Estratégias de Undersampling para Redução de Viés em Classificação de Texto Baseada em Transformers

Guilherme Fonseca
guilhermefonseca8426@aluno.ufsj.edu.br
UFSJ
Minas Gerais, Brazil

Gabriel Prenassi
prenassigabriel@aluno.ufsj.edu.br
UFSJ
Minas Gerais, Brazil

Washington Cunha
washingtoncunha@dcc.ufmg.br
UFMG
Minas Gerais, Brazil

Marcos André Gonçalves
mgoncalv@dcc.ufmg.br
UFMG
Minas Gerais, Brazil

Leonardo Rocha
lcrocha@ufsj.edu.br
UFSJ
Minas Gerais, Brazil

ABSTRACT

Automatic Text Classification (ATC) in unbalanced datasets is a common challenge in real-world applications. In this scenario, one (or more) class(es) is overrepresented, which usually causes a bias in the learning process towards these majority classes. This work investigates the effect of undersampling methods, which aim to reduce instances of the majority class, on the effectiveness of recent ATC methods. Through a systematic mapping of the literature, we selected and implemented 15 undersampling strategies. We also propose two new strategies and compare all 17 methods using RoBERTa as sentiment analysis classifier. Our results suggest that a set of undersampling approaches is capable of significantly reducing the learning bias of ATC methods towards the majority class on imbalanced datasets, without incurring any effectiveness loss, and with improvements in efficiency and reduction of carbon emissions.

KEYWORDS

Classificação de Texto, Transformers, Undersampling

1 INTRODUÇÃO

Classificação Automática de Texto (CAT) [16, 29] tem experimentado uma grande evolução nos últimos anos, com ênfase em estratégias supervisionadas motivadas por avanços em aprendizagem profunda baseados em *Transformers* [11, 25]. Essas estratégias se beneficiaram de aplicações que constantemente produzem grandes volumes de dados rotulados (por exemplo, redes sociais), nos quais os usuários podem classificar manualmente mensagens, anúncios e produtos, produzindo um grande volume de anotações. À medida que a quantidade de dados aumenta, estratégias de CAT baseadas em *Transformers* tornam-se mais eficazes, superando significativamente métodos tradicionais de CAT [30], tais como Regressão Logística, KNN, *Random Forest* e *SVM* [7].

Apesar dos enormes avanços de efetividade alcançados pelo uso de *Transformers* em CAT, o impacto dessas novas abordagens em problemas tradicionais de classificação, como o viés de aprendizado oriundo de coleções de dados desbalanceadas, tem sido pouco estudado [5]. Em coleções de dados desbalanceadas, as classes minoritárias podem ser sub-representadas no processo de aprendizado, resultando em modelos com baixa capacidade de generalização, enviesados para as classes majoritárias. Além de questões éticas relacionadas a modelos enviesados [14], existem diversos cenários em que a classe minoritária é a classe de interesse.

Por exemplo, na área de saúde, a predição de doenças é geralmente uma tarefa desbalanceada [42], dado que a maioria das pessoas são saudáveis ou pelo menos não apresentam a condição de interesse. Nesse caso, existe a tendência de um modelo treinado com uma amostra de dados reflita essa distribuição, sendo enviesado para a classe "saudável". Um modelo de classificação enviesado pode ser bastante acurado apenas predizendo a classe majoritária, porém o impacto de um falso negativo (pacientes portadores da doença erroneamente diagnosticados como não portadores) é muito alto[42].

Outro exemplo são aplicações relacionadas à análise de sentimento, foco do presente trabalho. Em cenários de produtos [13, 27] e pontos de interesse (POI) [18], usuários comumente tomam suas decisões de consumo analisando comentários e avaliações de outros usuários. Alguns usuários levam mais em consideração as avaliações classificadas como positivas (e.g., pontos positivos de um hotel), outros as avaliações negativas (por que não comprar certo produto?). Em ambas as situações a classe alvo pode ser a minoritária e o resultado de classificação de um modelo enviesado poderá influenciar de forma equivocada a decisão do usuário.

Assim, a primeira pergunta de pesquisa que procuramos responder é *PP1: Como algoritmos estado da arte recentes, baseados em Transformers, são afetados pelo desbalanceamento de classes em tarefas de análise de sentimentos? Há espaço para melhorias?*.

Para responder esta pergunta, comparamos o desempenho de seis métodos de classificação tradicionais (KNN, *Random Forest*, Regressão Logística, Support Vector Machine (SVM), XGBoost e LightGBM) e três métodos baseados em *Transformers* (RoBERTa [25], BART [23] e BERT [11]), avaliando tanto a efetividade (em termos de MacroF1) quanto o viés dos modelos resultantes (em termos de TPRGap, uma métrica que captura o viés a partir da diferença absoluta entre o TPR - *True Positive Rate* - das classes [21]).

Nossos resultados mostram que modelos baseados em *Transformers*, além de efetivos, possuem um viés menor quando comparados a modelos gerados por classificadores tradicionais. No entanto, nossos resultados também apontam que, apesar disso, ainda há espaço

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2024). Juiz de Fora, Brazil. Porto Alegre: Brazilian Computer Society, 2024.

ISSN 2966-2753

considerável para melhorias, principalmente em bases de dados que possuem um elevado grau de desbalanceamento (razão entre número de documentos da classe majoritária e o número de documentos da classe minoritária superior a 5).

Existem duas principais abordagens utilizadas para lidar com o desbalanceamento de dados. O *oversampling* consiste em criar novas amostras (geralmente sintéticas) da classe minoritária, para igualá-la em termos de instâncias à classe majoritária [15]. Essa abordagem incorre em um aumento significativo no tempo de geração dos modelos, uma vez que aumenta-se o total de instâncias a serem consideradas no aprendizado. Mais ainda, a geração sintética de novos dados, principalmente textuais, pode ser não trivial [12].

A abordagem alternativa para enfrentar o desbalanceamento é o *undersampling* (US), foco do presente trabalho, que são técnicas que reduzem instâncias da classe majoritária para equilibrar as classes. Até onde sabemos, não existem estudos na literatura que abordam como os métodos de *undersampling* interagem com os algoritmos de CAT do estado da arte baseados em *Transformers*. Assim, nossa segunda pergunta de pesquisa é *PP2: Métodos de undersampling, aplicados juntamente com classificadores baseados em Transformers, são capazes de reduzir o viés dos modelos de classificação? Qual o impacto dessa combinação na efetividade do modelo?*

Para responder a PP2, primeiramente realizamos uma revisão sistemática sobre os principais métodos de *undersampling* propostos na literatura. Identificamos e implementamos 14 métodos de *undersampling* que estão entre os mais utilizados. Apesar de terem fins diferentes, as áreas de seleção de instâncias (SI) [8] e *undersampling* são bastante relacionadas, pois ambas tratam de técnicas que visam selecionar um subconjunto de dados representativos – o que as difere são os objetivos da redução. Nesse sentido, adaptamos uma estratégia de SI, que é considerada estado da arte, para o cenário de *undersampling*, o E2SC [6]. Por fim, propomos duas novas estratégias de *undersampling*, que são contribuições desse trabalho: (1) a UBR (*Undersampling* Baseado em Redundância), que se concentra em remover instâncias da classe majoritária consideradas redundantes (muito similares à outras instâncias); e (2) E2SC-RL, uma variação do método de SI E2SC que realiza o cálculo das probabilidades das instâncias serem removidas por meio de Regressão Logística.

Investigamos o desempenho das 17 técnicas de *undersampling* em conjunto com o RoBERTa, classificador baseado em *Transformers* considerado estado da arte em análise de sentimentos [26]. De fato, *benchmarks* recentes [10, 26] demonstraram que as diferenças entre as novas versões desses *Transformers* (incluindo RoBERTa, BERT, DistilBERT, BART, AlBERT e XLNet) em diversos conjuntos de dados utilizados em nossos experimentos são muito pequenas. Nossos resultados apontaram que os métodos de *undersampling* NM1 [28], NM2 [28], E2SC [6], E2SC-RL (nossa proposta) e UBR (nossa proposta) foram capazes de reduzir o viés do modelo de classificação, mantendo sua efetividade.

