de IA, como o LangChain [Jeong 2023].

5.3. Recuperação de texto

O Recuperador analisa rapidamente grandes quantidades de documentos para encontrar informações relevantes para alimentar o Gerador com texto contextualizado, enriquecendo a saída do modelo da linguagem. Sem o Recuperador, o RAG seria como uma pessoa bem articulada que entrega informações irrelevantes [Sawarkar et al. 2024].

Ao implantar sistemas de recuperação, é essencial alcançar um equilíbrio entre eficácia e eficiência, garantindo que a latência, qualidade dos resultados e orçamento computacional permaneçam dentro dos limites necessários [Finardi et al. 2024]. Neste estudo,

[8]https://spacy.io/

[9]https://www.trychroma.com/

avaliamos os algoritmos de recuperação presentes no framework langchain: BM25, TF-IDF e por similaridade.

O algoritmo de busca por similaridade disponível no langchain compara os *embeddings* que têm maior similaridade com base no cálculo de cosseno [LangChain 2024]. Dada uma consulta de entrada, um recuperador é incorporado para buscar de maneira mais eficiente documentos relevantes por meio da métrica de similaridade de cosseno que pode ser calculada da seguinte forma [Rahutomo et al. 2012]:

$$Sim(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{k=1}^{t} w_{q_k} \cdot w_{d_k}}{\sqrt{\sum_{k=1}^{t} (w_{q_k})^2} \cdot \sqrt{\sum_{k=1}^{t} (w_{d_k})^2}} \tag{1}$$

O BM25 (do inglês, *Best Matching 25*) é um algoritmo clássico de recuperação baseado no modelo *bag-of-words* [Liu et al. 2024] e fundamentado em ponderação estatística para avaliar a relevância entre termos de pesquisa e documentos [Finardi et al. 2024]. Ele calcula a pontuação de frequência do termo inversa à frequência do documento (TF-IDF) entre a consulta e os documentos, levando em consideração tanto a frequência dos termos quanto o comprimento do documento. Essa abordagem permite ao BM25 filtrar de forma rápida e eficiente os documentos mais relevantes para a consulta a partir de um grande corpus de texto [Liu et al. 2024]. Nesse modelo, temos a consulta q, um fragmento de texto T, e a frequência do termo t_i no fragmento T, denotada como $f(t_i, T)$. O tamanho de T é representado por $|T|$, e o comprimento médio do fragmento é indicado por $avgTL$. O número total de fragmentos é denotado por N, e $n(t_i)$ é o número de fragmentos que contêm o termo t_i. Desse modo, a fórmula do BM25 pode ser expressa da seguinte maneira [Hu et al. 2024]:

$$BM25(q, T) = \sum_{i=1}^{n} \ln \left(\frac{N - n(t_i) + 0.5}{n(t_i) + 0.5} + 1 \right) \cdot \frac{f(t_i, T) \cdot (\kappa + 1)}{f(t_i, T) + \kappa \cdot \left(1 - b + b \cdot \frac{|T|}{\text{avgTL}}\right)} \tag{2}$$

Os parâmetros κ e b são ajustáveis, e a tarefa do modelo é retornar os k fragmentos de texto com as maiores pontuações BM25 para a consulta.

O algoritmo TF-IDF (do inglês, *Term Frequency-Inverse Document Frequency*) mede a relevância de uma palavra em um documento considerando sua frequência no texto (TF) e sua raridade no conjunto de documentos (IDF). Palavras comuns têm peso menor, enquanto termos raros e relevantes recebem maior pontuação, ajudando a filtrar informações importantes em buscas e recuperação de dados [Fan and Qin 1805]. Conforme a fórmula:

$$TFIDF = tf \cdot idf = \frac{t}{s} \cdot \log\left(\frac{M}{m} + 0.01\right) \tag{3}$$

Onde t indica o número de ocorrências da palavra no arquivo, s é a soma do número de ocorrências de todas as palavras no arquivo, M representa o número total de documentos e m representa o número de documentos que contêm os termos característicos [Fan and Qin 1805].

Nesse contexto, neste trabalho foi utilizada a técnica de *ensemble retriever* unindo os algoritmos de busca TF-IDF e BM25, ambos com peso 0.5, totalizando 1, buscando os 12 documentos mais relevantes de acordo com a consulta.

