Análise Comparativa de Métodos de Undersampling em Classificação Automática de Texto Baseada em Transformers

Guilherme Fonseca[1], **Washington Cunha**[2], **Leonardo Rocha**[1]

[1] Universidade Federal de São João del Rei, Brasil
[2] Universidade Federal de Minas Gerais, Brasil

guilhermefonseca8426@aluno.ufsj.edu.br, lcrocha@ufsj.edu.br
washingtoncunha@dcc.ufmg.br

Abstract. *Automatic Text Classification (ATC) in unbalanced databases is a common challenge in real-world applications. In this scenario, one of the classes is underrepresented, which could cause a bias in the learning process. This work investigates the effect of undersampling methods, which aim to reduce instances of the majority class, on the performance of recent ATC strategies based on transformers. We evaluated 15 existing undersampling strategies and one proposal in this work. Our results suggest that undersampling approaches are important for improving the performance of classification methods on imbalanced collections, not only reducing learning bias but also reducing training costs.*

Resumo. *Classificação Automática de Texto (CAT) em bases de dados desbalanceadas é um desafio comum em aplicações do mundo real. Nesse cenário, uma das classes é sub-representada, podendo provocar um viés no processo de aprendizado. Este trabalho investiga o efeito de métodos de undersampling, que visam reduzir instâncias da classe majoritária, no desempenho de estratégias de CAT recentes, baseada em transformers. Avaliamos 15 estratégias existentes de undersampling e uma proposta nesse trabalho. Nossos resultados sugerem que as abordagens de undersampling são importantes para melhorar o desempenho de métodos de classificação em coleções desbalanceadas, não apenas reduzindo o viés de aprendizado, mas também reduzindo o custo de treinamento.*

1. Introdução

A Classificação Automática de Texto (CAT) tem sido utilizada como uma importante ferramenta para a análise e organização de grandes volumes de documentos por serem capazes de mapear documentos textuais em um conjunto de categorias pré-definidas. A área de CAT tem experimentado uma grande evolução nos últimos anos, com ênfase em estratégias supervisionadas motivadas principalmente por avanços recentes em aprendizagem profunda baseadas em *transformers* [Devlin et al. 2018]. 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.

Dentre os desafios relacionados a essas abordagens, podemos citar: (i) a necessidade de grandes quantidades de dados classificados manualmente para realizar uma aprendizagem eficaz; e (ii) escalabilidade das soluções diante de coleções com milhões, às vezes bilhões, de documentos. Uma das principais razões para o sucesso das estratégias de CAT baseadas em *transformers* é que se beneficiarem em termos de desempenho

(eficácia) à medida que os dados aumentam, tornando-se mais eficazes do que os métodos tradicionais de aprendizado de máquina [Ng 2017]. Por isso, criou-se um senso comum de que os próprios algoritmos atuais são capazes de contornar problemas tradicionais de classificação, como o viés de aprendizado oriundo de coleções de dados desbalanceadas.

A desigualdade nas distribuições de classes em bases de dados é uma questão comum e que, ao longo da história de CAT, vem prejudicando o desempenho dos modelos de aprendizado de máquina, em que as classes minoritárias podem ser sub-representadas, o que resulta em baixa capacidade de generalização para essas classes e gera viés no processo de geração modelo. Existem duas principais abordagens utilizadas para lidar com o desbalanceamento de dados. Uma delas é o *oversampling*, que consiste em criar amostras da classe minoritária para igualar-se à classe majoritária [Han et al. 2005]. A outra abordagem utilizada para enfrentar o problema do desbalanceamento é o *undersampling* (foco do presente projeto), que consiste em técnicas destinadas a reduzir instâncias da classe majoritária para equilibrar as classes. Atualmente, não existem estudos na literatura que abordem como os métodos de *undersampling* interagem com os algoritmos de CAT mais atuais baseados em aprendizado profundo e *transformers*. Elaboramos a hipótese **HP1** de que *classificadores de aprendizado profundo baseados em transformers não conseguem, por si só, resolver totalmente o problema de bases de dados desbalanceadas, o viés gerado pelo aprendizado desigual entre as classes*.

