Muitas Classes Desbalanceadas? Não Classifique - Ranqueie! Uma Abordagem Baseada em *Retrieval-Augmented Generation (RAG)-labels* para Classificação Textual Multi-classe

Celso França[1], Ian Nunes[2], Thiago Salles[1], Washington Cunha[1], Gabriel Jallais[1] Leonardo Rocha[2] and Marcos André Gonçalves[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG) Belo Horizonte – MG – Brasil.

{celsofranca,tsalles,washingtoncunha,gabrieljallais,mgoncalv}@dcc.ufmg.br

[2]Departamento de Ciência da Computação – Universidade Federal de São João del-Rei (UFSJ) São João del-Rei – MG – Brasil.

ianqrbn06@aluno.ufsj.edu.br,lcrocha@ufsj.edu.br

Resumo. *Este artigo propõe uma abordagem para classificação multi-classe de textos em cenários semanticamente complexos e severo desbalanceamento, reformulando-a como uma tarefa de ranqueamento. A técnica explora a capacidade dos Grandes Modelos de Linguagem (LLMs) de representar semanticamente classes por meio de* ***RAG-labels****–descrições enriquecidas via Retrieval-Augmented Generation. Essa reformulação reduz o gap léxico-semântico entre textos e classes, permitindo aplicar a fusão de ranqueadores densos e esparsos. Nossa solução apresenta ganhos de até* ***70%*** *em MacroF1 frente a LLMs de última geração (e.g., LLama e DeepSeek), com redução de custo de até* ***10*** *vezes.*

Abstract. *This paper proposes an approach for multi-class text classification in scenarios of high semantic complexity and severe class imbalance, reformulating the task as a ranking problem. The technique exploits the ability of Large Language Models (LLMs) to semantically represent classes through* ***RAG-labels****–enriched descriptions generated via Retrieval-Augmented Generation. This reformulation reduces the lexical-semantic gap between texts and classes, allowing the fusion of dense and sparse rankers. Our solution presents gains of up to* ***70%*** *in MacroF1 compared to state-of-the-art LLMs (e.g., LLama and DeepSeek), with an average cost reduction of up to* ***10*** *times.*

1. Introdução

A Classificação Multi-Classe de Texto (CMCT) é um paradigma essencial de aprendizado de máquina (*ML*) para organizar e recuperar informação em grandes volumes de dados textuais. O objetivo é atribuir categorias semânticas predefinidas a documentos, com aplicações em domínios variados tais como negócios, saúde e educação [Sikosana et al. 2024]. Métodos tradicionais de *ML*, como *Support Vector Machines* e *Random Forests*, são eficientes e interpretáveis, mas limitados na captura de relações semânticas. Recentemente, propostas de *Small Language Models* (SLMs), como RoBERTa [Sy et al. 2024] e BART [Lewis et al. 2020a], buscam explorar representações densas e contextuais. Os *Large Language Models* (LLMs) evoluíram a partir dos SLMs,

com arquiteturas mais avançadas e pré-treinamento em larga escala. Evidências apontam esses modelos como estado-da-arte para várias tarefas de Processamento de Linguagem Natural (PLN), incluindo CMCT [Cunha et al. 2025].

Independentemente da abordagem, a CMCT enfrenta dois desafios centrais. O primeiro é o ***gap* léxico-semântico**, manifestado como (i) *vocabulary mismatch*, quando termos dos documentos não coincidem com os das classes–por exemplo, um documento sobre "desconto previdenciário"associado à classe "INSS", mesmo sem a sigla; e (ii) *ambiguidade semântica*, quando um termo que representa a classe pode indicar múltiplos conceitos no mundo real–por exemplo, "banco", que pode se referir a uma instituição financeira, um banco de dados ou um móvel, dependendo do contexto. Ambos exigem compreensão contextual além da pura correspondência (ou co-ocorrência) de palavras.

O segundo desafio é o **desbalanceamento** de classes em que poucas classes majoritárias concentram a maioria dos documentos, enquanto muitas classes minoritárias têm poucas instâncias, dificultando a generalização, causando vieses e agravando o impacto do *gap* léxico-semântico. SLMs e LLMs, mitigam indiretamente esses desafios ao explorar correlações semânticas entre rótulos e termos dos documentos. Embora mostrem bom desempenho em cenários *zero-shot*, o *fine-tuning* continua essencial para alcançar alta efetividade em CMCT [de Andrade et al. 2023], ainda que com custo computacional elevado, exigindo grandes volumes de dados rotulados, infraestrutura robusta e alto consumo energético [Dettmers et al. 2023].

Assim, é necessário buscar alternativas que reduzam o *gap* léxico-semântico entre textos e classes–especialmente em cenários desbalanceados–maximizando a eficiência computacional e minimizando impactos ambientais, sem comprometer a efetividade. Nesse contexto, nossas soluções propõem estratégias baseadas em definições claras e compreensivas das classes, viabilizando o *casamento* entre descrições de classes e documentos via similaridade semântica e léxica. Especificamente, propomos **RAG-Fuse**, um pipeline em dois estágios de recuperação e fusão que reformula a CMCT como uma tarefa de Recuperação de Informação (RI). *Nossa hipótese central é que a CMCT pode ser mais efetiva e eficiente sob a ótica da RI–especialmente em cenários com muitas classes e desbalanceamento–ao tratar o texto como uma consulta e as classes como documentos a serem ranqueados, desde que o gap semântico entre consultas e documentos tenha sido reduzido*.