Métodos de CAT baseados em *Transformers* demandam elevado custo computacional no processo de aprendizado dos modelos de classificação, resultando em longos tempos de execução e também contribuindo significativamente para a emissão de carbono na atmosfera [1]. Assim, precisamos investigar o impacto desse novo passo de pré-processamento na eficiência (i.e. tempo e custo computacional para geração e classificação) dos modelos. Dessa forma, nossa terceira pergunta de pesquisa é: *PP3: Qual o impacto da aplicação dessa etapa adicional de pré-processamento (undersampling) em termos de eficiência? E em termos da emissão de carbono?*.

Nossos resultados apontam que o uso de algumas das estratégias de *undersampling*, mais especificamente UBR, NM1, E2SC-RL, NM2 e E2SC, que foram capazes não apenas de reduzir o viés sem perda de efetividade, mas também diminuíram significativamente o tempo de treinamento dos modelos (50% em média), o que, consequentemente, contribuiu para redução na emissão de CO_2 (50%, em média) durante a geração e execução dos modelos de classificação.

Assim, as principais contribuições deste trabalho são:

- Mapeamento sistemático da literatura sobre métodos de *undersampling*, identificando e implementando 14 métodos que estão entre os mais utilizados. Identificamos e adaptamos também um método de seleção de instâncias para a tarefa de *undersampling*;
- Propostas de duas novas estratégias de *undersampling*;
- Avaliação das técnicas de *undersampling* em conjunto com classificadores baseado em *Transformers* sob três perspectivas: (1) efetividade da classificação; (2) eficiência (tempo); e (3) capacidade de generalização (viés).

2 LEVANTAMENTO DAS ESTRATÉGIAS

Nesta seção, detalhamos o processo de revisão sistemática da literatura (RSL) [7, 8] para selecionar quais estratégias de *undersampling* e seleção de instâncias que serão avaliadas.

2.1 Métodos de Undersampling

Recorremos ao mecanismo de pesquisa do Google Scholar para submeter a consulta e gerar nosso conjunto inicial de artigos. O Google Scholar foi escolhido devido à sua ampla cobertura, abrangendo as principais bibliotecas digitais de editoras como ACM, IEEE e Elsevier, além de repositórios de pré-impressão como Arxiv. A *string* de busca utilizada foi "*Undersampling*" e, para maximizar a abrangência da pesquisa, o mecanismo de busca não aplicou nenhum filtro de local ou ano. Com base nisso, coletamos, inicialmente, um total de 500 artigos únicos que, de alguma forma, utiliza alguma estratégia de *undersampling*.

Analisamos manualmente os 500 artigos, procurando identificar os mais pertinentes para o estudo. Um artigo foi considerado **relevante** caso utilizasse técnicas de *undersampling* para reduzir desbalanceamento, sendo que o método de *undersampling* empregado deveria ser explicitamente mencionado (citado). Identificamos 139 artigos relevantes e, a partir deles, enumeramos todas as técnicas de *undersampling* utilizadas, encontrando, ao todo, 32 técnicas diferentes. Uma tabela completa com uma descrição de todas as estratégias identificadas está disponível online (GitHub). Optamos por considerar em nossa avaliação aqueles que foram utilizados em mais de um dos trabalhos relevantes. Faremos nossa avaliação sobre os seguintes métodos:

- Links de Tomek (TL) [37]: dados dois exemplos e_i e e_j de diferentes classes, com $d(e_i, e_j)$ representando a distância entre e_i e e_j, um par $A(e_i, e_j)$ é chamado de link de Tomek se não houver nenhum exemplo e_l tal que $d(e_i, e_l) < d(e_i, e_j)$ ou $d(e_j, e_l) < d(e_i, e_j)$.

Se dois exemplos formam um link de Tomek, então ou um desses exemplos foi classificado manualmente errado ou ambos são exemplos pertencentes à fronteiras entre as classes e podem ser removidos.

- **Condensed Nearest Neighbors (CNN)[17]:** O conjunto de dados S é inicializado com um exemplo da classe majoritária e todos os exemplos da classe minoritária e um conjunto T é criado com os elementos que não pertencem a S. Cada exemplo de T é classificado pelo KNN usando S como conjunto de treinamento. Caso o KNN acerte a classe do exemplo, ele permanece em T; caso contrário, o exemplo é removido de T e colocado em S. Esse processo se repete até que não ocorram mais mudanças no conjunto S. Ao final, os elementos de T são descartados.
- **One-Sided Selection (OSS)[20]:** Combina o TL e uma variação do CNN. inicialmente, como no CNN, um conjunto S é inicializado com todas as instâncias da classe minoritária e uma da classe majoritária e um conjunto T com o restante dos elementos, depois as instâncias de T são classificadas com o KNN treinado em S e cada instância classificada erroneamente é colocada em S. No final, o TL é utilizado em S para identificar pares ambíguos na fronteira da classe.
- **Edited Nearest Neighbours (ENN)[39]:** insere todas as instâncias do conjunto original T no conjunto de solução S, utilizando o KNN de maneira iterativa para classificar todas as instâncias x dado que $x \in S$ e que x pertença a classe majoritária (considerando o conjunto $\{S - \{x\}\}$ como possíveis vizinhos). Por fim, remove as instâncias classificadas incorretamente.
- **Repeated Edited Nearest Neighbours (RENN)[36]:** ENN aplicado sucessivamente até que não seja possível remover mais pontos.
- **ALL k-NN [36]:** ENN aplicado sucessivamente, mas, a cada aplicação, o número de vizinhos a serem considerados aumenta.
- **Neighbourhood Cleaning Rule (NCR)[22]:** Utiliza o KNN para classificar todas as instâncias da base de dados. Caso a classe prevista seja diferente da classe real e a instância pertença à classe majoritária, a instância é eliminada. O NCR classifica também as instâncias da classe minoritária. Se a classificação estiver incorreta, o método elimina os vizinhos mais próximos da instância que pertencem à classe majoritária.
- **Near Miss (NM)[28]:** três métodos de *undersampling* são propostos. O NearMiss-1 (NM1) remove as instâncias da classe majoritária que têm a menor distância média entre as k instâncias da classe minoritária. O NearMiss-2 (NM2) seleciona os elementos da classe majoritária cuja distância média para os k pontos mais distantes da classe minoritária é a mais baixa. Já o NearMiss-3 (NM3) calcula, para cada instância da classe minoritária, as k instâncias da classe majoritária mais próximas e as mantém na base de dados.
- **SBC [41]:** Todo o conjunto de treino é dividido em N *clusters*. Para cada um dos *clusters*, o número de instâncias a serem selecionadas é calculado com base no número de amostras da classe majoritária e da classe minoritária que existem no *cluster*. Após isso, exemplos da classe majoritária são selecionados aleatoriamente. Por fim, o algoritmo combina as instâncias selecionadas de cada *cluster* com as da classe minoritária para formar um novo conjunto.
- **IHT [35]:** Utiliza um classificador (c) para obter o *instance hardness* (IH) de cada instância. O IH de uma instância é dado por $IH(< x_i, y_i >) = 1 - p(y_i|x_i, c)$ onde $p(y_i|x_i, c)$ denota a probabilidade, gerada pelo classificador c, da instância x_i pertencer à classe y_i. O IHT seleciona amostras da classe majoritária com baixa probabilidade de pertencerem à classe majoritária para serem removidas.
- **CC-NN [24]:** As instâncias da classe majoritária são divididas em N *clusters*, com N sendo o número de instâncias da classe minoritária. Após isso, o vizinho mais próximo do centróide de cada um dos *clusters* que pertença à classe majoritária é escolhido para compor, junto com as instâncias da classe minoritária, o conjunto final.
- **OBU [38]:** Utiliza o Fuzzy c-means para dividir os dados em 2 *clusters*, onde o *cluster* que tiver mais instâncias da classe minoritária é chamado de *CM*. Depois disso, o algoritmo remove todas as instâncias da classe majoritária cujo grau de pertencimento para o CM é menor que α (hiperparâmetro).

2.2 Métodos de seleção de instância

Apesar de terem fins diferentes, as áreas de seleção de instâncias (SI) e *undersampling* (US) são relacionadas, pois tratam de técnicas que visam selecionar um subconjunto de dados a ser usado no treinamento do modelo e que cumpram seus objetivos: (1) no caso de SI, melhorar a eficiência sem perda de efetividade; e (2) no caso de US, reduzir o viés da classe majoritária, mantendo a efetividade.

