Atribuição de Autoria: Um estudo comparativo entre a Classificação Tradicional de textos e uso de Aprendizado de Métrica Profundo

Walter do Espírito Santo Souza Filho, Edson Takashi Matsubara

Laboratório de Inteligência Artificial - Faculdade de Computação
Universidade Federal de Mato Grosso do Sul - UFMS
Cidade Universitária, Av. Costa e Silva, 79070-900 - MS - Brazil

walter21souza@gmail.com, edson.matsubara@ufms.br

1 Introdução

A identificação de autores de textos tem um papel importante, tanto na atualidade quanto num contexto histórico. Obras literárias ao longo do tempo passaram e ainda passam por diversos estudos, a fim de identificar seu real autor. Além disso, o plagiarismo também é um problema recorrente e, no contexto atual, o uso de modelos de linguagem, como o ChatGPT [1] e Bard [2], tornam essa prática cada vez mais acessível.

No campo de Processamento de Linguagem Natural (PLN), a tarefa de Atribuição de Autoria (AA) estuda a identificação do autor de um determinado texto ou documento. A ideia principal parte do pressuposto de que cada autor possui suas próprias características de escrita. Dessa forma, a proposta é que tais características sejam identificadas e aprendidas por um algoritmo, e que essas informações sejam utilizadas posteriormente para identificar o autor de um novo texto inédito.

O maior desafio na AA é justamente a identificação e extração de características importantes e relevantes para a identificação do autor. Um dos métodos tradicionais bem conhecidos, e que apresenta bons resultados, utiliza um modelo de linguagem baseado em transformers: BertAA (Fabien, 2020) [3]. Diversas variações desse mesmo modelo de linguagem foram surgindo ao longo do tempo, contribuindo para melhores resultados de Estado da Arte.

A proposta desse trabalho é justamente explorar uma nova adaptação do modelo tradicional, utilizando o mesmo modelo de linguagem, porém, voltado para um contexto de Aprendizado de Métrica Profundo. Nessa nova abordagem, o modelo de linguagem deixa de servir como um classificador, e passa a ser usado como um backbone que irá aprender métricas baseadas nas características de interesse. Para auxiliar nesse aprendizado de métrica, utilizamos juntamente durante o processo a loss ArcFace[4]. Na sequência, tais características extraídas são utilizadas por um algoritmo que utilize métricas de distância, fornecendo como resultado, a classificação final.

Com os resultados obtidos, foi possível observar melhorias na performance quando comparamos o método proposto com o método tradicional. Além disso, foram encontradas e discutidas relações entre os parâmetros variáveis do modelo proposto.

2 Método Tradicional

De modo geral, modelos como BertAA utilizam um modelo de representação de linguagem pré treinado, como o BERT (Devlin, 2019) [5]. A partir dele, é feito um ajuste fino (fine tuning) com o objetivo de que os pesos do modelo sejam ajustados para uma melhor especialização na tarefa de interesse.

Como mostrado na Figura 1 (inspirada no artigo original), uma sentença inicial passa pelo processo de tokenização, para a geração dos tokens. Nesse processo, também é adicionado um token especial de classificação no início ($[CLS]$). Em seguida, os tokens gerados são utilizados como dados de entrada no BERT. Seguindo sua estrutura, os tokens são transformados em embeddings E, passando pelas camadas internas.

Na última camada, o primeiro token C é especial e corresponde à classificação após o processamento. Esse token terá o tamanho correspondente à quantidade de classes de interesse, e o índice do maior elemento desse array será o resultado da classificação.

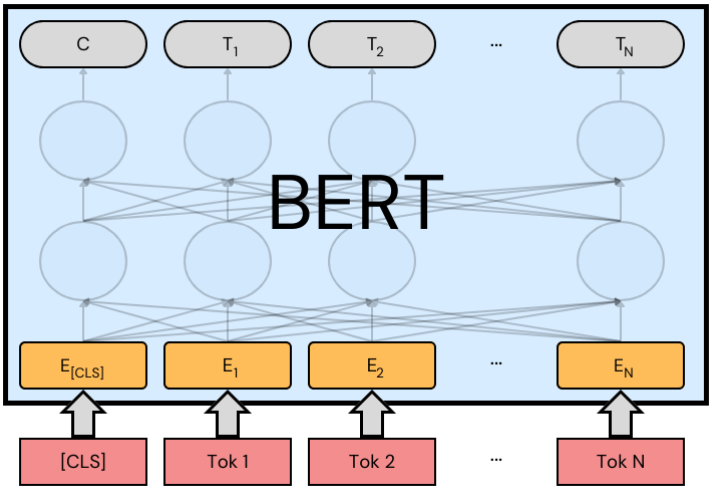

Figura 1: Arquitetura do modelo tradicional BERT.

