Muitas Classes Desbalanceadas? Escale na Nuvem! Um Relato de Uso da AWS para Classificação de Texto com RAG-labels

Celso França[1], **Leonardo Rocha**[2], **Marcos André Gonçalves**[1]

[1]Universidade Federal de Minas Gerais (UFMG), Brasil

[2]Universidade Federal de São João del Rei (UFSJ), Brasil

{celsofranca,mgoncalv}@dcc.ufmg.br, lcrocha@ufsj.edu.br

Resumo. *Este artigo apresenta a experiência usando a infraestrutura AWS para desenvolvimento e experimentação em larga escala do RAG-Fuse, combinando Amazon Bedrock para gerar representações semânticas de classes (RAG-labels), instâncias EC2 com GPU para o fine-tuning de Small Language Models (SLMs) e AWS S3 para armazenamento de dados e modelos. A arquitetura resultante viabiliza um pipeline escalável, eficiente e alinhado a aplicações do mundo real, sendo até* ***70% mais eficaz*** *e consumindo até* ***10 vezes menos*** *recursos do que LLMs com bilhões de parâmetros.*

Abstract. *This paper presents our experience using the AWS infrastructure to support large-scale development and experimentation of RAG-Fuse, combining Amazon Bedrock for generating class semantic representations (RAG-labels), EC2 GPU instances for fine-tuning Small Language Models (SLMs), and AWS S3 for data and model storage. The resulting architecture enables a scalable, efficient pipeline aligned with real-world applications, achieving up to* ***70% higher effectiveness*** *while consuming up to* ***10 times fewer*** *computational resources than LLMs with billions of parameters.*

1. Introdução

A Classificação Multi-Classe de Texto (CMCT) é uma tarefa fundamental no Processamento de Linguagem Natural (PLN), desempenhando um papel central na organização, indexação e recuperação de informação em domínios complexos, como saúde, educação e literatura científica [Sikosana et al. 2024]. Em cenários reais de aplicação, entretanto, a CMCT enfrenta pelo menos dois desafios estruturais. O primeiro é o ***gap*** **léxico-semântico**, que se manifesta principalmente na forma de *vocabulary mismatch* e *ambiguidade semântica*, exigindo mecanismos capazes de capturar relações contextuais além da mera correspondência lexical entre termos. O segundo desafio é o **desbalanceamento de classes**, em que poucas classes majoritárias concentram a maior parte das instâncias, enquanto numerosas classes minoritárias possuem poucos exemplos de treinamento. Esse cenário dificulta a generalização dos modelos, introduz vieses no processo de aprendizado e amplifica os efeitos do *gap* léxico-semântico.

Historicamente, abordagens tradicionais de *Machine Learning* (ML), como *Support Vector Machines* e *Random Forests*, foram amplamente utilizadas em tarefas de classificação textual devido à sua eficiência computacional e interpretabilidade. No entanto, esses métodos apresentam limitações na captura de relações semânticas profundas.

Com o avanço das representações contextuais, arquiteturas baseadas em *Small Language Models* (SLMs), como RoBERTa [Sy et al. 2024] e BART [Lewis et al. 2020a], passaram a explorar embeddings densos e contextualizados, proporcionando melhorias significativas na modelagem semântica. Posteriormente, os *Large Language Models* (LLMs) ampliaram esse paradigma ao combinar arquiteturas mais sofisticadas com pré-treinamento em larga escala. Apesar de seu elevado desempenho, especialmente em cenários *zero-shot*, o *fine-tuning* desses modelos ainda é frequentemente necessário para alcançar alta efetividade em CMCT [de Andrade et al. 2023]. Esse processo, entretanto, impõe custos computacionais consideráveis, demandando grandes volumes de dados rotulados, infraestrutura de alto desempenho e elevado consumo energético [Dettmers et al. 2023].