Apesar dos objetivos finais serem diferentes, partimos da hipótese de que há uma relação subjacente entre ambas as tarefas, principalmente para métodos de SI baseados em redução de redundância [8], que podem ser adaptados para remoção de instâncias redundantes da classe majoritária. Essa hipótese norteia a concepção do nosso novo método de US - UBR - descrito na próxima seção e também nos motivou a selecionar e adaptar trabalhos de SI para *undersampling*.

Uma varredura na literatura de SI aplicada a CAT nos revela que o método E2SC [6, 32] é o estado da arte. O método funciona em duas etapas. Na primeira, calcula as probabilidades de cada instância ser removida. Estas probabilidades são obtidas por meio da confiança do classificador KNN, que é calibrado (classificador cujas previsões de probabilidade de classe correspondem bem à acurácia do classificador) [34]. Na segunda etapa, o E2SC tenta estimar qual a taxa de redução ótima para a base de dados. Após isso, as instâncias são amostradas aleatoriamente, ponderadas pela probabilidade encontrada no primeiro passo. Para o nosso trabalho, realizamos uma modificação do E2SC, chamada E2SC_RL, que segue o mesmo princípio do E2SC, porém, em vez do KNN como classificador, utilizaremos Regressão Logística (RL). Optamos por essa abordagem ser um classificador igualmente calibrado e possuir baixo custo computacional, inferior ao KNN. Portanto, consideramos em nossos experimentos o E2SC e o E2SC_RL, ambos adaptados para remover apenas instâncias da classe majoritária. Além de todos as estratégias apresentadas nesta seção, apresentamos a seguir nossa nova proposta de estratégia de *undersampling*.

3 MÉTODO PROPOSTO

Nesta seção, apresentamos nossa segunda contribuição neste trabalho, uma nova abordagem denominada UBR (*Undersampling* Baseado em Redundância). Essa abordagem busca inspiração em técnicas de SI. Um par de documentos é considerado redundante se apresenta alta similaridade entre si. Nossa hipótese é que manter apenas uma das instâncias no conjunto de treinamento é suficiente, pois a presença de ambas não trará aumento significativo na aprendizagem do modelo. Ao se concentrar na redução de redundância na classe majoritária, obtemos um potencial de redução do desbalanceamento e, consequentemente, do viés para essa classe. O **Algoritmo 1** detalha o pseudocódigo do UBR.

Algoritmo 1: Algoritmo UBR

Input: X, α
Output: instanciasSel

$X_{Maj} \leftarrow obterInstancias(X, classe = majoritaria)$;
$X_{Min} \leftarrow obterInstancias(X, classe = minoritaria)$;
$instanciasSel \leftarrow X_{Min}$;
$D \leftarrow distanciaAproximadaNVizinhos(X_{Maj})$;
$clusters \leftarrow X_{Maj}$;
$N \leftarrow \|X_{Maj}\| - \|X_{Min}\|$;
while $N > 0$ **do**
  $A, B \leftarrow ParMaisSimilar(X_{Maj}, D)$;
  if $(\|clusters[A]\| > \alpha) OR (\|clusters[B]\| > \alpha)$ **then**
    continue ;
  end
  if $clusters[A] \neq clusters[B]$ **then**
    $clusters[A] \leftarrow clusters[A] \bigcup clusters[B]$;
    $deletar(cluster[B])$;
    $N = N - 1$;
  end
end
$escolhidos \leftarrow escolheRepresentante(clusters)$;
$instanciasSel \leftarrow instanciasSel \bigcup escolhidos$;

Dado X como o conjunto total de instâncias de treinamento, inicialmente dividimos X em dois conjuntos, X_{Maj} e X_{Min}, onde X_{Maj} consiste em instâncias de X pertencentes à classe majoritária, e X_{Min} consiste em instâncias de X pertencentes à classe minoritária. Para cada instância de X_{Maj}, calculamos a similaridade de cosseno da instância para os seus K vizinhos mais próximos e colocamos em uma lista ordenada D.

Por questões de otimização de tempo e de uso de memória, utilizamos uma versão aproximada do KNN [33]. Após isso, passamos a considerar cada instância de X_{Maj} como um *cluster* individual e realizamos N iterações, onde $N = X_{Maj} - X_{Min}$. Em cada iteração, buscamos em D o par de instâncias A e B pertencentes a X_{Maj} que apresenta a maior similaridade e que estão em *clusters* distintos e cujos *clusters* não sejam maiores que α (hiperparâmetro do método que controla a quantidade de vizinhos a ser avaliada), com o objetivo de unir os *clusters* aos quais A e B pertencem. Ao final, temos $|X_{Min}|$ *clusters* dentro do conjunto X_{Maj}, dos quais será selecionado aleatoriamente um representante para compor, juntamente com o conjunto X_{Min}, o novo conjunto de treinamento.[1]

dataset	# docs	# majoritária	# minoritária	RD	Nome
sentistrength_twitter_2L	2,289	1,340	949	1.41	A
vader_amazon_2L	3,610	2,128	1,482	1.44	B
english_dailabor_2L	1,227	739	488	1.51	C
debate_2L	1,979	1,249	730	1.71	D
sentistrength_youtube_2L	2,432	1,665	767	2.17	E
sentistrength_rw_2L	705	484	221	2.19	F
vader_twitter_2L	4,196	2,897	1,299	2.23	G
tweet_semevaltest_2L	3,060	2,223	837	2.66	H
sentistrength_digg_2L	782	572	210	2.72	I
sentistrength_myspace_2L	834	702	132	5.32	J
sentistrength_bbc_2L	752	653	99	6.60	K
digital_music_2L	162,989	158,985	4,004	39.71	L

Tabela 1: Coleções de dados utilizadas nos experimentos. A coluna "nome" contém como a base vai ser referenciada.

[1]Nossa técnica é limitada a problemas de classificação binários. Deixamos para o futuro a extensão da abordagem para problemas multi-rótulo.

4 CONFIGURAÇÃO EXPERIMENTAL

4.1 Base de dados

Consideramos 12 *datasets*, com diferentes níveis de desbalanceamento. A Tabela 1 mostra os *datasets* com o número de documentos, número de documentos pertencentes à classe majoritária e à minoritária, o nome pelo qual vamos nos referir a base de dados ao longo do trabalho e a razão de desbalanceamento (RD)[31], métrica que demonstra o quão desbalanceado é uma base de dados (quanto maior for a RD mais desbalanceado é a base de dados). O RD é calculado como sendo $RD = \frac{classe\ majoritaria}{classe\ minoritaria}$.

4.2 Método de Classificação de Texto

Consideramos métodos de CAT baseados em *Transformers* **BART** [23], **RoBERTa** [25] e **BERT**[11], que atualmente se apresentam como os melhores entre os métodos de classificação utilizados na literatura para tarefa de Análise de Sentimento [8]. Para ajustar os hiperparâmetros, utilizamos a mesma metodologia discutida em [8]. Assim, fixamos a taxa de aprendizado inicial como 5×10^{-5}, o número máximo de épocas como 20 e a paciência como 5 épocas. Por fim, realizamos um grid search em max_len (150 e 256) e batch_size (16, 32 e 64), pois esses valores especificados impactam diretamente na eficiência e efetividade do modelo. Utilizamos, também, 6 classificadores tradicionais: **KNN**, *Random Forest* [3], Regressão Logística (**RL**) [40], **SVM** [2], *XGBoost* (**XGB**) [4] e *LightGBM* (**LGBM**) [19].

4.3 Métricas e Protocolo Experimental

Nossa avaliação é feita sob três perspectivas: (1) efetividade da classificação; (2) capacidade de generalização dos modelos (viés); e (3) eficiência (tempo e emissão de CO_2). A efetividade é avaliada utilizando a *Macro Average F1* (MacroF1). A capacidade de generalização é medida pela métrica TPRGap apresentada em [9], definida na Equação 1, onde $TPR(i)$ é *true positive rate* da classe i, T é o número total de classes, N é o fator de normalização, que é igual ao número de pares de classes que comparamos $\binom{T}{2}$.

$$TPRGap = \sum_{i,j \epsilon T} \frac{|TPR(i) - TPR(j)|}{N} \quad (1)$$

A eficiência é medida com base no custo de cada método em termos do tempo total necessário para construir o modelo e realizar as clasificações. O *Speedup* é calculado como $S = \frac{T_{wo}}{T_w}$, onde T_w é o tempo total gasto na construção do modelo, mais o tempo da classificação, usando alguma abordagem de *undersampling*, e T_{wo} é o tempo total gasto na execução (modelo e classificação) sem a fase de *undersampling*. A emissão de CO_2 é o equivalente de dióxido de carbono gasto para treinamento de um modelo e classificação baseado em [21].

