COPPE
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia
UFRJ

APRENDIZADO DE MÁQUINA APLICADO A CLASSIFICAÇÃO DE DOCUMENTOS JURÍDICOS EM AMBIENTE DE BAIXO PODER COMPUTACIONAL

Izandro Monteiro Metello

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Setembro de 2022

APRENDIZADO DE MÁQUINA APLICADO A CLASSIFICAÇÃO DE DOCUMENTOS JURÍDICOS EM AMBIENTE DE BAIXO PODER COMPUTACIONAL

Izandro Monteiro Metello

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Geraldo Bonorino Xexéo

Aprovada por: Prof. Geraldo Bonorino Xexéo
Prof. Geraldo Zimbrão da Silva
Prof. Carlos Eduardo Ribeiro de Mello

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2022

Metello, Izandro Monteiro

Aprendizado de máquina aplicado a classificação de documentos jurídicos em ambiente de baixo poder computacional/Izandro Monteiro Metello. – Rio de Janeiro: UFRJ/COPPE, 2022.

XIV, 71 p. 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2022.

Referências Bibliográficas: p. 66 – 71.

1. Processamento de Linguagem Natural. 2. Intimações. 3. Aprendizado Profundo. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedico este trabalho a todos os que me ajudaram ao longo desta caminhada.

Agradecimentos

Agradeço primeiramente à São Benedito. A minha mãe que acredita no meu potencial e me apoia sempre. Agradeço a minha avó que enquanto estava entre nós me apoiou incondicionalmente. Agradeço à Alice, minha companheira amada que me apoia na dificuldade e na felicidade, que me da energia para seguir em frente. À minha irmã, Edeizi, que sinto saudades de ficar conversando sobre as coisas da vida. Agradeço a minhas gatas Jade e Selena por serem fofas e ficarem subindo na mesa alegrando meu dia. Agradeço ao meu orientador, professor Geraldo Xexéo, que nessa longa jornada sempre me ajudou e não desistiu de mim. Sou grato aos meus colegas de riso e choro Marcelle, Elaine, Arthur, Débora, Rafael, Airine, Ricardo. Agradeço a equipe CICIAR que me acolheu e com os quais sempre aprendo muito. Agradeço ao Gustavo Pinto, a Coppetec, e a galera da equipe Nexo, depois Siga, todo mundo que trabalhou comigo e pude colaborar. Agradeço ao CNPq pelo apoio e por acreditar no meu trabalho. Agradeço a todos de Cuiabá que acreditaram que eu podia fazer mais e melhor: meus amigos, especialmente tiozões Pedro Ivo, Fernando, Lorena. Nunca vou me esquecer também, professor Einstein Aguiar, fantástico! Agradeço a minha família do Rio: Guilherme, Vivian, Vanessa. Por conta de vocês nos sentimos em casa aqui também! Agradeço a Si Fu Úrsula Lima, André, Helena, que pude ter vida kung-fu e aprendi demais. Agradeço a todos que direta ou indiretamente me ajudaram a passar por essa pandemia e continuar lutando. Com todas essas pessoas eu aprendi que sempre que eu apostasse em mim mesmo nunca me decepcionaria.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

APRENDIZADO DE MÁQUINA APLICADO A CLASSIFICAÇÃO DE DOCUMENTOS JURÍDICOS EM AMBIENTE DE BAIXO PODER COMPUTACIONAL

Izandro Monteiro Metello

Setembro/2022

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

O retreino parcial de modelos BERT pode proporcionar grande parte do ganho de desempenho comparado ao retreino de todas as camadas do modelo e ao mesmo tempo economiza recursos computacionais. No caso da Caixa de Intimações Com Inteligência Artificial (CICIAR), o desempenho do retreino parcial do modelo BERT foi medido em relação ao modelo de referência.

O CICIAR é uma ferramenta desenvolvida para a Defensoria Pública do Estado do Rio de Janeiro e é integrada ao sistema Verde. O sistema Verde ajuda a gestão e as atividades diárias dos Defensores Públicos. Neste contexto, o CICIAR faz previsões de rótulos de intimações utilizando informações de contexto sem abri-las. A abertura da intimação inicia uma contagem regressiva de 10 dias, portanto, o sistema deve prever o rótulo sem abri-la.

Os experimentos foram realizados liberando uma fração de camadas do modelo para o treinamento e adaptando os hiper-parâmetros. A base de dados utilizada contém intimações e contextos das intimações em português brasileiro.

Os experimentos mostraram melhorias no desempenho da classificação com as mudanças propostas. As configurações de base utilizadas na aplicação têm uma precisão ponderada de 0,74. A melhor combinação aumenta este resultado para 0,826, refinando a última camada do modelo BERT e modificando os hiperparâmetros.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

MACHINE LEARNING APPLIED IN LEGAL TEXT CLASSIFICATION WITH LOW COMPUTING POWER

Izandro Monteiro Metello

September/2022

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

Partial retraining of layers BERT can deliver most of the increase in performance as complete retraining of BERT layer while saving computational resources. In the Caixa de Intimações Com Inteligência Artificial (CICIAR) context, the results of partially retraining the BERT model have been evaluated against a baseline model.

CICIAR is a tool developed for the Public Defender's Office of the State of Rio de Janeiro integrated with the Verde system. The Verde system helps with the management and daily tasks of Public Defenders. In this context, CICIAR makes predictions of subpoena labels using context information without opening them. Opening the subpoena starts a counting down of 10 days, so the system must predict the label without opening it.

The experiments have been performed by releasing a model fraction of layers for retraining and adapting the hyperparameters. The database used contains subpoenas and subpoena contexts in Brazilian Portuguese.

The experiments showed improvement in classification performance with the proposed changes. The baseline configurations used in the application have a weighted accuracy of 0.74. The best combination raises this result to 0.826 by refining the last layer of the BERT model and modifying hyperparameters.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Outrora as técnicas de classificação de texto utilizando eram utilizadas principalmente para sistemas de recuperação de informações. No entanto, à medida que os avanços tecnológicos surgiram ao longo do tempo a categorização de documentos e a classificação de texto têm sido usadas globalmente em muitas áreas, como medicina, ciências sociais, saúde, psicologia, direito, engenharia, etc KOWSARI *et al.* (2019).

O uso de tais técnicas é especialmente importante em áreas como o direito que é exclusivamente dependente da análise de extenso volume de informações textuais, motivo pelo qual tem-se explorado o potencial das técnicas de Processamento de Linguagem Natural (PLN) em otimizar a realização de muitas dessas tarefas (MASTELLA, 2020).

Nos últimos anos dentro do campo jurídico a prática da busca de informações evoluiu de consultas a acervos físicos como livros, revistas de Direito e legislação em cópia impressa para o uso de bibliotecas digitais. Como consequência, houve uma informatização de advocacias e departamentos jurídicos PEDRUZZI (2020).

No Brasil o setor privado e órgãos governamentais também estão migrando seus acervos para o meio digital. Como parte dessa transição, buscam incrementá-los com algumas técnicas de Inteligência Artificial (IA), que têm sido utilizadas para possibilitar a busca de informações de forma mais rápida com menos custo e maior precisão.

Considerando o grande volume de intimações recebidas todos os dias, o consequente impacto na carga de trabalho e rotina dos defensores públicos, a defensoria Pública do Rio de Janeiro (DPRJ) foi motivada a inovar.

A complexidade inerente do trabalho e a profusão diária de intimações preocupam os responsáveis pelo planejamento, que buscaram investigar como o uso de IA poderia colaborar no âmbito do direito (PARREIRAS *et al.*, 2022).

O crescente interesse da DPRJ pelo tema motivou uma parcela dos defensores a buscar como este tipo de tecnologia poderia impactar positivamente na produtividade do órgão e consequentemente melhorar o atendimento à comunidade. As

várias motivações encontradas para o uso de IA na DPRJ, algumas são:

- Melhor distribuição de trabalho;
- Melhoria na gestão;
- Possibilidade de antecipação das intimações;
- Possibilidade de criação de fluxos de trabalho; e
- Possibilidade de planejamento a longo prazo.

Neste contexto, a Caixa de Intimações com Inteligência Artificial (CICIAR) foi criado para auxiliar defensores públicos a tratar e priorizar o atendimento das intimações recebidas na plataforma digital usada para gerenciar o acompanhamento de processos, denominada Verde. O CICIAR categoriza as intimações sem oficialmente acessar as mesmas. O acesso inicia a contagem do prazo para atendê-las. A classificação prévia é a principal motivação para a criação do CICIAR. O Defensor Público tem um prazo de 10 dias para responder uma intimação ao abri-la. O CICIAR pode auxiliar o planejamento do Defensor Público antes mesmo de abrir o documento, podendo priorizar casos urgentes.

Este sistema depende do compartilhamento de conhecimento dos próprios Defensores para gerar os dados necessários para o treinamento da IA. Cabe a eles realizar a classificação das intimações que não foram rotuladas pelo sistema e a avaliar aquelas que já classificadas pelo modelo PARREIRAS *et al.* (2022).

1.1 Os custos de trabalhar com IA no Brasil

O custo do acesso a equipamentos de ponta para pesquisadores e profissionais da área de Inteligência Artificial é alto no Brasil. Por exemplo, o treinamento de um modelo BERT, segundo (BER, 2022), em máquinas virtuais do modelo Cloud TPUv3 pode consumir quatro dias utilizando de 4 até 16 máquinas em paralelo.

De acordo com a literatura o recomendado é realizar o pré-treino do modelo que em uma máquina pode chegar a consumir duas semanas chegando a um dispêndio de cerca de 2880 dólares, R$14.897,95 no câmbio atual (valores de 17/08/22). Em um processo de desenvolvimento de natureza exploratória como exemplo os trabalhos LEE *et al.* (2019), DEVLIN *et al.* (2018), RADFORD *et al.* (2018), que utilizam aprendizado profundo, descrevem o processo de escolha de hiper parâmetros sendo necessário executar dezenas de vezes os experimentos. Esse custo encarece pesquisas dessa natureza e restringe o desempenho de atividades.

De acordo MOLLOY (2021) soma-se a essas barreiras a escassez de placas de vídeo, a inflação por conta da pandemia de SARS-COV-2 e o alto valor do câmbio

do Dólar para o Real. A título de exemplo o custo de uma placa de vídeo de linha profissional com relevante memória de vídeo (48 GiB) capaz de executar tarefas de pré-treino do BERT-Large é de 4949 dólares, no câmbio atual R$25.581,88 (valores de 17/08/22). A realização do pré-treino só é possível com placas de 48 GiB de memória de vídeo dado o requisito para acomodar o *batch* recomendado de amostras. Além do Câmbio, para realizar compras de placas de vídeo desse porte no Brasil em lojas brasileiras o valor pode chegar até proibitivos R$65.400,00.

1.2 Objetivos

O objetivo dessa dissertação é avaliar o impacto no desempenho da classificação de texto no caso do do CICIAR por meio da aplicação do retreino de camadas do modelo pré-treinado BERT no processo de treinamento do classificador no contexto de restrição de recursos computacionais.

Especificamente, busca-se avaliar o desempenho da arquitetura de Transformadores e a implementação utilizada do BERT mediante a aplicação do retreino de camadas durante o processo. Durante o processo de experimentações, monitorar consumo de recurso computacional e discutir os resultados de classificação. Com base nos resultados de classificação encontrados no processo de experimentações, verificar o efeito da mudança no treinamento no contexto do CICIAR. Por fim, discutir o uso recursos computacionais e a viabilidade da implantação no sistema atual.

1.3 Organização

O capítulo 2 aborda a revisão bibliográfica dos algoritmos e técnicas usados nos treinos e avaliações, além do processo e requisitos para treinar os modelos. O capítulo 3 lista a metodologia e descrição dos dados, dos sistemas que fazem parte do contexto e da técnica usada para melhoria do classificador. O capítulo 4 é feita a listagem dos resultados de classificação e uso de recursos e a discussão das implicações. No capítulo 5 são feitas as conclusões e discussões dos resultados encontrados ao longo das experimentações.

Capítulo 2

Revisão Bibliográfica

A classificação de textos por meio de IA tem sido tratada por técnicas de Processamento de Linguagem Natural, utilizando abordagens como *Deep Learning*, e ferramentas como o BERT para gerar representações vetoriais que facilitam e possibilitam o tratamento computacional dos dados. No contexto de restrição de poder computacional e da necessidade de classificar documentos jurídicos, a seguir são apresentadas técnicas e ferramentas que fazem parte dos experimentos.

2.1 Processamento de Linguagem Natural

GOLDBERG (2016) afirma que por mais de uma década, as técnicas centrais de PLN foram dominadas por abordagens de aprendizagem de máquinas que utilizavam modelos lineares como máquinas vetoriais de suporte ou regressão logística, treinadas sobre vetores com muitas dimensões porém muito esparsas e que "recentemente o campo tem visto algum sucesso em mudar de tais modelos lineares comentradas esparsas para modelos de redes neuronais não lineares sobre entradas densas".

ALMEIDA e XEXÉO (2019) definem os *embeddings* que são representações densas de entradas de texto:

- *Word embeddings* são representações densas;
- São vetores de palavras distribuídos;
- Esses vetores têm comprimento fixo;
- Os vetores são construídos usando estatísticas de ocorrência de palavras, de acordo com a hipótese de distribuição.

Um modelo de linguagem é uma função ou um algoritmo para aprender uma função que captura as características estatísticas salientes da distribuição de sequências

de palavras em uma linguagem natural, permitindo tipicamente fazer previsões probabilísticas da próxima palavra dada anteriormente (BENGIO, 2008).

BENGIO *et al.* (2000) afirma que diferente dos modelos de n-grama baseados em classes, modelos de linguagem do tipo neural são capazes de reconhecer duas palavras semelhantes sem perder a capacidade de codifica-las como distintas uma da outra .