A **primeira contribuição** do trabalho é um mapeamento completo dos principais métodos de *undersampling*. Identificamos e implementamos 13 métodos que estão entre os mais utilizados na literatura. Apesar de terem fins diferentes, as áreas de seleção de instâncias (SI) e *undersampling* são bastante parecidas, pois ambas tratam de técnicas que visam selecionar um subconjunto de um conjunto de dados. Identificamos que em [Cunha et al. 2023a] foram adaptadas estratégias de *undersampling* para o problema de SI, sendo assim adaptamos duas estratégias de SI para o cenário de *undersampling*. Como a **segunda contribuição**, propomos uma nova abordagem de *undersampling* – totalizando 16 métodos. Como **terceira contribuição**, investigamos o desempenho das técnicas de *undersampling* em conjunto com o RoBERTa, classificador baseado em *transformers* considerado estado da arte. Consideramos três perspectivas: (1) **eficácia** da classificação; (2) **eficiência** (tempo); e (3) capacidade de generalização dos modelo (**viés**). Restringimos nosso foco a tarefas de classificação binária e de rótulo único em documentos textuais. Essa limitação foi estabelecida devido à predominância das técnicas de *undersampling* em cenários de classificação binária. Os melhores métodos de *undersampling* analisados foram UBR (nossa proposta), NM2, E2SC_RL, RUS e E2SC. Todos eles conseguiram, em termos de eficácia, manter-se estatisticamente iguais ao modelo treinado sem os métodos de *undersampling*. Além disso, conseguiram reduzir o viés do modelo, permitindo que aprendesse igualmente bem sobre todas as classes. Esses métodos também apresentaram uma significativa redução no tempo de treinamento do modelo. Assim, concluímos que as abordagens de *undersampling* são importantes para melhorar o desempenho de métodos de classificação em coleções desbalanceadas, não apenas reduzindo o viés de aprendizado, mas também reduzindo o custo de treinamento.

As implementações e experimentações foram realizadas pelo aluno Guilherme, sob a orientação do prof. Leonardo. A concepção do projeto e as análises de resultados foram feitas em conjunto, aluno e professor, com a colaboração do doutorando Washington.

2. Levantamento das Estratégias

Nessa seção, detalhamos o processo de análise sistemática da literatura para selecionar quais estratégias de *undersampling* e seleção de instâncias que serão avaliadas. Como primeira contribuição desse trabalho, implementamos todas as estratégias.[1]

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 *undersampling*.

Analisamos manualmente dos 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, sendo a estratégia randômica foi a mais utilizada (90% dos trabalhos relevantes). Uma tabela completa com uma descrição de todas as estratégias identificadas está disponibilizado online[2]. Por fim, de todos os métodos, optamos por considerar em nossa avaliação aqueles que foram utilizados em mais de um dos trabalhos relevantes. Faremos todas as análises sobre os seguintes métodos *undersampling*:

- Randômico (RUS): exemplos da classe majoritária são aleatoriamente removidos;

- Links de Tomek (TL) [Tomek 1976b]: dados dois exemplos e_i e e_j pertencentes a 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 é ruído ou ambos são exemplos fronteiriços e podem ser removidos;

- Condensed Nearest Neighbors (CNN)[Hart 1968]: 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 [3] 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)[Kubat et al. 1997]: Combina o TL e CNN. o TL é utilizado para identificar pares ambíguos na fronteira da classe, os quais são removidos na classe majoritária. O método CNN é então utilizado para eliminar exemplos redundantes da classe majoritária.

- Edited Nearest Neighbours (ENN)[Wilson 1972]: 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.

[1] `https://github.com/guilherme8426/Undersampling`

[2] `https://encurtador.com.br/dpH17`

[3] O KNN estima a classe de uma instância de acordo com seus k vizinhos mais próximos.

- Repeated Edited Nearest Neighbours (RENN)[Tomek 1976a]: O algoritmo ENN é aplicado sucessivamente até que não seja possível remover mais pontos.

- ALL k-NN (ALLKNN)[Tomek 1976a]: aplica o método ENN sucessivamente, mas com o diferencial que a cada passagem do ENN o número de vizinhos a serem considerados aumenta.

- Neighbourhood Cleaning Rule (NCL)[Laurikkala 2001]: 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 NCL classifica também 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)[Mani and Zhang 2003]: 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) [Yen and Lee 2006]: 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) [Smith et al. 2014]: Este algoritmo 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 a classe y_i. O IHT seleciona amostras da classe majoritária com baixa probabilidade de pertencerem à classe majoritária para serem removidas.

- (CC-NN) [Lin et al. 2017]: 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 centroide de cada um dos *clusters* que pertença a classe majoritária é escolhido para compor, junto com as instâncias da classe minoritária, o conjunto final.

- (OBU) [Vuttipittayamongkol et al. 2018]: 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 do método).

2.2. Métodos de seleção de instância

Apesar de terem fins diferentes, as áreas de seleção de instâncias (SI) e *undersampling* são bastante parecidas, pois ambas tratam de técnicas que visam selecionar um subconjunto de um conjunto de dados de tal forma que, quando treinado com o subconjunto selecionado, o modelo cumpra seu objetivo (melhorar a eficácia mantendo a eficiência em SI e melhorar a eficiência e a generalização do modelo em *undersampling*). Portanto, em nosso trabalho, também selecionamos trabalhos de SI e adaptamos para *undersampling*.