Os experimentos foram realizados na AWS. Para as bases de dados de A até K, as etapas *undersampling*, que demandam estritamente processamento em CPU, utilizaram uma instância do tipo **c6a.4xlarge** e a classificação, que demanda hardware especializado (GPU), instâncias do tipo **g4dn.xlarge**. Para a base L, que é bem maior que as demais, demandou um poder computacional maior e ambas as etapas foram executadas em instâncias do tipo **g5.4xlarge**. As bases de dados foram divididas utilizando o método de validação cruzada com 5 partições (base L) ou 10 partições (demais bases). As comparações foram realizadas utilizando o método estatístico Teste-T com correção de Bonferroni [8].

	RoBERTa		BART		BERT		SVM		LR		RF		XGB		LGBM		KNN	
dataset	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap	Macro F1	TPRGap
A	88.6(0.7)	0.063	**89.3(1.1)**	0.048	84.3(1.7)	0.050	71.8(2.5)	0.187	71.4(2.5)	0.228	67.0(2.4)	0.370	64.9(2.4)	0.417	63.2(3.2)	0.375	64.3(3.1)	0.478
B	**89.0(0.7)**	0.065	88.3(1.4)	0.079	86.9(0.8)	0.057	72.1(1.5)	0.257	72.9(1.5)	0.209	68.6(1.8)	0.422	67.6(1.4)	0.247	69.4(2.7)	0.250	65.9(2.8)	0.498
C	93.3(1.1)	0.041	**93.7(1.2)**	0.036	89.1(2.2)	0.063	79.3(3.1)	0.111	80.4(2.1)	0.117	75.7(2.8)	0.221	75.0(1.5)	0.171	68.9(5.9)	0.208	76.9(2.1)	0.166
D	**89.3(1.2)**	0.076	89.1(1.1)	0.085	85.5(2.0)	0.146	76.4(2.1)	0.213	75.6(3.6)	0.223	73.3(3.0)	0.271	71.5(1.8)	0.302	72.8(3.4)	0.296	72.7(3.2)	0.330
E	**89.7(1.9)**	0.096	88.9(1.7)	0.117	86.1(1.8)	0.134	79.0(1.7)	0.215	78.6(2.0)	0.245	72.6(4.0)	0.257	71.1(1.7)	0.472	71.7(3.7)	0.385	73.3(3.4)	0.263
F	87.3(3.4)	0.126	**88.0(3.3)**	0.128	80.9(2.5)	0.202	69.1(3.4)	0.421	68.2(2.8)	0.471	59.3(4.1)	0.698	63.1(5.0)	0.582	65.8(4.2)	0.454	58.8(2.2)	0.698
G	**94.2(1.0)**	0.160	93.7(1.0)	0.053	88.0(1.3)	0.108	82.0(1.0)	0.238	80.4(1.4)	0.320	72.6(1.9)	0.526	74.0(1.4)	0.476	74.0(2.8)	0.372	75.4(1.3)	0.454
H	**90.1(1.5)**	0.102	90.1(1.6)	0.099	86.4(1.9)	0.125	70.9(2.0)	0.404	71.9(1.7)	0.399	64.5(2.0)	0.615	64.0(1.9)	0.660	63.9(4.4)	0.633	60.8(1.5)	0.654
I	**83.8(5.0)**	0.190	81.6(5.6)	0.198	79.1(3.6)	0.290	67.0(5.6)	0.539	63.0(6.8)	0.619	54.9(5.1)	0.731	56.7(5.2)	0.665	56.4(5.3)	0.598	55.2(6.4)	0.800
J	**83.2(3.4)**	0.333	83.0(5.8)	0.283	79.8(4.9)	0.350	68.1(3.8)	0.610	63.1(6.4)	0.733	59.8(4.6)	0.809	59.2(3.5)	0.803	58.4(10.5)	0.763	56.5(4.4)	0.875
K	**81.0(4.5)**	0.350	78.0(6.1)	0.410	76.4(4.4)	0.447	50.5(5.0)	0.944	53.3(4.7)	0.914	54.3(5.7)	0.888	53.4(5.3)	0.898	55.7(11.2)	0.833	46.4(0.1)	0.998
L	87.8(2.5)	0.308	**88.6(1.2)**	0.296	85.3(0.7)	0.383	78.7(0.3)	0.601	78.2(0.7)	0.562	78.9(0.4)	0.561	72.0(0.3)	0.698	73.6(0.7)	0.668	60.9(0.5)	0.866
Média:		0.159		0.153		0.196		0.395		0.420		0.531		0.533		0.486		0.590

Tabela 2: Resultados de Macro-F1 e TPRGap dos classificadores. Células em negrito são os maiores valores numéricos para uma base de dados e células em verde são estatisticamente equivalentes à classificação de maior valor numérico.

5 ANÁLISE DOS RESULTADOS

5.1 PP1: Comparação entre métodos de CAT

Para responder a **PP1** (*Como algoritmos estado da arte recentes, baseados em Transformers, são afetados pelo desbalanceamento de classes em tarefas de análise de sentimentos? Há espaço para melhorias?*), comparamos os classificadores tradicionais **KNN**, **RF**, **RL**, **SVM**, **XGB** e **LGBM** com os baseados em *Transformers* **RoBERTa**, **BART** e **BERT**.

A Tabela 2 apresenta os resultados de MacroF1 e TPRGap para estes classificadores. Como primeira análise, podemos destacar a superioridade, em termos de *efetividade*, dos classificadores baseados em *Transformers* quando comparados aos classificadores tradicionais – estes foram inferiores (estatisticamente) aos *Transformers* em todas as 12 bases de dados consideradas, resultado condizente com a literatura [7]. Já dentre os classificadores baseados em *Transformers*, o RoBERTa e o BART se destacam, ambos com resultados de classificação estatisticamente equivalentes em todas bases. O BERT, por sua vez, é estatisticamente equivalente ao RoBERTa e ao BART em apenas 4 das 12 bases de dados. Por fim, quando analisamos os valores numéricos absolutos de cada classificador, o modelo RoBERTa produz os maiores valores de MacroF1 em 8 coleções enquanto o BART o faz em 4 delas, reforçando a ideia da literatura [8] de que o RoBERTa produz resultados condizentes com o estado da arte atual de análise de sentimentos. Por isso, nas análises das próximas seções, passaremos a utilizar apenas o **RoBERTa**.

Focando agora no viés (TPRGap médio) das abordagens (quanto menor TPRGap, menor viés), observamos que os *Transformers* apresentam menor viés quando comparados aos métodos de CAT tradicionais. Para os classificadores tradicionais, o método que obteve os melhores valores foi o SVM com um TPRGap médio de 0.395, o que é mais que o dobro do TPRGap médio dos métodos baseados em *Transformers* que conseguiram 0.159 (RoBERTa), 0.153 (BART) e 0.196 (BERT). Esse resultado é muito interessante e até onde sabemos não foi reportado na literatura - *a boa capacidade dos Transformers de lidar com desbalanceamento de dados.*

Apesar disso, os *Transformers* ainda apresentam resultados de TPRGap alto em bases de dados que têm um desbalanceamento elevado, como é o caso das bases J, K e L, com RD igual a 5.32, 6.60 e 39.71, respectivamente. Isso nos mostra que ainda há espaço para melhorias, isto é, espaço para usar técnicas capazes de reduzir o viés dos modelos *Transformers*.

Portanto, os classificadores baseados em *Transformers* conseguem gerar modelos que, além da efetividade estado da arte, apresentam um viés consideravelmente menor quando comparados aos classificadores tradicionais. Além disso, observamos que, mesmo para os *Transformers*, ainda há espaço considerável de melhoria. Esse espaço é explorado a seguir.

5.2 PP2: Efetividade e Viés de CAT com undersampling

Na Tabela 3, apresentamos os resultados de efetividade obtidos por meio da aplicação dos métodos de US juntamente com o classificador RoBERTa. A coluna "NoUnder" apresenta o resultado sem o *undersampling* do conjunto de treinamento. Um ponto importante é que, para todos os métodos que permitem hiperparametrização no que tange a quantidade de instâncias a serem removidas (UBR, E2SC, E2SC_RL, NM1, NM2, IHT e CC_NN), limitamos a remoção de no máximo 50% da base de dados, pois trabalhos recentes relacionados à seleção de instâncias [6] apontam que esse é o limite empírico de redução onde ainda é possível não ocorrer perdas na efetividade. Os outros métodos não foram modificados, seguindo a política de remoção própria.