Os modelos de linguagem neurais compartilham a força estatística entre uma palavra (e seu contexto) e outras palavras e contextos similares. A representação distribuída que o modelo aprende para cara palavra, segundo (BENGIO *et al.*, 2000), "permite este compartilhamento 2.1, permitindo que o modelo trate as palavras que têm características em comum de forma semelhante".

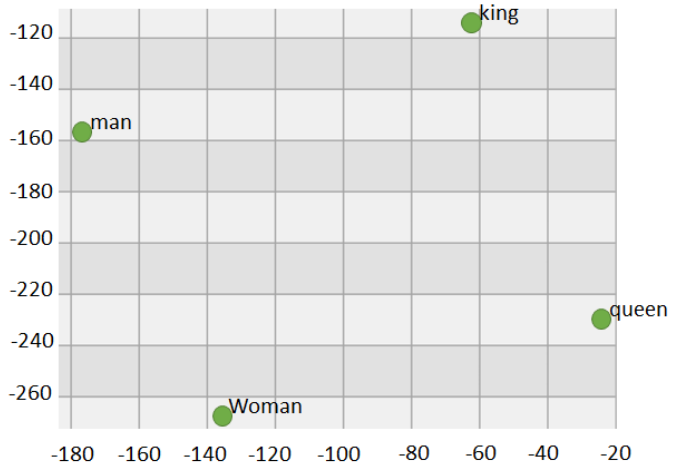

Figura 2.1: Modelos neurais mapeiam informações semânticas. Fonte: Adaptado de (ALAMMAR, 2017)

2.2 Processamento de texto com Redes Neurais

2.2.1 Recurrent Neural Network

Uma Rede Neural Recorrente (do inglês *Recurrent Neural Network* - RNN), Figura 2.2, é uma extensão de uma rede neural *feed-forward* convencional capaz de aceitar entradas de tamanhos variados CHUNG *et al.* (2014). A RNN consegue processar sequências de tamanho variável por meio de uma estrutura recorrente na camada oculta. A ativação da camada oculta depende da entrada anterior. Dada uma sequencia $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T)$ a RNN atualiza os estados recorrentes h_t:

$$\mathbf{h}_t = \begin{cases} 0, & t = 0 \\ \phi(h_{t-1}, x_t), & \text{caso contrário} \end{cases} \tag{2.1}$$

onde ϕ é uma função não linear. A RNN pode ter saída variável $\mathbf{y} = (\mathrm{y}_1, \mathrm{y}_2, ..., \mathrm{y}_T)$. A atualização dos valores da camada oculta da RNN é feita por:

$$\mathbf{h}_t = (g(W\mathbf{x}_t + U\mathbf{x}_{t-1})), \tag{2.2}$$

onde g é uma função suave, limitada como funções sigmoide ou tangente hiperbólica.

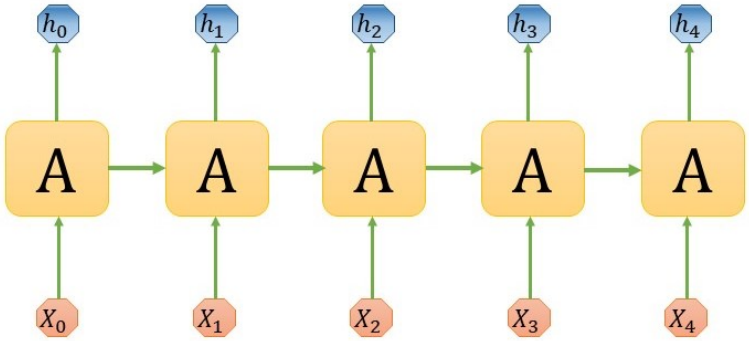

Figura 2.2: Rede neural recorrente. Fonte: Adaptado de (CHUNG *et al.*, 2015).

BENGIO *et al.* (1994) notaram que é difícil treinar RNNs para aprender dependências com grande distância porque os gradientes tendem a desaparecer ou explodir.

2.2.2 *Long Short-Term Memory*

Modelos de Linguagem baseados em redes neurais recorrentes, introduzidos por MIKOLOV *et al.* (2010), têm topologia semelhante a de redes *feedforward*, porém com pesos que geram caminhos recorrentes em camadas internas. Como consequência, o resultado processamento da rede para uma dada palavra depende de palavras que antecedem o termo atual.

Uma das vantagens da RNN comparado a redes *feedforward* é a redução da dependência de um contexto de tamanho pré-definido, podendo tirar proveito de dependências entre termos com maior distância. Porém, segundo BENGIO *et al.* (1994) ao mesmo tempo que as RNN facilitam a utilização de uma informação de grande extensão para treino, foi verificado que os algoritmos de treino baseados em gradiente podem falhar no aprendizado de pesos da RNN quando termos dependentes com longa distância são explorados. Isso se deve ao fato de, quando há um aumento na distancia das dependência, o calculo do gradiente fica cada vez mais instável.

Os valores dos gradientes podem explodir ou decair exponencialmente com o incremento do tamanho do contexto. Como solução para este problema, uma nova

arquitetura foi proposta para aprimorar a RNN(HOCHREITER e SCHMIDHUBER, 1997), com subsequentes evoluções propostas por GERS *et al.* (2000) e GERS *et al.* (2002).

O resultado dessa RNN aprimorada é referido como rede neural *Long Short-Term Memory* (LSTM), ilustrada na Figura 2.3.

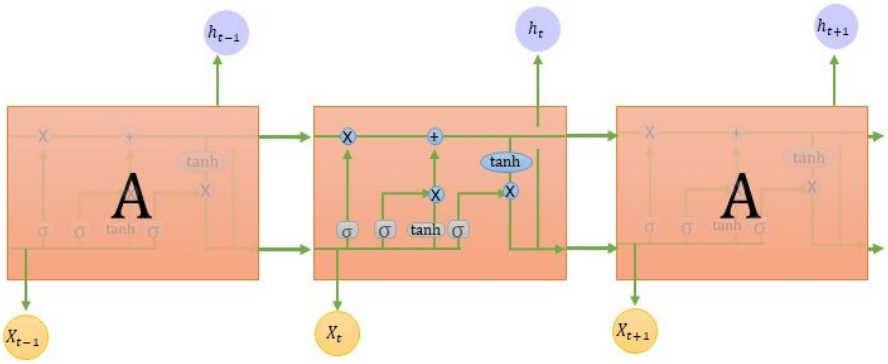

Figura 2.3: Rede neural recorrente com células LSTM. Fonte: Adaptado de (OLAH, 2018)

O treinamento da rede é feito com o *Stochastic Gradient Descendent* (BOTTOU, 2012). Para treinar redes *feedforward* o gradiente é calculado com o algoritmo *backpropagation* (RUMELHART *et al.*, 1985). Para obter os gradientes de uma rede neural recorrente (RNN) existem outras versões do *Backpropagation Through Time* (BPTT).

2.2.3 Transformadores

A arquitetura de Transformadores (foi apresentada em 2017 VASWANI *et al.* (2017) para "superar limitações de tamanho de contexto por meio do mecanismo de atenção" (SINGH e MAHMOOD, 2021). Desde então surgiram modelos como GPT (RADFORD *et al.*, 2018) e o BERT (DEVLIN *et al.*, 2018). Modelos Competitivos de *sequence transduction* normalmente têm estruturas de codificador-decodificador (VASWANI *et al.*, 2017).

Nessa estrutura, o codificador mapeia uma entrada que consiste em uma sequência de símbolos $(x_1, ..., x_n)$ para uma sequência de elementos contínuos $Z = (z_1, ..., z_n)$. Dado um Z, o decodificador então gera uma sequencia de saída um elemento por vez. O Transformador, Figura 2.4, segue essa mesma arquitetura usando uma pilha de camadas de completamente conectadas para o codificador e o decodificador (VASWANI *et al.*, 2017).

Segundo VASWANI *et al.* (2017), mecanismos de atenção se tornaram parte de vários modelos em tarefas como modelagem de sequências ou *transduction models*,

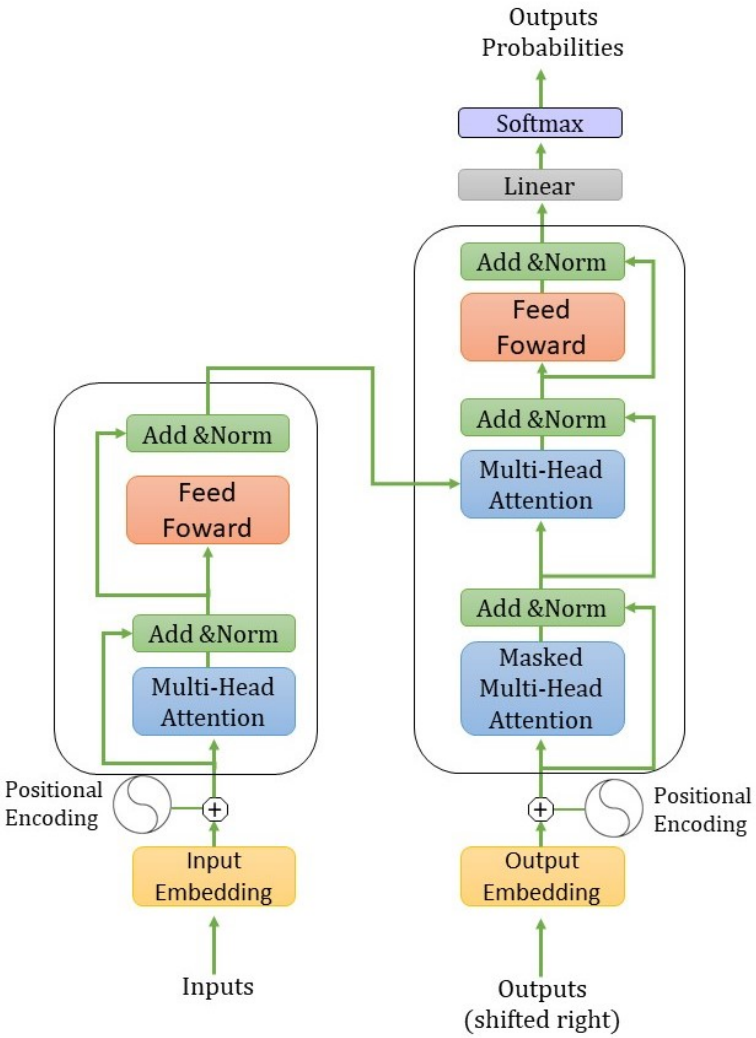

Figura 2.4: Estrutura Codificador - Decodificador. Fonte: Adaptado de (ALAMMAR, 2018a)

permitindo a modelagem de dependências independente de suas distâncias em uma sequência de entrada ou saída.

Arquiteturas como Extended Neural GPU, ByteNet e ConvS2S foram construídos pensando na redução da complexidade computacional, porém ainda existe um crescimento linear ou logarítmico com o aumento das distâncias entre posições relacionadas em uma dada entrada.

Na arquitetura de Transformadores, esse custo é reduzido a um número constante de operações. Em alguns casos mecanismos de atenção são aplicados em modelos associados a redes recorrentes. No caso dos Transformadores, é empregado o conceito de *self-attention* porém sem o uso de mecanismos de convolução ou RNNs. A Figura 2.5 demonstra a entrada do primeira camada que tem uma camada de *embeddings*.

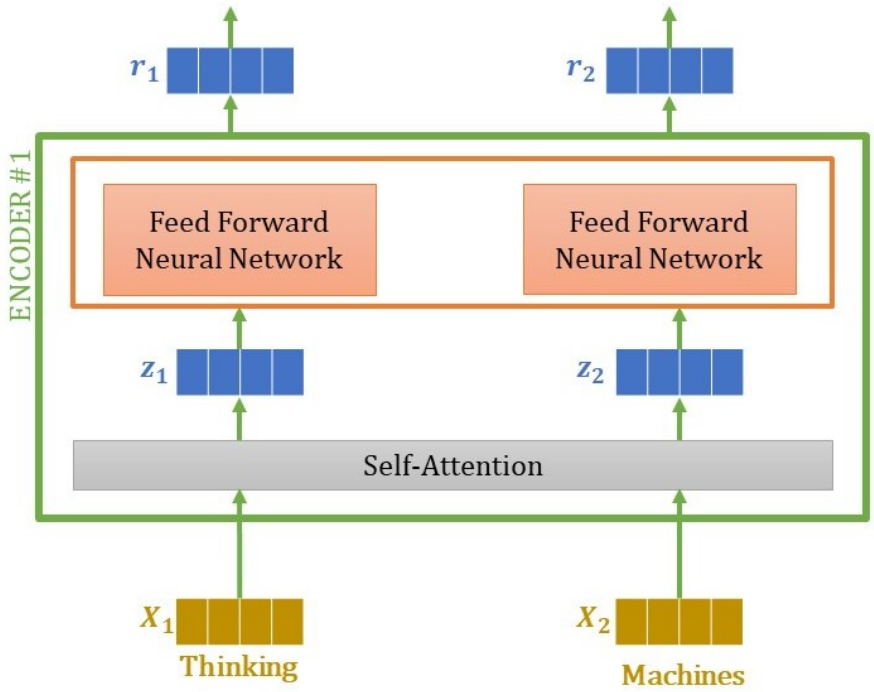

Figura 2.5: A primeira camada do codificador tem uma camada de *embedding* para converter os *tokens* para uma representação densa. Fonte: Adaptado de (ALAMMAR, 2018a)

Arquitetura do modelo

Nessa arquitetura, o codificador mapeia sequências de representações de símbolos

$$\mathbf{x} = (x_1, x_2, ..., x_n) \tag{2.3}$$

para uma sequencia de representações contínuas:

$$\mathbf{z} = (z_1, z_2, ..., z_n) \tag{2.4}$$

Dado um $\mathbf{z}$, o decodificador gera então uma saída de símbolos, um elemento por vez. A cada passo o modelo é auto-regressivo, consumindo o símbolo gerado anteriormente como uma entrada adicional para gerar o próximo.