3 Aprendizado de Métrica Profundo

Uma métrica d é uma função não-negativa que tem o objetivo de estabelecer uma relação de "distância"entre dois pontos A e B. Para isso, a métrica deve obedecer 3 propriedades fundamentais:

- Não-negativa: $d(A, B) \geq 0$, sendo $d(A, B) = 0$ se $A = B$;
- Desigualdade triangular: $d(A, B) \leq d(A, C) + d(C, B)$;
- Simetria: $d(A, B) = d(B, A)$.

Utilizando esse conceito, tem-se a abordagem de Aprendizado de Métrica, que utiliza essa relação que a função de métrica proporciona, para estabelecer uma similaridade entre os dados de interesse, como mostrado na Figura 2.

Porém, quando trata-se de Aprendizado de Métrica Profundo, uma rede neural é responsável por aprender quais são as características mais representativas dos dados, e utilizá-las para criar essa função de similaridade.

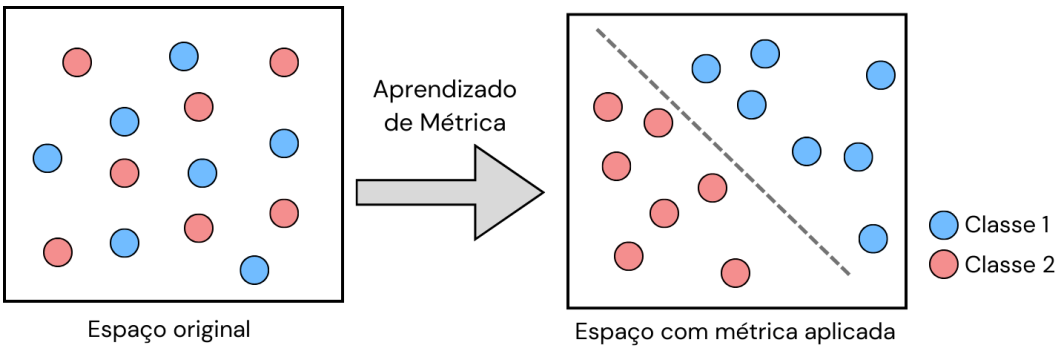

Figura 2: Transformação de espaço utilizando aprendizado de métrica.

4 Método Proposto

4.1 Estrutura Geral

Neste trabalho, propomos um novo método para a tarefa de Atribuição de Autoria, inspirado tanto na classificação tradicional quanto no aprendizado de métrica profundo. Para isso, tomamos como ponto de partida um modelo de linguagem baseado em transformers, como por exemplo o BERT ou DistilBERT [6], realizando algumas modificações em suas últimas camadas, com o objetivo de utilizá-lo num contexto de aprendizado de métrica. Seguindo essa ideia, o esquema mostrado na Figura 3 descreve o fluxo de dados.

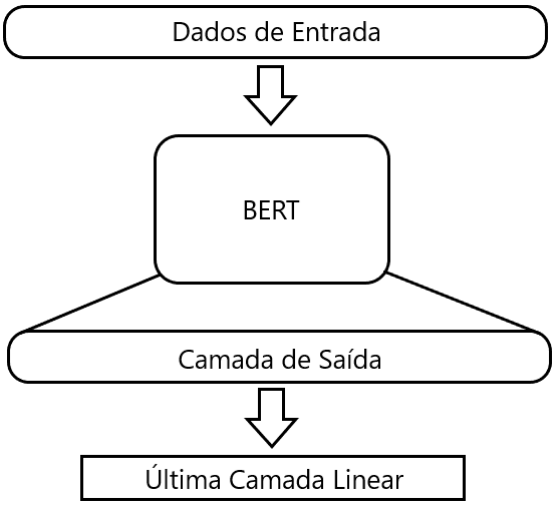

Figura 3: Esquema do modelo proposto.