Cada um dos algoritmos discutidos possui características específicas que os tornam adequados para diferentes cenários de recuperação de informações. O método de similaridade baseada em *embeddings* é especialmente poderoso para capturar relações semânticas complexas entre os termos, permitindo a recuperação de documentos mesmo quando a consulta e o texto não compartilham as mesmas palavras, mas são semanticamente próximos. No entanto, em cenários como a análise forense de logs, onde os termos de busca costumam ser técnicos, objetivos e muitas vezes literais (como nomes de processos, comandos ou mensagens de erro), algoritmos baseados em frequência de termos, como TF-IDF e BM25, apresentaram melhor desempenho. O TF-IDF é eficaz por valorizar termos raros, que geralmente são mais discriminativos em consultas específicas de logs. Já o BM25, por incorporar normalização pelo comprimento dos fragmentos e permitir ajuste fino por meio dos parâmetros κ e b, consegue balancear a relevância dos termos frequentes sem penalizar textos curtos ou longos demais.

Assim, a escolha por combinar TF-IDF e BM25 em um *ensemble retriever* visa unir as vantagens de ambos: a capacidade do TF-IDF de destacar termos raros e a robustez estatística do BM25, resultando em uma recuperação mais precisa e alinhada às características dos dados de logs forenses.

Outra técnica empregada é a compressão de contexto. Como a linguagem natural é inerentemente redundante, um mesmo texto pode ser representado em diferentes comprimentos dentro de um LLM, preservando a informação essencial [Ge et al. 2024, Jiang et al. 2023]. Estudos anteriores já demonstraram que reduzir a entrada do modelo por meio da compactação do contexto pode atenuar o tempo de geração no RAG [Rau et al. 2024]. Além disso, esse método pode melhorar o desempenho de tarefas subsequentes, permitindo o uso de comandos (*prompts*) mais longos e aumentando a eficiência de inferência dos LLMs [Jiang et al. 2023]. Para isso, foi utilizado o modelo menor Gemma2:2b para extrair as partes mais relevantes dos documentos, empregando a classe *LLMChainExtractor* do LangChain em conjunto com a *ContextualCompressionRetriever*, responsável pela compressão.

5.4. Geração de diálogo

Pesquisas anteriores mostraram que o uso estratégico de técnicas de engenharia de *prompt* pode aprimorar de forma significativa o desempenho dos Grandes Modelos de Linguagem (LLMs), otimizando sua capacidade de gerar respostas mais precisas e coerentes [Wei et al. 2022, Kojima et al. 2022]. O trabalho de Li et al. destaca o papel crítico do contexto na melhoria do desempenho dos Grandes Modelos de Linguagem. O estudo revela que fornecer orientações contextualmente relevantes pode melhorar significativamente a precisão e a robustez das respostas do modelo [Li et al. 2023].

Após a recuperação dos documentos relevantes, seus conteúdos e metadados são formatados em um texto estruturado e incorporados a um *prompt* de contextualização. Para este trabalho, foi utilizado o seguinte *prompt*: “Utiliza os seguintes elementos de contexto para responder à pergunta no final. Mantenha a resposta o mais completa possível citando os textos fontes. Caso a resposta não seja conhecida, informe que não há uma resposta disponível em vez de tentar gerar uma.”Em seguida, o usuário faz sua pergunta, que é processada pelo sistema de recuperação e adicionada ao contexto junto aos documentos relevantes. Por fim, o contexto é processado pelo modelo de LLM, que fornece a resposta.