Diferentemente das abordagens de *ML*, *SLMs* e *LLMs*, que aprendem uma função de probabilidade *indireta* entre os labels das classes e os textos, o **RAG-Fuse** combina técnicas *diretas* de ranqueamento e fusão de *ranking* (Figura 1). Visando reduzir o *gap* semântico entre textos e classes, essencial à proposta, introduzimos os **RAG-labels** – descrições de classes geradas via *Retrieval Augmented Generation* (RAG) [Lewis et al. 2020b] e *otimização de prompt* [Chen et al. 2024]. Uma vez reduzido esse *gap*, aplicamos algoritmos de fusão de *ranking* [Bassani 2023], estratificados entre classes majoritárias e minoritárias, para combinar *rankings* de recuperadores densos — voltados à semântica — e esparsos — focados em aspectos léxicos e sintáticos — explorando sua complementariedade para mitigar o desbalanceamento.

Para avaliar a efetividade e a eficiência do **RAG-Fuse**, comparamos seu desempenho considerando cenários de desbalanceamento com amplos espaços de classes e doze

abordagens de CMCT como *baselines*: 3 abordagens baseadas em *ML*, 4 abordagens baseadas em *SLMs*, 5 abordagens baseadas em *LLMs open-source* e 1 abordagem baseada em clusterização de classes (*CB*), especialmente projetada para cenários com muitas classes. Especificamente, investigamos as seguintes questões de pesquisa:

QP1 – Qual a eficácia do **RAG-Fuse** na tarefa de CMCT em relação às abordagens baseadas em *ML*, *SLMs*, *LLMs* e *CB*?
QP2 – Como o custo computacional do **RAG-Fuse** se compara às abordagens descritas?
QP3 – Qual a magnitude do *gap* léxico-semântico entre textos e classes? O **RAG-Fuse** é capaz de reduzi-lo (e como)?

Nossos resultados experimentais mostram que as abordagens baseadas em *ML* apresentam o menor tempo total (treinamento + inferência), mas foram significativamente inferiores em termos de efetividade (*Macro-F1*), especialmente em cenários desbalanceados. Por outro lado, os SLMs e os LLMs alcançaram bons resultados em termos de efetividade, mas com um aumento considerável no custo computacional. Em contraste, o **RAG-Fuse** consistentemente se mostrou mais eficaz do que todas as abordagens baseadas em *ML* e *SLMs* em todos os *datasets*. Mais notavelmente, o **RAG-Fuse** também superou os *LLMs*, como **LLaMA 3.1** e **DeepSeek**, com ganhos de efetividade de até 70% sobre as as LLMs em alguns conjuntos de dados e com um tempo até 10 vezes menor, sendo a abordagem que oferece o melhor equilíbrio entre efetividade e eficiência. Aprofundando na análise dos resultados do **RAG-Fuse**, observamos que todos os componentes da solução, i.e., *RAG-labels* e as fusão de *rankings* esparsos e densos são essenciais para a efetividade da mesma, tornando-a robusta e adaptável à heterogeneidade dos cenários de CMCT.

Sumarizando, as principais contribuições deste artigo são: (i) introdução dos **RAG-labels** como estratégia para redução do *gap* léxico-semântico entre textos e classes; (ii) proposta de uma **abordagem dual de recuperação e fusão** para CMCT que ajuda a tratar, além do gap semântico, o desbalanceamento de classes; e (iii) uma ampla **validação experimental** com vários conjuntos de dados distintos e classificadores do estado-da-arte, incluindo LLMs de última geração, sob as óticas de efetividade e eficiência.

2. Revisão da Literatura

2.1. Abordagens baseadas em Machine Learning (ML)

Métodos clássicos de CMTC, como *SVM* [Sun et al. 2009], *Logistic Regression* [Cunha et al. 2023] e *Random Forest* [Zhou et al. 2016], usam representações esparsas como *TF-IDF*, sendo valorizados por sua simplicidade e baixo custo. Apesar de eficazes em cenários menos complexos, essas abordagens dependem fortemente de engenharia de atributos e não capturam contexto semântico [Zhou et al. 2016].

2.2. Abordagens baseadas em Small Language Models (SLMs)

Small Language Models utilizam *Transformers* e mecanismos de atenção para modelar contexto. Embora superem abordagens tradicionais em CMTC, apresentam custo computacional mais elevado. Utilizamos os modelos com melhor desempenho (Macro-F1) conforme análise em [Cunha et al. 2023]: *RoBERTa* [Sy et al. 2024], *BERT* [Devlin et al. 2019], *BART* [Lewis et al. 2020a] e *XLNet* [Yang et al. 2019].

2.3. Abordagens baseadas em Large Language Models (LLMs)

Large Language Models (LLMs), como *LLaMA* [Touvron et al. 2023] e *DeepSeek* [Guo et al. 2025], operam com bilhões de parâmetros e podem ser adaptados para múltiplas tarefas de PLN via *in-context learning* ou *fine-tuning*. Embora poderosos, exigem alto custo computacional. Para CMTC, utilizamos versões ajustadas de LLMs open-source com desempenho robusto [de Andrade et al. 2024]: *DeepSeek*, *LLaMA*, *Mistral* [Jiang 2024] e *BloomZ* [Muennighoff et al. 2022].

2.4. Abordagens baseadas em Clusterização de Grandes Conjuntos Classes (CB)

Modelos como *Match-XML* [Ye et al. 2024], *LightXML* [Jiang et al. 2021], *AttentionXML* [You et al. 2019] e *XR-Transformer* [Zhang et al. 2021] seguem um processo hierárquico de agrupamento, associação e atribuição de scores às classes, sendo altamente eficazes para problemas com muitas classes. Contudo, sua dependência de agrupamentos fixos limita a aplicabilidade em cenários de CMTC, especialmente aqueles com alto desbalanceamento. Em contraste, o **RAG-Fuse** segmenta dinamicamente o espaço de classes para cada texto, promovendo adaptabilidade semântica. Utilizamos o recentemente proposto *Match-XML* [Ye et al. 2024] como baseline em nossos experimentos.