No esquema acima, após a sentença inicial de interesse passar pelo processo de tokenização, temos os Dados de Entrada. Esses dados passam por todo processamento do modelo de linguagem, percorrendo suas camadas internas e blocos de transformers até que, por fim, passe pela última Camada de Saída. Nesse momento, no Método Tradicional, o fluxo de dados poderia ainda ser processado por mais uma camada, chamada de Camada de Classificação. Em nosso modelo, retiramos essa camada, caso ela exista, e inserimos uma nova Última Camada Linear. Nela, temos como resultado os Embeddings, que devem possuir as características mais relevantes da sentença inicial, aprendida pelo modelo de linguagem escolhido.

Vale ressaltar que, nesse modelo proposto, a estrutura interna do modelo de linguagem será fixa, sem adaptações. Dessa forma, temos a liberdade de manipular apenas a maneira com que os Dados de Entrada são apresentados ao modelo, e também como deve ser a construção e comportamento da Última Camada Linear. Sendo assim, dois principais atributos se destacam: tamanho dos tokens (dimensão do vetor de entrada), e tamanho dos embeddings (dimensão do vetor da última camada linear). Um pouco mais adiante, na Seção de Resultados 6, discutiremos os impactos de cada um desses atributos, e como eles se relacionam.

Após todo esse processo, os embeddings gerados são, por sua vez, utilizados em algum algoritmo que use métricas de distância, como o KNN (Guo, 2004) [7]. Apesar de sua simplicidade, o KNN apresenta resultados bem relevantes, quando aliado ao aprendizado de máquina gerado pelo modelo proposto.

4.2 Loss ArcFace

A ArcFace tem origem pelos métodos utilizados em tarefas de reconhecimento de faces. Dois desses principais métodos são baseados em Softmax [8] e na loss Triplet (Schroff, 2015)[9]. Apesar dos dois métodos obterem resultados ótimos em tarefas de reconhecimento de faces, ambos possuem alguns lados negativos.

Para a loss Softmax, o principal ponto negativo é que as características aprendidas pelo treinamento apesar de serem boas para a classificação de um conjunto fechado, não são discriminativas o suficiente para um problema de classificação de conjunto aberto.

Já para o caso da loss Triplet, seu principal problema é que, para um conjunto grande de dados, a quantidade existente de combinações de triplas cresce muito rápido, causando um aumento significativo no número de iterações que a loss precisa processar.

Sendo assim, temos o surgimento do ArcFace como uma maneira de justamente aproveitar esses resultados ótimos já obtidos, porém, evitando os pontos negativos existentes.

A ideia principal da loss ArcFace é melhorar a representação das características das amostras em um espaço discriminativo. Com isso, o objetivo é garantir que a representação de dados de classes distintas estejam bem separadas uma das outras, enquanto que as representações de dados de uma mesma classe estejam mais próximas.

Para isso, a loss ArcFace toma como base a função Softmax, fazendo uma adaptação que insere uma margem angular. A construção da equação segue do seguinte modo:

Tomando Softmax como base, uma das definições mais usadas é a seguinte:

$$L_1 = -\log \frac{e^{W_{y_i}^T x_i}}{\sum_{j=1}^{N} e^{W_{y_i}^T x_i}} \tag{1}$$

Em 1, x_i é o i-ésimo dado de entrada, que pertence à classe y_i. W_{y_i} é o vetor peso correspondente à classe y_i, e N é a quantidade total de classes.

Fazendo a equivalência $W_{y_i}^T x_i = \|W_{y_i}\| \, \|x_i\| \cos\theta_{y_i}$, adotando $\|W_{y_i}\| = 1$ (pela normalização l2), e $\|x_i\| = s$ (aplicando normalização l2 e fazendo uma escala para s), temos o resultado em 2:

$$L_2 = -\log \frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{N} e^{s\cos\theta_j}} \tag{2}$$

É então adicionada uma penalidade de margem angular m, entre x_i e W_{y_i}. Com isso, em 3 temos a equação final que descreve a função loss ArcFace.

$$L_3 = -\log \frac{e^{s\cos(\theta_{y_i}+m)}}{e^{s\cos(\theta_{y_i}+m)} + \sum_{j=1,j\neq y_i}^{N} e^{s\cos\theta_j}} \tag{3}$$

Existe também uma variação mais genérica que engloba outras duas losses: SphereFace e CosFace. Combinando as penalidades de margens de cada uma das três losses, temos o seguinte resultado em 4:

$$L_4 = -\log \frac{e^{(s\cos(m_1\theta_{y_i}+m_2)-m_3)}}{e^{(s\cos(m_1\theta_{y_i}+m_2)-m_3)} + \sum_{j=1,j\neq y_i}^{N} e^{s\cos\theta_j}} \tag{4}$$

Sendo m_1 (margem multiplicativa) a penalidade para SphereFace, m_2 (margem aditiva) para ArcFace e m_3 (margem cosseno) para CosFace. Pode-se notar então que variando o valor das penalidades, podemos transitar entre as losses, aproveitando as características específicas que cada uma delas possui.

