Importante: Artigo apenas **para fins pedagógicos**. Os resultados **não** devem ser considerados como referência para outras finalidades, pois **podem conter erros**.

Predição de Densidade de Vogais utilizando BERTimbau

Nahim Alves de Souza
nahim@usp.br
Instituto de Matemática e Estatística
Universidade de São Paulo

Resumo

Após o surgimento do BERT, diversos modelos foram criados com base na mesma arquitetura. O BERTimbau é um desses modelos, sendo desenvolvido especialmente para o idioma português brasileiro. Neste trabalho o BERTimbau foi refinado e treinado para a tarefa de predição da densidade de vogais em sentenças de língua portuguesa. Os resultados mostraram que o modelo conseguiu bom desempenho tanto nas tarefas de classificação, quanto na tarefa de regressão.

1 Introdução

Recentemente, o advento do ChatGPT[1] popularizou o uso de modelos de linguagem entre diversos públicos, até mesmo entre aqueles que não são especialistas da área de Inteligência Artificial. No entanto, existem muitos outros modelos de linguagem com arquitetura similar ao GPT que, embora não tenham a mesma popularidade, têm sido muito utilizados em diversas aplicações nos últimos anos.

Esses outros modelos, também baseados na arquitetura de *Encoder-Decoder* dos *Transformers* (Vaswani et al., 2017), têm sido amplamente difundidos, principalmente, pela sua capacidade de transferência de aprendizado. Isto é, são modelos pré-treinados em extensas bases de dados que permitem, posteriormente, o refinamento (*fine-tuning*) para serem aplicados em tarefas específicas. Além disso, o fácil acesso a esses modelos através de bibliotecas como a HuggingFace[2], proporcionou uma disseminação ainda maior desses modelos.

Até hoje, o BERT (Devlin et al., 2018) é um dos modelos mais populares na literatura, pois além do seu ótimo desempenho e da sua capacidade de ser refinado para outras tarefas, ele também serviu de base para a criação de muitos modelos de linguagem pré-treinados, tais como RoBERTa, DistilBERT, ALBERT, BERTimbau, entre outros.

A arquitetura desses modelos segue a proposta original do BERT, variando nos parâmetros de treinamento e nos dados de entrada, de acordo com o contexto para o qual estão sendo pré-treinados. O BERTimbau, por exemplo, é um modelo desenvolvido especialmente para o idioma português brasileiro (Souza et al., 2019), e que agora pode ser refinado para tarefas nesse idioma.

Neste trabalho, o desempenho do modelo BERTimbau foi avaliado em tarefas de regressão e classificação, mais especificamente, na tarefa de predição da densidade de vogais de sentenças em português. Para essa avaliação, o modelo foi refinado com base no córpus de avaliações de produtos da B2W[3], onde estão disponíveis milhares sentenças em português brasileiro.

[1] `https://chat.openai.com/`
[2] `https://huggingface.co/`
[3] `https://github.com/americanas-tech/b2w-reviews01`

Nas seções seguintes, será apresentado um breve resumo sobre o funcionamento dos modelos de linguagem mais recentes. Em seguida, serão apresentadas tarefas de avaliação do modelo BERTimbau, acompanhadas dos experimentos e uma discussão sobre os resultados obtidos.

2 Modelos de Linguagem

Um modelo de linguagem pode ser definido como um modelo probabilístico cuja função é predizer a próxima palavra em sequência de palavras (Jamil, 2023). Considere, por exemplo, a sequência de palavras "A capital de Pernambuco é". Nesse caso, o modelo pode ter o objetivo de calcular a probabilidade da próxima palavra ser "Recife", isto é, a probabilidade condicional P("Recife"|"A capital de Pernambuco é"). A mesma abordagem também pode ser empregada em outras tarefas como, por exemplo, *Question-Answering* (predizendo a sequência de palavras que responde a uma pergunta) ou *Neural Machine Translation* (tradução de textos de um idioma para outro).

2.1 Encoder-Decoder

As redes neurais recorrentes (RNNs) foram a referência para a construção de modelos de linguagem por um bom tempo, pois elas deram origem um mecanismo capaz de capturar o contexto em torno de uma palavra (Lindemann et al., 2021).