3. Abordagem Proposta

Assumindo um conjunto de documentos textuais $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ e um espaço de classes $\mathcal{C} = \{c_1, c_2, \ldots, c_L\}$, a CMTC busca aprender uma função $f(d, c) : |\mathcal{D}| \times |\mathcal{C}| \rightarrow \mathbb{R}$ que define um *grau de associação* (ou relevância) entre cada classe c e um documento d. Abordagens tradicionais de *ML*, *SLMs* e *LLMs* aproximam essa função ajustando **indiretamente** os parâmetros que maximizam a probabilidade condicional $P(c_1, c_2, \ldots, c_L|e_d)$, onde e_d é uma representação vetorial (esparsa ou densa) do texto em d.

Em contraste, reformulamos CMTC como uma tarefa de RI, mapeando d para uma consulta de busca e tratando cada classe c como um "documento". Dessa forma, a função f é aproximada por uma função de ranqueamento que mede **diretamente** a similaridade léxico-semântica entre o texto (consulta) e as classes candidatas, permitindo recuperar as classes mais relevantes para cada documento. A partir dessa reformulação, propomos o **RAG-Fuse**, um pipeline complementar de recuperação e fusão em dois estágios, projetado para lidar com os desafios do *gap* léxico-semântico e desbalanceamento.

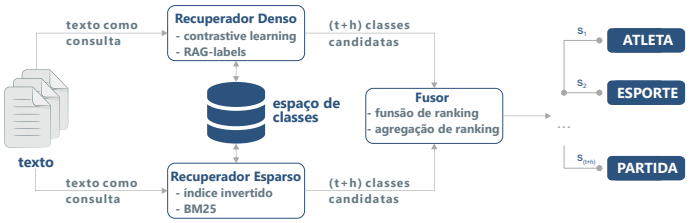

Figura 1. RAG-Fuse: *Pipeline* de Recuperação e Fusão

No primeiro estágio, nossa abordagem realiza uma divisão dinâmica do espaço de classes *para cada documento*, utilizando recuperadores esparsos e densos de forma complementar para gerar conjuntos candidatos de classes relevantes. Conforme Figura 1, o **recuperador esparso** calcula o escore de relevância de um par $(texto, classe)$ com base em representações esparsas e de alta dimensionalidade, baseadas em *bag-of-words*. Já o **recuperador denso** utiliza um algoritmo de vizinho mais próximo aproximado (ANN)

sobre representações densas de textos e classes, aprendidas a partir de *transformers* (SLMs) ajustados e armazenadas em um espaço de *embedding* compartilhado. Para tratar do **desbalanceamento** entre as classes, os recuperadores esparso e denso segmentam o espaço de classes em dois subconjuntos: (i) subconjunto $\mathcal{H} \subset \mathcal{C}$ das classes majoritárias e; (ii) o subconjunto $\mathcal{T} \subset \mathcal{C}$ de classes minoritárias. Isso é necessário porque, assim como na maioria das estratégias de CMTC, os ranqueadores tendem a apresentar um viés para classes majoritárias, dado que as mesmas têm um maior número de documentos que podem ser "casados"com a descrição da classe.

No segundo estágio, aplicamos uma técnica de fusão de *rankings* que combina os resultados dos recuperadores *esparsos* e *densos*. Partimos da premissa de que uma única abordagem de recuperação pode não contornar o ***gap* léxico-semântico**, falhando em recuperar todas as classes relevantes às consultas textuais [Arabzadeh et al. 2021]. Embora recuperadores densos geralmente superem os esparsos em eficácia [Penha and Hauff 2023], eles têm dificuldades em capturar correspondências léxicas devido à dependência de representações semânticas [Lin et al. 2023]. Essa limitação decorre da incompatibilidade entre a tarefa de ranqueamento e os objetivos de pré-treinamento dos *transformers*, que priorizam o entendimento semântico [Llordes et al. 2023]. Embora o ajuste fino (*fine-tuning*) mitigue o problema, ele persiste [Lin et al. 2023]. Recuperadores esparsos podem superar os densos em cenários específicos [Lin et al. 2023] ou atuar como mecanismos complementares, recuperando classes ignoradas pelos densos. Experimentos de ablação (Seção 5.3) demonstram que ambos ranqueadores são complementares e essenciais para nossa estratégia.

Nosso pipeline mitiga eficazmente as limitações de cada recuperador, enquanto explora suas vantagens, combinando a capacidade de correspondência exata do recuperador esparso com os recursos semânticos do recuperador denso. Por fim, nosso pipeline atribui um escore de relevância mais refinado para os pares $(texto, classe)$ na etapa de fusão, combinando os *rankings* gerados pelos recuperadores denso e esparso–cada um deles composto por t classes minoritárias e h classes majoritárias, como mostrado na Figura 1. Em seguida, os *rankings* fundidos–segmentados nos subconjuntos de classes $\mathcal{T}$ e $\mathcal{H}$–são agregados para compor o *ranking* final das classes (Seção3.3).

3.1. RAG-Labels

O *gap* léxico-semântico ocorre entre os textos dos documentos, geralmente longos, e as classes, compostas por um ou dois termos frequentemente ambíguos. Essa ambiguidade exige análise abrangente do texto para correta interpretação [Wang et al. 2023]. Para mitigar o problema, o **RAG-Fuse** introduz os **RAG-labels** – descrições das classes geradas por *LLMs* com base no contexto recuperado. Para maximizar a qualidade da geração dos RAG-labels, propomos uma abordagem em duas etapas: (i) otimizar o *prompt* de descrição utilizando o próprio *LLM*; e (ii) preencher o *prompt* otimizado com contextos relevantes via *Retrieval-Augmented Generation* (RAG).