Realizando uma busca na literatura, encontramos o trabalho [Cunha et al. 2023a] onde é proposto o método E2SC que se apresenta como o estado-da-arte em SI por meio de uma comparação bem vasta e completa com diversas estratégias de SI da literatura. O método funciona em duas etapas. Na primeira etapa, calcula as probabilidades de cada instância ser removida. Estas probabilidades são obtidas através da confiança

de um classificador calibrado (KNN). 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 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 porque a regressão logística é um classificador igualmente calibrado e possui baixo custo computacional, semelhante ao KNN . Portanto, consideramos em nossos experimentos E2SC e E2SC_RL. Além de todos as estratégias apresentadas nessa seção, apresentamos a seguir nossa nova proposta de estratégia de *undersampling* que também será considerada em nossa avaliação.

3. Método proposto

Nesta seção, apresentamos nossa segunda contribuição nesse trabalho, uma nova abordagem denominada de UBR (Undersampling Baseado em Redundância), que se concentra em retirar instâncias redundantes da classe majoritária. Um par de documentos é considerado redundante se apresenta alta similaridade entre si. Manter apenas uma das instâncias no conjunto de treinamento é suficiente, pois a presença de ambas não trará aumento significativo na aprendizagem do modelo devido a alta similaridade entre elas. O **Algoritmo 1** mostra em mais detalhes o pseudo código do UBR, seguido pela explicação detalhada de cada passo.

Algorithm 1: Algoritmo UBR

Input: X
Output: instanciasSel

$X_{Maj} \leftarrow obterInstancias(X, classe = majoritaria)$;
$X_{Min} \leftarrow obterInstancias(X, classe = minoritaria)$;
$instanciasSel \leftarrow X_{Min}$;
$similaridade \leftarrow similaridadeParAPar(X_{Maj})$;
$clusters \leftarrow X_{Maj}$;
$N \leftarrow \|X_{Maj}\| - \|X_{Min}\|$;
while $N > 0$ **do**
    $A, B \leftarrow instanciasMaisSimilares(X_{Maj}, similaridade)$;
    if $clusters[A] \neq clusters[B]$ **then**
        $clusters[A] \leftarrow clusters[A] \bigcup clusters[B]$;
        $deletar(cluster[B])$;
        $N -= 1$;
    end
end
$representantes \leftarrow escolheRepresentateParaCadaCluster(clusters)$;
$instanciasSel \leftarrow instanciasSel \bigcup representantes$;

Dado X como o conjunto total de instâncias de treinamento, nossa abordagem inicialmente divide 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. Em seguida, calculamos a similaridade de cosseno par a par dentro do conjunto X_{Maj} e passamos a considerar cada instância de X_{Maj} como um *cluster* individual. Posteriormente, realizamos N iterações, onde $N = \|X_{Maj}\| - \|X_{Min}\|$. Em cada iteração, buscamos o par de instâncias A e B pertencentes a X_{Maj} que apresenta a

maior similaridade e que estão em *clusters* distintos, com o objetivo de unir os *clusters* aos quais A e B pertencem. Ao final deste processo, teremos N *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.

4. Configuração Experimental

Nessa seção apresentamos o ambiente experimental que será utilizado para realizar a comparação entre todas as estratégias de *undersampling* discutidas na seções anteriores.

4.1. Bases de dados

Consideramos 13 conjuntos de dados, com diferentes níveis de desbalanceamento, conforme detalhado na Tabela 1.

Conjunto de Dados	# de Documentos	% Classe Majoritária	Nome
debate	1979	63%	A
english_dailabor	1227	60%	B
nikolaos_ted	727	56%	C
sentistrength_bbc	752	87%	D
sentistrength_digg	782	73%	E
sentistrength_mys	834	84%	F
sentistrength_rw	705	69%	G
sentistrength_twitter	2289	59%	H
sentistrength_yt	2432	68%	I
tweet_semevaltest	3060	73%	J
vader_amazon	3610	59%	K
vader_nyt	4946	55%	L
vader_twitter	4196	69%	M

Tabela 1. Detalhes das Coleções de dados utilizadas nos experimentos. A coluna "nome" contém como a base vai ser referenciada no trabalho.

4.2. Método de Classificação de Texto

Utilizaremos em nossos experimentos o método **RoBERTa**, que atualmente se apresenta como o melhor entre os métodos de classificação utilizados na literatura [Cunha et al. 2023b]. Para ajustar os hiperparâmetros, utilizamos a mesma metodologia discutida em [Cunha et al. 2023b]. 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 e 32), pois esses valores especificados impactam diretamente na eficiência e eficácia do modelo.

4.3. Métricas e Protocolo Experimental

Nossa avaliação dos métodos de *undersamplig* é feita sob três perspectivas importantes: (1) eficácia da classificação; (2) eficiência (tempo); e (3) capacidade de generalização dos modelo (viés). A eficácia é avaliada utilizando a Macro Average F1 (MacroF1). A eficiência é medida com base no custo de cada método em termos do tempo total necessário para construir o modelo. O *Speedup* é calculado como $S = \frac{T_{wo}}{T_w}$, onde T_w é o tempo total gasto na construção do modelo usando a abordagem de *undersampling*, e T_{wo} é o tempo total gasto na execução sem a fase de *undersampling*. Por fim, para analisar a capacidade de generalização dos modelos , utilizamos a métrica TPRGap apresentada em [Czarnowska et al. 2021] e definida na equação abaixo, 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 $\binom{T}{2}$.