Nesse contexto, abordagens recentes têm buscado alternativas que conciliem efetividade semântica e eficiência computacional. O estado da arte nesse sentido é o RAG-Fuse [França et al. 2025], uma abordagem que difere das estratégias tradicionais de *ML*, *SLMs* e *LLMs*. Enquanto essas abordagens aprendem uma função de probabilidade *indireta* entre textos e rótulos de classe, o **RAG-Fuse** reformula a CMCT como um problema de recuperação de informação, combinando técnicas de ranqueamento e de fusão de *rankings* (Figura 1). Para reduzir o *gap* léxico-semântico entre documentos e classes, a abordagem introduz os **RAG-labels**, descrições enriquecidas de classes geradas por meio de *Retrieval Augmented Generation* (RAG) [Lewis et al. 2020b] combinada com *prompt optimization* [Chen et al. 2024]. Uma vez enriquecida a representação semântica dos rótulos, algoritmos de fusão de *ranking* [Bassani 2023] são aplicados de forma estratificada entre classes majoritárias e minoritárias, permitindo combinar os resultados de recuperadores densos, especializados em relações semânticas, e de recuperadores esparsos, os quais são focados em correspondências léxicas e sintáticas. Essa complementariedade contribui para reduzir simultaneamente o *gap* semântico e os efeitos do desbalanceamento.

A computação em nuvem desempenhou um papel fundamental no desenvolvimento e na experimentação do **RAG-Fuse**, permitindo o uso eficiente e responsável de recursos computacionais. Nesse contexto, este artigo apresenta um relato de experiência sobre a implementação e experimentação do **RAG-Fuse** na infraestrutura da **AWS**. O objetivo principal é demonstrar como a orquestração de serviços gerenciados de Inteligência Artificial, em particular o **Amazon Bedrock**, combinada à elasticidade computacional do **Amazon EC2**, viabilizou a condução de experimentos em cenários de alta complexidade computacional.

Ao longo deste relato, descrevemos como o uso do Amazon Bedrock eliminou a necessidade de provisionar e gerenciar infraestrutura dedicada para hospedar LLMs responsáveis pela geração dos *RAG-labels*, permitindo que o foco da pesquisa permanecesse na lógica algorítmica. Paralelamente, instâncias Amazon EC2 otimizadas com suporte a GPU foram empregadas para o *fine-tuning* dos SLMs, proporcionando paralelismo computacional e reduzindo significativamente o tempo total de execução dos experimentos. Os resultados operacionais indicam que a adoção de infraestrutura em nuvem pode democratizar o acesso a *pipelines* avançados de IA generativa para grupos acadêmicos de pesquisa. Esse ganho foi particularmente evidente devido à possibilidade de executar múltiplos experimentos em paralelo, acelerando o ciclo de obtenção de resultados e tomada de decisões de pesquisa.

2. Visão Geral da Abordagem RAG-Fuse

O **RAG-Fuse** [França et al. 2025] é um *pipeline* complementar de recuperação e fusão em dois estágios, projetado para lidar simultaneamente com os desafios do *gap* léxico-semântico e do desbalanceamento de classes. A proposta reformula a CMCT como um problema de Recuperação de Informação (RI). Nesse paradigma, em vez de aprender uma função de probabilidade indireta que associa textos a rótulos de classe, o modelo interpreta o texto de entrada como uma consulta e o conjunto de classes como uma coleção de "documentos" candidatos a serem ranqueados com base em sua similaridade léxico-semântica.

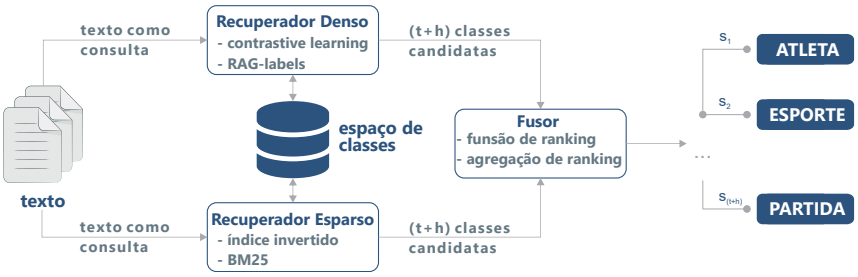

Figura 1. Arquitetura geral do pipeline RAG-Fuse [França et al. 2025].