Essa característica deu origem a outras arquiteturas, baseadas nas RNNs, tais como a *Encoder-Decoder*, que ganhou um grande destaque, especialmente pelo seu desempenho em tarefas de tradução de textos (Luong et al., 2015).

O ápice dessa popularidade, no entanto, ocorreu a partir do surgimento de mecanismos de atenção neural(Olah & Carter, 2016) e dos *Transformers*(Vaswani et al., 2017), que substituíram o uso das RNNs, por mecanismos muito mais eficientes.

A Figura 1 ilustra simplificadamente a arquitetura *encoder-decoder*. O *Encoder* é responsável por codificar a sentença de entrada, enquanto o *Decoder* recebe a sentença codificada e gera a saída a partir dela. Note que o *Encoder* e o *Decoder* podem ser treinados separadamente, visto que suas tarefas são independentes - o *Decoder* precisa apenas das sentenças codificadas durante o treinamento e não das originais.

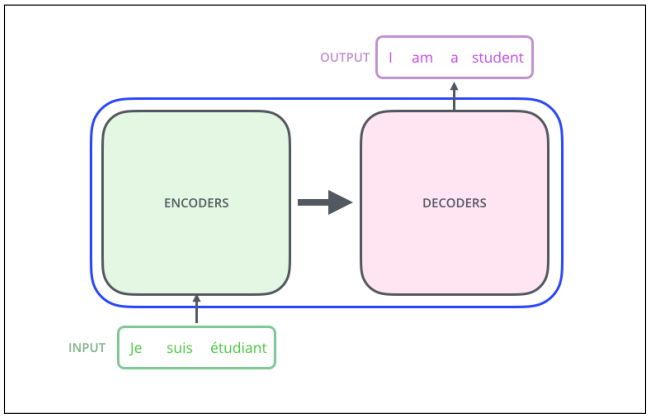

Figura 1: Ilustração da arquitetura *Encoder-Decoder* (Allamar, 2018).

2.2 Transformers

O conceito de *Transformers* foi apresentado por Vaswani et al. (2017) no famoso artigo "Attention is All You Need". Resumidamente, pode-se dizer que um *transformer* é um tipo de rede neural desenvolvida para capturar informações de dados sequenciais, assim como as redes neurais recorrentes (RNNs).

Todavia, a grande vantagem dos *Transformers*, está no fato de que a sua arquitetura (baseada na *Encoder-Decoder*) possibilita a captura eficiente das informações de correlação de uma sequência de dados, através de um mecanismo de Atenção Neural (Olah & Carter, 2016), que elimina a necessidade de recorrência na rede. Isso permite que os modelos baseados em *Transformers* sejam muito mais eficientes, quando comparados às RNNs, podendo ser treinados com um conjunto de dados significativamente maior.

A Figura 2 apresenta a arquitetura de um *Transformer*. Em ambas as partes, *encoder* e *decoder*, a entrada é constituída por uma sequência de vetores contendo as entradas codificadas, isto é, os *embeddings* (Bengio et al., 2000). Na etapa seguinte, denominada *Positional Encoding*, são acrescentadas informações sobre a posição de cada elemento na sequência. Por fim, o modelo um mecanismo chamado de *Self-Attention*, gera uma representação contextual de cada elemento em relação aos demais na sequência.