Podemos observar que os métodos UBR, NM1, E2SC_RL, NM2, E2SC, TL e OSS conseguem empate estatístico com a classificação sem *undersampling* (i.e., com o treino completo desbalanceado) em todas as coleções analisadas. Isso demonstra que todas as técnicas listadas acima tem a capacidade de balancear a base sem causar perdas de efetividade. Os demais métodos não obtiveram bons resultados em relação aos anteriores, perdendo em 4 (IHT), 3 (OBU, SBC, RENN e ALLKNN), 2 (NM3) ou em 1 (NCR, ENN) base(s), respectivamente. Os métodos CNN e CC_NN são estatisticamente equivalentes ao US em 11 de 12 bases de dados, porém, no maior conjunto de dados (L), ambos tiveram um tempo de *undersampling* que ultrapassou o tempo de treinamento do modelo de classificação sem o *undersampling*, sendo, portanto, desconsiderados para essa análise devido a sua impraticabilidade.

Na Tabela 4 apresentamos os resultados para a métrica TPRGap, a qual mede o viés dos modelos. A coluna "NoUnder" apresenta o resultado sem o *undersampling*, enquanto que as demais apresentam o TPRGap dos modelos com *undersampling*. As cores do fundo das células representam o quanto os modelos conseguiram reduzir o viés do modelo comparados ao "NoUnder". Ou seja, quanto maior o tom de verde, maior a redução do viés, e quanto mais vermelho,

dataset	NoUnder	UBR	NM1	E2SC_RL	NM2	E2SC	NM3	IHT	OBU	SBC	NCR	TL	OSS	RENN	ALLKNN	ENN	CNN	CC_NN
A	88.6(0.7)	88.6(0.8)	88.9(0.8)	88.8(1.1)	89.0(0.8)	88.6(1.2)	88.5(1.0)	87.7(1.2)	85.1(1.4)	87.6(1.5)	87.2(2.0)	**89.2(1.2)**	88.3(0.6)	85.6(0.9)	87.2(2.1)	85.6(0.9)	88.2(1.5)	88.7(1.4)
B	89.0(0.7)	88.7(1.1)	88.3(1.2)	89.1(1.1)	88.2(0.7)	88.6(1.4)	51.3(13.5)	87.8(1.5)	85.6(1.4)	87.9(1.2)	89.4(1.4)	88.7(0.9)	88.7(0.9)	83.1(8.9)	70.2(5.1)	83.4(9.0)	88.1(2.0)	**90.0(1.2)**
C	93.3(1.1)	94.3(1.4)	94.2(1.5)	93.4(1.3)	94.1(1.1)	93.8(1.6)	93.9(1.4)	92.0(1.2)	92.3(1.6)	89.3(3.3)	92.4(1.4)	93.4(1.7)	94.3(1.5)	92.5(1.7)	91.7(2.0)	92.5(1.7)	93.7(1.1)	**94.5(1.4)**
D	**89.3(1.2)**	87.6(1.5)	88.0(1.5)	88.7(1.7)	86.3(1.5)	88.1(1.7)	87.7(2.0)	84.3(3.2)	80.7(1.6)	87.7(1.2)	86.7(1.5)	88.4(1.5)	89.1(1.4)	83.4(2.1)	83.6(3.4)	83.4(2.1)	88.2(1.0)	81.7(13.8)
E	89.7(1.9)	87.9(1.6)	88.4(1.8)	89.4(1.7)	89.3(1.9)	89.1(1.9)	89.0(1.7)	82.3(1.4)	88.2(2.3)	79.2(4.2)	68.7(7.2)	**89.9(1.6)**	**89.9(1.6)**	55.2(3.5)	58.6(3.5)	55.2(3.5)	89.6(1.5)	89.3(1.6)
F	87.3(3.4)	82.3(3.5)	83.1(4.4)	85.4(3.5)	86.7(3.7)	**88.6(3.6)**	86.3(2.9)	80.6(2.2)	77.1(5.1)	86.9(3.0)	87.4(2.8)	88.4(4.0)	88.2(3.8)	81.6(3.1)	84.6(5.2)	87.7(3.6)	86.7(3.2)	84.1(3.2)
G	**94.2(1.0)**	92.7(0.9)	92.6(1.1)	93.1(1.2)	92.5(1.2)	92.6(1.1)	92.9(1.3)	87.9(1.1)	86.7(2.4)	92.0(0.9)	93.2(1.0)	93.4(1.4)	93.8(1.1)	91.5(1.0)	93.1(0.9)	92.9(0.8)	93.0(1.0)	93.0(1.0)
H	90.1(1.5)	88.6(1.6)	89.7(2.1)	88.5(1.5)	89.2(1.8)	89.3(1.8)	88.9(1.6)	83.0(2.0)	88.3(1.7)	88.1(0.9)	90.1(1.5)	90.3(1.1)	**90.6(1.6)**	86.8(1.9)	88.7(1.8)	89.2(1.3)	89.8(1.8)	88.8(1.5)
I	83.8(5.0)	82.9(4.5)	80.5(4.3)	80.6(4.7)	81.3(4.1)	81.3(4.7)	81.0(3.9)	74.1(4.5)	77.6(3.3)	81.0(4.6)	84.0(4.9)	**87.2(4.7)**	85.4(3.7)	75.8(4.4)	77.3(5.5)	81.2(5.3)	83.0(5.0)	82.4(4.6)
J	83.2(3.4)	80.1(4.2)	81.0(4.7)	81.5(5.2)	80.8(5.5)	82.9(5.0)	79.4(5.3)	74.5(3.7)	81.8(6.0)	60.2(2.9)	**84.5(4.6)**	83.2(4.8)	**84.5(4.6)**	80.7(3.9)	82.4(4.1)	84.1(3.3)	81.5(3.6)	79.9(9.5)
K	**81.0(4.5)**	79.3(3.1)	78.0(4.1)	78.7(4.2)	75.0(5.0)	79.2(4.7)	74.0(3.2)	73.3(4.7)	71.7(4.4)	71.0(4.5)	79.8(5.1)	78.7(4.6)	77.2(5.0)	79.4(5.0)	77.4(6.1)	77.8(5.9)	77.8(4.3)	76.7(3.7)
L	87.8(2.5)	81.9(4.8)	87.1(1.3)	**88.7(0.3)**	85.1(3.5)	**88.7(0.3)**	51.0(1.7)	60.3(1.5)	77.3(9.1)	30.9(1.7)	80.1(21.4)	80.0(21.4)	72.7(26.5)	69.2(3.5)	67.9(2.6)	79.8(21.2)	-	-

Tabela 3: Macro-F1 do RoBERTa utilizando as abordagens de *undersampling*. Células em negrito são os maiores valores numéricos para uma base de dados. Células em verde representam resultados que são estatisticamente equivalentes à classificação sem *undersampling* (NoUnder). Células com "-" representam métodos que resultaram em um tempo superior ao tempo de classificação da mesma base sem o *undersampling* e, portanto, desconsiderado.