Conforme ilustrado na Figura 1, o pipeline é estruturado em dois estágios complementares. No primeiro estágio, o **RAG-Fuse** realiza uma divisão dinâmica do espaço de classes *para cada documento de entrada*. Essa etapa utiliza recuperadores esparsos e densos de forma complementar para gerar conjuntos candidatos de classes potencialmente relevantes. No segundo estágio, os *rankings* produzidos pelos recuperadores são combinados por meio de uma estratégia de fusão de *rankings*. Ao integrar recuperadores com propriedades complementares, o pipeline explora simultaneamente a capacidade de correspondência lexical precisa dos modelos esparsos e o poder de generalização semântica dos modelos densos. Como resultado, a etapa de fusão produz um escore de relevância mais robusto para cada par $(texto, classe)$.

2.1. Redução do Gap Léxico-Semântico via RAG-labels

Uma das principais contribuições do RAG-Fuse é a introdução dos *RAG-labels*, descrições textuais enriquecidas e contextualizadas das classes. Em vez de representar cada classe apenas por um identificador ou um rótulo curto, o método gera descrições semânticas mais completas, capazes de capturar relações conceituais relevantes entre classes e documentos. A geração desses rótulos enriquecidos combina *Retrieval-Augmented Generation* (RAG) [Lewis et al. 2020b] com um mecanismo iterativo de *prompt optimization* [Chen et al. 2024]. Inicialmente, documentos representativos são recuperados a partir do conjunto de treinamento e utilizados como contexto para a geração de descrições semânticas das classes. Em seguida, um LLM atua simultaneamente como gerador de texto e otimizador de instruções, refinando iterativamente os *prompts* para produzir descrições mais informativas e discriminativas. Essa etapa é computacionalmente intensiva, pois envolve múltiplas chamadas de inferência a modelos de bilhões de parâmetros. Quando executado localmente para centenas ou milhares de classes, o processo rapidamente esgota a memória de GPUs convencionais e impõe um alto custo computacional, tornando necessária a utilização de infraestrutura escalável.

2.2. Recuperação Dual e Fusão de Rankings

Uma vez enriquecido o espaço de classes por meio dos *RAG-labels*, o pipeline realiza a recuperação de candidatos utilizando dois mecanismos complementares:

- **Recuperador Esparso:** especializado em correspondência léxica e sintática exata, baseado em índices invertidos de alta dimensionalidade e na função de ranqueamento BM25.
- **Recuperador Denso:** focado em similaridade semântica profunda, obtida por meio do ajuste fino (*fine-tuning*) de SLMs, como BERT ou RoBERTa. O treinamento é realizado com aprendizado contrastivo utilizando a função de perda *Normalized Temperature-Scaled Cross Entropy*. Nesse processo, textos e classes são projetados em um espaço vetorial compartilhado, permitindo a realização de buscas eficientes via algoritmos de vizinhos mais próximos aproximados (HNSW).

Por fim, os resultados obtidos pelos dois recuperadores são normalizados e combinados por meio do algoritmo clássico de fusão de *rankings* CombMNZ. Essa estratégia explora a complementariedade entre os sinais léxicos e semânticos produzidos pelos dois sistemas de recuperação, resultando em uma estimativa mais robusta da relevância entre textos e classes.

A natureza computacionalmente intensiva dessas etapas, particularmente o treinamento iterativo de modelos baseados em *transformers* e a execução de buscas vetoriais em larga escala, torna desafiadora a execução do pipeline em ambientes locais restritos. Esse cenário motiva diretamente a adoção de uma arquitetura baseada em computação em nuvem.

3. Arquitetura da Solução na AWS

A utilização de um ambiente de pesquisa utilizando infraestrutura AWS foi um fator decisivo para viabilizar a execução do *pipeline* RAG-Fuse e a avaliação das linhas de base em um tempo experimental viável. A arquitetura foi concebida para desacoplar a inferência intensiva de *Large Language Models* (LLMs) das etapas de treinamento e indexação das representações densas, permitindo otimizar simultaneamente o custo computacional e o tempo total de processamento. Para isso, a solução foi estruturada em três componentes principais da AWS: armazenamento escalável de dados (Amazon S3), orquestração de inferência para LLMs (Amazon Bedrock) e processamento acelerado por GPU (Amazon EC2).