O Transformador segue essa arquitetura geral usando camadas completamente conectadas com *self-attention*, para o codificador e o decodificador. A arquitetura de Transformadores especifica um codificador com uma pilha de $N = 6$ camadas idênticas. Cada camada tem duas sub-camadas, sendo a primeira o mecanismo *multi-head* de *self-attention* e a segunda é uma camada completamente conectada do tipo de rede densa completamente conectada.

Entre cada uma das camadas é usado uma conexão residual, seguida da camada de normalização. Todas as sub-camadas do modelo e as camadas *embedding* têm

dimensão de 512. O decodificador é também composto por uma pilha de $N = 6$ camadas idênticas.

Além das duas sub-camadas em cada camada do codificador, o decodificador adiciona uma terceira sub-camada de *Multi-Head Attention* ao longo da saída do codificador. De maneira semelhante ao codificador, são usadas conexões residuais em volta de cada sub-camada.

A função de atenção pode ser descrita como o mapeamento de uma consulta (*Query*) e um conjunto de pares de chave-valor onde a *Query*, chaves e valores e a saída são todos vetores. A saída é computada como uma soma ponderada de valores, onde o peso atribuído a cada valor é computado pela função de compatibilidade da *Query* com a chave correspondente.

Self-Attention

O mecanismo de atenção ilustrado na Figura 2.6 utilizado na arquitetura de Transformadores consiste em *queries* e *keys* de dimensão d_k e os valores para a dimensão d_v.

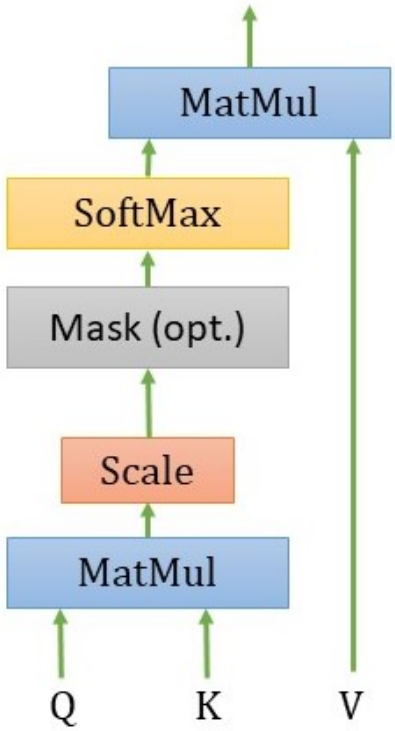

Figura 2.6: Exemplo da estrutura da camada de *self-attention*. Fonte: Adaptado de (VASWANI *et al.*, 2017)

A *self-attention* se baseia em criar três vetores *Queries*, *Keys* e *Values* para cada *token* da entrada do codificador. Cada *token* na primeira camada é convertido para

um *embedding*. Os vetores *Queries*, *Keys* e *Values* são gerados pela multiplicação de cada *embedding* por uma matriz correspondente W_q, W_k e W_v.

Em seguida é feito o cálculo do score com o produto interno dos vetores de *Query* e *Key*. Para cada *Query* é feita uma multiplicação com todas as *Keys* de outros *tokens* da sequência de entrada, gerando uma matriz de Q elementos por K chaves.

Ao resultado dessas operações, os scores são divididos por raiz da dimensão dos vetores de *Query*, *Key* e *Value* e a para cada linha correspondente à cada Q_n é aplicado *softmax* nos resultados (figura 2.7). Na arquitetura proposta por (VASWANI *et al.*, 2017), a dimensão desses vetores é 64. Os *scores* são divididos por 8. Forma-se uma matriz Z para cada cabeça de atenção do modelo.

A *self-attention* refere-se essa associação da atenção a um *token* referente a outro *token*.

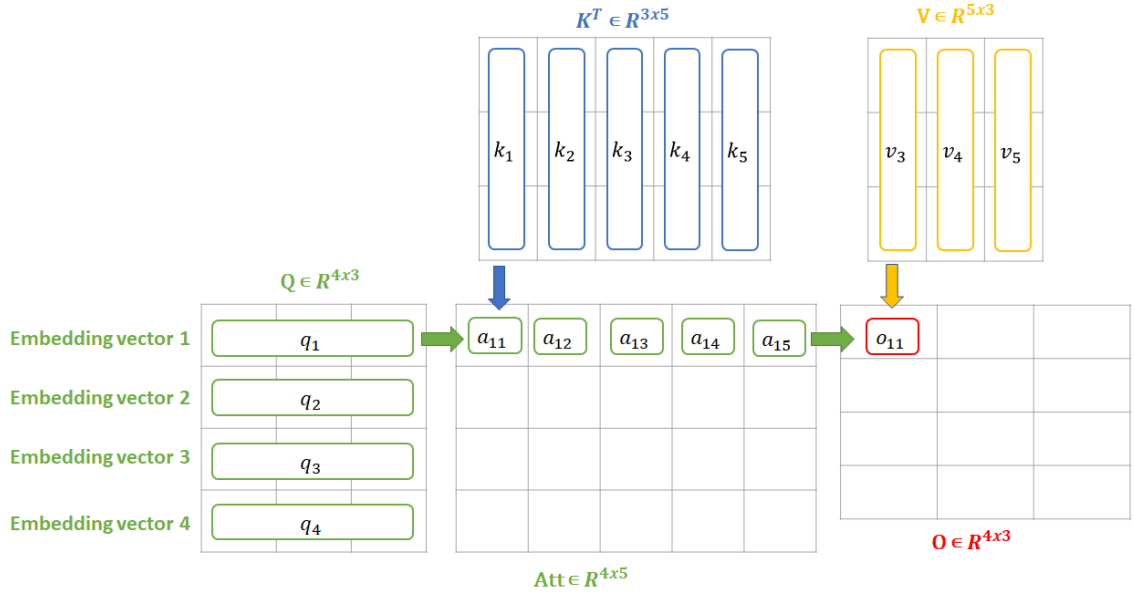

Figura 2.7: Operação da camada Self-Attention. Fonte: Adaptado de (ADALOGLOU, 2021)

Multi-headed Self-Attention

A arquitetura de Transformadores especifica várias *heads* (figura 2.8) de atenção. Cada mecanismo desses gera uma matriz de saída Z_n. Essas matrizes são concatenadas e esse resultado é multiplicado por uma matriz de pesos treinada com o modelo.

As dimensões dessa matriz de pesos são planejadas para que o resultado dessa operação Z' tenha a mesma dimensão das matrizes Z_n resultantes dos mecanismos de atenção. Na arquitetura de Transformadores, esse é o final da camada de atenção e essa saída é conectada a camada densa completamente conectada.

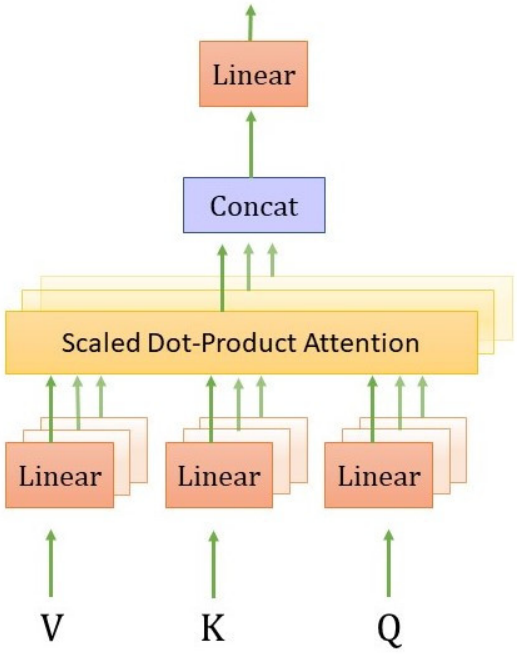

Figura 2.8: Exemplo da estrutura da camada de *multi-head self-attention*. Fonte: Adaptado de (VASWANI *et al.*, 2017)

2.2.4 Codificação de posição

Acrescenta-se um vetor a cada *embedding* de entrada e estes vetores seguem um padrão específico que o modelo aprende, ajudando a determinar a posição de cada palavra ou a distância entre as diferentes palavras na sequência. O objetivo é que a adição destes valores aos *embeddings* proporcione distâncias significativas entre os vetores de *embeddings*, já que eles são projetados em vetores de *Query*, *Key* e *Value* e durante a operação de atenção (figura 2.9) (SHIV e QUIRK, 2019).

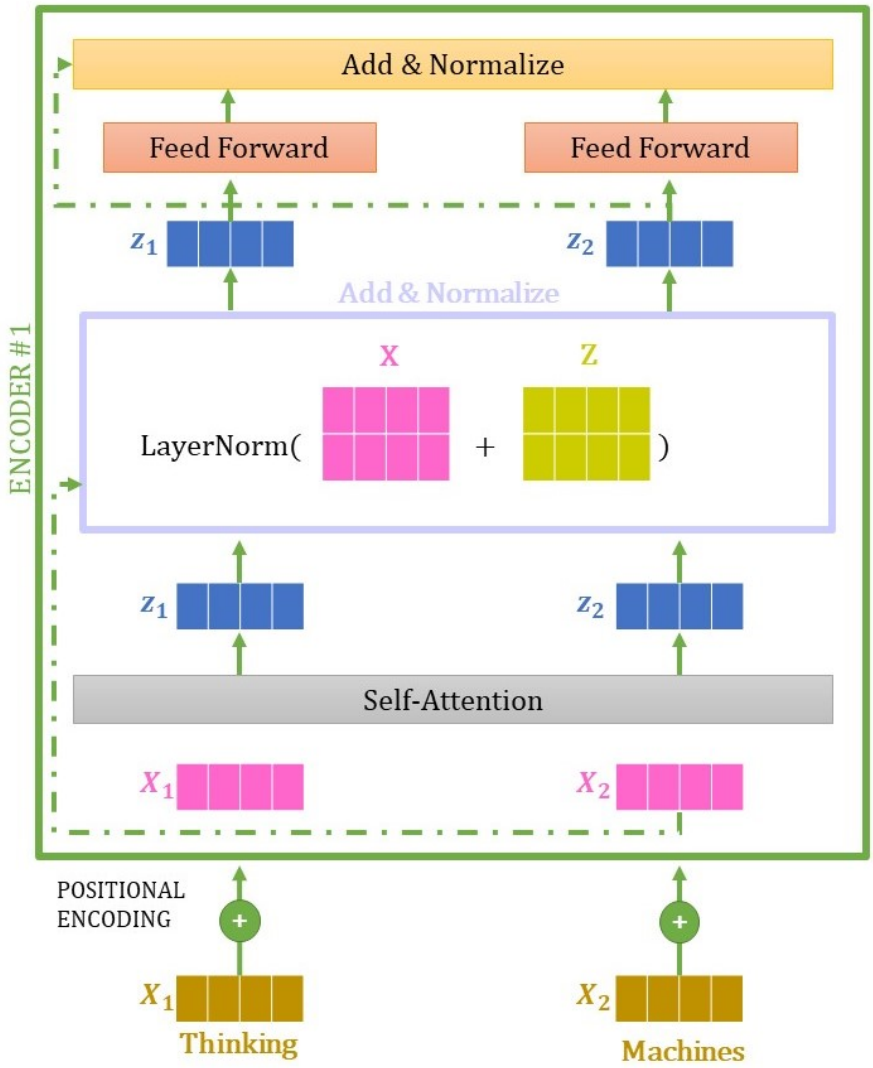

Figura 2.9: Codificação de posição e residuais. Fonte: Adaptado de (ALAMMAR, 2018a)

2.2.5 BERT

A arquitetura de Transformadores VASWANI *et al.* (2017); SINGH e MAHMOOD (2021) trouxe mudanças a geração de *embeddings* como o uso de mecanismos de atenção para superar limitações de contexto. Em 2018 RADFORD *et al.* (2018); SINGH e MAHMOOD (2021) introduziram modelos como GPT e BERT que se beneficiam da arquitetura de Transformadores foram apresentados e se posicionaram como estado da arte SINGH e MAHMOOD (2021); YANG *et al.* (2019).

Na Figura 2.10, BERT (do inglês *Bidirectional Encoder from Transformers* é uma implementação da arquitetura de Transformadores. Segundo DEVLIN *et al.* (2018), existem duas estratégias principais para utilizar um modelo de linguagem pré-treinado. A primeira estratégia é abordagem de *fine-tuning*, fazendo o refino em

todos os parâmetros pré-treinados.

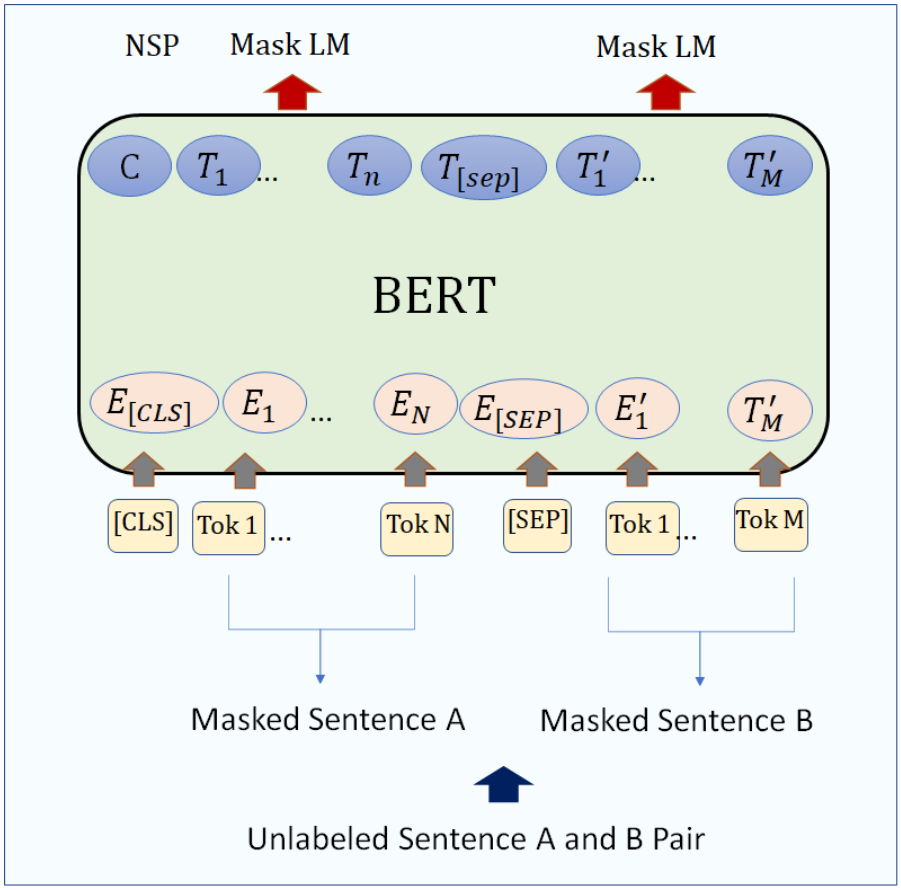

Figura 2.10: Arquitetura BERT. Fonte: Adaptado de (DEVLIN *et al.*, 2018)

A segunda estratégia é o processo de pré-treino. O pré-treino é um treinamento inicial do modelo inspirado na tarefa Cloze TAYLOR (1953) chamado *Masked Language Model* (MLM) ilustrado na Figura 2.11 (SALAZAR *et al.*, 2019). DEVLIN *et al.* (2018) afirma que essa forma de treino estimula o aprendizado da estrutura bidirecional do *Tranformer* do BERT.