dataset	NoUnder	UBR	NM1	E2SC_RL	NM2	E2SC	NM3	IHT	OBU	SBC	NCR	TL	OSS	RENN	ALLKNN	ENN	CNN	CC_NN
A	0.063	0.010	0.004	0.002	0.005	0.003	0.011	0.072	0.114	0.039	0.077	0.035	0.036	0.141	0.098	0.141	0.047	0.011
B	0.065	0.043	0.026	0.034	0.013	0.033	0.692	0.075	0.100	0.012	0.048	0.060	0.060	0.188	0.445	0.166	0.027	0.023
C	0.041	0.002	0.003	0.000	0.018	0.007	0.022	0.059	0.019	0.149	0.038	0.029	0.025	0.061	0.062	0.061	0.002	0.011
D	0.076	0.005	0.043	0.019	0.019	0.006	0.020	0.107	0.097	0.013	0.032	0.052	0.042	0.143	0.108	0.143	0.006	0.070
Média	0.061	0.015	0.019	0.014	0.014	0.012	0.186	0.078	0.083	0.053	0.049	0.044	0.041	0.133	0.178	0.128	0.021	0.029
E	0.096	0.037	0.012	0.001	0.010	0.031	0.026	0.197	0.023	0.245	0.403	0.093	0.090	0.634	0.593	0.634	0.032	0.024
F	0.126	0.006	0.047	0.029	0.054	0.025	0.036	0.132	0.144	0.024	0.043	0.108	0.134	0.111	0.034	0.010	0.000	0.011
G	0.160	0.002	0.003	0.006	0.001	0.001	0.012	0.122	0.094	0.017	0.018	0.063	0.062	0.037	0.007	0.004	0.012	0.012
H	0.102	0.030	0.000	0.006	0.021	0.020	0.001	0.158	0.007	0.035	0.028	0.085	0.077	0.072	0.021	0.007	0.016	0.024
I	0.190	0.037	0.027	0.040	0.036	0.006	0.063	0.212	0.066	0.029	0.018	0.118	0.131	0.158	0.136	0.046	0.006	0.037
Média	0.135	0.023	0.018	0.017	0.025	0.017	0.027	0.164	0.067	0.070	0.102	0.093	0.099	0.202	0.158	0.140	0.013	0.022
J	0.333	0.148	0.176	0.222	0.193	0.221	0.111	0.085	0.264	0.352	0.256	0.336	0.304	0.216	0.248	0.262	0.112	0.247
K	0.350	0.182	0.209	0.215	0.210	0.226	0.064	0.059	0.173	0.047	0.237	0.361	0.400	0.287	0.324	0.324	0.193	0.329
Média	0.342	0.165	0.192	0.218	0.202	0.224	0.087	0.072	0.219	0.199	0.247	0.349	0.352	0.251	0.286	0.293	0.153	0.288
L	0.308	0.182	0.207	0.214	0.208	0.216	0.198	0.067	0.245	0.619	0.434	0.443	0.579	0.045	0.042	0.403	-	-
Média Total	0.159	0.057	0.063	0.066	0.066	0.066	0.105	0.112	0.112	0.132	0.136	0.149	0.162	0.174	0.177	0.183	-	-

Tabela 4: TPRGap dos modelos gerados pelo classificador RoBERTa em conjunto com as abordagens de *undersampling* utilizando o classificador RoBERTa. Quanto mais verde a célula, maior a redução do viés. Quanto mais vermelho, maior o aumento do viés.

maior o agravamento do viés. Para auxiliar essa análise, dividimos nossas bases em 4 grupos com relação a seu grau de desbalanceamento (RD). No primeiro grupo são as bases de dados que têm um RD até 2 (bases A, B, C, D), o segundo contém as bases com RD maiores que 2 e menores que 5 (bases E, F, G, H e I), o terceiro aquelas com RD maior que 5 e menor que 10 (bases J e K). Por fim, no último grupo, temos apenas a base de dados L, com um RD de 39.71.

Em relação ao viés médio total calculado, os métodos que conseguiram a maior redução de viés dos modelos em todas as 12 coleções foram UBR (viés total médio de 0.057), NM1 (0.063), E2SC_RL (0.066), NM2 (0.066) e E2SC (0.066), com uma redução de quase 3 vezes comparada ao NoUnder (0.159).

Os métodos RENN, ALLKNN, ENN, que já estão entre os piores em termos de efetividade, também desempenham mal em relação ao enviesamento do modelo, aumentando o viés médio. Os métodos TL e OSS, apesar de apresentarem bons resultados em termos de efetividade, demonstram um baixo desempenho em relação ao enviesamento, piorando o modelo em alguns casos (em 3 datasets cada) e tendo um TPRGap médio total próximo ao NoUnder – 0.149 (TL) e 0.162 (OSS).

Por fim, observamos também que o método UBR – que obteve os melhores resultados quanto a média total de TRPGap – se mostra bastante eficaz individualmente por base, principalmente naquelas mais desbalanceadas. Por exemplo, nas bases do grupo 3 (bases K e J) e 4 (L), o UBR reduziu o viés do modelo NoUnder em cerca de duas vezes, estando sempre muito próximo do menor viés geral alcançado por qualquer método para essas bases. Observação similar é válida para os outros dois grupos: UBR sempre aparece entre os melhores resultados. O E2SC também é bastante competitivo, apresentando os melhores resultados médios para os grupos 1 e 2 e estando, também, entre os melhores nos demais grupos.

Sumarizando, conseguimos, com as análises reportadas acima, responder positivamente à **PP2** (*Métodos de undersampling, aplicados juntamente com classificadores baseados em Transformers, são capazes de reduzir o viés dos modelos de classificação? Qual o impacto dessa combinação na efetividade do modelo?*), pois os métodos UBR, NM1, E2SC_RL, NM2 e E2SC são capazes de significativamente reduzir o viés do modelo, sem perda de efetividade em todas as bases.

5.3 PP3: Eficiência de CAT com undersampling

Analisamos aqui os métodos de US em relação à sua eficiência, buscando verificar qual o impacto dessa nova etapa de pré-processamento dos dados no tempo total e na emissão total de CO_2 proveniente do treinamento dos modelos. A Tabela 5 apresenta o *speedup* produzido pelos métodos de US. Conforme mencionamos na Seção 4.3, o *speedup* é calculado pela razão entre o tempo total gasto na construção do modelo, mais o tempo da classificação, usando alguma abordagem de *undersampling* pelo tempo total gasto na execução (modelo e classificação) sem a fase de *undersampling*.

dataset	UBR	NM1	E2SC_RL	NM2	E2SC	NM3	IHT	OBU	SBC	NCR	TL	OSS	RENN	ALLKNN	ENN	CNN	CC_NN
A	1.209	1.076	1.094	1.135	1.178	1.096	1.181	1.374	1.216	1.210	0.929	0.961	1.376	1.230	1.415	1.243	1.193
B	1.203	1.163	1.273	1.114	1.099	1.588	1.230	1.342	1.364	0.988	0.962	0.990	1.513	1.452	1.476	1.005	1.186
C	1.315	1.237	1.400	1.096	1.253	1.247	1.115	1.214	1.638	1.223	0.873	0.885	1.039	1.120	1.070	1.361	1.202
D	1.558	1.469	1.712	1.450	1.456	1.493	1.402	1.282	1.637	1.516	1.185	1.150	1.854	1.447	1.966	1.519	1.500
E	1.450	1.312	1.569	1.312	1.361	1.400	1.182	1.381	1.769	1.607	0.937	0.948	1.611	1.641	1.663	1.176	1.467
F	1.534	1.299	1.371	1.450	1.666	1.544	1.627	1.491	1.507	1.413	1.068	1.065	1.416	1.632	1.504	1.601	1.529
G	1.527	1.530	1.518	1.374	1.354	1.414	1.298	1.329	1.444	1.122	0.946	0.995	1.220	1.159	1.270	1.310	1.521
H	1.749	1.659	1.950	1.619	1.788	1.757	1.668	1.607	1.728	1.319	1.055	1.067	1.771	1.587	1.395	1.494	1.867
I	1.657	1.923	1.635	1.601	1.609	1.519	1.634	1.331	1.564	1.401	1.034	1.014	1.516	1.751	1.537	1.510	1.612
J	1.487	1.730	1.617	1.523	1.876	2.128	1.535	1.302	2.861	0.877	1.003	0.800	0.998	0.977	0.990	2.283	1.608
K	1.695	1.691	1.802	1.621	1.741	3.054	1.602	1.411	3.433	1.487	1.107	0.972	1.220	1.341	1.335	2.377	1.442
L	2.662	2.709	2.903	2.867	3.039	39.486	2.103	1.777	12.498	1.038	0.892	1.318	0.946	1.713	1.230	-	-
Média	1.587	1.566	1.654	1.513	1.618	4.811	1.465	1.404	2.722	1.267	0.999	1.014	1.373	1.421	1.404	-	-

Tabela 5: Resultados de Speedup no custo total (tempo) para geração dos modelos utilizando o classificador RoBERTa em conjunto com as abordagens de *undersampling*. Quanto mais verde a célula, maior a redução no tempo total de treinamento em relação a abordagem sem *undersampling*. Quanto mais vermelho, maior o tempo.