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

Os experimentos foram realizados na AWS. As estratégias de *undersampling*, que demandam estritamente de processamento em CPU, utilizaram uma instância do tipo **c6a.4xlarge** e a classificação, que demanda de hardware especializado (GPU), instâncias do tipo **g4dn.xlarge**. Todas as bases de dados foram divididas utilizando o método de validação cruzada com 10 partições, todas as comparações foram realizadas utilizando o método estatístico Teste T com correção de Bonferroni [Cunha et al. 2023b].

5. Análise dos Resultados Experimentais

Nessa seção apresentamos os resultados da análise comparativa entre as estratégias de *undersampling* aplicadas em algoritmos de aprendizado profundo para classificação automática de texto, nossa terceira contribuição nesse trabalho.

5.1. Análise da Eficácia

Na Tabela 2 apresentamos os resultados obtidos por meio da aplicação dos métodos de *undersampling* ao classificador RoBERTa. A coluna "NoUnder" apresenta o resultado sem o *undersampling* do conjunto de treinamento. Observamos na Tabela 2 que os métodos UBR, NM2, E2SC_RL, RUS, E2SC, CNN, CC_NN, TL, OSS conseguem empate estatístico em todas as bases de dados analisadas. Além disso, os métodos OSS e TL são os que obtêm o melhor resultado nos conjuntos de dados. O método OSS alcança o melhor resultado em 4 dos 13 conjuntos de dados, enquanto o TL obtém o melhor resultado em 3 deles. Em contrapartida, os demais métodos não obtiveram bons resultados em relação aos anteriores, perdendo em 5 (IHT), 3 (OBU e RENN), 2 (ALLKNN, ENN, NCR, SBC) ou em 1 (NM1, NM3) base de dados, respectivamente. Estratégias de *undersampling* têm como objetivo mais que melhorar a eficácia do modelo, mas também aprimorar a generalização do mesmo, ou seja, a capacidade de aprendizado do modelo sobre todas as classes. Dessa forma,além da eficácia, é necessário examinar se o modelo conseguiu aprender melhor sobre a classe minoritária, conforme discutido na próxima seção.