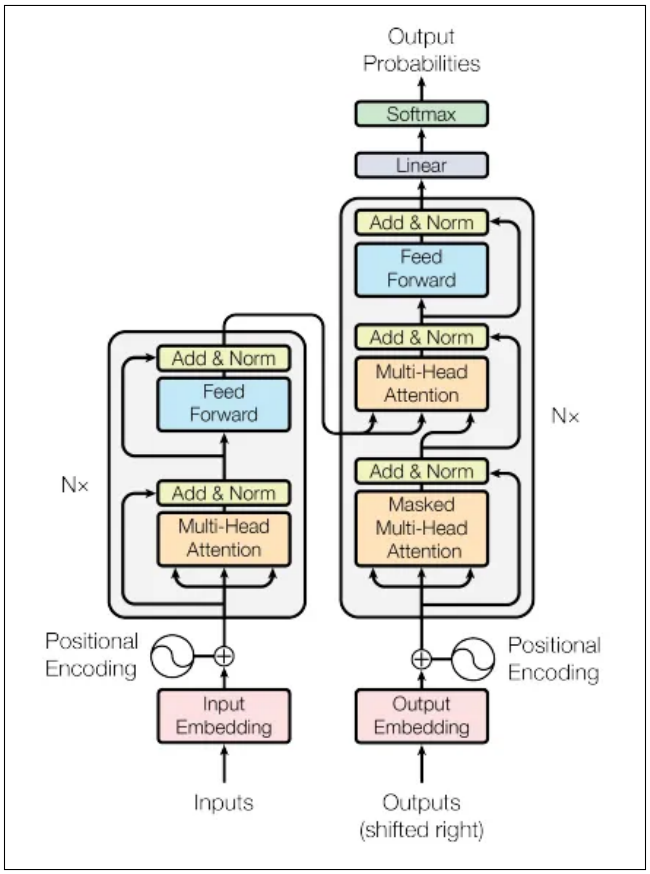

Figura 2: Arquitetura do modelo *Transformer*. O bloco à esquerda representa o *Encoder* e à direta o *Decoder*. Observe que a arquitetura é flexível, e possibilita a combinação sequencial de um número N de blocos, tanto no *Encoder* quanto no *Decoder*.

Na camada de atenção (*Multi-Head Attention*) a entrada (X) é replicada em três matrizes $K = XW^K$, $Q = XW^Q$ e $V = XW^V$, denominadas *Key*, *Query* e *Value*, respectivamente. Essas matrizes são combinadas através de uma multiplicação e uma função *softmax*: $Attention(K, Q, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$ - onde d_k é a dimensão de das chaves (K). Durante o treinamento, os pesos W^K, W^Q, e W^V são iterativamente atualizados, produzindo uma matriz que representa a correlação entre os elementos da sequência.

2.3 Transferência de Aprendizado

Apesar da eficiência dos *Transformers*, a dificuldade de conseguir dados de qualidade para o treinamento, juntamente com a necessidade massiva de recursos computacionais, contribuiu para o avanço de técnicas de transferência de aprendizado (*transfer learning*) (Zhuang et al., 2019).

O objetivo da transferência de aprendizado é permitir a reutilização de uma rede neural pré-treinada, que pode ser especializada através de uma operação de ajuste fino, mais conhecida como *fine-tuning*. Para isso, uma ou mais camadas são adicionadas à rede pré-treinada e um conjunto menor de dados de treinamento é utilizado para treinar essa nova rede. Essa nova etapa de treinamento, no entanto, é muito mais rápida, uma vez que a parte mais extensa do treinamento foi realizada previamente.

Em 2018, Howard & Ruder propuseram uma arquitetura para a utilização dessas técnicas em tarefas de processamento de linguagem natural (NLP). A partir de então, surgiram diversos modelos pré-treinados para NLP, cada um deles com sua própria arquitetura, muitas delas no modelo *encoder-decoder* dos *Transformers*, como é o caso do do GPT (Radford & Narasimhan, 2018) e do BERT (Devlin et al., 2018).

2.4 BERT e BERTimbau

O BERT (*Bidirectional Encoder Representation from Transformers*) é um modelo de linguagem pré-treinado, baseado no *encoder* do *Transformer* (Devlin et al., 2018). A principal característica desse modelo reside no seu treinamento busca capturar o contexto anterior e posterior de cada palavra da sentença. Diferentemente de outros modelos como o GPT, o BERT não foi treinado para tarefas de *prompt* ou geração de texto, ao invés disso ele é focado na possibilidade de ser refinado para outras tarefas através das técnicas de *fine-tuning* (Jamil, 2023).

A arquitetura do modelo BERT é composta por uma sequência *encoders* combinados, podendo ser de tamanho 12 (BERT-base) ou 24 (BERT-large). O número de neuronios na camada de *Feed Forward* também varia (3072 ou 4096), assim como o número de cabeças de atenção neural (12 ou 16). Em relação ao *Transformer* original, o tamanho do vetor de *embeddings* é maior (768 e 1024).