dataset	NoUnder	UBR	NM1	E2SC_RL	NM2	E2SC	NM3	IHT	OBU	SBC	NCR	TL	OSS	RENN	ALLKNN	ENN	CNN	CC_NN
A	55.537	45.884	51.580	50.740	48.897	47.114	50.650	46.966	40.362	43.922	45.838	59.770	57.725	40.322	45.119	39.218	40.892	44.769
B	83.539	69.377	71.793	65.550	74.933	75.923	52.539	67.865	62.154	59.487	84.470	86.764	84.280	55.137	57.481	56.526	66.378	66.234
C	34.265	26.042	27.693	24.463	31.241	27.324	27.460	30.724	28.192	20.567	27.991	39.227	38.706	32.962	30.580	32.013	24.429	27.903
D	93.391	59.893	63.529	54.531	64.380	64.097	62.535	66.566	72.778	56.550	61.574	78.756	81.160	50.329	64.525	47.462	57.948	61.228
E	55.470	38.202	42.243	35.322	42.254	40.692	39.576	46.877	40.118	30.730	34.474	59.146	58.448	34.394	33.765	33.315	41.039	35.803
F	36.458	23.745	28.052	26.563	25.116	21.848	23.591	22.375	24.414	23.945	25.780	34.099	34.192	25.725	22.307	24.220	22.403	23.493
G	171.805	112.413	112.145	113.060	124.901	126.778	121.423	132.204	129.121	116.158	152.980	181.434	172.539	140.629	148.092	135.182	117.260	108.297
H	78.329	44.724	47.164	40.116	48.316	43.737	44.516	46.895	48.635	41.999	59.281	74.156	73.297	44.120	49.254	56.096	43.587	39.587
I	22.538	13.581	11.704	13.769	14.060	13.986	14.816	13.768	16.906	14.135	16.061	21.784	22.209	14.847	12.854	14.647	14.468	13.676
J	21.994	14.773	12.700	13.592	14.429	11.708	10.321	14.307	16.869	7.486	25.071	21.912	27.473	22.023	22.498	22.193	9.291	13.296
K	23.693	13.934	13.980	13.114	14.579	13.568	7.724	14.747	16.740	6.504	15.897	21.369	24.348	19.383	17.626	17.710	9.510	15.948
L	2,552.335	942.877	938.264	876.957	879.055	834.978	60.623	1,209.117	1,414.486	117.465	2,324.505	2,760.191	1,831.865	1,210.973	1,245.247	1,943.847	-	-

Tabela 6: Emissão de Carbono (CO_2) para geração dos modelos utilizando o classificador RoBERTa em conjunto com as abordagens de *undersampling*. Quanto mais verde a célula, maior a redução da emissão em relação a abordagem sem *undersampling*.

Podemos observar que, com exceção do TL, todos os métodos, em média, conseguiram manter ou reduzir o tempo em comparação com o RoBERTa aplicado aos dados originais (sem *undersampling*). Os métodos UBR, NM1, E2SC_RL, NM2 e E2SC, que nas análises anteriores se mostraram superiores quantos aos critérios de efetividade e redução do enviesamento, conseguem também um bom desempenho no critério de eficiência. Os *speedups* alcançados, respectivamente, de 1.587, 1.566, 1.654, 1.513 e 1.618 são muito bons. Juntamente com o NM3 (4.811) e SBC (2.722), esses são os métodos com maior ganho de *speedup*. Contudo, vale lembrar que o NM3 e o SBC geram perdas de efetividade para algumas coleções (Tabela 3) ao produzir os respectivos ganhos de *speedup*.

Por fim, na Tabela 6, apresentamos os valores de emissão de CO_2 (em g) produzido pelos métodos de *undersampling*. Assim como causaram perda de eficiência, os métodos TL e OSS também produzem um aumento de emissão de CO_2 para algumas bases. Já todos os demais métodos, em menor ou maior grau, geram alguma redução de emissão. Para as bases de dados de A a K, a emissão de carbono é muito pequena quando comparada a base L, que é ordens de magnitude maior que as demais. Por essa mesma razão, o tempo de processamento de L é muito maior, mesmo ela sendo executada em uma máquina de maior poder computacional. Por isso, nossa análise nesse critério focará nessa base. Ao analisar a emissão dos métodos de *undersampling* para a base de dados L, observamos que os métodos UBR (emissão 942.877 g CO_2), NM1 (938.264 g CO_2), E2SC_RL (876.957 g CO_2), NM2 (879.055 g CO_2) e E2SC (834.978 g CO_2), os mesmos métodos destacados nas análises anteriores, também conseguem reduzir mais que pela metade a emissão em comparação com o NoUnder (2,552.335 g CO_2). Traduzindo esses números para exemplos ilustrativos, podemos dizer que a emissão antes era equivalente a 2.73 meses de uma árvore sequestrando carbono ou a emissão de 14.40 km percorridos por um carro [21]. Já a emissão após o *undersampling*, considerando o UBR como exemplo, seria equivalente a 1.03 meses de sequestro de carbono feito por uma árvore ou a emissão de de 5.41 Km percorridos por um carro de passageiros. Se individualmente essa parece ser uma redução pequena, se considerarmos milhões de máquinas ao redor do mundo rodando processos similares todos os dias, dezenas ou centenas de vezes, essa redução pode passar a ser considerável.

Voltando à nossa **PP3** (*Qual o impacto da aplicação dessa etapa adicional de pré-processamento (undersampling) em termos de eficiência? E em termos da emissão de carbono?*), temos que os métodos UBR, NM1, E2SC_RL, NM2 e E2SC conseguem reduzir o enviesamento do modelo, mantendo a efetividade e reduzindo o tempo de treinamento dos modelos e, consequentemente, as emissões de CO_2.

5.4 Discussão Final

Dadas as análises apresentadas nesta seção, onde os métodos de *undersampling* foram avaliados quanto à sua efetividade, sua capacidade reduzir o enviesamento dos modelos e sua eficiência, podemos concluir que os melhores métodos de *undersampling* analisados foram UBR, NM1, E2SC_RL, NM2 e E2SC. Todos eles conseguiram reduzir o enviesamento do modelo para a classe majoritária sem

produzir nenhum tipo de perda de efetividade global (em termos de MacroF1). Esses métodos também apresentaram uma redução no tempo de treinamento do modelo com uma consequente redução na emissão de CO_2. Das estratégias que se destacaram, **duas** foram originalmente propostas neste trabalho - UBR e E2SC_RL. Em particular a **UBR** foi aquela que apresentou os resultados mais consistentes de redução de enviesamento sem perda de efetividade, com um bom speedup e redução de emissão de CO_2, sendo nossa recomendação final para o problema, se tivermos de fazer uma.

6 CONCLUSÕES E TRABALHOS FUTUROS

O impacto do desbalanceamento de classes, relacionado ao viés de um classificador para a classe majoritária, em estratégias do estado da arte de CAT baseadas em *Transformers*, tem sido pouco discutido na literatura [5]. Neste trabalho, apresentamos uma avaliação detalhada de métodos de *undersampling* (US) aplicados em conjunto com algoritmos baseados em *Transformers* na tarefa de Análise de Sentimento. Primeiramente, realizamos uma análise comparativa entre métodos de classificação baseados em *Transformers* e tradicionais sob duas perspectivas: efetividade e enviesamento. Essa análise revelou que, além de mais efetivos, como conhecidos, os *Transformers* são capazes de lidar de forma mais adequada com o problema do viés que os algoritmos tradicionais. Nossos resultados experimentais também indicaram que ainda existe espaço para melhoria, principalmente em bases de dados com um desbalanceamento maior.

Baseado em um mapeamento da literatura sobre *undersampling*, selecionamos e implementamos os 14 métodos mais utilizados, além de adaptarmos diretamente um método de seleção de instâncias para a tarefa de *undersampling* devido a conexões entre as tarefas. Propomos, também, duas novas estratégias de US: E2SC_RL e UBR, totalizando em um conjunto de 17 métodos a serem comparados.

Uma avaliação experimental vasta, utilizando esses 17 métodos e 12 bases de dados revelou que um conjunto de cinco métodos de *undersampling* – UBR (nossa proposta), NM1, E2SC_RL (nossa proposta), NM2, E2SC – foram capazes de reduzir o viés dos modelos de CAT quando comparados com modelos sem *undersampling* sem perdas de efetividade (qualidade da classificação). Além disso, esses mesmos métodos produziram uma redução significativa no tempo de treino e na emissão de CO_2 no treinamento dos modelos *Transformers*. Entre esses 5 métodos, o UBR apresentou os resultados mais consistentes considerando todos os critérios analisados.