dataset	NoUnder	UBR	NM2	E2SC_RL	RUS	E2SC	CNN	CC_NN	TL	OSS	NM1	NM3	ALLKNN	ENN	NCR	SBC	OBU	RENN	IHT
A	89.0(1.7)	87.6(1.5)	86.3(1.5)	88.7(1.7)	87.6(1.9)	88.1(1.7)	88.2(1.0)	81.7(13.8)	88.4(1.5)	**89.1(1.4)**	88.0(1.5)	87.7(2.0)	83.6(3.4)	83.4(2.1)	86.7(1.5)	87.7(1.2)	80.7(1.6)	83.4(2.1)	84.3(3.2)
B	93.3(1.1)	94.3(1.4)	94.1(1.1)	93.4(1.3)	94.1(1.2)	93.8(1.6)	93.7(1.1)	**94.5(1.4)**	93.4(1.7)	94.3(1.5)	94.2(1.5)	93.9(1.4)	91.7(2.0)	92.5(1.7)	92.4(1.4)	89.3(3.3)	92.3(1.6)	92.5(1.7)	92.0(1.2)
C	80.0(2.6)	80.0(2.3)	80.5(2.5)	80.9(3.3)	80.9(2.9)	80.3(2.9)	**82.6(1.2)**	81.3(2.0)	80.8(2.4)	80.3(2.8)	81.3(2.8)	80.6(3.2)	79.7(2.4)	76.1(2.4)	76.9(2.0)	80.1(2.7)	75.3(1.8)	76.1(2.4)	79.7(3.0)
D	**81.0(4.5)**	68.6(5.0)	68.7(4.8)	74.2(5.0)	74.3(3.7)	75.2(4.2)	77.8(4.3)	72.6(5.6)	78.7(4.6)	77.2(5.0)	67.0(4.8)	74.0(3.2)	77.4(6.1)	77.8(5.9)	79.8(5.1)	71.0(4.5)	71.7(4.4)	79.4(5.0)	58.3(4.5)
E	83.8(5.0)	82.9(4.5)	81.3(4.1)	80.6(4.7)	83.3(4.1)	81.3(4.7)	83.0(5.0)	82.4(4.6)	**87.2(4.7)**	85.4(3.7)	80.5(4.3)	81.0(3.9)	77.3(5.5)	81.2(5.3)	84.0(4.9)	81.0(4.6)	77.6(3.3)	75.8(4.4)	74.1(4.5)
F	82.8(5.1)	75.3(3.5)	80.0(4.3)	78.2(3.3)	79.5(6.6)	79.9(5.7)	81.5(3.6)	68.0(7.7)	83.2(4.8)	**84.5(4.6)**	75.6(5.0)	79.4(5.3)	82.4(4.1)	84.1(3.3)	**84.5(4.6)**	60.2(2.9)	81.8(6.0)	80.7(3.9)	63.5(4.1)
G	87.3(3.4)	82.3(3.5)	86.7(3.7)	85.4(3.5)	85.2(3.3)	**88.6(3.6)**	86.7(3.2)	84.1(3.2)	88.4(4.0)	88.2(3.8)	83.1(4.4)	86.3(2.9)	84.6(5.2)	87.7(3.6)	87.4(2.8)	86.9(3.0)	77.1(5.1)	81.6(3.1)	80.6(2.2)
H	88.6(0.7)	88.6(0.8)	89.0(0.8)	88.8(1.1)	88.7(1.6)	88.6(1.2)	88.2(1.5)	88.7(1.4)	**89.2(1.2)**	88.3(0.6)	88.9(0.8)	88.5(1.0)	87.2(2.1)	85.6(0.9)	87.2(2.0)	87.6(1.5)	85.1(1.4)	85.6(0.9)	87.7(1.2)
I	89.7(1.9)	87.9(1.6)	89.3(1.9)	89.4(1.7)	89.0(1.9)	89.1(1.9)	89.6(1.5)	89.3(1.6)	**89.9(1.6)**	**89.9(1.6)**	88.4(1.8)	89.0(1.7)	58.6(3.5)	55.2(3.5)	68.7(7.2)	79.2(4.2)	88.2(2.3)	55.2(3.5)	82.3(1.4)
J	90.1(1.5)	88.6(1.6)	89.2(1.8)	88.5(1.5)	89.7(1.4)	89.3(1.8)	89.8(1.8)	88.8(1.5)	90.3(1.1)	**90.6(1.6)**	89.7(2.1)	88.9(1.6)	88.7(1.8)	89.2(1.3)	90.1(1.5)	88.1(0.9)	88.3(1.7)	86.8(1.9)	83.0(2.0)
K	89.0(0.7)	88.7(1.1)	88.2(0.7)	89.1(1.1)	88.5(0.9)	88.6(1.4)	88.1(2.0)	**90.0(1.2)**	88.7(0.9)	88.7(0.9)	88.3(1.2)	51.3(13.5)	70.2(5.1)	83.4(9.0)	89.4(1.4)	87.9(1.2)	85.6(1.4)	83.1(8.9)	87.8(1.5)
L	**85.3(1.0)**	84.1(1.0)	83.6(1.3)	84.1(1.2)	84.7(0.7)	84.9(0.7)	84.0(1.1)	85.2(1.0)	84.1(2.3)	85.0(1.0)	78.8(12.1)	84.8(1.1)	84.6(1.1)	79.1(1.8)	81.1(1.2)	84.5(1.0)	82.7(1.5)	79.1(1.8)	85.1(1.2)
M	**94.2(1.0)**	92.7(0.9)	92.5(1.2)	93.1(1.2)	93.0(1.0)	92.6(1.1)	93.0(1.0)	93.0(1.0)	93.4(1.4)	93.8(1.1)	92.6(1.1)	92.9(1.3)	93.1(0.9)	92.9(0.8)	93.2(1.0)	92.0(0.9)	86.7(2.4)	91.5(1.0)	87.9(1.1)

Tabela 2. Resultados de Macro-F1 das abordagens de undersampling utilizando o classificador RoBERTa. Células em verde são estatisticamente equivalentes à classificação sem undersampling (NoUnder). Células em negrito representam o melhor resultado da base de dados.

5.2. Análise da generalização do modelo

A Tabela 3 apresenta os resultados do TPRGap. A coluna "NoUnder" apresenta o resultado sem o *undersampling*, enquanto 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": quanto mais próximo do verde, maior a redução do viés, e quanto mais vermelho, maior o agravamento do viés.

Em relação ao viés médio calculado, temos que os métodos que conseguiram os menores vieses e, por consequência, conseguiram reduzir mais o viés em relação ao "NoUnder", foram UBR (viés médio de 0,020), NM2 (0.021), E2SC_RL (0,022), RUS (0.025) e E2SC (0.029). Dentre esses, o E2SC_RL consegue reduzir o viés em todas as bases de dados, o NM2 e o E2SC conseguem reduzir o viés em 12 dos 13 conjuntos de dados, enquanto o UBR e o RUS consegue fazer isso em 11. Os métodos RENN, ALLKNN, ENN e IHT, que já estão entre os piores em termos de eficácia, também desempenham mal em relação à generalização do modelo, com todos aumentando o viés médio. Os métodos TL e OSS, apesar de apresentarem bons resultados em termos de eficácia, demonstram um baixo ganho em relação à generalização. Na maioria das bases de dados, eles reduzem pouco o viés do modelo, apresentando viés médio de 0.11 e 0.10, respectivamente.