Originalmente, o BERT foi treinado com dados de língua inglesa, limitando seu uso a esse idioma. Visando abranger o maior número possível de idiomas, diversos modelos baseados no BERT surgiram ao longo dos anos. Para o português brasileiro, Souza et al. propuseram um modelo chamado BERTimbau (Souza et al., 2019), que adota a mesma arquitetura de pré-treinamento do BERT, ajustando apenas alguns parâmetros e os dados de treinamento.

O modelo foi treinado com cerca de 3.5 milhões de documentos em português brasileiro, provenientes do córpus brWaC (Wagner Filho et al., 2018). Esse modelo atingiu resultados acima dos padrões do estado-da-arte, nas tarefas de *Named Entity Recognition*, *Sentence Textual Similarity* e *Recognizing Textual Entailment*, tornando-se uma referência dentre os modelos de língua portuguesa.

3 Avaliação do modelo BERTimbau

Neste trabalho, com o objetivo de investigar o desempenho do modelo BERTimbau, foi proposto utilizar o modelo na predição de densidade de vogais em sentenças em língua portuguesa. As sentenças foram obtidas no córpus da B2W, que contém cerca de 130 mil avaliações de produtos feitas por consumidores em uma plataforma de comércio eletrônico. Para este trabalho, entrentanto, apenas um subconjunto (aleatório) de 10 mil dessas avaliações foi utilizado[4].

Cada avaliação do córpus corresponde a uma sentença. A densidade de vogais se refere ao valor DV = (num. de vogais / (num. de vogais + num de consoantes)), que é a proporção do número de vogais em relação ao número de letras de cada sentença, ou seja, apenas as vogais e consoantes são consideradas no cálculo, os símbolos e números na sentença são desconsiderados.

[4]O subconjunto dos dados está disponível em: `https://github.com/alan-barzilay/NLPortugues/tree/master/Semana%2003/data`

A proposta foi dividida em três tarefas, sendo uma de regressão e duas de classificação:

- **Tarefa 1 - Regressão**: predizer o valor final corresponde ao DV, descrito acima.
- **Tarefa 2 - Classificação com dados quantizados não balanceados**: predizer a classe a qual a sentença pertence, de acordo com a densidade de vogais, sendo:
  - C0 - sentenças onde $DV < \frac{1}{3}$;
  - C1 - sentenças onde $\frac{1}{3} \leq DV < \frac{2}{3}$;
  - C2 - sentenças onde $DV \geq \frac{2}{3}$.
- **Tarefa 3 - Classificação com dados quantizados balanceados**: predizer a classe a qual a sentença pertence, sendo 3 classes, onde cada classe contém $\frac{1}{3}$ das sentenças do córpus ordenadas pela densidade de vogais.

Os textos das avaliações estão disponíveis no campo `review_text` da base original, mas a densidade de vogais e os rótulos para a classificação precisaram ser calculados durante em uma etapa de pré-processamento dos dados.

Para isso, foram selecionadas todas as linhas com valores não-nulos na coluna `review_text` e para cada uma dessas linhas foi adicionada uma coluna `density`, correspondente à densidade de vogais (DV). Em seguida, foram criadas duas novas colunas `label1` e `label2` correspondendo, respectivamente, aos rótulos das tarefas 2 e 3.

O número de tokens por sentença foi fixado em 64, visto que poucas avaliações ultrapassam esse número de palavras, como pode ser visto no histograma da Figura 3. Assim, um *padding* será acrescentado nas sentenças mais curtas, enquanto as sentenças maiores serão truncadas para este tamanho.

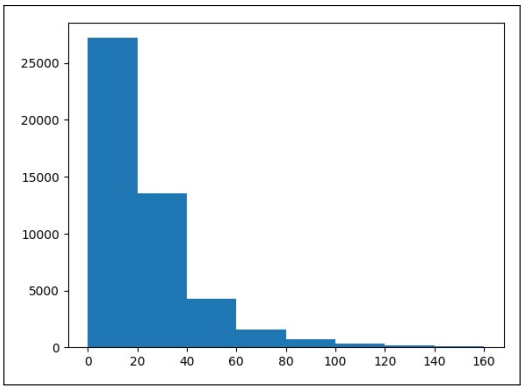

Figura 3: Histograma da número de palvras por sentença.