Como trabalhos futuro, visamos estender o presente estudo considerando, também, outros cenários de CAT, como multiclasses e/ou hierárquico, além de avaliar outros algoritmos de CAT além do RoBERTa, tais como BERT e BART. Ademais, considerando que a eficácia dos LLMs recentes em comparação com modelos anteriores baseados em *Transformer*, como o RoBERTa, para análise de sentimentos e propósitos de CAT ainda não está clara [10], e que, quando LLMs superam alguns *Transformers* de 1ª e 2ª geração, os ganhos são tipicamente de apenas alguns pontos percentuais [10], consideramos incerto se esses ganhos marginais se traduzem em benefícios práticos em aplicações do mundo real. Desta forma, em trabalhos futuros, planejamos realizar uma análise completa sobre o custo-benefício dos LLMs em relação aos *Transformers* de 1ª e 2ª geração, de modo a habilitar a aplicação dos métodos de *undersampling* como etapas de pré-processamento de LLMs recentes.

AGRADECIMENTOS

Este trabalho foi financiado por CNPq, CAPES, Fapemig, FAPESP, CIIA-Saúde e AWS.

REFERENCES

[1] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051* (2020).
[2] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. 1992. A training algorithm for optimal margin classifiers. In *5th COLT*.
[3] Leo Breiman. 2001. Random forests. *Machine learning* (2001).
[4] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd KDD*.
[5] Washington Cunha, Sérgio D. Canuto, Felipe Viegas, Thiago Salles, Christian Gomes, Vítor Mangaravite, Elaine Resende, Thierson Rosa, Marcos André Gonçalves, and Leonardo Rocha. 2020. Extended pre-processing pipeline for text classification: On the role of meta-feature representations, sparsification and selective sampling. *IP&M*. (2020).
[6] Washington Cunha, Celso França, Guilherme Fonseca, Leonardo Rocha, and Marcos André Gonçalves. 2023. An Effective, Efficient, and Scalable Confidence-Based Instance Selection Framework for Transformer-Based Text Classification. In *the 46th ACM SIGIR*.
[7] Washington Cunha, Vítor Mangaravite, Christian Gomes, Sérgio Canuto, Felipe Viegas, Celso França, Wellington Santos Martins, Jussara M Almeida, et al. 2021. On the cost-effectiveness of neural and non-neural approaches and representations for text classification: A comprehensive comparative study. *IP&M* (2021).
[8] Washington Cunha, Felipe Viegas, Celso França, Thierson Rosa, Leonardo Rocha, and Marcos André Gonçalves. 2023. A Comparative Survey of Instance Selection Methods applied to NonNeural and Transformer-Based Text Classification. *Comput. Surveys* (2023).
[9] Paula Czarnowska, Yogarshi Vyas, and Kashif Shah. 2021. Quantifying social biases in NLP: A generalization and empirical comparison of extrinsic fairness metrics. *TACL* (2021).
[10] Claudio MV de Andrade, Fabiano M Belém, Washington Cunha, Celso França, Felipe Viegas, Leonardo Rocha, and Marcos André Gonçalves. 2023. On the class separability of contextual embeddings representations–or "The classifier does not matter when the (text) representation is so good!". *IP&M* (2023).
[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[12] Georgios Douzas, Maria Lechleitner, and Fernando Bacao. 2022. Improving the quality of predictive models in small data GSDOT: A new algorithm for generating synthetic data. *Plos one* (2022).
[13] Vinicius HS Durelli, Rafael S Durelli, Andre T Endo, Elder Cirilo, Washington Luiz, and Leonardo Rocha. 2018. Please please me: does the presence of test cases influence mobile app users' satisfaction?. In *Proceedings of the XXXII Brazilian Symposium on Software Engineering*.
[14] Xavier Ferrer, Tom van Nuenen, Jose M. Such, Mark Coté, and Natalia Criado. 2021. Bias and Discrimination in AI: A Cross-Disciplinary Perspective. *IEEE Technology and Society Magazine* (2021).
[15] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. 2005. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*. Springer.
[16] Xiao Han, Yuqi Liu, and Jimmy Lin. 2021. The simplest thing that can possibly work:(pseudo-) relevance feedback via text classification. In *Proceedings of the 2021 ACM SIGIR ICTIR*.
[17] Peter Hart. 1968. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory* (1968).
[18] Antônio Júnior, Pablo Cecilio, Felipe Viegas, Washington Cunha, Elisa Albergaria, and Leonardo Rocha. 2022. Evaluating topic modeling pre-processing pipelines for portuguese texts. In *WebMedia*.
[19] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in NeurIPS* (2017).
[20] Miroslav Kubat, Stan Matwin, et al. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*. Citeseer.
[21] Loïc Lannelongue, Jason Grealey, and Michael Inouye. 2021. Green algorithms: quantifying the carbon footprint of computation. *Advanced science* (2021).
[22] Jorma Laurikkala. 2001. Improving identification of difficult small classes by balancing class distribution. In *8th Conference on AIME*.
[23] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

[24] Wei-Chao Lin, Chih-Fong Tsai, Ya-Han Hu, and Jing-Shang Jhang. 2017. Clustering-based undersampling in class-imbalanced data. *Info. Sciences* (2017).
[25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
[26] Hongxia Lu, Louis Ehwerhemuepha, and Cyril Rakovski. 2022. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. *BMC medical research methodology* (2022).
[27] Washington Luiz, Felipe Viegas, Rafael Alencar, Fernando Mourão, Thiago Salles, Dárlinton Carvalho, Marcos Andre Gonçalves, and Leonardo Rocha. 2018. A Feature-Oriented Sentiment Rating for Mobile App Reviews. In *the World Wide Web Conference (WWW '18)*.
[28] Inderjeet Mani and I Zhang. 2003. kNN approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*. ICML.
[29] Luiz Felipe Mendes, Marcos Gonçalves, Washington Cunha, Leonardo Rocha, Thierson Couto-Rosa, and Wellington Martins. 2020. " Keep it Simple, Lazy"–MetaLazy: A New MetaStrategy for Lazy Text Classification. In *Proceedings of the 29th ACM International CIKM*.
[30] Andrew Ng. 2017. Machine learning yearning. *URL: http://www. mlyearning. org/(96)* (2017).
[31] Albert Orriols-Puig and Ester Bernadó-Mansilla. 2009. Evolutionary rule-based systems for imbalanced data sets. *Soft Computing* (2009).
[32] Andrea Pasin, Washington Cunha, Marcos André Gonçalves, and Nicola Ferro. 2024. A Quantum Annealing Instance Selection Approach for Efficient and Effective Transformer Fine-Tuning. In *ICTIR*.
[33] Alexander Ponomarenko, Nikita Avrelin, Bilegsaikhan Naidan, and Leonid Boytsov. 2014. Comparative analysis of data structures for approximate nearest neighbor search. *Data analytics* (2014).
[34] Sivaramakrishnan Rajaraman, Prasanth Ganesan, and Sameer Antani. 2022. Deep learning model calibration for improving performance in class-imbalanced medical image classification tasks. *PloS one* (2022).
[35] Michael R Smith, Tony Martinez, and Christophe Giraud-Carrier. 2014. An instance level analysis of data complexity. *Machine learning* (2014).
[36] Ivan Tomek. 1976. An experiment with the edited nearest-neighbor rule. (1976).
[37] Ivan Tomek. 1976. Two Modifications of CNN. *IEEE Transactions on Systems, Man, and Cybernetics* (1976).
[38] Pattaramon Vuttipittayamongkol, Eyad Elyan, Andrei Petrovski, and Chrisina Jayne. 2018. Overlap-based undersampling for improving imbalanced data classification. In *19th IDEAL*. Springer.
[39] Dennis L Wilson. 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems* (1972).
[40] Raymond E Wright. 1995. Logistic regression. (1995).
[41] Show-Jane Yen and Yue-Shi Lee. 2006. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *ICIC Kunming, China, August 16–19, 2006*. Springer.
[42] Bruna Stella Zanotto, Ana Paula Beck da Silva Etges, Avner Dal Bosco, Eduardo Gabriel Cortes, Renata Ruschel, Washington Luiz, et al. 2021. Stroke outcome measurements from electronic medical records: cross-sectional study on the effectiveness of neural and nonneural classifiers. *JMIR Medical Informatics* (2021).