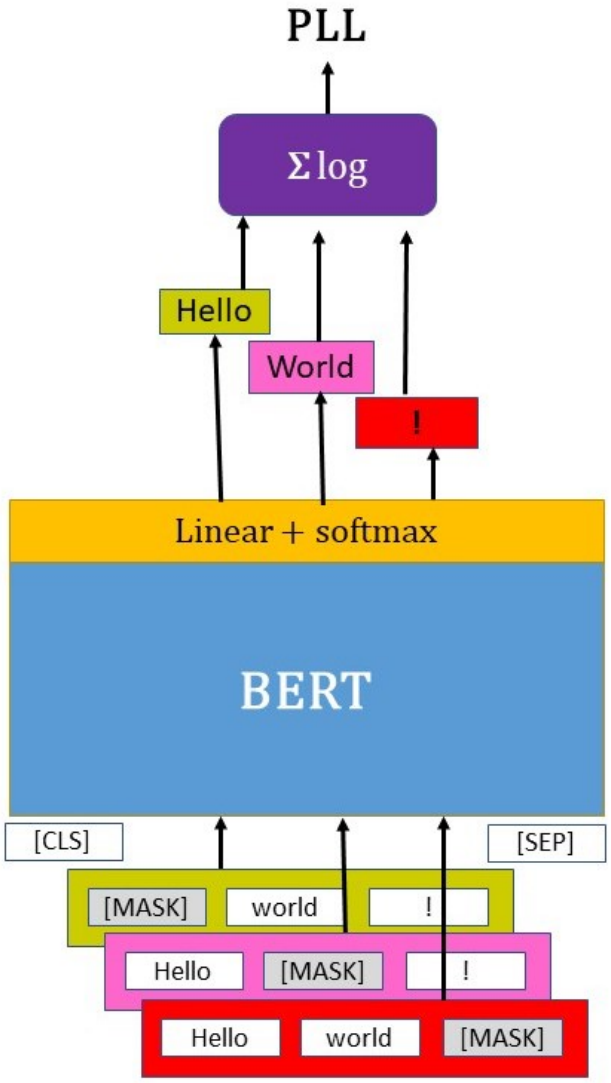

Figura 2.11: Masked Language Model - MLM. Fonte: Adaptado de (SALAZAR *et al.*, 2019)

O método MLM faz substituições como a tarefa Cloze, trocando 15% dos *tokens* da entrada por um *token* [MASK], uma palavra errada ou mantém a palavra original. O objetivo de fazer a predição do vocabulário original.

Outra tarefa utilizada no treinamento é a *"Next Sentence Prediction"* (NSP) SUN *et al.* (2021) ilustrado na Figura 2.12). Em diversas atividades é necessária a compreensão entre sentenças. Para isso, DEVLIN *et al.* (2018) faz o pré-treino com sentenças extraídas de um corpus, com duas sentenças que são 50% das vezes sequencia uma da outra ou não.

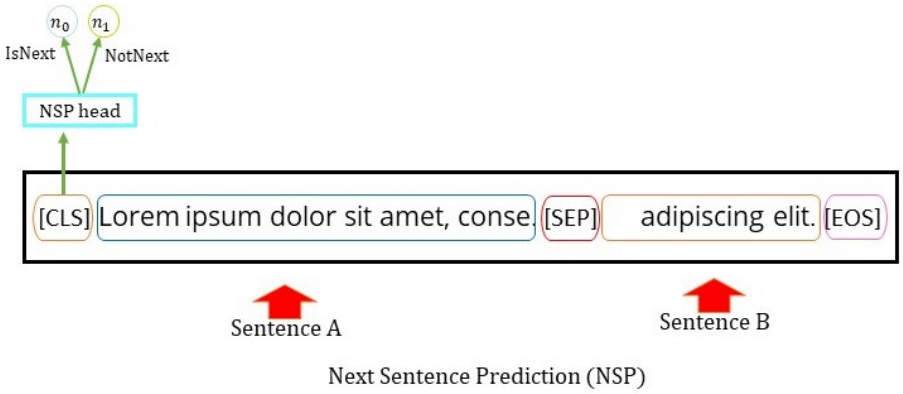

Figura 2.12: Predição de próxima sentença. Fonte: Adaptado de (SUN *et al.*, 2021)

A notação utilizada por VASWANI *et al.* (2017) para descrever a arquitetura é L para número de camadas do Transformadores, o tamanho da camada oculto como H, e o número de cabeças de *self-attention* como A. As configurações geradas, ilustradas na Figura 2.13, foram os modelos BERT-BASE (L=12, H=768, A=12, total de parâmetros = 110 milhões) e BERT-LARGE (L=24, H=1024, A=16, total de parâmetros = 340 Milhões). A dimensão da camada oculta é um vetor de saída de cada camada correspondente a cada *token* de entrada.

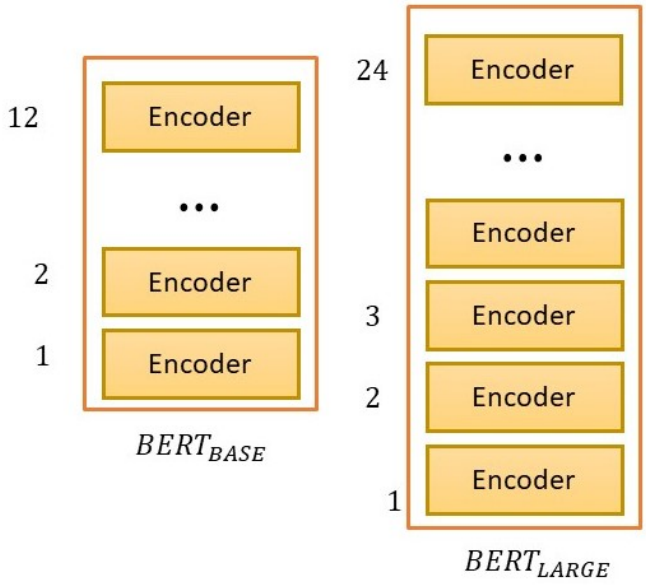

Figura 2.13: Configurações das dimensões BERT Base e Large. Fonte: Adaptado de (ALAMMAR, 2018b)

RADFORD *et al.* (2018) lista tarefas que podem ser aplicadas para refino de um Transformador que incluem classificação, inferência, similaridade e responder perguntas (figura 2.14).

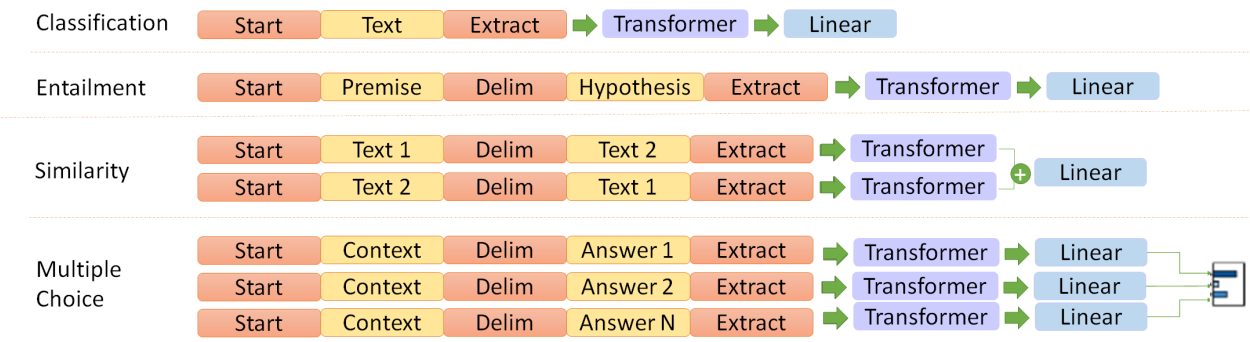

Figura 2.14: Exemplos de aplicação do BERT. Fonte: Adaptado de (ALAMMAR, 2018b)

2.2.6 Processo de treinamento

A tabela 2.1 mostra o uso de memória necessária para o pré-treino do modelo BERT-Base e BERT-Large.

Tipo	BERT-Base	BERT-Large
Modelo	110M * 4 bytes/1024^3	340M * 4 bytes/1024^3
Total Modelo:	**0,41GB**	**1,27GB**
Forward	110M * 8 bytes*16/1024^3	340M * 8 bytes*16/1024^3
Total Forward:	**13,11GB**	**40,53GB**
Backward	110M * 8 bytes/1024^3	340M * 8 bytes/1024^3
Total backward:	**0,82GB**	**2,53GB**
Total	**14,34GB**	**44,33GB**

Tabela 2.1: Quantidade de memória necessária para treinar o modelo por completo. Fonte:CICIAR (2021a).

2.2.7 Modelo BERT Inglês

O treinamento de um modelo BERT normalmente exige muito recurso computacional. O treino executado por (DEVLIN *et al.*, 2018) inclui as bases BookCorpus (ZHU *et al.*, 2015) e *Wikipedia* em inglês, totalizando 16GB de texto descompactado.

Para executar o treino completo de refino a partir de um modelo pré-treinado do modelo BERT-Base, (DEVLIN *et al.*, 2018) apontam requisitos de hardware como sendo pelo menos uma placa de vídeo equipada com memória entre 12GiB e 16GiB.

O modelo BERT-Large demanda uma disponibilidade ainda maior de memória na placa de vídeo. O treino completo do modelo utiliza ainda mais recursos. De acordo com a documentação, disponível no endereço (BER, 2022), em máquinas virtuais Cloud TPU pode consumir quatro dias, utilizando entre 4 e 16 máquinas

em paralelo.

2.2.8 Modelos alternativos em Inglês

O modelo RoBERTa, sigla para *Robustly optimized BERT approach* (LIU *et al.*, 2019), foi treinado com uma base de 160GB de texto utilizando maquinas DGX-1 cada uma com 8 placas de vídeo NVidia V100 de 32GB. LEE *et al.* (2019) resume a diferença do RoBERTa em um pré-treino com sentenças maiores usando mais dados e removendo a tarefa de *Next Sentence Prediction*.

Além do modelo RoBERTa, que é um aprimoramento do modelo em relação ao BERT inicial, existem variações do modelo como ALBERT (LAN *et al.*, 2019) que visa manter resultados em várias tarefas com desempenho similar ao BERT mas com tamanho de modelo reduzido. O BioBERT (LEE *et al.*, 2020) é treinado com um corpus de conteúdo biomédico em contraste com modelos como BERT, RoBERTa, ALBERT que são treinados com corpus genéricos de um idioma. O autor destaca a melhora do desempenho em tarefas desse domínio.

2.2.9 Modelo TinyBERT Inglês

TinyBERT, ilustrado na Figura 2.15 é um modelo derivado de BERT e é gerado a partir da técnica de *Knowledge Distillation* (KD) (JIAO *et al.*, 2019). A técnica de KD consiste em transferir o conhecimento de um modelo *"teacher"* para um modelo menor *"student"* e foi proposta por HINTON *et al.* (2015).

O modelo TinyBERT tem 4 camadas e 14.5 milhões de parâmetros que, comparado ao BERT-Base com 12 camadas e 109 milhões de parâmetros, consegue manter parte do desempenho nas tarefas avaliadas. Nos experimentos o modelo TinyBERT superou o BERTtiny TURC *et al.* (2019) em todas as tarefas GLUE com melhoria de 6,8% em média.