Nessa etapa, também foi decidido descartar as sentenças cuja densidade tinha valor zero, visto que elas provocavam uma explosão no valor da métrica MAPE, devido a uma divisão por zero. Observando a Figura 4, nota-se que essas sentenças são apenas *outliers* (menos de 0,1% das amostras), assim o descarte dessas amostras foi a solução mais simples a ser adotada.

Por fim, duas outras colunas colunas foram adicionadas para registar a densidade de vogais na primeira e na última palavra da sentença: `first_word_density` e `last_word_density`. Esses valores são utilizados durante o cálculo do *baseline* de avaliação da **Tarefa 1**, descrita posteriormente.

3.1 Métricas de Avaliação

Na Tarefa 1, foram utilizadas as seguintes métricas para avaliar o modelo de regressão:

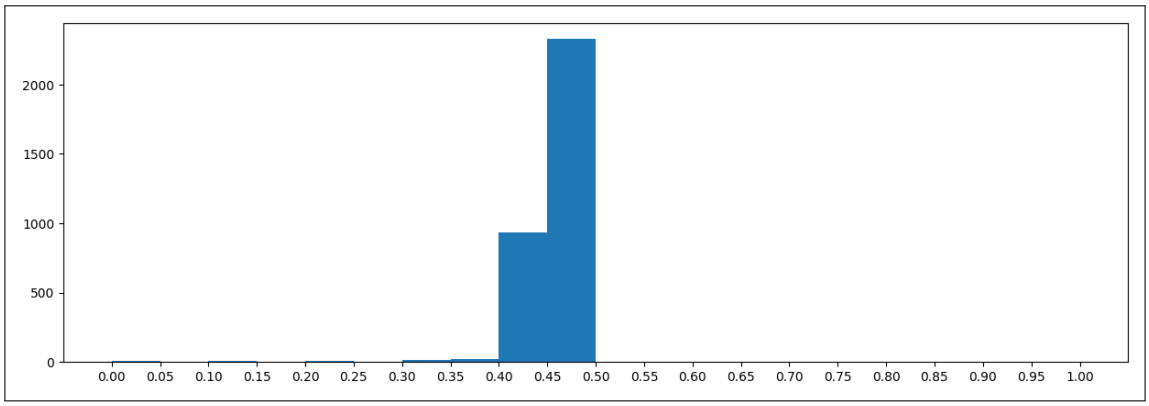

Figura 4: Histograma da distribuição da densidade de vogais no córpus. Pode-se observar que a imensa maioria das sentenças possui um valor de DV entre 0,4 e 0,5.

- **MSE - erro quadrático médio:** média do quadrado das distâncias entre os valores previstos e os valores reais.
- **RMSE - raiz quadrada do erro quadrático médio:** similar ao MSE, no entanto, o RMSE é menos sensível a valores extremos (*outliers*) do que o MSE, pois os valores não são elevados ao quadrado.
- **MAE - erro absoluto médio:** média do erro absoluto entre os valores previstos e os valores reais, é semelhante ao MSE, entretanto, não penaliza tanto os *outliers* quanto o MSE, pois no MAE os valores não são elevados ao quadrado.
- **MAPE - erro percentual absoluto médio:** representa a média das porcentagens de erro absoluto entre os valores previstos e os reais.
- R^2 **- r-quadrado ou coeficiente de determinação:** medida que representa o quanto da variabilidade dos dados o modelo consegue explicar. Quanto mais próximo de 1, melhor o modelo, mas os valores podem ser negativos, pois o modelo pode ser arbitrariamente ruim. Se o modelo sempre predisser uma saída constante, o valor de R^2 será 0.
- **Correlação de Pearson:** valor entre -1 e 1 que mede a força e a direção da relação entre o modelo e os dados Um valor próximo 0 indica menor correlação entre o modelo, valores positivos indicam uma correlação positiva, enquanto valores negativos indicam uma relação negativa.