3.1.1. Otimização de Prompt para Descrição de Classes

A eficácia dos *LLMs* depende fortemente da qualidade dos *prompts* utilizados [Chen et al. 2024]. Assim, propomos um processo de otimização com o objetivo de alinhar as capacidades dos LLMs à tarefa específica de descrição de classes, por meio do refinamento iterativo do *prompt*. Nossa abordagem consiste em estruturar o processo

de otimização em linguagem natural, utilizando um $meta_{prompt}$ que incorpora *prompts* específicos previamente gerados para a tarefa e suas respectivas avaliações, conforme detalhado no Algoritmo 1. Os parâmetros necessários incluem um $task_{prompt}$, um $meta_{prompt}$, um LLM e um conjunto de $samples$. O $task_{prompt}$ é a instrução que orienta o LLM na geração de descrições contextualizadas das classes. Já o $meta_{prompt}$ contém instruções que guiam o LLM no processo de refinamento do $task_{prompt}$. O $meta_{prompt}$ é conceitualmente análogo ao processo de *stochastic gradient descent* (SGD) utilizado na otimização de redes neurais profundas, mas adaptado para o contexto de otimização de *prompts*. Assim como o SGD atualiza iterativamente os pesos do modelo, o $meta_{prompt}$ refina iterativamente o $task_{prompt}$ incorporando métricas de acurácia obtidas nas gerações anteriores e utilizando amostras ($samples$) para guiar a otimização. Utilizamos o *Llama 3.1 8B* [Dubey et al. 2024] como LLM, atuando em dois papéis: otimizador de prompts ao responder ao $meta_{prompt}$ e gerador de descrições de classes ao responder ao $task_{prompt}$.

Algoritmo 1: Otimizador de Prompt LLM

Parameters: $LLM, samples, task_{prompt}, meta_{prompt}, epochs$
1 **for** $epoch \in epochs$ **do**
2 $sims \leftarrow [\ ]$
3 **for** $sample \in samples$ **do**
4 $target\_label \leftarrow sample.target\_label$
5 $target\_desc \leftarrow sample.target\_description$
6 $pred\_desc = generate(target\_label, task_{prompt}, LLM)$
7 $sims \leftarrow sims \cup \{cosine\_sim(pred\_desc, target\_desc)\}$
8 **end**
9 $task_{prompt} \leftarrow update(\overline{sims}, task_{prompt}, meta_{prompt}, LLM)$
10 **end**
11 **return** $task_{prompt}$

Para cada conjunto de dados de CMCT propomos a utilização de um subconjunto pequeno e uniformemente selecionado (máximo 256 amostras), visando manter a eficiência. Cada amostra é composta por trios $(context, target\_label, target\_description)$, onde o *context* representa um conjunto de textos associados à $target\_label$, e a $target\_description$ foi gerada pelo *GPT-4* com base no *contexto*. O uso do *GPT-4* como provedor externo de descrições de classes evita uma otimização trivial, que ocorreria caso o mesmo LLM fosse utilizado tanto para gerar a $target\_description$ quanto para responder ao $task_{prompt}$. Especificamente, a cada etapa do processo de otimização, o LLM prediz uma descrição de classe $pred_{desc}$ utilizando uma $target\_label$ e o $task_{prompt}$ atual, que contém o contexto correspondente àquela $target\_label$ (linha 6). Em seguida, calculamos a similaridade de cosseno entre a descrição predita e a descrição alvo (linha 7). Ao final de cada época (linha 9), o $task_{prompt}$ é atualizado com base no $meta_{prompt}$, nas similaridades médias $\overline{sims}$ e utilizando o LLM como otimizador.

3.1.2. Retrieval-Augmented Generation (RAG)

Nossa proposta inclui uma abordagem *RAG* na qual garantimos que as descrições de classes geradas estejam contextualizadas nos textos associados às respectivas classes. Dessa forma, recuperamos os textos associados a cada classe-alvo como contexto no $task_{prompt}$ otimizado. Por fim, utilizamos o modelo *Llama 3.1 8B da Meta* e o $task_{prompt}$ otimizado, enriquecido com o contexto relevante, para predizer a descrição da classe.

3.2. Etapa de Recuperação

Na primeira etapa do pipeline, os *recuperadores* esparsos e densos particionam o espaço de classes ao selecionar, para cada texto, o subconjunto mais promissor de classes. Para

lidar com o *desbalanceamento*, os recuperadores devem atender a dois requisitos: (i) efetividade na identificação de classes candidatas promissoras e (ii) eficiência diante de um espaço de classes desbalanceado. A recuperação realiza um particionamento dinâmico das classes, levando em conta as nuances individuais de cada texto. Essa flexibilidade é essencial para reduzir perdas de efetividade, em especial nas classes minoritárias.

3.2.1. Recuperador Esparso Léxico

O recuperador esparso primeiro constrói um índice de busca utilizando *textos com classes conhecidas*, conforme ilustrado na Figura 2(a). Dado um texto t como consulta, recuperamos do índice outros textos similares a t e retornamos suas classes como candidatas. Em seguida, atribuímos a cada classe candidata uma pontuação igual ao maior valor de relevância entre t e cada texto recuperado (uma classe pode estar associada a múltiplos textos recuperados). Por fim, selecionamos as k classes mais bem ranqueadas como resposta. Visando a efetividade, empregamos a popular função de ranqueamento Best Match 25 (**BM25**) [Askari et al. 2023], amplamente utilizada em motores de busca modernos [Askari et al. 2023]. O BM25 avalia a relevância considerando a frequência dos termos e o comprimento do texto, utilizando uma função de saturação para normalizar a frequência dos termos. O recuperador esparso garante eficiência ao utilizar um *índice invertido* [Askari et al. 2023] para armazenar estatísticas de termos pré-computadas, reduzindo significativamente o tempo necessário para calcular as pontuações de relevância.