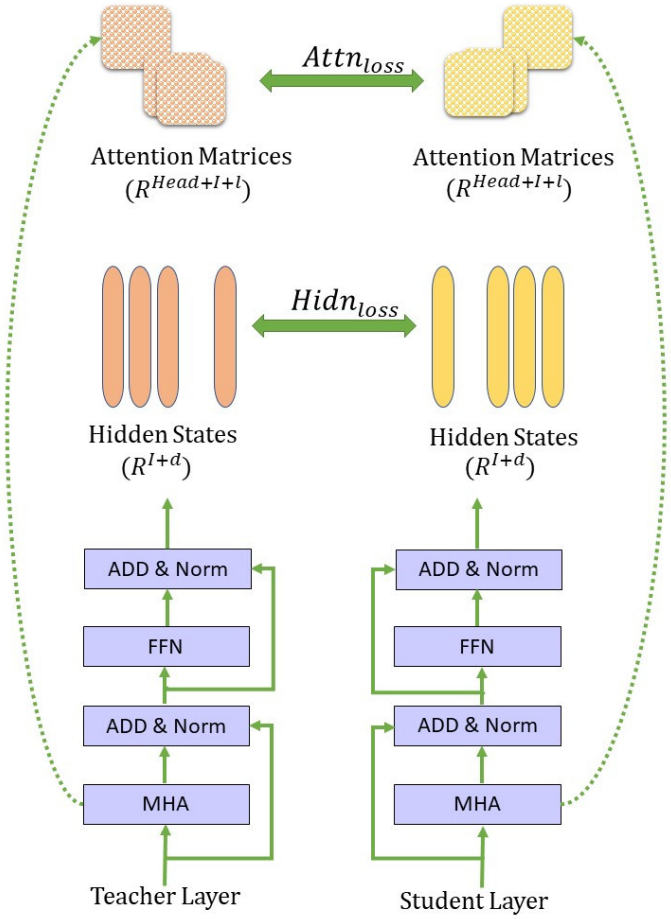

Figura 2.15: Arquitetura de aprendizado TinyBERT. Fonte: Adaptado de (JIAO *et al.*, 2019)

2.2.10 Modelo BERT Português

Mais de 100 idiomas têm modelos BERT correspondentes[1]. Em Português podemos utilizar o BERTimbau (SOUZA *et al.*, 2020) com modelos pré-treinados para Tensorflow e PyTorch dos tipos BASE e LARGE. O BERTikal (POLO *et al.*, 2021) é uma versão do BERT disponível para Português com refino em corpus jurídico.

[1] `https://github.com/google-research/bert`

2.2.11 Refinamento parcial

Os modelos pré-treinados como BERT-Base e BERT-Large têm respectivamente cerca de 110 milhões e 340 milhões de parâmetros, exigindo grande recurso computacional para treinamento e refino. O treinamento pode demorar entre dias ou semanas DEVLIN *et al.* (2018). Já a atividade de refino do modelo pode consumir entre horas e dias, a depender do Hardware utilizado.

Para o refino do modelo LEE *et al.* (2019) afirma que apenas um quarto das camadas precisa necessariamente ser refinadas para obter 90% da qualidade original de um refino completo. Os experimentos foram executados em placas de vídeo NVIDIA Tesla V100 em modelos pré-treinados de BERT DEVLIN *et al.* (2018) e RoBERTa LIU *et al.* (2019). Para o BERT foi escolhida a configuração de 3 épocas de refino e para RoBERTa foi escolhido a configuração de 10 épocas. As taxas de aprendizado usadas foram 1.00×10^{-5}. Os testes foram feitos com o GLUE benchmark que envolve as seguintes tarefas:

- CoLA - Corpus of Linguistic Acceptability WARSTADT *et al.* (2019).
- MRPC - Microsoft Research Paraphrase Corpus DOLAN e BROCKETT (2005).
- STS-B - Semantic Textual Similarity Benchmark CER *et al.* (2017).
- QQP - Quora Question Pairs IYER *et al.* (2017).
- SS2 - Sentiment Treebank SOCHER *et al.* (2013).
- RTE - Recognizing Textual Entailment BENTIVOGLI *et al.* (2009).
- MNLI - Multigenre Natural Language Inference WILLIAMS *et al.* (2017).
- QNLI - Question Natural Language Inference ALISHAHI *et al.* (2019).

LEE *et al.* (2019) notou que quando todos os componentes estão congelados exceto a camada de saída específica da tarefa, para a maioria das tarefas o desempenho do modelo refinado atinge apenas 64% da qualidade original. A cada camada adicional liberada para refino o desempenho aumenta drasticamente.

A tarefa com menor impacto foi o conjunto de dados SST-2 para BERT-Base que obteve acurácia de 84.9 com todas as camadas congeladas. Na camadas congeladas até a 9ª camada e três camadas retreinando a acurácia foi de 90.8. A configuração com retreino de todas as todas as 12 camadas a acurácia foi 92.7.

A tarefa com maior impacto foi CoLA com BERT-Base com MCC (Matthews Correlation Coefficient) de 29.4 com as camadas congeladas até a 12ª. Na configuração com congelamento até a 9ª camada e três camadas retreinando o MCC foi de 47.5. Com o todas as 12 camadas retreinando o MCC foi de 58.3.

2.3 Classificação de documentos em português

O experimento realizado por ZAMPIERI e GEBRE (2012) em 2012 explora a identificação de variedade linguística aplicado ao Português brasileiro e europeu. O experimento foi feito com testes de n-gramas de dois a seis utilizando distribuição probabilística de Laplace e a biblioteca NLTK para gerar o *language model*. Foram utilizados um corpus Brasileiro com textos publicados pela Folha de São Paulo de 2004 e um corpus Português com textos do Diário de notícias de 2007. A classificação de um dado texto foi feita com a comparação dos *language models* e o que mais se aproxima determina a classe do texto. Nos testes o experimento com 4-gramas se destacou com 0.998 de acurácia.

Em 2014 TOLEDO *et al.* (2014) aplicou classificação de texto em Português Brasileiro na identificação de discurso textual de adultos sadios. Foram utilizadas descrições de 144 indivíduos, com no mínimo três anos de educação. O SVM-RBF apresentou melhores resultados para a classificação binária em quatro de um total de seis testes, em comparação com Multi-Layer Perceptron, NaiveBayes, SimpleLogistic, JRip e J48. Em 2017, BRUM e NUNES (2017) propôs o corpus TweetSentBR de Tweets em Português Brasileiro. Foram anotados 15000 sentenças do seguimento de programas de TV. As classes escolhidas são positivo, neutro e negativo. Foram feitos experimentos de *baseline*, registrando 80,99% na medida F.

(HARTMANN *et al.*, 2017) faz um estudo do uso de *Word Embbedings* em Português. Na publicação são realizados experimentos com FastText, Glove, Wang2Vec e Word2Vec. Foi utilizado um corpus com 17 fontes, com gêneros como enciclopédicos, informativos e didáticos. Foram feitas comparações entre as técnicas na parte do corpus em Português Brasileiro com a parte em Português Europeu, o GloVe mostrou melhor desempenho em avaliações de analogias de sintática e semântica.

O trabalho de SANTOS *et al.* (2020) busca sinais de problemas mentais como depressão, ansiedade, anorexia e pensamentos suicidas em Tweets no idioma Português Brasileiro.

(SOUZA *et al.*, 2020) traz um modelo pré-treinado do BERT-Base e BERT-Large para o Português, chamado de BERTimbau. Seguindo o padrão dos modelos Base, as dimensões são com 12 camadas, 768 de dimensão ocultas, 12 cabeçotes de atenção e 110 milhões de parâmetros. Para o BERT-Large são 24 camadas, 1024 de dimensão oculta, 16 cabeçotes de atenção e 330 milhões de parâmetros treináveis. A sentença de entrada tem tamanho máximo de 512 tokens. O vocabulário de treino utilizado contém 30 mil sub-palavras e 2 milhões de sentenças em Português. O treino foi feito com um milhão de *steps* utilizando taxa de aprendizado de 1.00×10^{-4} além de empregarem *warm up* de taxa de aprendizado e decaimento sucessivo. Foi utilizada uma instância TPU v3-8. O treinamento do BERTimbau Base durou 4 dias, com 8

épocas. O treinamento do BERTimbau Large durou 7 dias. Os autores afirmam que o BERTimbau foi capaz de apresentar desempenho semelhante a modelos treinados para outros idiomas.

CABRAL *et al.* (2021) pesquisaram técnicas para detectar desinformação em mensagens de WhatsApp para o idioma Português Brasileiro. Foi construído um corpus com exemplos positivos e negativos com metadados como visualizações, número de comentários, data de publicação. A coleta de dados foi feita buscando grupos públicos, com no mínimo 100 usuários. Foram selecionados 59 grupos para coleta das mensagens e metadados. Os autores citam o cuidado com a privacidade dos usuários e que foram criados números identificadores anônimos. Foram rotuladas por humanos especialistas 5284 mensagens. Nos testes feitos para avaliar desempenho de classificadores, destaca-se F-score de 0,733 utilizando LSVM.

2.4 Classificação de documentos jurídicos

De acordo com MASTELLA (2020) o Poder Judiciário Brasileiro enfrenta um cenário árduo devido a diversos fatores que culminam na lentidão no ciclo de atendimento às demandas jurídicas da população como por exemplo:

- O crescente aumento no ingresso de processos;
- A limitada disponibilidade de servidores a ser mantida ou reduzida;
- Uma baixa vazão dos processos.

Diversas melhorias vêm sendo propostas e executadas a fim de tornar o Sistema mais eficiente e uma delas é a completa implantação do processo eletrônico em substituição aos processos físicos que pode facilitar o acesso às informações processuais. Entretanto muitas das tarefas realizadas ainda são manuais impactando negativamente a gestão otimizada dos recursos disponíveis e celeridade do processo.

As técnicas para classificação automática de textos têm despertado o interesse de pesquisadores dos mais diversos domínios do conhecimento dentre os quais pode-se destacar as Ciências Jurídicas. O Direito inerentemente depende da análise de um grande volume de informações textuais, motivo pelo qual tem-se explorado o potencial das técnicas de PLN em otimizar a realização de muitas dessas tarefas.

Contudo, é importante considerar que a legislação varia conforme o país, estado, município jurisdicional além de ser escrita no respectivo idioma de origem. Frente ao exposto, mostra-se relevante a realização de pesquisas com técnicas de PLN aplicadas a textos jurídicos do país de interesse, considerando-se o respectivo idioma MASTELLA (2020).

Os experimentos de MASTELLA (2020) utilizaram Redes Neurais Recorrentes Bi-LSTM para classificar petições em Português Brasileiro com classificadores nas configurações únicos e *Emsemble* em uma base de dados de 107 mil documentos com uma métrica F1 de 0,846 se destacando no processo.

(POLO *et al.*, 2021) disponibilizou modelos pré-treinados com Phraser, Word2Vec, Doc2Vec, FastText e BERT para documentos jurídicos em Português Brasileiro. Foram utilizados quatro conjuntos de dados com 3,14 GB, 0,74 GB, 0,77 GB e 1,12 GB de texto.

LEME (2021) desenvolveu uma pesquisa comparando combinações de topologias de Redes Neurais e técnicas de Aprendizado de Máquina para classificar tipo de processos eletrônicos do CADE. A base de dados utilizada foi de 1275 processos de 2015 à 2018 e convertidos para texto com técnica de OCR. Foi utilizado validação cruzada com k=5. As combinações foram feitas com BERT, CNN (Rede Neural Convolucional), BOW (Bag Of Words), LSTM e BiLSTM. A melhor combinação foi com BOW + BERT + LSTM atingindo acurácia de 92,9%.

Foram desenvolvidas ferramentas para tratar problemas de PLN em documentos Legais, como identificação de partes em processos (NGUYEN *et al.*, 2018), classificação de documentos (SOUZA *et al.*, 2020; DA SILVA *et al.*, 2018), predição de seguimento jurídico (SULEA *et al.*, 2017).

2.5 Desempenho computacional em tarefas de *Deep-Learning*

O código para treinamento[2] de um modelo BERT, em máquinas virtuais Cloud TPU pode consumir quatro dias utilizando entre 4 e 16 nós em paralelo. Os autores recomendam realizar o pré-treino do modelo utilizando uma máquina alugada na nuvem. Esse processo pode durar duas semanas e custar cerca de 1600 dólares.

Para tarefa de refino de modelo BERT placas de video com 16GiB de memória são suficientes. O treinamento inicial exige placas com no mínimo 48GiB de memória. Para a arquitetura de *Tensor processing unit* (TPU), a memória mínima recomendada é de 64GiB. A tabela 2.2 lista produtos de infraestrutura e plataforma como serviço.

Os serviços oferecidos pelo Google Colaboratory, ou Colab, têm entre 12GiB e 16GiB de memória de placa de vídeo disponível e estão listados na tabela 2.3. Nessa faixa de recursos, assim como na tabela 2.2 a máquina virtual P5000 com VRAM de 16GiB, é possível executar o ajuste fino do BERT.

Para pré-treinos que geram modelos do zero é necessário uma quantidade maior

[2] `https://github.com/google-research/bert`

Serviço	Preço por hora	Mensal
Colab	gratuito	gratuito
Colab+	não se aplica	58 reais
Colab Pro	não se aplica	258 reais
Paperspace P5000 16GiB VRAM	0,78 dólares por hora	569 dólares
Paperspace A6000 48GiB VRAM	1,89 dólares por hora	1379 dólares
Google Cloud 64GiB TPU V2	4,5 dólares por hora	3285 dólares
Google Cloud 128GiB TPU V3	8 dólares por hora	5840 dólares

Tabela 2.2: Custos de serviços para execução de treinamentos. Fonte: De autoria própria.

de memória de vídeo ou memória integrada no caso das plataformas TPU. Como a arquitetura das maquinas TPU é unificada e diferente das placas de vídeo, o requisito de memória é diferente sendo 64GiB de RAM como apontando pelos autores.

	Colab Free	Colab Pro	Colab Pro+
Garantia de acesso	baixa	alta	mais alta
Placa de vídeo	K80	K80, T4 e P100	K80, T4 e P100
Memória RAM	16GiB	32GiB	52GiB
Tempo de execução	12 horas	24 horas	24 horas
Execução *background*	Não	Não	Sim
Público	Casual	Regular	Uso intensivo

Tabela 2.3: Serviços oferecidos pelo Colab. Fonte: De autoria própria

O custo de uma placa de vídeo de linha profissional com relevante quantidade de memória de vídeo é superior à 1800 dólares, como descrito na tabela 2.4, para tarefas de refino do BERT. Para realizar pré-treino uma placa de vídeo com a quantidade recomendada de memória para acomodar o batch de amostras está na faixa de 4999 dólares.

Placa de vídeo	Preço
Nvidia P5000 16GiB VRAM	1844 dólares
Nvidia A6000 48GiB VRAM	4999 dólares
Nvidia GTX2070 8GiB	499 dólares
Nvidia RTX3060 12GiB	529 dólares

Tabela 2.4: Custo de placas disponíveis no mercado. Fonte: De autoria própria.