No caso das Tarefas 2 e 3, outras métricas foram utilizadas para avaliar a qualidade do modelo de classificação:

- **Acurácia:** representa a porcentagem de predições que o modelo conseguiu acertar.
- **Sensibilidade:** representa a taxa de verdadeiros positivos, isto é, de todas as amostras pertencentes a uma classe X, quantas o modelo conseguiu classificar corretamente.
- **Especificidade:** representa a taxa de verdadeiros negativos, isto é, das amostras que não pertencem uma classe X, quantas o modelo conseguiu classificar corretamente (que pertenciam a outras classes).

A seção seguinte descreve os resultados obtidos durante os experimentos considerando as métricas apresentadas para cada uma das tarefas.

4 Experimentos e Resultados

O mesmo conjunto de dados foi utilizado em todas as tarefas. A proporção utilizada para o particionamento dos conjuntos de treinamento, validação e teste também foi a mantida igual - 75%, 15% e 15%, respectivamente. A implementação foi baseada no modelo apresentado por Jiang (2022), posteriormente, os parâmetros e métricas foram ajustados conforme a necessidade.

4.1 Tarefa 1 - Regressão de densidade de vogais

Na avaliação do desempenho do modelo de regressão, alguns *baselines* foram utilizados como referência, cada um deles corresponde ao desempenho obtido caso um valor fixo para todas as predições. Os três valores fixados como *baseline* foram: a densidade de vogais do córpus todo (DV-córpus), a densidade de vogais da primeira palavra (DV-primeira) e a densidade de vogais da última palavra (DV-última).

A Tabela 1 apresenta os valores das métricas para cada *baseline*, juntamente com os valores de desempenho do modelo refinado a partir do BERTimbau. O modelo foi refinado para a tarefa de regressão utilizando 10 épocas de treinamento, com os parâmetros: `weight_decay=0.001`, `learning_rate=5e-5` e `metric_for_best_model='mse'` e tamanho do *batch* igual a 64. Um número entre 10 e 35 épocas testado nos experimentos, no entanto, o decidiu-se manter o número 10, pois o modelo começava a sofrer *overfitting* ao se acrescentarem mais épocas.

Tabela 1: Comparação do desempenho do modelo de regressão em relação aos *baselines*. Os melhores valores de cada métrica estão destacados em **negrito**. A correlação de Pearson retornou valor *nan* para DV-córpus, pois não faz sentido medir a correlação com um vetor onde todos os valores são iguais.

	DV-córpus	**DV-primeira**	**DV-última**	**Modelo**
MSE	**0,0014**	0,0476	0,0222	0,0016
RMSE	**0,0377**	0,2183	0,1488	0,0394
MAE	**0,0233**	0,1412	0,0933	0,0333
MAPE	**0,0621**	0,2956	0,1940	0,0739
R^2	**0**	-32,5110	-14,5810	-0,0910
Pearson	*nan*	0,2170	0,2810	**0,7494**

As medidas de erro obtidas nos experimentos, indicam que o modelo conseguiu uma aproximação muito próxima dos valores reais. Apesar disso nota-se que o modelo superou o *baseline* DV-córpus apenas na medida de correlação de Pearson. Isso indica que o modelo foi capaz de encontrar uma certa correlação entre os dados. No entanto, a correlação encontrada não foi suficiente para que as medidas de erro (MSE, RMSE, MAE, MAPE e R^2) superassem o *baseline* DV-córpus.

Acredita-se que esses resultados ocorreram não devido a um mau desempenho do modelo, mas devido à característica da distribuição dos dados (Figura 4). O valor da DV do córpus é 0,48, assim ele está muito próximo do valor da DV da maior parte das sentenças, logo as medidas de erro retornarão valores pequenos para este *baseline*.

Com base nos outros dois *baselines* (DV-primeira e DV-última), é possível observar que o modelo foi melhor do que um outro modelo que atribuísse sempre um valor correspondente à primeira ou à última palavra. Desse modo, pode-se concluir que o modelo gerado a partir do BERTimbau obteve resultados satisfatórios na tarefa de regressão.