5 Procedimento Experimental

Essa Seção apresenta o conjunto de métricas utilizado para realizar a avaliação e comparação de performance entre os métodos de classificação discutidos até então. Além disso, também introduz o conjunto de losses utilizados para os treinos de deep metric learning.

5.1 Métricas

Dentre as métricas bem conhecidas da biblioteca padrão do Scikit-Learn[1], foram utilizadas: accuracy, precision, recall e F1-Score (Macro).

Para melhor entendimento das equações que definem cada umas das métricas citadas, são adotadas algumas notações seguindo a matriz de confusão na Figura 4:

Uma matriz de confusão representa o resultado da classificação do modelo de forma visual por meio de uma tabela. As linhas representam as classes verdadeiras, e as colunas a classe predita pelo modelo. Dessa forma, é possível notar que as classificações corretas acabam se localizando na diagonal principal da matriz de confusão.

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Figura 4: Matriz de Confusão.

[1] `https://scikit-learn.org/`

Accuracy indica uma performance mais generalizada do modelo. Ela mede o total de classificações corretas do modelo, dentre todas as classificações.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN} \tag{5}$$

Precision verifica o quão acertivo o modelo é, medindo dentre as classificações de classe Positivo, quantas foram corretas.

$$Precision = \frac{VP}{VP + FP} \tag{6}$$

Recall mede dentre o conjunto real da classe Positivo, quantas foram corretamente preditas como Positivo.

$$Recall = \frac{VP}{VP + FN} \tag{7}$$

A métrica F1-Score é uma média harmônica entre Precision e Recall, resultando em uma contribuição relativa entre elas.

$$\begin{aligned} F_1 &= 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \\ F_1 &= \frac{VP}{Vp + \frac{1}{2} \cdot (FP + FN)} \end{aligned} \tag{8}$$

5.2 Datasets

Dois datasets foram escolhidos para o caso de estudo: IMDb62 e C50. Para ambos conjuntos de dados, foi feita uma separação de dados de treinamento e de testes. Isso ocorreu apenas uma única vez e se manteve fixa durante todos os treinamentos e validações realizados.

Além disso, a quantidade de amostras utilizadas por iteração (batch size) escolhida foi 64. Entretanto, em alguns testes foi preciso reduzir para 48 e 16, por causa da quantidade de armazenamento necessária para o treinamento ultrapassar a que tínhamos disponível na placa de vídeo utilizada.

5.2.1 IMDb62

O conjunto de textos IMDb (Internet Movie Database) é utilizado amplamente na área de análise de sentimentos e também classificação de textos. Conta com um conjunto de avaliações e críticas de filmes produzidas por diversos autores, coletadas do site IMDb.

Neste trabalho utilizamos uma versão reduzida do conjunto original de textos, o IMDb62. Nele, temos um total de 62 autores, com 1000 textos para cada autor, totalizando 62000 textos.

5.2.2 C50

Esse dataset é, novamente, uma versão reduzida do RCV1 (Lewis, 2004) [10], um conjunto de notícias manualmente categorizadas, disponibilizado pela agência de notícias Reuters para fins de pesquisa. Nessa versão reduzida, temos os primeiros 50 autores quando classificados em relação ao tamanho total dos artigos. Tanto a parte de treino quanto de teste, possuem cada uma 2500 textos (50 textos para cada autor).

5.3 Modelo de Linguagem

Tanto para o método tradicional quanto para o método proposto, optamos por utilizar o modelo de linguagem DistilBERT, por ser um modelo menor (66 milhões de parâmetros - 40% a menos que o modelo BERT), mais rápido e computacionalmente mais barato (em torno de 60% mais rápido que o BERT) [6].

6 Resultados

Foram realizados diversos testes para os datasets IMDb62 e C50, variando os parâmetros de tamanho do token, tamanho do embedding, e também a seed de inicialização dos pesos utilizados pelos modelos. Após os treinamentos, foi medido a Accuracy, Precision, Recall e F1-Score de cada um deles. Em seguida, os resultados das métricas foram agrupados pelo tamanho do token e realizada uma média aritmética simples.