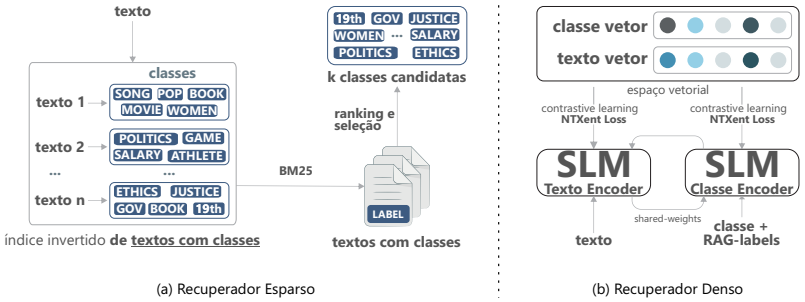

Figura 2. (a) o recuperador esparso explora um índice invertido de textos rotulados e BM25 para recuperar classes candidatas. (b) o recuperador denso utiliza uma arquitetura *bi-encoder*, codificando separadamente os textos e classes em um espaço vetorial compartilhado.

3.2.2. Recuperador Denso Semântico

O recuperador denso–Figura 2(b)–visa maximizar a efetividade via *fine-tuning* de um SLM, projetando textos e classes em um espaço vetorial compartilhado, onde vetores de classes relevantes se aproximam do vetor do texto. A arquitetura processa separadamente o texto e a classe enriquecida (classe + **RAG-labels**) com um *SLM*, extraindo representações pela concatenação dos quatro últimos estados ocultos do token [CLS] [Devlin et al. 2019]. Utiliza-se a *Normalized Temperature-Scaled Cross Entropy Loss* [Liu et al. 2023], uma função de perda contrastiva que aproxima pares semanticamente similares e afasta os dissimilares. Para eficiência, o recuperador adota uma arquitetura *bi-encoder* (*encoders* independentes para texto e classe), o que permite desacoplar indexação e busca. Durante a indexação, as classes são representadas vetorialmente; na busca, recuperam-se as k classes mais relevantes por meio de *Approximate Nearest Neighbors*, usando o algoritmo HNSW [Bruch et al. 2023]. Com essa ar-

quitetura, o **RAG-Fuse** atinge complexidade logarítmica em relação ao tamanho do índice [França and others. 2025]. O recuperador mitiga o *gap* léxico-semântico e o *desbalanceamento*, realizando interações granulares via atenção, o que refina a representação das classes e aprimora a capacidade do **RAG-Fuse** de avaliar sua relevância com texto.

3.3. Estágio de Fusão

Os *rankings* gerados pelos dois recuperadores esparsos e densos são fundidos gerando uma ordenação final que abrange tanto classes minoritárias $(c_m \in \mathcal{T})$ quanto majoritárias $(c_M \in \mathcal{H})$. Nossa proposta é apresentada no Algoritmo 2 (*Fuse and Aggregate*) que processa os *rankings* densos e esparsos, cada um estratificados em tipos de classe $c \in \mathcal{T} \cup \mathcal{H}$. Durante a etapa de normalização (linhas 3–6), o algoritmo aplica a normalização dos *rankings* de forma independente para cada classe c. Na etapa de fusão (linhas 8–10), um método de fusão baseado em ranking combina os *rankings* normalizados de cada classe c [Bassani 2023], gerando *rankings* intermediários fundidos. Por fim, na etapa de agregação (linha 12), o algoritmo consolida os *rankings* disjuntos de $\mathcal{T}$ e $\mathcal{H}$ em um único ranking final de classes para cada texto, agregando seus escores fundidos.

Algoritmo 2: Fuse and Aggregate

```
   Parameters: dense, sparse rankings
1  norm_dense, norm_sparse, ranking ← {}, {}, {}
   /* Normalization step: normalize dense and sparse rankings */
3  for c ∈ T ∪ H do
4  |  norm_dense[c] ← normalize(dense[c])
5  |  norm_sparse[c] ← normalize(sparse[c])
6  end
   /* Fusion step: fuse normalized T and H rankings */
8  for c ∈ T ∪ H do
9  |  ranking[c] ← fuse(norm_dense[c], norm_sparse[c])
10 end
   /* Aggregation step: aggregate T and H rankings */
12 return aggregate(ranking[T], ranking[H])
```

4. Metodologia Experimental

Para avaliar a efetividade do **RAG-Fuse** sob a perspectiva dos desafios da CMTC, conduzimos experimentos extensivos utilizando quatro conjuntos de dados amplamente adotados e representativos na literatura [Cunha et al. 2024, Cunha et al. 2025]: (i) *OHSUMED* (documentos médicos); (ii) *ACM* (artigos científicos em Ciência da Computação), (iii) *TWITTER* (conteúdo de redes sociais); e (iv) *REUTERS* (artigos de notícias). A Tabela 1 resume as principais características desses conjuntos, destacando sua diversidade em termos de escala e estrutura de classes.

Tabela 1. Estatísticas dos Conjuntos de Dados. Número de textos (N), número de classes (C), densidade (palavras por documento) e distribuição de classes.

DATASET	N	C	DENSIDADE	DISTRIBUIÇÃO
OHSUMED	18,302	23	154	Desbalanceado
ACM	24,897	11	65	Desbalanceado
TWITTER	6,997	6	28	Desbalanceado
REUTERS	13,327	90	171	Extremamente Desbalanceado

Comparamos o desempenho do **RAG-Fuse** com métodos estado-da-arte para CMTC (Seção 2): abordagens de ML (*SVM*, Random Forests (*RF*) e Logistic Regression (*LR*); abordagens baseadas em SLMs (*RoBERTa*, *BERT*, *BART* e *XLNet*); abordagens baseadas em LLMs (*BloomZ*, *LLaMa-2*, *LLaMa-3.1*, *Mistral* e *DeepSeek*); e abordagem

baseada CB (Match-XML). Esses modelos abrangem uma variedade de técnicas e arquiteturas, oferecendo uma base sólida de comparação.