4.2 Tarefa 2 - Classificação com dados quantizados não balanceados

Para esta tarefa, visto que a maior parte das sentenças possui densidade de vogais entre 0,4 e 0,5 (Figura 4), é esperado que a imensa maioria das sentenças seja agrupada em uma única classe (onde $\frac{1}{3} \leq DV < \frac{2}{3}$) após o particionamento dos dados. De fato, a distribuição das amostras foi de 37 amostras na primeira classe, 9946 amostras na segunda e apenas 8 amostras na terceira.

Neste caso, dado que as classes estão extremamente desbalanceadas, basta que o modelo atribua o rótulo correspondente à classe de maior amostragem para ter uma acurácia maior que 99%. Os resultados na Tabela 2 mostram que o modelo conseguiu atingir essa acurácia, conforme previsto.

Tabela 2: Desempenho do modelo de classificação para a Tarefa 2, após 10 épocas de treinamento. São apresentadas as métricas obtidas no conjunto de teste, separadas por classe (C0, C1 e C2) e total.

	Acurácia	**Sensibilidade**	**Especificidade**
C0	0,9953	1,0000	0,9953
C1	0,9947	0,9953	0,8750
C2	0,9993	0,5000	1,0000
Total	**0,9947**	**0,8318**	**0,9568**

Durante os experimentos da Tarefa 2, também foram utilizadas 10 épocas de treinamento e os demais parâmetros utilizados na Tarefa 1 foram mantidos. Foi possível observar que logo na primeira época de treinamento, o modelo obteve acurácia de 0,99, sendo que a sensitividade (verdadeiros positivos) da classe C1 foi de 1,0, no entanto a especificidade (verdadeiros negativos) foi de apenas 0,25. Isso indica que o modelo conseguiu classificar corretamente todas as amostras da classe C1, mas atribuiu o rótulo de C1 para muitas amostras pertencentes às outras classes.

Após a execução das demais épocas de treinamento, observou-se que o modelo conseguiu melhorar a sensibilidade e a especificidade total, chegando a 0,95 para ambas as métricas, na partição de validação. Esse resultado ressalta a importância da utilização de múltiplas métricas na avaliação de um modelo, pois as métricas de sensibilidade e especificidade conseguiram capturar um comportamento que não poderia ser visualizado observando apenas a acurácia, visto que esta permaneceu praticamente inalterada durante o treinamento.

4.3 Tarefa 3 - Classificação com dados quantizados balanceados

Para esta tarefa, as amostras foram ordenadas pela densidade de vogais, e em seguida particionadas em 3 grupos de mesmo tamanho. Como se vê na Figura 5, devido ao grande número de amostras que possuem uma densidade de vogais entre 0,4 e 0,5, as três classes contêm amostras com valores nessa faixa.

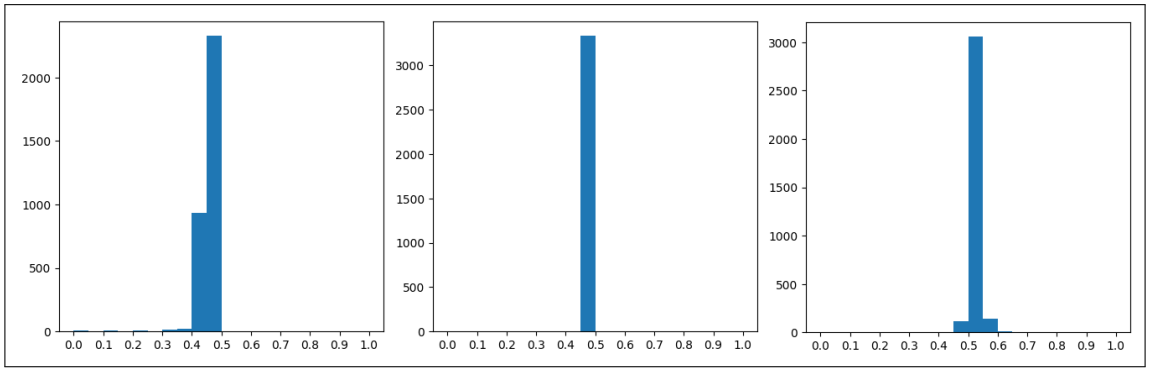

Figura 5: Histograma da distribuição das amostras nas classes balanceadas C0, C1 e C2, respectivamente.