Soma-se a essas barreiras a escassez de placas de vídeo e o aumento de preços por conta da pandemia de COVID MOLLOY (2021) e o câmbio do Dólar para o Real.

Capítulo 3

Metodologia

Nesta seção são apresentados ferramentas e configurações do ambiente utilizados. Busca-se medir a alteração desempenho do classificador do CICIAR com modificações arquiteturais e retreino de camadas dentro do cenário de restrição de poder computacional. Para tanto foram avaliados os classificadores de base e os classificadores modificados com retreino de camadas BERT. O classificador de base utiliza frases pré-processadas com BERT-Base. Os classificadores modificados têm a camada BERT-Base acoplada com camadas recebendo treinamento com o treino do classificador. A base de dados utilizada é o *Data Warehouse* do CICIAR com os documentos pré-processados, limpos e estruturados em tabelas. A seguir serão detalhados o conjunto de dados utilizado, os experimentos e os resultados.

3.1 Verde

O Verde é um sistema de assistência às atividades da Defensoria Pública do Estado do Rio de Janeiro. O sistema atende aos órgãos da Defensoria Pública do Estado do Rio de Janeiro (DPRJ) e foi lançado em 1º de junho de 2016. O Verde oferece recursos como:

- Cadastro unificado de assistidos;
- Cadastro de casos;
- Cadastro de lembretes;
- Emissão de documentos;
- Cadastro de lembretes;
- Gerenciamento;
- Recebimento de intimações;

- E outros.

Em 2021 foram abertos mais de 300 mil casos. Cenários com grande quantidade de dados podem tirar proveito da aplicação de técnicas de Inteligência artificial.

3.2 CICIAR

O CICIAR (Caixa de Intimações Com Inteligência Artificial) foi criado para auxiliar os defensores públicos a tratar e priorizar o atendimento de intimações recebidas na caixa de intimações do Verde. O CICIAR faz previsões dos rótulos das intimações e essa classificação pode auxiliar o planejamento do defensor em quando abrir o documento (PARREIRAS *et al.*, 2022).

O CICIAR categoriza as intimações sem oficialmente acessar as mesmas. A classificação prévia é a principal motivação para a criação do CICIAR. Ao acessar uma intimação, se inicia a contagem do prazo para atendê-las. O Defensor Público tem prazo de 10 dias para responder uma intimação ao abri-la. A capacidade de prever sobre o que pode ser cada documento é valiosa por auxiliar o planejamento do Defensor Público antes mesmo de abrir o documento, podendo priorizar casos urgentes.

O CICIAR está implantado em mais de 300 órgãos da Defensoria Pública do Estado do Rio de Janeiro. A base de dados contém 57397 processos com um total de 214 mil intimações e para a Defensoria foram destinadas 68 mil intimações. Foram extraídas 27 milhões de frases dos documentos para criação de representações com modelo BERT.

A geração de dados para treinamento do classificador depende do compartilhamento de conhecimento de especialistas. O Defensor Público é o responsável por esse fornecimento de informação ao classificar ou dar rótulos para frases das intimações e de seus contextos.

A base de dados contém 42 mil frases classificadas. São classificadas em média 2601 intimações por dia. Estão sendo registrados centenas de *feedbacks* semanais dos Defensores Públicos em relação às classificações. No ambiente de produção estão habilitados dois rótulos com acurácia superior a 0.93 (tabela 3.1) com previsão de expansão para mais quatro rótulos. Os rótulos só são aplicados caso haja uma confiança de mais de 70% na classificação.

O CICIAR está em constante mudança com o aumento da base de dados, melhoria constante dos modelos, aprimoramento e incremento das heurísticas.

O desenvolvimento do sistema CICIAR depende de pesquisa intensiva para geração das soluções técnicas. É necessário criar frequentemente experimentos para validar novas ideias e aprimorar existentes.

Rótulo	Precisão	Revocação	Acurácia
Sentença	0.91	0.86	0.96
Marcam Data	0.84	0.98	0.93

Tabela 3.1: Qualidade da classificação de rótulos. Fonte: De autoria própria. Fonte: CICIAR (2021b)

3.2.1 Classificação de documentos no CICIAR

A classificação de documentos dentro do CICIAR é feita seguindo critérios adotados pelos Defensores Públicos. Os contextos são documentos que entre a última Decisão, Despacho ou Sentença do Juiz e uma intimação, como ilustrado na Figura 3.1.

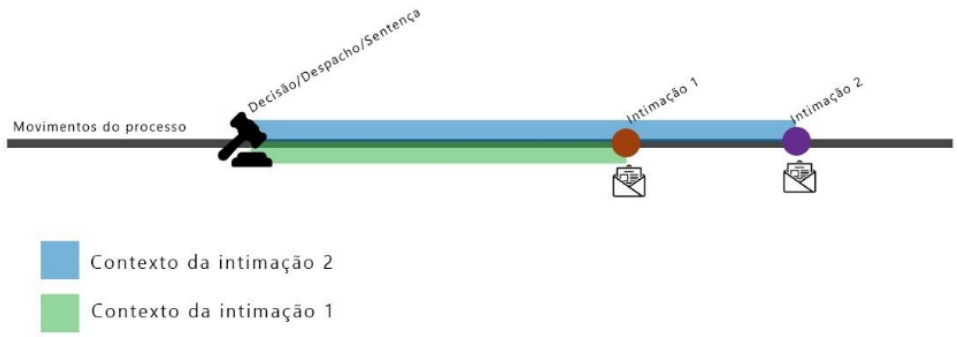

Figura 3.1: Contextos de intimações. Fonte: (CICIAR, 2020)

As frases de um contexto e alguns documentos vinculados. As frases que pertencem a uma intimação ou a um contexto são classificadas referente a essa mesma intimação.

Os rótulos disponíveis são:

- Em Branco,
- Nada,
- Outros,
- Decisão Interlocutória,
- Decisão final,
- Marcam Data,
- Prazos/Manifestações e
- Prisão.

O processo de rotulagem não exige que se rotule todas as frases contidas ou referentes a uma intimação, apenas as frases mais relevantes para prever o tipo de intimação que será inferido pelo classificador.

3.2.2 Ciclo de operação

Para não sobrecarregar com acessos à API do Verde, o CICIAR possui uma janela de execução de 12 horas, de 19h às 7h, o que é equivalente à 720 minutos. Dentro dessa janela de execução, a Figura 3.2 indica o atividade do sistema em 66% do tempo.

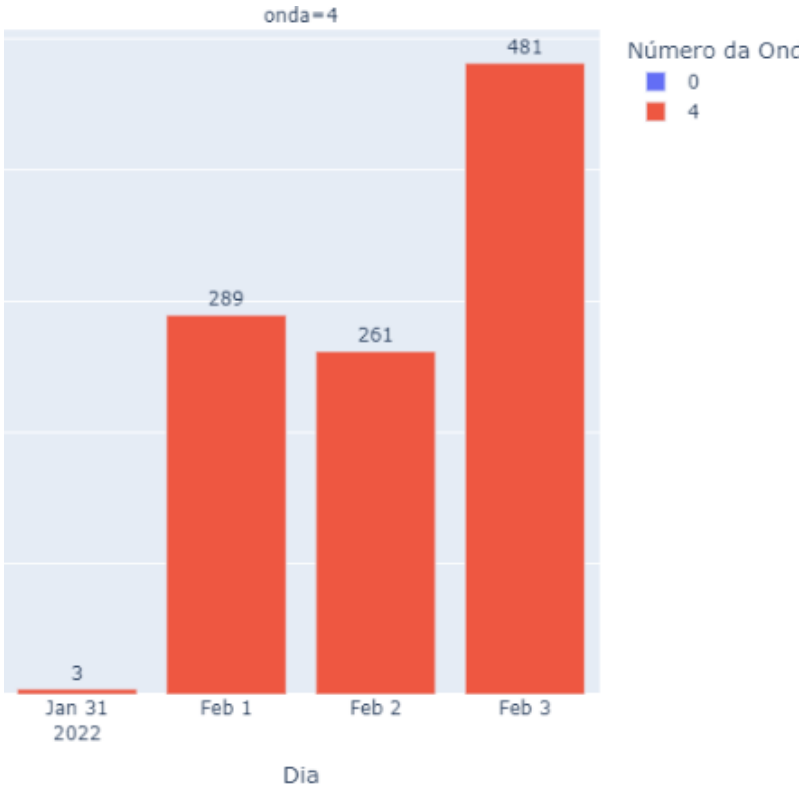

Figura 3.2: Tempo de atividade. Fonte: (CICIAR, 2022)

O sistema de produção está hospedado na nuvem. O uso médio, segundo relatório feito de CPU, Figura 3.3 foi de cerca de 0,5%. O uso de rede, Figura 3.4 registrado, exceto dia 6 de fevereiro, domingo, foi acima de de 140MiB.

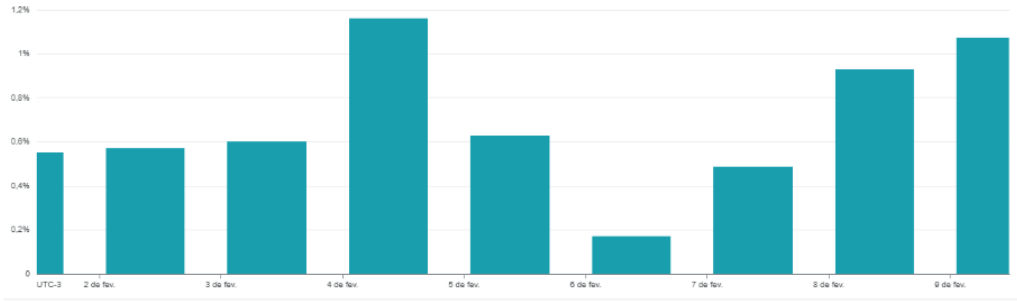

Figura 3.3: Média do uso de CPU no ambiente de produção. Fonte: (CICIAR, 2022)

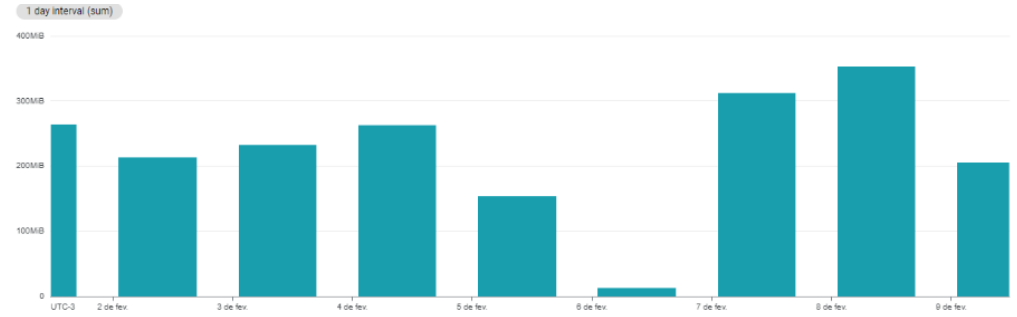

Figura 3.4: Média do uso da interface de rede no ambiente de produção. Fonte: (CICIAR, 2022)

3.3 Origem dos dados

Foi utilizada a base de dados do CICIAR, que é gerada através da coleta de processos do Tribunal de Justiça do Estado do Rio de Janeiro. As frases rotuladas são então utilizadas para treinar o classificador. A base utilizada contém 2861 processos. Nesse conjunto, existem 8750 frases de intimações e 8159 frases de contextos de intimações. As frases de intimação se originam de documentos de intimações, e frases de contexto se originam de documentos entre uma decisão, despacho ou sentença de um Juiz.

Os rótulos utilizados para classificação das sentenças são:

- Decisão Interlocutória,
- Decisão Final do Juiz,
- Marcam Data,
- Nada,
- Outro,
- Prazos / Manifestações e

- Prisão.

A tabela 3.3 detalha o significado de cada rótulo.

Rótulos	**Descrição**
Em branco	"Esta frase é irrelevante"
Nada	"Esta frase é irrelevante, mas pode causar confusão para um leigo. Preciso deixar claro para o computador que não é relevante."
Outros	Alimentos Provisórios, Guarda Provisória, Resultado de Penhora, Deferimento/Indeferimento de Tutela de Urgência.
Interlocutória	Alimentos Provisórios, Guarda Provisória, Resultado de Penhora, Deferimento/Indeferimento de Tutela de Urgência.
Decisão Final	Acórdão, Sentença, Extinção do Processo, Homologação, Ciência de Sentença.
Prazos / Manifestações	Alegações Finais, Memoriais, Razões de recurso em sentido escrito, Razões de apelação, Contra-razões, Plano de Adjudicação, Réplica, Resposta a Acusação, Primeiras declarações, Plano de partilha, Solicitação de esclarecimento/manifestação, Determinação de emenda inicial
Prisão	Relaxamento Prisão, Prisão Revogada, Prisão Decretada, Decretação de Prisão, Alvará de Soltura, Relaxamento Prisão com outra medida cautelar

Tabela 3.2: Descrição e exemplos de cada rótulo. Fonte: De autoria própria.

A extração do contexto é feita a partir da ultima decisão do Juiz. As frases foram rotuladas por especialistas. No processo de rotulagem o Defensor Público especialista pode selecionar uma frase de uma intimação que define o tema daquela intimação, além de poder selecionar que é uma frase que não diz nada. É esperado que de dezenas de frases somente algumas mais relevantes sejam rotuladas.

Os gráficos 3.5 e 3.6 mostram a distribuição das frases por rótulo para frases de intimações e de contextos, respectivamente.