dataset	NoUnder	UBR	NM2	E2SC_RL	RUS	E2SC	CNN	CC_NN	TL	OSS	NM1	NM3	ALLKNN	ENN	NCR	SBC	OBU	RENN	IHT
A	0,07562	0,00526	0,01853	0,01854	0,10067	0,00593	0,00617	0,07018	0,05219	0,04227	0,04284	0,01988	0,10759	0,14337	0,03171	0,01340	0,09738	0,14337	0,10717
B	0,04085	0,00212	0,01763	0,00010	0,00291	0,00687	0,00153	0,01066	0,02859	0,02518	0,00329	0,02174	0,06245	0,06050	0,03813	0,14865	0,01905	0,06050	0,05852
C	0,03626	0,04169	0,00124	0,01709	0,00030	0,00363	0,01509	0,04500	0,06196	0,06571	0,05489	0,02340	0,05508	0,28919	0,22409	0,03702	0,16325	0,28919	0,10262
D	0,35029	0,00532	0,06155	0,10423	0,05708	0,06278	0,19329	0,07545	0,36113	0,39997	0,04434	0,06393	0,32423	0,32425	0,23746	0,04699	0,17321	0,28657	0,30775
E	0,19016	0,03673	0,03624	0,04019	0,03429	0,00625	0,00591	0,03717	0,11754	0,13073	0,02724	0,06280	0,13647	0,04629	0,01828	0,02853	0,06560	0,15752	0,21150
F	0,33307	0,01217	0,02457	0,02040	0,00347	0,07266	0,11219	0,24206	0,33647	0,30434	0,02356	0,11102	0,24844	0,26183	0,25559	0,35174	0,26444	0,21590	0,33633
G	0,12616	0,00625	0,05449	0,02920	0,02544	0,10534	0,00003	0,01117	0,10764	0,13407	0,04659	0,06990	0,03360	0,01001	0,04322	0,02391	0,14439	0,11132	0,13242
H	0,06283	0,01018	0,00468	0,00194	0,01095	0,00299	0,04734	0,01091	0,03505	0,03556	0,00389	0,01052	0,09839	0,14146	0,07711	0,03903	0,11441	0,14146	0,07175
I	0,09617	0,03728	0,00977	0,00084	0,01166	0,03139	0,03178	0,02367	0,09276	0,09025	0,01233	0,02581	0,59280	0,63410	0,40318	0,24493	0,02269	0,63410	0,19679
J	0,10245	0,03027	0,02146	0,00612	0,00050	0,01952	0,01640	0,02391	0,08453	0,07735	0,00050	0,00093	0,02120	0,00710	0,02815	0,03456	0,00668	0,07195	0,15786
K	0,06542	0,04332	0,01336	0,03388	0,02650	0,03294	0,02745	0,02328	0,05979	0,05959	0,02551	0,69208	0,44500	0,16559	0,04797	0,01200	0,09956	0,18819	0,07457
L	0,02431	0,03269	0,02148	0,01678	0,04924	0,03420	0,11090	0,04219	0,02369	0,01909	0,12359	0,02736	0,09631	0,29716	0,20889	0,07850	0,13915	0,29716	0,05811
M	0,15996	0,00226	0,00062	0,00629	0,01478	0,00088	0,01212	0,01232	0,06303	0,06229	0,00311	0,01205	0,09334	0,09803	0,01809	0,01669	0,09423	0,03696	0,12216
Média	0,12796	0,02043	0,02197	0,02274	0,02598	0,02964	0,04463	0,04831	0,10957	0,11126	0,03167	0,08780	0,17807	0,19068	0,12553	0,08277	0,10800	0,20263	0,14904

Tabela 3. Resultados de TPRGap das 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.

Com essas análises feitas, podemos confirmar nossa hipótese de pesquisa **HP1** de que classificadores de aprendizado profundo baseados em *transformers* não conseguem, por si só, resolver totalmente o problema de bases de dados desbalanceadas, o viés gerado pelo aprendizado desigual entre as classes. Notamos que para resolver esse problema, podemos utilizar os métodos de *undersampling*, que fazem com que o modelo generalize seu conhecimento das classes e, consequentemente, reduza o viés do modelo, resolvendo assim o problema das bases de dados desbalanceadas.