6.1 IMDb62

Os resultados obtidos pelo método tradicional utilizando DistilBERT são mostrados na Tabela 1. Nela, a primeira coluna define a quantidade de tokens utilizada, a segunda coluna mostra o tamanho do lote, e nas colunas seguintes estão os diferentes tipos de métricas utilizadas para a avaliação do modelo.

De modo semelhante, na Tabela 2 estão os resultados para o método proposto. Entretanto, existe um novo parâmetro variável que é o tamanho do embedding, sinalizado na segunda coluna.

Tabela 1: Método Tradicional - Relação entre o tamanho do token e as métricas utilizadas para avaliação (IMDb62 - DistilBERT).

token size	batch size	accuracy	f1_score	precision	recall
256	48	0.932882	0.933815	0.937788	0.932882
128	64	0.845048	0.849338	0.862671	0.845048
64	64	0.734946	0.736888	0.756170	0.734946

Tabela 2: Método Proposto - Relação entre o tamanho do token, embedding e as métricas utilizadas para avaliação (IMDb62 - ArcFace loss).

token size	embedding size	batch size	accuracy	f1_score	precision	recall
512	128	16	0.945715	0.951835	0.947612	0.945743
256	256	48	0.954513	0.955352	0.955650	0.954563
128	256	64	0.887021	0.892874	0.890480	0.887029
	128	64	0.880113	0.888449	0.884132	0.880093
	64	64	0.874661	0.885011	0.879036	0.874658
64	256	64	0.742609	0.747981	0.750616	0.742602
	128	64	0.736353	0.742417	0.746168	0.736341
	64	64	0.727881	0.734331	0.738370	0.727859

6.2 C50

Seguindo a mesma apresentação para o conjunto de dados IMDb62, nas Tabelas 3 e 4 estão os resultados do método tradicional e proposto, respectivamente.

Tabela 3: Método Tradicional - Relação entre o tamanho do token e as métricas utilizadas para avaliação (C50 - DistilBERT).

token size	batch size	accuracy	f1_score	precision	recall
128	64	0.751466	0.749749	0.763619	0.751466
64	64	0.749333	0.730352	0.751791	0.727999
32	64	0.704356	0.699843	0.724343	0.704356
16	64	0.669333	0.668835	0.678630	0.669333

Tabela 4: Método Proposto - Relação entre o tamanho do token, embedding e as métricas utilizadas para avaliação (C50 - ArcFace loss).

token size	embedding size	batch size	accuracy	f1_score	precision	recall
128	128	64	0.761367	0.780557	0.762623	0.767998
	64	64	0.762174	0.779406	0.763549	0.765745
	32	64	0.763788	0.777830	0.764691	0.769251
	16	64	0.751950	0.765352	0.753025	0.757383
64	128	64	0.756793	0.776376	0.758241	0.762413
	64	64	0.747646	0.764762	0.749105	0.754957
	32	64	0.745763	0.767204	0.746790	0.751629
	16	64	0.740651	0.760754	0.741265	0.745036
32	128	64	0.730697	0.748254	0.732130	0.737056
	64	64	0.735808	0.758746	0.737284	0.740517
	32	64	0.734194	0.753411	0.735401	0.735605
	16	64	0.724778	0.747159	0.725617	0.735344
16	128	64	0.671240	0.695567	0.672191	0.673784
	64	64	0.677966	0.697351	0.678611	0.685408
	32	64	0.672047	0.694365	0.673426	0.680447
	16	64	0.672585	0.695988	0.673549	0.683863

7 Discussão dos Resultados

Considerando os resultados fornecidos pelos treinamentos utilizando os modelos de interesse para os conjuntos de dados IMDb62 e C50, é possível observar, em sua totalidade, um ganho de performance a favor do método proposto. Fixando um tamanho de token para ambos modelos, e variando o tamanho do embedding para o modelo proposto, nota-se uma melhora nos resultados das métricas em todos os casos de teste.

Além disso, as relações entre os tamanhos de token e embedding se tornam evidentes quando comparadas com os respectivos resultados. Como os dados de entrada do modelo são justamente os tokens gerados pelo processo de tokenização, existe uma relação direta entre a quantidade de tokens utilizada e a quantidade de informação relevante da sentença inicial que eles armazenam.

Já no caso dos embeddings, eles são os dados de saída do modelo proposto que, posteriormente, é utilizado por um algoritmo baseado em métricas de distância. Sendo assim, os resultados apontam que dados com uma maior dimensão (maior tamanho de embedding) geraram um espaço métrico mais discriminativo, possibilitando alcançar melhores resultados.