6. Resultados

A avaliação da estrutura RAG nesta pesquisa utilizou um método de pontuação inspirado em abordagens da literatura, como o trabalho de G-Eval [Liu et al. 2023]. Seguindo a metodologia de Finardi et al., aplicou-se um sistema de pontuação dividido em cinco categorias para comparar a similaridade contextual entre textos, já que métricas tradicionais como BLEU e ROUGE não capturam adequadamente a equivalência semântica entre sentenças reformuladas [Finardi et al. 2024]. Além disso, conforme Li et al., a calibração correta permite automatizar a avaliação, usando LLMs para autoavaliar a qualidade de seu desempenho [Li et al. 2023]. Para avaliar o desempenho do modelo, utilizou-se o GPT-4 na geração de perguntas e na atribuição de pontuações às respostas, com base em prompts específicos descritos no Apêndice A, garantindo uma avaliação alinhada ao conteúdo semântico das respostas geradas. O sistema de pontuação adotado segue a seguinte escala [Finardi et al. 2024]:

- **Pontuação 1**: A resposta não tem relação com a referência.
- **Pontuação 3**: A resposta tem pouca relevância, mas não está alinhada com a referência.
- **Pontuação 5**: A resposta tem relevância moderada, mas contém imprecisões.
- **Pontuação 7**: A resposta está alinhada com a referência, mas tem omissões pequenas.
- **Pontuação 10**: A resposta está completamente precisa e se alinha perfeitamente com a referência.

Adicionalmente, foi calculada a precisão do retorno de documentos relevantes pela estrutura RAG, conforme a fórmula:

$$\text{Precisão} = \frac{\text{Número de documentos relevantes retornados}}{\text{Número total de documentos retornados}} \tag{4}$$

Para avaliar o desempenho da estrutura RAG, foram geradas diversas perguntas e respostas baseadas em logs do sistema. Abaixo, são apresentados alguns exemplos respondidos pela estrutura desenvolvida neste artigo:

1. **Qual foi a mensagem registrada pelo sistema no evento gerado em ”2020-11-01 06:38:50”pelo “SecurityCenter”?**
   - **Resposta:** “updated quick heal firewall status successfully to security_product_state_on..”Essa mensagem indica que o estado do Quick Heal Firewall foi atualizado com sucesso para “security_product_state_on”, o que sugere que a configuração de segurança do firewall foi alterada.
2. **Qual foi o horário de encerramento da transação do Windows Installer para o arquivo `c2rintloc.en-us.16.msi` em 2020-11-13?**
   - **Resposta:** O horário de término foi às 06:50:51, conforme indicado na mensagem: “ending a windows installer transaction”.
3. **Qual foi o ID do processo cliente associado ao encerramento da transação do Windows Installer para o arquivo `c2rint64.16.msi` em “2020-10-26”?**
   - **Resposta:** O ID do processo cliente registrado para o término da transação do Windows Installer referente ao arquivo ‘c2rint64.16.msi’ em ‘2020-10-26’ é *11616*.

4. **Qual é a sequência de eventos relacionados ao serviço de Software Protection Platform entre `2020-11-10 16:30:50` e `2020-11-13 18:07:08`, incluindo os motivos de re-agendamento e os resultados das verificações de status de licenciamento?**
    - **Resposta:** Infelizmente, não consegui encontrar informações relevantes sobre o serviço de Software Protection Platform nesse intervalo. No entanto, algumas observações podem ser feitas:
        - O evento "successfully scheduled software protection service for re-start"indica que o serviço foi re-agendado para um novo horário de execução. A data do re-agendamento é 2020-11-07 10:29:53z.
        - O motivo do re-agendamento não está explicitamente detalhado, mas a mensagem "rulesengine..."sugere que o re-agendamento foi realizado por um mecanismo de regras.
        - Não há informações sobre os resultados das verificações de status de licenciamento no registro de evento.
5. **Qual foi o erro identificado no evento relacionado ao serviço `TileRepositoryS-1-5-18` na data `2020-11-07` às `18:21`?**
    - **Resposta:** O erro identificado foi: "error -1023 (0xfffffc01) occurred while opening logfile c:\windows \system32 \config \systemprofile \appdata \local \tiledatalayer \database \edb.log."
6. **Qual problema foi identificado no evento do `igfxCUIService2.0.0.0` registrado no timestamp `2020-11-01 06:38:49` com o Event ID `0`?**
    - **Resposta:** O problema identificado no evento do 'igfxCUIService2.0.0.0' registrado no timestamp '2020-11-01 06:38:49' é a falta de descrição do evento, que pode estar relacionada à falta de informações no registro ou permissões insuficientes para acessar os arquivos necessários.