3.1. Armazenamento e Gerenciamento de Dados com Amazon S3

A condução dos experimentos exigiu o gerenciamento eficiente de grandes volumes de dados textuais (incluindo os conjuntos OHSUMED, ACM, TWITTER e REUTERS), além dos artefatos gerados durante o treinamento, como índices de recuperação e *checkpoints* de modelos treinados. Para suportar esse fluxo de dados, o Amazon S3 foi adotado como *data lake* centralizado do projeto.

Todo o processo de sincronização de dados, desde o envio das bases de treinamento até o armazenamento dos pesos finais dos modelos (*model weights*), foi realizado por meio da interface de linha de comando da AWS (AWS CLI). Essa estratégia permitiu automatizar a transferência de dados entre os *buckets* do S3 e os volumes Amazon

EBS anexados às instâncias EC2. Como resultado, foi possível minimizar gargalos de entrada/saída (I/O) e reduzir períodos de ociosidade das GPUs durante operações de leitura e escrita de dados.

3.2. Orquestração de LLMs via Amazon Bedrock

Um dos principais gargalos computacionais da abordagem proposta está na geração em larga escala dos *RAG-labels*. Esse processo envolve múltiplas chamadas de inferência a LLMs de bilhões de parâmetros ao longo do ciclo iterativo de *prompt optimization*. A execução dessa etapa em infraestrutura autogerenciada implicaria custos elevados de provisionamento e manutenção de recursos computacionais.

Para contornar essa limitação, adotamos o Amazon Bedrock como serviço gerenciado para acesso a modelos fundacionais, especificamente o Llama 3.1 8B. A integração foi implementada em Python utilizando o SDK `boto3`. Por meio de chamadas programáticas à API do Bedrock, o sistema enviava o contexto recuperado dos documentos de treinamento juntamente com instruções estruturadas, recebendo como resposta as descrições enriquecidas das classes.

Essa arquitetura *serverless* eliminou completamente a necessidade de provisionar instâncias dedicadas apenas à inferência de LLMs, reduzindo significativamente a complexidade operacional da infraestrutura. Além disso, permitiu escalar a geração de *RAG-labels* sob demanda, mantendo baixa latência e alta disponibilidade.

3.3. *Fine-Tuning* com Amazon EC2

Com os dados centralizados no S3 e os *RAG-labels* gerados via Bedrock, as etapas de treinamento foram executadas em instâncias Amazon EC2 com suporte a GPU. Para maximizar a eficiência computacional, as cargas de trabalho foram divididas em duas frentes distintas, dimensionadas de acordo com os requisitos de memória e paralelismo de cada modelo.

- **Instância p3.2xlarge (Treinamento do RAG-Fuse):** O treinamento dos recuperadores densos baseados em SLMs, como BERT e RoBERTa, utilizando aprendizado contrastivo, demanda capacidade computacional significativa, porém com requisitos moderados de VRAM em comparação ao treinamento de LLMs. Para o *pipeline* do RAG-Fuse, uma instância `p3.2xlarge` (equipada com uma GPU NVIDIA Tesla V100 de 16 GB) apresentou o melhor equilíbrio entre custo e desempenho. Essa configuração foi suficiente para executar as etapas de treinamento dos encoders densos, construção de índices vetoriais e fusão de *rankings*.
- **Instância p3.8xlarge (*Fine-Tuning* dos Baselines LLMs):** Para garantir uma comparação rigorosa com abordagens baseadas em LLMs, realizamos o *fine-tuning* de modelos *open-source* utilizados como *baselines*. Devido ao tamanho desses modelos e à necessidade de aplicar técnicas de *Parameter-Efficient Fine-Tuning* (PEFT), como QLoRA, a carga computacional foi migrada para uma instância `p3.8xlarge`. Essa instância possui uma topologia multi-GPU com quatro GPUs NVIDIA Tesla V100, permitindo paralelizar o treinamento e reduzir significativamente o tempo necessário para executar as 10 partições da validação cruzada.

Essa arquitetura híbrida, que combina serviços gerenciados para inferência de LLMs e instâncias especializadas para treinamento de modelos, permitiu explorar de forma eficiente os recursos da nuvem. Como resultado, tornou-se possível executar experimentos de grande escala de forma reprodutível, mantendo controle sobre custos computacionais e tempo total de execução.