5.3. Análise do speedup

Por fim, analisamos os métodos de *undersampling* em relação ao tempo que eles agregam ao treinamento do modelo. Para avaliarmos isso, utilizamos o *Speedup*. A Tabela 4 nos mostra o *Speedup* produzido pelos métodos de *undersampling*, sendo que a cor das células indica o quão mais rápido (células em verde) ou mais lento (células em vermelho) foi a execução dos métodos em comparação com o método sem o *undersampling*.

dataset	UBR	NM2	E2SC_RL	RUS	E2SC	CNN	CC_NN	TL	OSS	NM1	NM3	ALLKNN	ENN	NCR	SBC	OBU	RENN	IHT
A	1,555200221	1,449872028	1,711766029	1,313962103	1,450949443	1,519014265	1,499513496	1,185338595	1,150125745	1,469424777	1,492743265	1,446589438	1,966386117	1,515756031	1,637036202	1,282441319	1,854393593	1,402193906
B	1,312857776	1,09618715	1,399751571	1,123389128	1,24333915	1,360903033	1,202327722	0,8731271485	0,884820259	1,236653313	1,247103779	1,119874171	1,069796236	1,223274784	1,637901621	1,214429471	1,038975915	1,114500154
C	1,226596237	1,007158636	1,20134376	1,165989336	1,087838578	1,467541143	0,9910764738	0,980828542	0,9547779645	1,126742856	1,173851964	1,034830322	1,160261536	1,426285301	1,270549396	1,25901506	1,100413606	1,14010102
D	3,192294048	2,640772931	2,948934792	3,045314204	2,748866649	2,37712012	2,852684148	1,106877673	0,9715602448	3,118253936	3,053826103	1,34115665	1,335080943	1,486799815	3,433008363	1,411198951	1,219748944	2,981429166
E	1,653584412	1,601049175	1,634669523	1,569876147	1,585513271	1,510314654	1,612383276	1,033770502	1,013900119	1,922867144	1,519423714	1,750712127	1,536832664	1,401476215	1,563978764	1,331157781	1,516110875	1,634028959
F	2,488779651	2,159909221	2,42394204	2,061359638	1,888080792	2,283087694	2,42211846	1,003147329	0,8001252343	2,435393042	2,128457801	0,9768111587	0,9904521912	0,8760762004	2,861302773	1,302355523	0,9978077218	1,607722809
G	1,532099359	1,450144261	1,371187762	1,394015667	1,520553956	1,600700467	1,528724848	1,068375912	1,065418278	1,298538982	1,442939485	1,632306443	1,503690001	1,412660504	1,507234911	1,491394066	1,415762908	1,626917263
H	1,20531643	1,134802243	1,093816708	1,096734287	1,171842837	1,242530039	1,192799837	0,9285457011	0,9613305935	1,076007495	1,09570989	1,229783977	1,414702516	1,210301958	1,215629879	1,373855105	1,375968822	1,181405901
I	1,443032291	1,311512108	1,568898922	1,357905116	1,353849886	1,175690675	1,467150393	0,9372071449	0,9482897905	1,312084409	1,400384782	1,640550551	1,663018205	1,606719547	1,769201749	1,380551582	1,610831465	1,182081481
J	1,730184183	1,619014604	1,950086739	1,878591316	1,777694929	1,493789194	1,866602595	1,055282936	1,067495951	1,658995889	1,757462935	1,587340736	1,39459701	1,319433854	1,728216718	1,60743696	1,770909914	1,667875834
K	1,196781774	1,113734501	1,273495471	1,114431418	1,095251354	1,005224075	1,186430672	0,9620309201	0,9902202144	1,162764624	1,588457566	1,4516276	1,476137449	0,987928543	1,363870727	1,342419188	1,513152287	1,22996007
L	1,040107213	0,979249296	1,10352304	1,04415206	1,063920594	0,9700916408	1,029276664	0,9763645487	0,9535910302	0,9252925979	1,007842752	1,147503533	1,548504308	1,465484942	1,20001587	1,370207471	1,517627453	1,149477283
M	1,656555436	1,50087843	1,658258435	1,464709306	1,47517157	1,430906517	1,661993331	1,033599562	1,086759485	1,671748116	1,544101515	1,083498522	1,27872899	1,225491234	1,577864155	1,451588914	1,33234894	1,418119128
Média	1,633337618	1,46648343	1,641513446	1,510033056	1,497144078	1,495147194	1,577929378	1,011115116	0,9883396083	1,570366706	1,573254273	1,341737325	1,410629859	1,319870687	1,751216241	1,370619338	1,404927111	1,487370229

Tabela 4. Resultados de Speedup no custo total (tempo) das abordagens de undersampling utilizando o classificador RoBERTa. 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.

A primeira informação relevante ao analisarmos a Tabela 4 é que, com exceção do OSS, todos os métodos conseguiram, em média, ou manter (tl) ou reduzir (outros

métodos) o tempo de treinamento do modelo em comparação com o treinamento do RoBERTa sem técnicas de *undersampling*. Os métodos UBR, NM2, E2SC_RL, RUS e E2SC, que, com as análises anteriores, têm se mostrado os melhores métodos, conseguem um *Speedup* muito bom, alcançando respectivamente 1.63, 1.46, 1.64, 1.51 e 1.49 de *Speedup*. Em outras palavras, podemos dizer que os métodos UBR, NM2, E2SC_RL, RUS e E2SC conseguem melhorar o viés do modelo, fazendo-o generalizar melhor, mantendo a eficácia e reduzindo o tempo necessário para treinar o modelo.