Para as abordagens baseadas em *ML*, adotamos *TF-IDF* como representação textual devido à sua simplicidade e eficiência em CMTC. Alternativas como embeddings estáticos podem resultar em perdas de eficácia quando comparadas ao uso de *TF-IDF + SVM*, enquanto embeddings contextuais podem ser 1.5x a 31.1x mais lentos [Cunha et al. 2025]. Os hiperparâmetros explorados foram: SVM: parâmetro $C \in \{0.125, 0.5, 1, 2, 8, 32, 128, 512, 2048, 8192\}$; LR: $C \in \{0.001, 0.01, 0.1, 1\}$; RF: número de árvores geradas $\in \{50, 100, 200\}$ e $max_{feat} \in \{0.08, 0.15, 0.30\}$. Dado o elevado número de hiperparâmetros referentes aos modelos *SLMs* e *LLMs*, a realização de uma busca exaustiva via validação cruzada não foi viável. Para os *SLMs*, seguimos a metodologia proposta em [Cunha et al. 2023], fixando a taxa de aprendizado inicial em $5e^{-5}$, o número máximo de épocas em 20 e a paciência em 5 épocas. Além disso, realizamos uma busca nos parâmetros *max_len* ({128, 256}) e *batch_size* ({16, 32}), pois essas configurações impactam diretamente a eficiência e a eficácia dos modelos. Para os LLMs, adotamos a estratégia de quantização em 4 bits, combinada com *QLoRA* e *PEFT*, permitindo um ajuste fino eficiente em hardware local [de Andrade et al. 2024]. Fixamos a taxa de aprendizado inicial em $2e^{-4}$, o número de épocas em 4, o *batch_size* em 4 e *max_len* em 256.

Adotamos uma definição formal para particionar o conjunto de classes em subconjuntos disjuntos $\mathcal{H}$ e $\mathcal{T}$. Seja $\mathcal{D} = \{(d_i, c_i)\}_{i=1}^{N}$ um conjunto de dados de CMTC. Suponha que o conjunto de classes $\{c_1, \ldots, c_L\}$ esteja ordenado por frequência de ocorrência em ordem decrescente. Ao definirmos um limiar l, as l classes mais frequentes $\{c_1, \ldots, c_l\}$ formam o conjunto $\mathcal{H}$, enquanto as $L-l$ classes restantes $\{c_{l+1}, \ldots, c_L\}$ compõem o conjunto $\mathcal{T}$. O limiar l é determinado com base no princípio de Pareto, classificando como $\mathcal{T}$ as classes que compõem os 80% menos frequentes e como $\mathcal{H}$ as 20% mais frequentes.

Para o recuperador esparso, configuramos os parâmetros do BM25 como $b = 0{,}75$ e $k = 1{,}5$ [Askari et al. 2023]. Para o recuperador denso, realizamos o ajuste fino do SLM (BERT ou RoBERTa) para representar textos e classes como embeddings em um espaço vetorial compartilhado, utilizando a função de perda *Normalized Temperature-Scaled Cross Entropy* [Liu et al. 2023] como objetivo de aprendizagem. A taxa de aprendizado varia entre $5e^{-3}$ e $5e^{-5}$ por meio de uma política cíclica de taxa de aprendizado. Na etapa de fusão, utilizamos o algoritmo CombMNZ e a normalização ZMUV [Bassani 2023] em todos os conjuntos de dados, conforme os resultados de 60 experimentos envolvendo a combinação de 10 algoritmos de fusão por ranqueamento com 6 estratégias de normalização [França et al. 2025].

Reportamos os resultados obtidos por meio de uma validação cruzada de 10 partições. A efetividade é avaliada utilizando *Macro-F1*, devido ao desbalanceamento de classes. Para análise de significância estatística dos resultados, utilizamos o teste t pareado com 95% de confiança com correção de Bonferroni [França and others. 2025]. Avaliamos os tempos de execução dos algoritmos utilizando uma única máquina: um Intel Core i7-5820K (12 threads, 3.30GHz), 64GB de RAM e uma GPU NVIDIA RTX 3090 com 24GB de memória. Visando a reprodutibilidade, disponibilizamos todos os códigos, modelos e dados em: `https://github.com/celsofranssa/RAG-Fuse`.

5. Resultados Experimentais

5.1. QP1 – Efetividade

Os resultados de efetividade estão resumidos na Tabela 2. Observa-se que nenhum método de classificação tradicional (*ML*) nem as abordagens baseadas em *SLMs* alcançaram os melhores desempenhos em qualquer conjunto de dados, apesar de no TWITTER as SLMs RoBERTa e BART apresentarem desempenho competitivo em relação às LLMs. Sumarizando, as abordagens com *LLMs* estabeleceram os *baselines* mais competitivos nos conjuntos OHSUMED, ACM e TWITTER. Já no conjunto REUTERS, o método MatchXML, baseado em clusterização (*CB*), destacou-se como o *baseline* mais efetivo, provavelmente devido ao maior número de classes desse conjunto de dados.

Tabela 2. 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)
	RAG-Fuse	**83.5(0.4)**	**76.8(1.1)**	**84.7(1.1)**	**71.6(2.8)**

O **RAG-Fuse** apresenta os melhores resultados absolutos de Macro-F1 em **todos** os datasets, sendo o único entre todos os métodos a atingir esse objetivo. 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.

5.2. QP2 – Eficiência

A Figura 3 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 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.