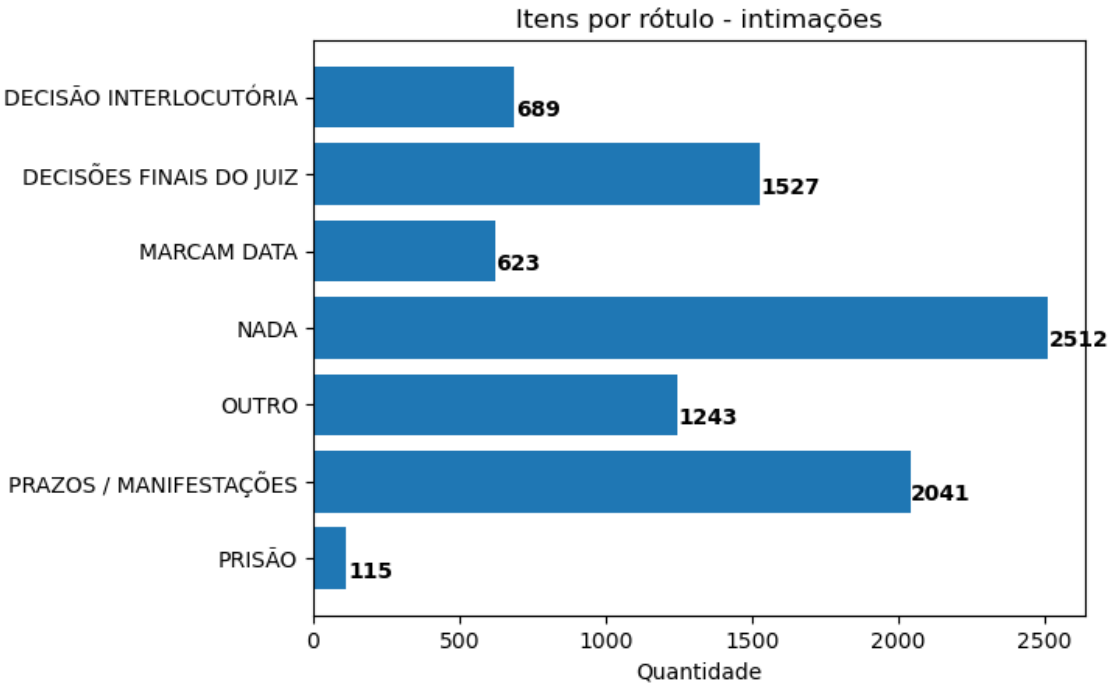

Figura 3.5: Distribuição das frases de intimações por rótulos. Fonte: De autoria própria.

Quantidade de frases por classe de contexto:

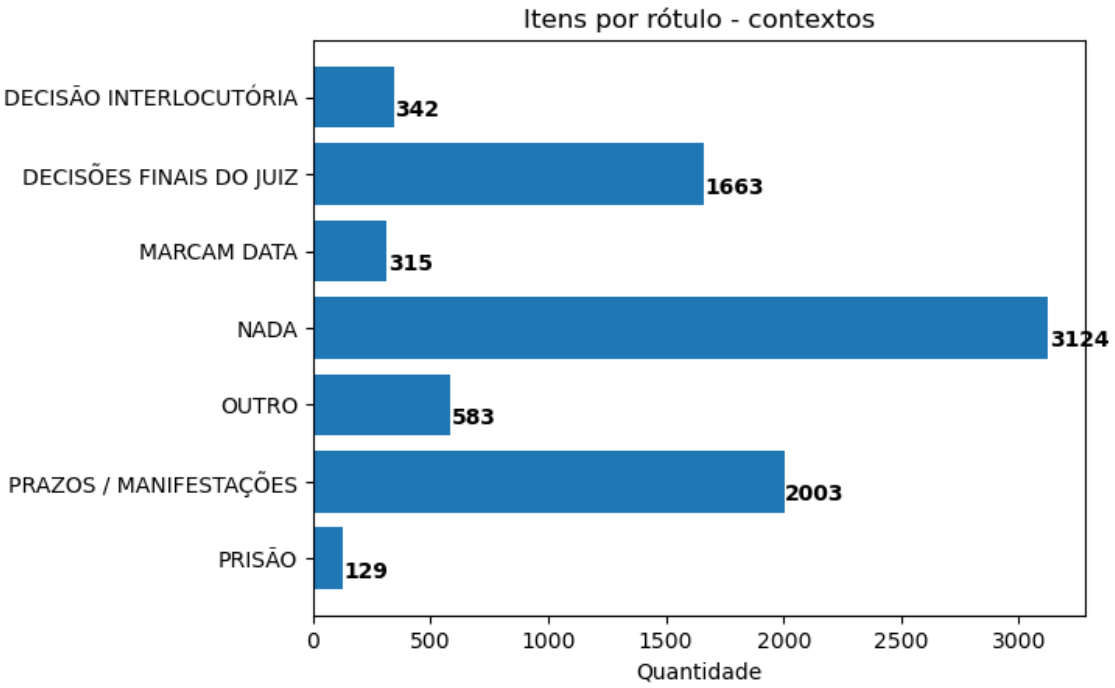

Figura 3.6: Distribuição das frases de contextos por rótulos. Fonte: De autoria própria.

A tabela 3.3 mostra a distribuição proporcional de cada classe na soma dos conjuntos de frases de intimação e de contexto.

Classe	Proporção
Decisão Interlocutória	6%
Decisões Finais do Juiz	18,8%
Marcam Data	5,5%
Nada	33,3%
Outro	10,7%
Prazos / Manifestações	23,9%
Prisão	1,4%

Tabela 3.3: Distribuição das frases total do conjunto de dados. Fonte: De autoria própria.

Verifica-se que o conjunto de dados é muito desbalanceado entre as classes, como a classe "PRISÃO"contendo 1,4% dos elementos do conjunto de dados. A tabela 3.4 mostra informações sobre o conjunto de dados utilizados.

Total de frases de intimação	8750
Total de frases de contexto	8159
Conjunto de teste	1313

Tabela 3.4: Estatísticas do conjunto de dados. Fonte: De autoria própria.

Rótulos	**Exemplo**
Nada	"NO MAIS, A SENTENÇA VERGASTADA.", "Contudo, diante do iter criminis percorrido, restando claro que o delito esteve próximo da consumação, na forma do artigo 14, II, do CP, a reprimenda deve ser reduzida na fração de 1/2 (metade), para 2 (dois) anos e 8 (oito) meses de reclusão e 6 (seis) dias multa."
Outros	"Anote-se o patrocinio da parte ré pela Defensoria Pública Tabelar.", "Cumpra-se o v. acórdão."
Decisão Final	"Custas rateadas igualmente pelas partes, conforme §2º do art. 90, do NCPC, observada a gratuidade de justiça que ora também defiro ao réu.", "Justiça do Estado do Rio de Janeiro, por unanimidade, em negar provimento aos recursos, mantendo-se a sentença nos seus atuais termos."
Marcam data	"Juíza, foi designado o dia 21/10/2019, às 14:15h, para realização da Audiência Especial de Conciliação.", "Terça-Feira, às 10:00h, publicada no DOe de 27/09/2019 às fls. 142/150."
Prazos / Manifestações	"No mesmo prazo, venha cópia do anexo referido no documento de fl. 12, isto é, a íntegra da notificação dirigida ao réu.", "Diante da certidão de fls. 496, à parte interessada para tomar ciência da remessa do mandado de fls. 488."
Prisão	"Expeçam-se alvarás de soltura clausulado.", "Assim, mantenho a custódia cautelar dos acusados."
Decisão Interlocutória	"O que se vê do processo é que não há questões processuais pendentes, razão pela qual dou o feito por saneado.", "Decreto a perda da prova para o depoimento pessoal do autor, uma vez que não houve o recolhimento das custas para a intimação."

Tabela 3.5: Exemplos de frase para cada rótulo. Fonte: De autoria própria.

3.4 DW e pré-processamento dos dados

Os documentos dos processos são consultados a partir de API do sistema Verde e são salvos na codificação UTF-8. Em seguida, passam por um processo de substituição de siglas e abreviaturas para palavras completas como "fls."vira folhas e "fl."vira folha. O processo de separação de frases é feito utilizando "splitline"do Python. A modelagem do DW foi feita a partir do manual do CNJ e do Web Service do Tibunal de Justiça. A Figura 3.7 ilustra a sequencia do processo desde a chegada do aviso de intimação. A Figura 3.8 ilustra a operação diária dos sistemas.

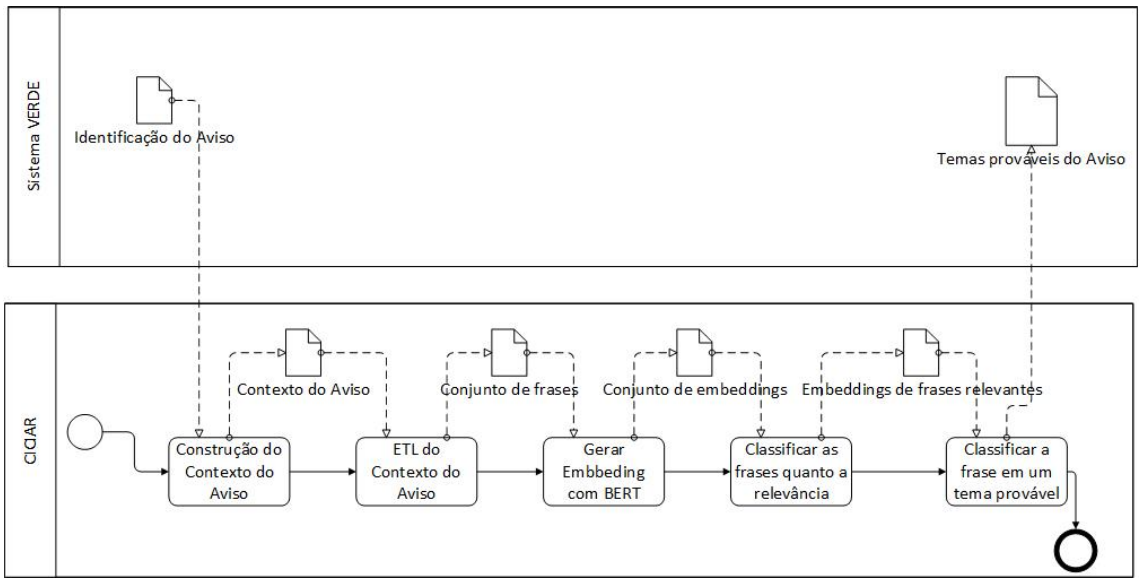

Figura 3.7: Processo de predição de classe de uma intimação. Fonte: Adaptado de (PARREIRAS *et al.*, 2022)

3.5 Ambiente de desenvolvimento

O experimento foi criado em Python 3 VAN ROSSUM e DRAKE JR (2014), dentro do serviço Colab da Google, na modalidade *local runtime*. A tabela 3.5 mostra as bibliotecas mais relevantes utilizadas para preparo dos dados e aprendizado.

3.6 Treinamento do classificador

Os experimentos realizados utilizam dados do DW, tanto as frases quanto as representações vetoriais de BERT-Base. A limpeza dos dados é listada na tabela 3.7. A versão BERT-Base e BERT-Large utilizados são do modelo BERTimbal. Os treinamentos dos classificadores de rede neural são feito com frases pré-processadas e os experimentos com frases em linguagem natural. Dessa forma, é possível avaliar

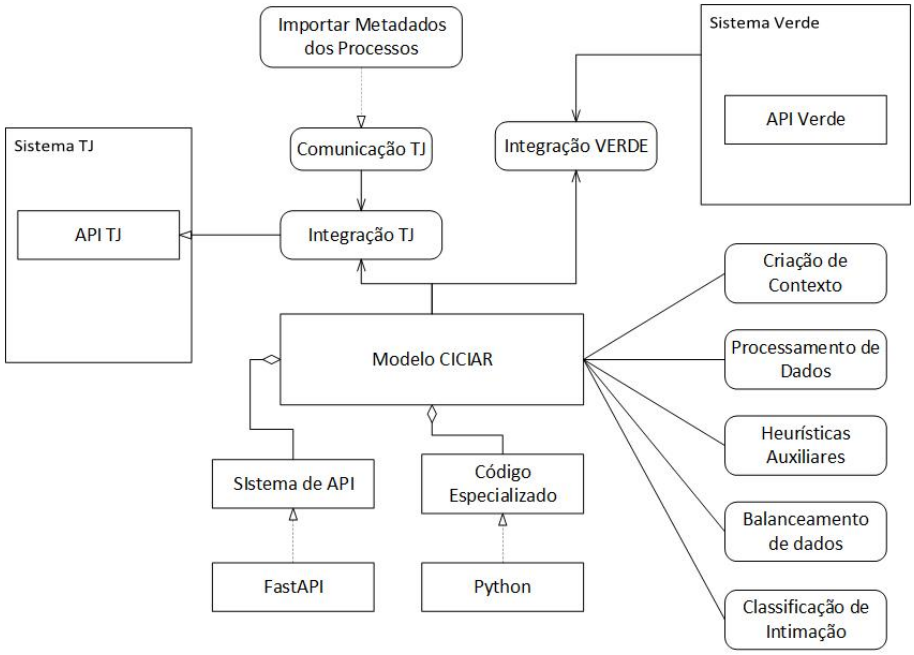

Figura 3.8: Estrutura organizacional dos componentes do ambiente CICIAR na operação diária. Fonte: Adaptado de (PARREIRAS *et al.*, 2022)

Ferramenta	Função
Pandas	Carregamento do conjunto de dados, limpeza e visualização
PyTorch	Treinamento do modelo
Matplotlib	Visualização e plotagem de resultados
Sklearn	Segmentação do conjunto dados de forma balanceada
Transformers	Tokenizador, carga de modelo BERT, optimizer ADAMW
Seaborn	Geração de gráficos

Tabela 3.6: Principais bibliotecas e ferramentas utilizadas. Fonte: De autoria própria.

de forma isolada o impacto das alterações propostas. O AutoTokenizer foi utilizado para gerar a frase tokenizada e com as máscaras, necessários para utilização da rede BERT. Para o treino foi utilizado uma função de *loss* F1-score. Os experimentos executados incluem a classe "Nada", tanto para o treinamento com frases pré-processadas quanto para o classificador com camada BERT.