Os resultados obtidos na avaliação da estrutura RAG demonstram um desempenho consistente e promissor. A média de pontuação atribuída pelo GPT-4 às respostas geradas foi de 8,17, indicando um alto alinhamento com as referências semânticas. Embora a precisão na recuperação de documentos relevantes tenha apresentado uma média de 0,3194, essa métrica não reflete de maneira significativa a eficácia do sistema. Tal resultado é esperado, pois o número de documentos retornados foi fixado como uma constante, enquanto, em diversas perguntas, a quantidade de documentos relevantes disponíveis era inferior a esse valor. Ademais, os documentos recuperados passam posteriormente por uma etapa de compressão, na qual conteúdos irrelevantes e marginalmente relevantes são descartados, assegurando que apenas as informações mais pertinentes sejam utilizadas na geração das respostas.

Em contrapartida, o cálculo do Recall mostrou uma média de 0,8256, sugerindo uma boa cobertura das informações pertinentes. Esses resultados reforçam a eficácia da combinação de LLMs com a técnica RAG, demonstrando seu potencial para auxiliar na análise forense ao processar logs e responder a perguntas técnicas de maneira precisa e contextualizada.

7. Conclusão

Neste trabalho, foi apresentada uma abordagem para análise forense de logs utilizando LLMs com a técnica RAG com uma arquitetura que possa ser executada localmente por

meio do framework LanChain e modelo Llama 3.2. Desta forma, o foco é possibilitar, dado a interação com a LLM, responder as questões do usuário, recuperando informações relevantes dado um grande volume de logs que preserve a precisão semântica e interoperabilidade.

A estrutura proposta foi capaz de superar limitações que as LLMs tradicionais ainda são um desafio, como as "alucinações" e dificuldade de lidar com dados não presentes em sua base de treinamento. Ao termos incorporado a recuperação semântica com indexação vetorial e compressão de contexto, a solução foi capaz de oportunizar respostas contextualizadas, preservando a rastreabilidade das informações e o aspecto crítico da investigação forense.

Os resultados, puderam confirmar que a combinação entre LLMs e a técnica RAG, quando executadas de forma local com uma arquitetura modular e base vetorial, melhora significativamente a capacidade de análise automática dos logs. Com a pontuação de 8,17 e um recall elevado de 0,8256, reforça a solução proposta como eficaz. Embora, a precisão na recuperação de documentos relevantes tenha sido moderada com 0,3194, visto a abrangência mediante ao contexto sensível.

Para mais, este trabalho reforça o potencial que as LLMs possibilitam alinhadas a área de cibersegurança, quando integradas a arquiteturas flexíveis, interpretáveis e independentes de arquitetura externa. Dada a aplicação do RAG, abre-se espaço para novas investigações na resposta a incidentes, auditoria automatizada, detecção de ameaças, como também análise de conformidade.

Desta forma, temos como possibilidades futuras as seguintes propostas: i) aplicação de arquitetura de logs em fontes diferentes, como em ambientes em nuvem; ii) agrupamento dos logs com base em determinada categoria ou caracteristica compartilhada; iii) possibilidade de incremento no mecanismo de extração semântica automática de metadados; e iv) possibilidade de construção de interface para utilização de usuário não técnicos.

Referências

Adnan, K. and Akbar, R. (2019). An analytical study of information extraction from unstructured and multidimensional big data. *Journal of Big Data*, 6(1):1–38.

Ahmad, R., Alsmadi, I., Alhamdani, W., and Tawalbeh, L. (2023). Zero-day attack detection: a systematic literature review. *Artificial Intelligence Review*, 56(10):10733–10811.

Da Silva, E. H. M., dos Santos, E. M. F., de Barros Monteiro, M. L., Bezerra, S. L., and de Miranda, S. C. (2024). Chattcu: Inteligência artificial como assistente do auditor. *Revista do TCU*, 153:19–45.

Fan, H. and Qin, Y. (2018/05). Research on text classification based on improved tf-idf algorithm. In *Proceedings of the 2018 International Conference on Network, Communication, Computer Engineering (NCCE 2018)*, pages 501–506. Atlantis Press.

Finardi, P., Avila, L., Castaldoni, R., Gengo, P., Larcher, C., Piau, M., Costa, P., and Caridá, V. (2024). The chronicles of rag: The retriever, the chunk and the generator.

Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., and Wang, H. (2024). Retrieval-augmented generation for large language models: A survey.

Ge, T., Jing, H., Wang, L., Wang, X., Chen, S.-Q., and Wei, F. (2024). In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*.

Hadi, M. U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M. B., Akhtar, N., Wu, J., Mirjalili, S., et al. (2023). Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*.

Hu, J., Zhou, Y., and Wang, J. (2024). Intrinsic evaluation of rag systems for deep-logic questions.

IBM (2023). What is langchain? Disponível em: `https://www.ibm.com/think/topics/langchain`. Acesso em: 19 de Janeiro de 2025.

International Telecommunication Union (ITU) (2023). Global offline population steadily declines to 2.6 billion people in 2023. Disponível em: `https://www.itu.int/itu-d/reports/statistics/2023/10/10/ff23-internet-use/`. Acesso em: 03 de Janeiro de 2025.

Jeong, C. (2023). A study on the implementation of generative ai services using an enterprise data-based llm application architecture. *Advances in Artificial Intelligence and Machine Learning*, 03(04):1588–1618.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. (2023). LLMLingua: Compressing prompts for accelerated inference of large language models. In Bouamor, H., Pino, J., and Bali, K., editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376, Singapore. Association for Computational Linguistics.

Juvekar, K. and Purwar, A. (2024). Introducing a new hyper-parameter for rag: Context window utilization.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Kwon, D., Kim, H., Kim, J., Suh, S. C., Kim, I., and Kim, K. J. (2019). A survey of deep learning-based network anomaly detection. *Cluster Computing*, 22:949–961.

LangChain (2024). Select by similarity. Disponível em: `https://python.langchain.com/v0.1/docs/modules/model_io/prompts/example_selectors/similarity/`. Acesso em: 19 de Janeiro de 2025.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Li, X., Tang, H., Chen, S., Wang, Z., Maravi, A., and Abram, M. (2023). Context matters: Data-efficient augmentation of large language models for scientific applications.

Liu, F., Kang, Z., and Han, X. (2024). Optimizing rag techniques for automotive industry pdf chatbots: A case study with locally deployed ollama models.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. (2023). G-eval: Nlg evaluation using gpt-4 with better human alignment.

Maryamah, M., Irfani, M. M., Tri Raharjo, E. B., Rahmi, N. A., Ghani, M., and Raharjana, I. K. (2024). Chatbots in academia: A retrieval-augmented generation approach for improved efficient information access. In *2024 16th International Conference on Knowledge and Smart Technology (KST)*, pages 259–264.

Melz, E. (2023). Enhancing llm intelligence with arm-rag: Auxiliary rationale memory for retrieval augmented generation.

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., and Mian, A. (2024). A comprehensive overview of large language models.

Nayerifard, T., Amintoosi, H., Bafghi, A. G., and Dehghantanha, A. (2023). Machine learning in digital forensics: A systematic literature review.

Oliner, A., Ganapathi, A., and Xu, W. (2012). Advances and challenges in log analysis. *Communications of the ACM*, 55(2):55–61.

Padilha, R., Theóphilo, A., Andaló, F. A., Vega-Oliveros, D. A., Cardenuto, J. P., Bertocco, G., Nascimento, J., Yang, J., and Rocha, A. (2021). A inteligência artificial e os desafios da ciência forense digital no século xxi. *Estudos Avançados*, 35(101):113–138.

Petukhova, A., Matos-Carvalho, J. P., and Fachada, N. (2025). Text clustering with large language model embeddings. *International Journal of Cognitive Computing in Engineering*, 6:100–108.

Rahutomo, F., Kitasuka, T., Aritsugi, M., et al. (2012). Semantic cosine similarity. In *The 7th international student conference on advanced science and technology ICAST*, volume 4, page 1. University of Seoul South Korea.

Rau, D., Wang, S., Déjean, H., and Clinchant, S. (2024). Context embeddings for efficient answer generation in rag.

Sawarkar, K., Mangal, A., and Solanki, S. R. (2024). Blended rag: Improving rag (retriever-augmented generation) accuracy with semantic search and hybrid query-based retrievers.