4. Metodologia Experimental

Para avaliar a efetividade do **RAG-Fuse** conduzimos uma avaliação experimental abrangente utilizando quatro conjuntos de dados: (i) *OHSUMED*, composto por documentos médicos; (ii) *ACM*, contendo artigos científicos da área de Ciência da Computação; (iii) *TWITTER*, formado por textos curtos oriundos de redes sociais; e (iv) *REUTERS*, constituído por artigos jornalísticos.

Comparamos o **RAG-Fuse** com métodos de estado da arte para CMCT. Utilizamos métodos clássicos de aprendizado de máquina (ML), tais como *Support Vector Machines* (SVM), *Random Forests* (RF) e *Logistic Regression* (LR). Entre os modelos baseados em SLMs, utilizamos *RoBERTa*, *BERT*, *BART* e *XLNet*. Também incluímos abordagens baseadas em LLMs, como *BloomZ*, *LLaMA-2*, *LLaMA-3.1*, *Mistral* e *DeepSeek*.

Os resultados são reportados por meio de validação cruzada estratificada com 10 partições. A efetividade dos métodos é medida pela métrica *Macro-F1*. Para verificar a significância estatística das diferenças observadas entre os métodos, aplicamos o teste t pareado com nível de confiança de 95%, utilizando correção de Bonferroni para controlar o erro do tipo I em múltiplas comparações [França et al. 2025].

Todos os experimentos foram executados na infraestrutura descrita na Seção 3. A fim de garantir a reprodutibilidade e transparência científica, todo o código-fonte, modelos treinados e scripts experimentais foram disponibilizados publicamente em: `https://github.com/celsofranssa/RAG-Fuse`.

5. Resultados Experimentais

Os resultados de efetividade estão resumidos na Tabela 1. O **RAG-Fuse** apresenta os melhores resultados absolutos de Macro-F1 em **todos** os datasets.

O **RAG-Fuse** empata estatisticamente com as duas versões do *LLama* e com o *Mistral* na OHSUMED e com o *LLama-3.1* e *Mistral* na ACM. No Twitter, o único método competitivo com o **RAG-Fuse** é o *BloomZ*. Finalmente na REUTERS, onde a tarefa é particularmente desafiadora devido ao elevado número de classes e desbalanceamento, o **RAG-Fuse** se destaca, sendo estatisticamente superior a todos os outros métodos sendo pelo menos **37%** melhor que o segundo melhor método (*MatchXML*) nesse dataset e **70%** melhor que as LLMs e SLMs mais efetivas.

A Figura 2 ilustra a relação entre eficiência computacional (medida pelo inverso do tempo de treinamento e inferência) e efetividade (Macro-F1) dos modelos avaliados. O **RAG-Fuse** destaca-se consistentemente no quadrante superior direito do gráfico, evidenciando uma combinação de alta efetividade e boa eficiência. Modelos tradicionais de *ML* como *SVM*, *LR* e *RF*, embora mais eficientes (posicionados à direita), apresentam desempenho significativamente inferior em Macro-F1. Abordagens com *LLMs*, como *LLaMa-3.1*, *Mistral* e *DeepSeek*, atingem alta efetividade, mas com elevado custo

Tabela 1. Efetividade: Macro-F1 e Intervalo de Confiança (IC) de cada classificador. Valores em negrito indicam empates estatísticos (teste t com correção de Bonferroni) por conjunto de dados.