5.4. Discussão Final

Dadas as análises apresentadas anteriormente, onde os métodos de *undersampling* foram avaliados quanto à sua eficácia, sua capacidade de melhorar a generalização dos modelos e sua capacidade de reduzir o tempo necessário para treinar o modelo, podemos concluir que os melhores métodos de *undersampling* analisados foram UBR, NM2, E2SC_RL, RUS e E2SC. Isso porque todos eles conseguiram, em termos de eficácia, manter-se estatisticamente iguais ao modelo treinado sem os métodos de *undersampling*. Além disso, conseguiram reduzir o viés do modelo, permitindo que aprendesse igualmente bem sobre todas as classes. Esses métodos também apresentaram uma significativa redução no tempo de treinamento do modelo. Importante ressaltar que dessas estratégias que se destacaram, duas foram originalmente propostas para o cenário de seleção de instâncias.

6. Conclusões e Trabalhos Futuros

Nesse trabalho apresentamos uma avaliação detalhada de métodos de *undersampling* aplicadas em conjunto com algoritmos de classificação automática de texto (CAT). Com o avanço o de algoritmos de CAT de aprendizado profundo baseado em *transformers*, criou-se um senso comum que os próprios algoritmos eram capazes de contornar problemas tradicionais de classificação, como o viés de aprendizado oriundo de coleções de dados desbalanceadas. No presente trabalho, refutamos esse senso comum com a hipótese que, apesar dos classificadores mais avançados atingirem valores de eficácia muito bons, eles ainda não são capazes, por si só, de resolver o problema do viés de aprendizado proveniente de bases de dados desbalanceadas.

Para isso, apresentamos uma análise comparativa detalhada dos principais métodos de *undersampling* aplicadas em conjunto com uma estratégia de CAT baseada em *transformers*, tida como estado-da-arte (i.e. RoBERTa). Ao todo, selecionamos e implementamos os 15 métodos de *undersampling* mais utilizados na literatura (primeira contribuição), além de propormos uma nova estratégia (segunda contribuição). Nossa avaliação se seu sob três perspectivas: eficácia, redução de viés e tempo de treinamento. Como resultado de nossas análises (terceira contribuição), chegamos a um conjunto de cinco métodos de *undersampling*, que consideramos como os melhores. São eles: o UBR, NM2, E2SC_RL, RUS e o E2SC. Concluímos também que ao utilizarmos algum desses métodos em conjunto de classificadores baseados em *transformers*, o resultado é um modelo com um tempo de treinamento menor e que, apesar de manter a eficácia, consegue generalizar melhor o conhecimento sobre todas as classes.

Como trabalho futuro, visamos estender o presente estudo considerando também outros cenários de classificação automática de texto, como multi classes e/ou hierárquico, além de avaliar outros algoritmos de CAT além do RoBERTa, tais como BERT e BART.

Referências

Cunha, W., França, C., Fonseca, G., Rocha, L., and Gonçalves, M. A. (2023a). An effective, efficient, and scalable confidence-based instance selection framework for transformer-based text classification. In *Proceedings of the 46th ACM SIGIR*.

Cunha, W., Viegas, F., França, C., Rosa, T., Rocha, L., and Gonçalves, M. A. (2023b). A comparative survey of instance selection methods applied to nonneural and transformer-based text classification. *ACM Computing Surveys*.

Czarnowska, P., Vyas, Y., and Shah, K. (2021). Quantifying social biases in nlp: A generalization and empirical comparison of extrinsic fairness metrics. *TACL*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Han, H., Wang, W.-Y., and Mao, B.-H. (2005). Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, page 179. Citeseer.

Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *8th Conference on Artificial Intelligence in Medicine in Europe, AIME 2001 Cascais, Portugal, July 1–4, 2001, Proceedings 8*, pages 63–66. Springer.

Lin, W.-C., Tsai, C.-F., Hu, Y.-H., and Jhang, J.-S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, 409:17–26.

Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126, pages 1–7. ICML.

Ng, A. (2017). Machine learning yearning. *URL: http://www. mlyearning. org/(96)*, 139.

Smith, M. R., Martinez, T., and Giraud-Carrier, C. (2014). An instance level analysis of data complexity. *Machine learning*, 95:225–256.

Tomek, I. (1976a). An experiment with the edited nearest-nieghbor rule.

Tomek, I. (1976b). Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11):769–772.

Vuttipittayamongkol, P., Elyan, E., Petrovski, A., and Jayne, C. (2018). Overlap-based undersampling for improving imbalanced data classification. In *19th IDEAL 2018, Madrid, Spain, November 21–23, 2018*, pages 689–697. Springer.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421.

Yen, S.-J. and Lee, Y.-S. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *ICIC Kunming, China, August 16–19, 2006*, pages 731–740. Springer.