Silva, E. M. D. and Avanço, L. (2024). Visibilidade em cibersegurança: Uma pesquisa exploratória. In *20th CONTECSI-INTERNATIONAL CONFERENCE ON INFORMATION SYSTEMS AND TECHNOLOGY MANAGEMENT VIRTUAL*.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models.

Vazquez, F. J. B. (2024). Política de resposta a incidentes cibernéticos e estratégias de aderência à legislação brasileira. *Dataset Reports*, 3(1):114–119.

Wang, Z., Liu, J., Zhang, S., and Yang, Y. (2024). Poisoned langchain: Jailbreak llms by langchain.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., ichter, b., Xia, F., Chi, E., Le, Q. V., and Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Yang, H., Zhang, M., Wei, D., and Guo, J. (2024). Srag: Speech retrieval augmented generation for spoken language understanding. In *2024 IEEE 2nd International Conference on Control, Electronics and Computer Technology (ICCECT)*, pages 370–374.

Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., and Chen, E. (2024). A survey on multimodal large language models. *National Science Review*, 11(12).

Zhao, P., Zhang, H., Yu, Q., Wang, Z., Geng, Y., Fu, F., Yang, L., Zhang, W., Jiang, J., and Cui, B. (2024). Retrieval-augmented generation for ai-generated content: A survey.

Şakar, T. and Emekci, H. (2025). Maximizing rag efficiency: A comparative analysis of rag methods. *Natural Language Processing*, 31(1):1–25.

A. Prompts Utilizados

Neste apêndice, apresentamos os prompts utilizados na avaliação do modelo.

Prompt 1: Geração de Pergunta

Você receberá um documento contendo logs de sistema e deverá gerar uma única pergunta com base nesse documento.
A pergunta deve ser clara, específica e detalhada para que tenha apenas uma resposta correta, baseada exclusivamente nos logs fornecidos.
Instruções:
- A pergunta deve ser formulada de forma técnica e precisa, levando em conta o contexto dos logs.
- A resposta para a pergunta deve ser um trecho exato dos logs fornecidos, sem modificações ou interpretações.
- A pergunta deve exigir uma resposta com mais de 4 palavras.
- Evite perguntas genéricas ou vagas. Certifique-se de que a pergunta explora informações relevantes dos logs, como eventos, erros, timestamps, IPs, IDs de processos, status de conexões, entre outros.
Início dos logs: `{document}`

Prompt 2: Avaliação da Pergunta

Você receberá uma pergunta gerada pelo modelo e a resposta extraída dos logs. Seu objetivo é comparar a pergunta com a resposta e atribuir uma pontuação numérica de acordo com o sistema de avaliação descrito abaixo.
Sistema de Pontuação - Nota 1: A pergunta não tem relação com os logs ou está completamente errada. - Nota 3: A pergunta tem pouca relevância, mas não está alinhada com a resposta correta. - Nota 5: A pergunta tem relevância moderada, mas contém imprecisões. - Nota 7: A pergunta está alinhada com a resposta, mas tem pequenas omissões ou poderia ser mais clara. - Nota 10: A pergunta é completamente precisa, clara e alinhada perfeitamente com a resposta.
Pergunta gerada: {`pergunta`}
Resposta extraída dos logs: {`resposta`}
Avalie com muito cuidado e atenção, levando o tempo necessário para fornecer uma pontuação de qualidade.
Exemplos de Pontuação
- Nota 1: [“Quantos usuários acessaram o sistema?”, “Erro fatal no kernel às 03:45:12.”]
- Nota 3: [“Qual erro ocorreu no sistema?”, “O serviço foi reiniciado manualmente por admin às 14:23:09.”]
- Nota 5: [“Qual erro crítico ocorreu no sistema?”, “Ocorreu um erro para o usuário root.”]
- Nota 7: [“O usuário admin encontrou algum erro de permissão?”,“O usuário admin tentou acessar um recurso restrito sem permissão.”]
- Nota 10:[“O usuário admin tentou acessar um recurso restrito sem permissão?”, “O usuário admin tentou acessar um recurso restrito sem permissão.”]
Responda apenas com um número correspondente à pontuação.