		DATASET			
	MODEL	**OHSUMED**	**ACM**	**TWITTER**	**REUTERS**
ML	**SVM**	72.9(1.5)	67.7(1.5)	64.0(1.1)	33.2(2.3)
	LR	72.0(1.4)	66.7(1.3)	63.1(1.1)	41.5(2.6)
	RF	56.7(1.2)	60.1(1.2)	45.5(1.3)	27.0(1.6)
SLMs	**RoBERTa**	77.8(1.2)	70.3(1.4)	78.4(1.8)	41.9(2.2)
	BERT	76.4(1.2)	71.8(1.0)	64.5(1.9)	40.2(2.8)
	BART	77.6(0.7)	70.8(0.7)	79.0(2.1)	42.2(2.1)
	XLNet	77.6(1.0)	69.9(0.9)	76.4(2.1)	41.3(2.6)
LLMs	**BloomZ**	81.5(1.0)	72.9(1.7)	80.0(1.8)	40.7(2.2)
	LLaMa-2	**82.2(0.9)**	74.9(1.9)	78.8(1.9)	41.8(2.5)
	LLaMa-3.1	**83.1(1.1)**	**77.8(0.9)**	78.6(1.6)	41.5(2.6)
	Mistral	**83.1(0.8)**	**76.3(1.4)**	79.3(2.4)	41.5(2.4)
	DeepSeek	82.0(0.9)	75.2(1.3)	78.4(1.8)	41.7(2.7)
IR CB	**MatchXML**	74.5(4.2)	66.8(2.7)	75.5(3.3)	52.2(2.4)
IR	**RAG-Fuse**	**83.5(0.4)**	**76.8(1.1)**	**84.7(1.1)**	**71.6(2.8)**

computacional (posicionadas à esquerda). O **RAG-Fuse** apresenta a melhor relação eficácia-eficiência: é 8 vezes em média mais rápido que os *LLMs*, mantendo ou superando seu desempenho. Em alguns cenários (e.g., ACM e REUTERS), é mais de 10 vezes mais eficiente, mantendo a melhor efetividade entre todos os métodos testados.

6. Conclusão e Discussão

Neste relato de experiência, apresentamos o **RAG-Fuse**, uma abordagem para CMCT que reformula o problema sob a perspectiva de Recuperação de Informação (RI). Além da contribuição algorítmica, demonstramos como uma arquitetura baseada em computação em nuvem torna viável a execução desse *pipeline* em escala. Em particular, a orquestração entre o Amazon Bedrock, responsável pela geração semântica dos *RAG-labels* por meio de LLMs, e o Amazon EC2, utilizado para o treinamento e a fusão de recuperadores esparsos e densos, mostrou-se fundamental para atender às demandas computacionais do método.

Os experimentos conduzidos em quatro *benchmarks* públicos evidenciam ganhos substanciais de desempenho, alcançando melhorias de até **37%** em Macro-F1 em relação ao segundo melhor método avaliado, e de até **70%** quando comparado a LLMs executados localmente em cenários com forte desbalanceamento de classes. Além disso, a infraestrutura baseada na AWS permitiu reduzir significativamente o tempo total de execução dos experimentos e o custo computacional associado, viabilizando a execução paralela de múltiplos experimentos e acelerando o ciclo de experimentação científica.

6.1. Discussão e Lições Aprendidas

O uso de instâncias EC2 (`p3.2xlarge` e `p3.8xlarge`) dedicadas exclusivamente ao treinamento e ao processamento vetorial profundo, combinado com a delegação da geração e otimização de *prompts* ao serviço gerenciado Amazon Bedrock, permitiu evitar a ociosidade de recursos computacionais de alto custo. Paralelamente, o Amazon S3