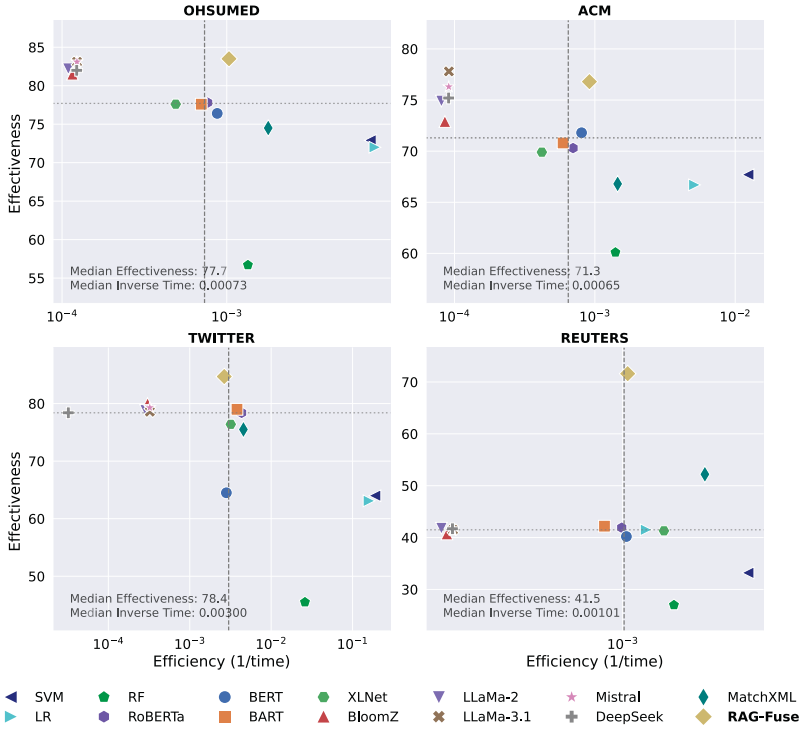

Figura 3. Relação entre eficiência (inverso do tempo em segundos) e eficácia (Macro-F1). Modelos mais desejáveis estão posicionados no quadrante superior direito (alta eficácia e alta eficiência).

5.3. QP3 – *Gap* Léxico-Semântico (Análise de Ablação)

Para mensurar o gap léxico-semântico, avaliamos o alinhamento entre os temas dos textos e de suas classes por meio de duas perspectivas: léxica e semântica. Adotamos uma abordagem de recuperação de informação, tratando as classes como documentos indexados e os textos como consultas. Na perspectiva léxica, utilizamos os métodos *TF-IDF* e *BM25* para ranqueamento. Já na perspectiva semântica, empregamos representações vetoriais extraídas de um modelo *transformer* em configuração zero-shot, utilizando similaridade cosseno para ranquear as classes. O alinhamento entre texto e classe é avaliado por meio do *Macro-F1*, comparando a classe mais bem ranqueada com a classe real.

Tabela 3. Medição do *gap* léxico-semântico por meio da Macro-F1

	DATASET			
GAP	**OHSUMED**	**ACM**	**TWITTER**	**REUTERS**
Léxico	17.0(0.7)	14.4(0.5)	0.0(0.0)	38.8(1.2)
Semântico	3.3(0.3)	13.0(0.6)	17.8(1.2)	0.2(0.1)

Tabela 4. Macro-F1 por dataset no estudo de ablação sobre os componentes do RAG-Fuse.

	DATASET			
MODEL	**OHSUMED**	**ACM**	**TWITTER**	**REUTERS**
RAG-Fuse	**83.5(0.4)**	**76.8(1.1)**	**84.7(1.1)**	**71.6(2.8)**
- RAG-labels	44.7(10.2)	58.1(9.3)	**82.6(2.1)**	61.0(12.3)
- sparso	79.5(0.7)	68.6(1.7)	77.1(1.7)	66.6(2.2)
- denso	70.6(1.0)	67.5(1.9)	66.3(1.8)	59.8(2.3)

A Tabela 3 mostra a magnitude do *gap* nas duas perspectivas. No TWITTER, por exemplo, o *gap* léxico é máximo (não há nenhuma correspondência direta de vocabulário entre o texto e a classe correspondente). Um caso ilustrativo é o da classe *sports_&_gaming*, atribuída a um *tweet* que comemora uma vitória em um jogo de *softball*, mas não menciona nenhum dos termos que caracteriza a classe. Na OHSUMED e REUTERS, o desalinhamento é predominantemente semântico. Na REUTERS a busca

densa apresenta desempenho insatisfatório devido ao número elevado de classes e à sobreposição semântica entre elas. Na OHSUMED, o vocabulário técnico da área médica contribui para um *gap* semântico pronunciado. Juntos, os resultados para essas duas métricas revelam o desalinhamento estrutural (léxico e semântico) que motiva o uso de técnicas mais sofisticadas de descrição e recuperação de classes como as do **RAG-Fuse**.

Para demonstrar que os três principais componentes do **RAG-Fuse** – *RAG-labels*, recuperadores esparso e denso – são fundamentais para reduzir o *gap* léxico-semântico e o impacto do desbalanceamento de classes, realizamos uma avaliação do impacto da remoção de cada componente na qualidade final do resultado (análise de ablação). A Tabela 4 mostra os efeitos na *Macro-F1* ao desabilitar cada um desses componentes. Contrastamos esses resultados com a solução completa (linha **RAG-Fuse**).

Os resultados confirmam nossa hipótese de que a fusão entre *rankings* é crucial para minimizar o *gap* léxico-semântico. Em todos os conjuntos de dados, a fusão superou os recuperadores isolados. A fusão obteve ganhos de **5%** em *OHSUMED*, **12%** em *ACM*, **10%** em *TWITTER* e **8%** em *REUTERS* sobre o recuperador denso, e **18%**, **14%**, **28%** e **20%**, respectivamente, sobre o recuperador esparso. Esses resultados reforçam a complementaridade dos recuperadores. Quanto aos *RAG-labels*, os ganhos foram significativos, com aumentos de **18%** em *REUTERS*, **87%** em *OHSUMED* e **32%** em *ACM*. O *TWITTER* foi o único *dataset* onde desabilitar os *RAG-labels* emparelhou estatisticamente com o **RAG-Fuse**. Como demonstrado na Tabela 3, embora o *gap* léxico seja o máximo possível neste conjunto, o *gap* semântico é o menor dentre os cenários avaliados. As seis classes presentes no *TWITTER* (tais como *sports_&_gaming*, *business_&_entrepreneurs* e *science_&_technology*) são **semanticamente** bem definidas e mais facilmente separáveis. Isso permite que o recuperador denso explore com maior eficácia as associações semânticas aprendidas entre as classes e os termos dos documentos, reduzindo a necessidade de enriquecimento contextual via *RAG-labels*.