Além disso, na Tabela 2, é notório uma perda de eficiência entre o treinamento de tamanho de token 256 (batch size 48) e 512 (batch size 16). A generalização de dados é muito importante durante o treinamento, pois em um cenário real (ou teste), o algoritmo processará dados nunca antes vistos. Entretanto, ao diminuir o tamanho da quantidade de amostras, o grau de generalização dos dados também diminui [11]

[12]. Dessa forma, é esperado um resultado inferior no cenário de testes.

Considerando também a estrutura dos dois conjuntos de textos testados (IMDb62 e C50), o modelo proposto conseguiu obter bons resultados mesmo com a diferença existente entre as quantidades de textos para cada autor. Dessa forma, mostrando uma boa adaptação mesmo no cenário com uma quantidade reduzida de exemplares por autores.

8 Conclusão

Este trabalho mostra que uma abordagem no âmbito do Aprendizado de Métrica Profundo para a Atribuição de Autoria é possível e traz melhorias nos resultados quando comparado ao Método Tradicional. A simplicidade de aplicação do método proposto torna-o uma opção viável e acessível, especialmente com os avanços contínuos nos modelos de linguagem existentes. Também forneceu um melhor compreendimento sobre como os diferentes parâmetros de token, embedding e quantidade de amostras se relacionam com os resultados obtidos, reafirmando algumas das premissas já conhecidas. Apesar dos resultados positivos, é importante ressaltar que a tarefa de Atribuição de Autoria ainda enfrenta desafios significativos, especialmente quando lida com textos curtos, estilo de escrita variável ou manipulações intencionais. A qualidade dos resultados está intrinsecamente ligada à qualidade e representatividade dos conjuntos de dados utilizados para treinamento, ressaltando a importância da disponibilidade de dados diversificados e abrangentes.

Referências

[1] OpenAI. *ChatGPT: OpenAI's Conversational Language Model*. `https://openai.com/research/chatgpt/`. Acessado em: 16 de novembro de 2023. 2021.

[2] Google AI. *Bard: Um grande modelo de linguagem do Google AI*. 2023. URL: `https://ai.googleblog.com/2023/01/lamda-language-model-for-dialogue-and.html`.

[3] Maël Fabien et al. "BertAA : BERT fine-tuning for Authorship Attribution". Em: *Proceedings of the 17th International Conference on Natural Language Processing (ICON)*. Indian Institute of Technology Patna, Patna, India: NLP Association of India (NLPAI), dez. de 2020, pp. 127–137. URL: `https://aclanthology.org/2020.icon-main.16`.

[4] Jiankang Deng et al. "ArcFace: Additive Angular Margin Loss for Deep Face Recognition". Em: vol. 44. 10. Institute of Electrical e Electronics Engineers (IEEE), out. de 2022, pp. 5962–5979. DOI: `10.1109/tpami.2021.3087709`. URL: `https://doi.org/10.1109%2Ftpami.2021.3087709`.

[5] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". Em: 2019. arXiv: `1810.04805 [cs.CL]`.

[6] Victor Sanh et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. arXiv: 1910.01108 [cs.CL].

[7] Gongde Guo et al. "KNN Model-Based Approach in Classification". Em: (ago. de 2004).

[8] Aston Zhang et al. *Dive into Deep Learning*. https://d2l.ai. 2020.

[9] Florian Schroff, Dmitry Kalenichenko e James Philbin. "FaceNet: A unified embedding for face recognition and clustering". Em: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun. de 2015. DOI: 10.1109/cvpr.2015.7298682. URL: https://doi.org/10.1109%2Fcvpr.2015.7298682.

[10] David D. Lewis et al. "RCV1: A New Benchmark Collection for Text Categorization Research". Em: *J. Mach. Learn. Res.* 5 (dez. de 2004), pp. 361–397. ISSN: 1532-4435.

[11] Fengxiang He, Tongliang Liu e Dacheng Tao. "Control Batch Size and Learning Rate to Generalize Well: Theoretical and Empirical Evidence". Em: *Advances in Neural Information Processing Systems*. Ed. por H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a70712a252123c40d2adba6a11d84-Paper.pdf.

[12] Samuel L. Smith et al. *Don't Decay the Learning Rate, Increase the Batch Size*. 2018. arXiv: 1711.00489 [cs.LG].