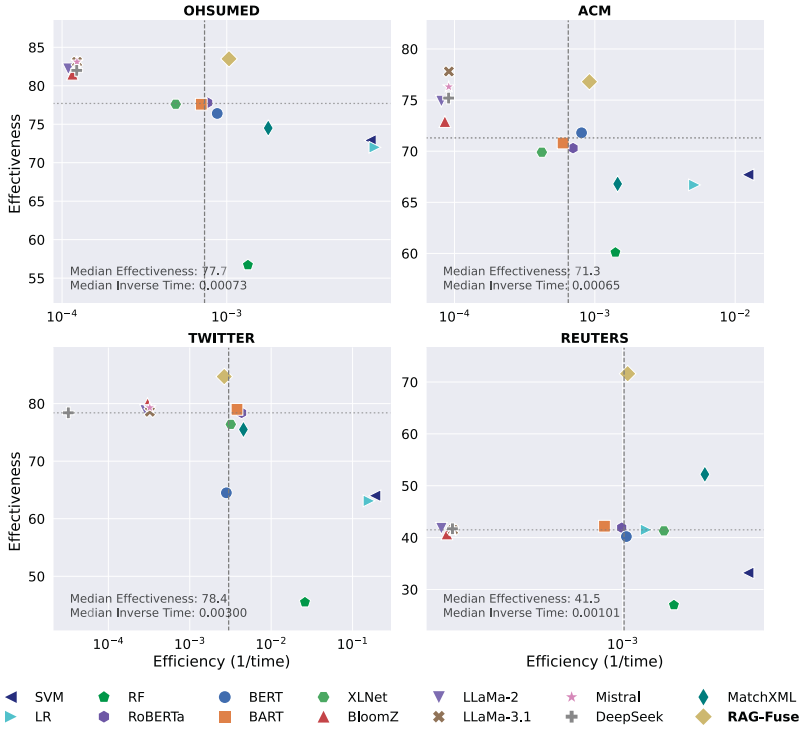

Figura 2. Relação entre eficiência (inverso do tempo em segundos) e eficácia (Macro-F1). Modelos mais desejáveis estão posicionados no quadrante superior direito (alta eficácia e alta eficiência) [França et al. 2025]

atuou como camada central de armazenamento de dados experimentais, reduzindo os gargalos de I/O na transferência dos artefatos do *pipeline*. Essas práticas demonstraram-se essenciais para garantir robustez, eficiência e estabilidade na execução de experimentos em larga escala.

6.2. Direções Futuras

Como direções futuras, pretendemos investigar estratégias de atualização dinâmica dos *RAG-labels* em domínios altamente dinâmicos, bem como explorar mecanismos de fusão adaptativa baseados na confiança dos recuperadores. Além disso, planejamos estender a arquitetura proposta para cenários de maior complexidade estrutural, como *Extreme Multi-label Text Classification* (XMTC) e *Hierarchical Text Classification* (HTC). Esses cenários apresentam desafios adicionais relacionados à escala do espaço de rótulos e à estrutura hierárquica das classes, reforçando a importância de infraestruturas elásticas de computação em nuvem para suportar a experimentação e o desenvolvimento em larga escala.

Agradecimentos

Este trabalho foi apoiado por: CNPq, CAPES, Instituto Nacional de Ciência e Tecnologia em Inteligência Artificial Responsável para Linguística Computacional, Tratamento e Disseminação de Informação (INCT-TILD-IAR), FAPEMIG, AWS, Google, NVIDIA, CIIASaúde e FAPESP.

Declaração sobre uso de Inteligência Artificial

Ferramentas de Inteligência Artificial generativa foram utilizadas apenas como apoio à revisão gramatical e à melhoria da clareza textual de trechos do manuscrito. Todo o conteúdo técnico, experimental e científico foi produzido e validado pelos autores.

Referências

Bassani, E. (2023). ranxhub: An online repository for information retrieval runs. In *SIGIR*, page 3210–3214.

Chen, Y. et al. (2024). PRompt optimization in multi-step tasks (PROMST): Integrating human feedback and heuristic-based sampling. In *EMNLP*, pages 3859–3920.

de Andrade, C. M. et al. (2023). On the class separability of contextual embeddings representations – or "the classifier does not matter when the (text) representation is so good!". *Information Processing Management*, 60(4).

Dettmers, T. et al. (2023). Qlora: Efficient finetuning of quantized llms. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115.

França, C., Nunes, I., Salles, T., Cunha, W., Jallais, G., Rocha, L., and Gonçalves, M. A. (2025). Muitas classes desbalanceadas? não classifique-ranqueie! uma abordagem baseada em retrieval-augmented generation (rag)-labels para classificação textual multiclasse. In *Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 264–277. SBC.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2020a). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th ACL*, pages 7871–7880, Online.

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. (2020b). Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, pages 9459–9474.

Sikosana, M., Ajao, O., and Maudsley-Barton, S. (2024). A comparative study of hybrid models in health misinformation text classification. OASIS '24, page 18–25.

Sy, C. Y., Maceda, L. L., Canon, M. J. P., and Flores, N. M. (2024). Beyond bert: Exploring the efficacy of roberta and albert in supervised multiclass text classification. *International Journal of Advanced Computer Science & Applications*, 15(3).