6. Conclusão e Direções Futuras

Neste trabalho, propusemos o **RAG-Fuse**, uma abordagem inovadora para CMCT reformulada como tarefa de RI. A proposta baseia-se em dois componentes principais: (i) geração de *RAG-labels* via otimização de prompts com LLMs, enriquecendo semanticamente as descrições das classes; e (ii) fusão de rankings oriundos de recuperadores esparsos e densos, com tratamento diferenciado para classes majoritárias e minoritárias. Experimentos em quatro benchmarks públicos mostraram ganhos de até **37%** em Macro-F1 sobre o segundo melhor método (e até **70%** sobre LLMs) em cenários desbalanceados com muitas classes. O RAG-Fuse também superou as LLMs com até **10** vezes menos custo computacional, evidenciando sua eficácia e escalabilidade. Como direções futuras, destacamos a atualização contínua dos RAG-labels em domínios dinâmicos, a aplicação do pipeline em tarefas como Modelagem de Tópicos e recomendação, e o estudo de fusões adaptativas baseadas na confiança dos recuperadores e na dificuldade das amostras.

Acknowledgments

Este trabalho foi apoiado por CNPq, Capes, Fapemig, Fapesp, AWS e Instituto Nacional de Ciência e Tecnologia em Inteligência Artificial Responsável para Linguística Computacional, Tratamento e Disseminação de Informação (INCT-TILD-IAR; 408490/2024-1).

Referências

Arabzadeh, N. et al. (2021). Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. In *CIKM*, page 2862–2866.

Askari, A. et al. (2023). Injecting the bm25 score as text improves bert-based re-rankers. In *ECIR*, page 66–83.

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

Bruch, S. et al. (2023). An approximate algorithm for maximum inner product search over streaming sparse vectors. *TOIS*, 42(2):1–43.

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

Cunha, W. et al. (2023). A comparative survey of instance selection methods applied to nonneural and transformer-based text classification. *ACM*.

Cunha, W., Moreo, A., Esuli, A., Sebastiani, F., Rocha, L., and Gonçalves, M. A. (2024). A noise-oriented and redundancy-aware instance selection framework. *ACM TOIS*.

Cunha, W., Rocha, L., and Gonçalves, M. A. (2025). A thorough benchmark of automatic text classification: From traditional approaches to large language models. *arXiv*.

de Andrade, C., Cunha, W., Reis, D., Pagano, A. S., Rocha, L., and Gonçalves, M. A. (2024). A strategy to combine 1stgen transformers and open llms for automatic text classification. *arXiv*.

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.

Devlin, J. et al. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J. and othes, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Dubey, A. et al. (2024). The llama 3 herd of models. *arXiv*.

França, C. and others. (2025). Optimizing tail-head trade-off for extreme multi-label text classification (xmtc) with rag-labels and a dynamic two-stage retrieval and fusion pipeline. In *Proceedings of the 48th International ACM SIGIR conference on Research and Development in Information Retrieval*.

França, C., Rabbi, G., Salles, T., Cunha, W., Rocha, L., and Gonçalves, M. A. (2025). Ranking-based fusion algorithms for extreme multi-label text classification (xmtc).

Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Jiang, F. (2024). Identifying and mitigating vulnerabilities in llm-integrated applications. Master's thesis, University of Washington.

Jiang, T. et al. (2021). Lightxml: Transformer with dynamic negative sampling for high-performance extreme multi-label text classification. In *AAAI*, volume 35, pages 7987–7994.

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.

Lin, S.-C. et al. (2023). Aggretriever: A Simple Approach to Aggregate Textual Representations for Robust Dense Passage Retrieval. *TACL*, 11:436–452.

Liu, J. et al. (2023). A contrastive learning framework for safety information extraction in construction. *Advanced Engineering Informatics*, 58:102194.

Llordes, M. et al. (2023). Explain like i am bm25: Interpreting a dense model's ranked-list with a sparse approximation. In *SIGIR*, page 1976–1980. ACM.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T. L., Bari, M. S., Shen, S., Yong, Z.-X., Schoelkopf, H., et al. (2022). Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Penha, G. and Hauff, C. (2023). Do the findings of document and passage retrieval generalize to the retrieval of responses for dialogues? In *ECIR*, pages 132–147.

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.

Sun, A., Lim, E.-P., and Liu, Y. (2009). On strategies for imbalanced text classification using svm: A comparative study. *Decision Support Systems*, 48(1):191–201.

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).

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Wang, J., Chen, Z., Qin, Y., He, D., and Lin, F. (2023). Multi-aspect co-attentional collaborative filtering for extreme multi-label text classification. *Knowledge-Based Systems*, 260(2):1–11.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Ye, H., Sunderraman, R., and Ji, S. (2024). Matchxml: An efficient text-label matching framework for extreme multi-label text classification. *IEEE TKDE*, 36(9):4781–4793.

You, R. et al. (2019). Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. In Wallach, H. et al., editors, *NeurIPS*, volume 32, pages 1–11.

Zhang, J. et al. (2021). Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In *NeurIPS*, volume 34, pages 7267–7280.

Zhou, Q., Zhou, H., and Li, T. (2016). Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features. *Knowledge-based systems*, 95:1–11.