Os parâmetros de treinamento para este modelo permaneceram os mesmos das tarefas anteriores. Alguns experimentos alterando os parâmetros de `learning_rate`, `weight_decay` e tamanho do *batch* foram realizados, no entanto, como o desempenho do modelo variou pouco, optou-se por manter os valores dos parâmetros inalterados.

A Tabela 3 mostra os resultados obtidos pelo modelo após o treinamento. Caso o modelo apenas atribuísse todas as amostras para uma única classe, a acurácia seria de 0,33. Caso o modelo encontrasse um determinado

limiar e classificasse as amostras com DV maior que o limiar para classe C2 e os menores para a classe C0 (errando todos os da classe C1), a acurácia seria em torno de 0,66. Como a acurácia final foi de 0,72, pode-se concluir que o modelo obteve um desempenho satisfatório.

Tabela 3: Desempenho do modelo de classificação para a Tarefa 3, após 10 épocas de treinamento. São apresentadas as métricas obtidas no conjunto de teste, separadas por classe (C0, C1 e C2) e total.

	Acurácia	**Sensibilidade**	**Especificidade**
C0	0,8566	0,6855	0,9453
C1	0,7298	0,6908	0,7500
C2	0,8612	0,8004	0,8895
Total	**0,7238**	**0,7256**	**0,8616**

5 Conclusão

A partir dos experimentos e dos resultados obtidos, foi possível verificar que o modelo BERTimbau pode ser refinado tanto para tarefas de regressão quanto de classificação, apesar de originalmente ele ser treinado em tarefas de classificação. O trabalho demonstrou que nas três tarefas propostas, os modelos construídos a partir do BERTimbau obtiveram desempenho satisfatório, de acordo com as métricas analisadas.

Uma outra contribuição notável deste trabalho foi destacar a importância da utilização de diferentes métricas de avaliação durante a construção de um modelo de aprendizado. Na Tarefa 1, o Coeficiente de Pearson foi fundamental para verificar que o modelo treinado havia obtido resultados melhores do que o *baseline* DV-córpus. Semelhantemente, na Tarefa 2, as métricas de sensibilidade e especificidade permitiram a identificação da melhoria do modelo ao longo do treinamento, apesar da acurácia total permanecer inalterada.

Referências

Jay Allamar. The illustrated transformer, 2018. URL `http://jalammar.github.io/illustrated-transformer/`.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp (eds.), *Advances in Neural Information Processing Systems*, volume 13. MIT Press, 2000. URL `https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

Jeremy Howard and Sebastian Ruder. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146, 2018. URL `http://arxiv.org/abs/1801.06146`.

Umar Jamil. Bert explained from scratch, 2023. URL `https://github.com/hkproj/bert-from-scratch`.

Jinhang Jiang. Linear regression with hugging face, 2022. URL `https://towardsdatascience.com/linear-regression-with-hugging-face-3883fe729324`.

Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021. ISSN 2212-8271. doi: https://doi.org/10.1016/j.procir.2021.03.088. URL `https://www.sciencedirect.com/science/article/pii/S2212827121003796`. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL `http://arxiv.org/abs/1508.04025`.

Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016. doi: 10.23915/distill.00001. URL http://distill.pub/2016/augmented-rnns.

Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. *Preprint*, 2018. URL https://api.semanticscholar.org/CorpusID:49313245.

Fábio Souza, Rodrigo Nogueira, and Roberto Lotufo. Portuguese named entity recognition using bert-crf. *arXiv preprint arXiv:1909.10649*, 2019. URL http://arxiv.org/abs/1909.10649.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jorge A. Wagner Filho, Rodrigo Wilkens, Marco Idiart, and Aline Villavicencio. The brWaC corpus: A new open resource for Brazilian Portuguese. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (eds.), *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL https://aclanthology.org/L18-1686.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. URL http://arxiv.org/abs/1911.02685.