3.6.1 Hiperparâmetros e configurações dos experimentos

Foram utilizado 50 épocas para o treino das redes neurais. O BERT-Base utilizado foi BERTimbal, com 12 camadas e 768 de dimensão de saída. O tamanho do *batch* utilizado é de 16. Foi utilizado Validação Cruzada tipo k-fold com k=5, cada fold treinando 80% dos dados e validando em 20%. A listagem dos hiperparâmetros é feita na tabela 3.8.

O treinamento, ilustrado na imagem 3.9, utilizou frases de intimação e de con-

Técnica de limpeza	Propósito
Quebra de linha	Separar em linhas individuais documentos que vem em uma única linha, como em XML
Substituição de siglas	São trocadas por palavras para auxiliar o tokenizador e o classificador
Remoção de frases indesejadas	remoção de número de páginas, e frases de controle que não fazem parte do sistema.

Tabela 3.7: Processos de limpeza. Fonte: De autoria própria.

Parâmetros	Valor
Batch size	16
época	50
dropout	0.1
Camadas BERT-Base	12
Dimensão saída BERT-Base	768
Classes	7
Conjunto de teste:	1313
Conjunto de Validação Cruzada:	15596
k da Validação Cruzada:	5
Conjunto de treino:	12476
Conjunto de validação:	1488

Tabela 3.8: Configurações dos experimentos. Fonte: De autoria própria.

texto. A validação e o teste foram feitos com de frases de intimações.

Do total das 16909 frases, sendo 8750 frases de intimação e 8159 frases de contexto, foi particionado os 15% de teste restando 7437 frases de intimação e as 8159 frases de contexto para o conjunto de treino e validação, totalizando 15596 frases.

O treinamento utilizou 80% das frases de intimação e de contexto, totalizando 12476 frases. A validação utilizou 80% das frases do conjunto de intimação após a retirada das frases de teste: 80% de 7437 frases, totalizando 1488 frases.

3.7 Processo de treinamento atual

O pré-processamento de sentenças e o treinamento dos classificadores que fazem parte da rotulagem das sentenças são atividades que demandam grande poder de processamento computacional.

Atualmente fazê-los em duas etapas trás benefícios em relação ao tempo de treinamento dos classificadores uma vez que como a rede BERT é utilizada com os pesos estáticos no processo de vetorização das sentenças o resultado da vetorização é sempre o mesmo. Dessa forma podemos reaproveitar as representações vetoriais das frases e possível armazená-las convertidas e treinar os classificadores. O tempo

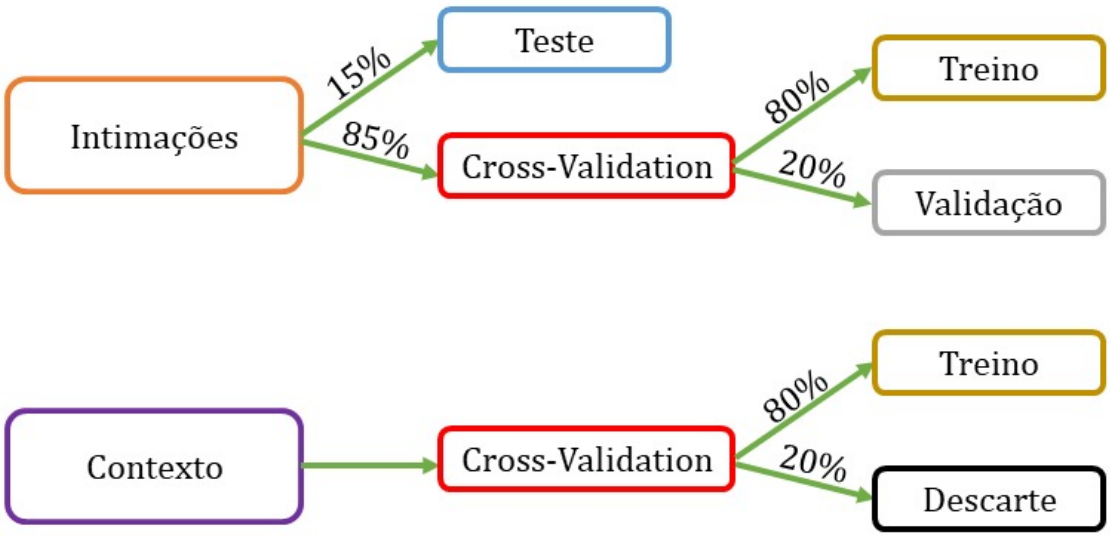

Figura 3.9: Particionamento dos conjuntos de treino, validação e teste. Fonte: De autoria própria.

de treino dos classificadores é reduzido de horas para minutos.

3.8 Organização do processo de avaliação dos classificadores

Propõe-se uma abordagem utilizando as sentenças diretamente no modelo sem o passo de pré-processamento.

Apesar de perder o benefício do ganho de tempo e de economia de recursos podemos ganhar qualidade na classificação ao poder ajustar parâmetros da rede BERT, conservando a capacidade de executar os treinamentos em computadores de baixo poder computacional.

A arquitetura é composta pela rede BERT se acoplando à rede dos classificadores. Com isso o modelo passa a receber a sentença na forma de *tokens* e máscara. Na estrutura proposta será avaliado o desempenho da aplicação do retreino de camadas junto ao treinamento do classificador.

LEE *et al.* (2019) demonstra como a liberação das últimas camadas da rede BERT pode ser benéfica para a qualidade da classificação e ao mesmo tempo consumindo menos recursos que o *fine-tunning* completo, já que menos camadas são retreinadas. Liberar uma fração de camadas do BERT para retreino pode aumentar a desempenho de classificação de um modelo se comparado ao uso do modelo com pesos congelados.

Ao utilizar o modelo pré-treinado é possível treinar classificadores e últimas camadas do modelo BERT em placas de vídeo com 8GiB ou 12GiB de memória que

são mais acessíveis. Dentro desse cenário avaliar o impacto no desempenho do classificador dentro de recursos computacionais de baixo custo.

O processo de avaliação de desempenho da técnica de retreino parcial de camadas do modelo BERT com o classificador pode ser visualizado Figura 3.10. O processo foi desenvolvido dentro do contexto do CICIAR utilizando a base de dados pré-processados, código fonte para processamento e treinamento do modelo. Com base nos estudos e materiais de referência foram construídos os experimentos para avaliar o impacto da técnica de retreino parcial de camadas no treinamento de modelos do CICIAR.

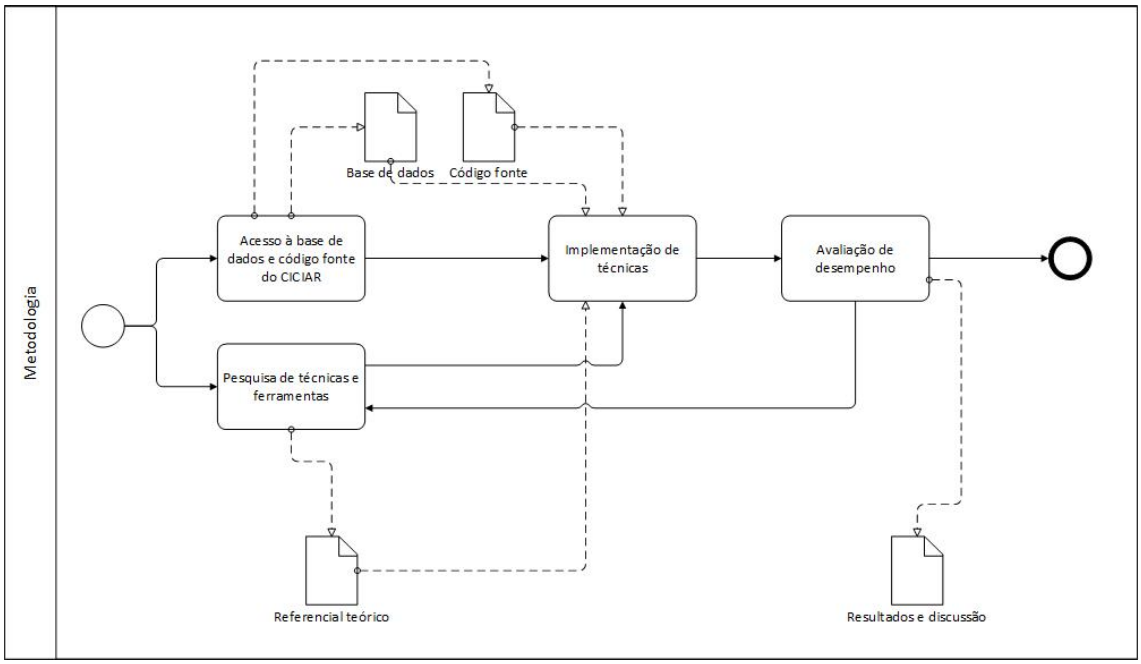

Figura 3.10: Processo de avaliação de desempenho da técnica de retreino parcial do modelo BERT. Fonte: De autoria própria.

Capítulo 4

Resultados e Discussão

Nesta seção serão apresentados os resultados dos experimentos executados. Os quinze experimentos foram executados utilizando a base de dados de treinamento do CICIAR de intimações e contextos de intimações rotulados. Os experimentos podem ser divididos em dois grupos: pré-processado, que utiliza frases convertidas previamente pelo BERT-Base e inserido diretamente no no classificador a ser treinado, e o segundo grupo com experimentos que recebem a frase transformada em *tokens* e tem o BERT e o classificador treinando juntos. O segundo grupo explora o retreino de camadas e pode ser dividido pelo número de camadas que treinam com o classificador.

Os experimentos do tipo "Pré-processado" são o ponto de partida sendo que a configuração de hiperparâmetro inicial é a 1.00×10^{-3}. Os experimentos "BERT-Base uma camada", "Bert-Base duas camadas" e "Bert-Base três camadas" utilizam o recurso de refino de camadas, sendo que "uma camada" representa a última camada liberada para retreino com as onze primeiras congeladas, "duas camadas" são duas últimas camadas liberadas para retreino com dez camadas congeladas e "três camadas" são três últimas camadas liberadas para retreino com nove camadas congeladas.

Os experimentos visam avaliar a aplicação da técnica de refino de camadas do modelo BERT integrado ao classificador no caso da classificação de frases do CICIAR.

4.1 Resultados do treinamento dos classificadores

4.1.1 Treinamento do classificador multi-classe com frases pré-processadas com BERT-Base

O resultado do treino classificadores com o pre-processamento das frases podem ser verificados nas tabelas de precisão 4.1, revocação 4.2 e medida F_1 4.3 nos experi-

mentos de nome Pré-processado. Cada linha mostrando o experimento com variação de Taxa de Aprendizado de 1.00×10^{-3}, 1.00×10^{-4}, 1.00×10^{-5} e 1.00×10^{-6}. Os experimentos dessas linhas foram feitos utilizando as representações vetoriais pré-processadas das frases. A evolução da *Loss* de validação do treinamento é mostrada nas imagens a seguir.

A Figura 4.1 mostra a evolução do treinamento com Taxa de Aprendizado de valor 1.00×10^{-3}, no qual a menor *Loss* média de validação de 0.801 é atingida em média na época 11. A partir da época 12 o modelo mostra um aumento da *Loss* de validação com o passar do treinamento, até a época 50.

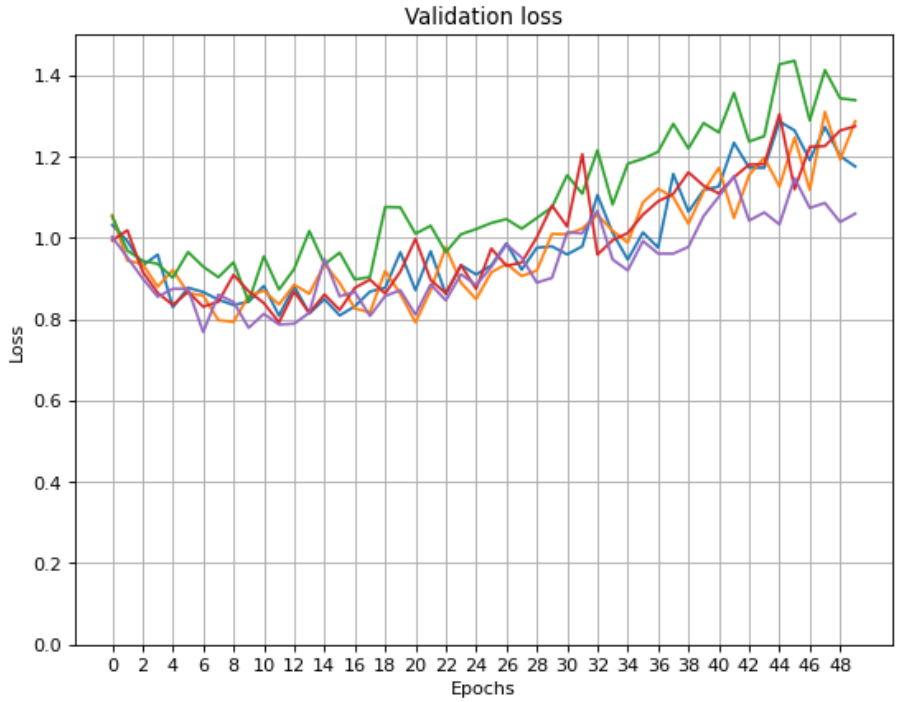

Figura 4.1: Frases pré-processadas, TA = 1.00×10^{-3}. Fonte: De autoria própria.

A Figura 4.2 mostra a evolução do treinamento com o Taxa de Aprendizado de valor 1.00×10^{-4}, no qual a menor *Loss* média de validação de 0.746 é atingida em média na época 44.

A Figura 4.3 mostra a evolução do treinamento com Taxa de Aprendizado de valor 1.00×10^{-5}, no qual a menor *Loss* média de validação de 0.911 é atingida em média na época 48.

A Figura 4.4 mostra a evolução do treinamento com Taxa de Aprendizado de valor 1.00×10^{-6} no qual a *Loss* mínima é atingida na época 46.

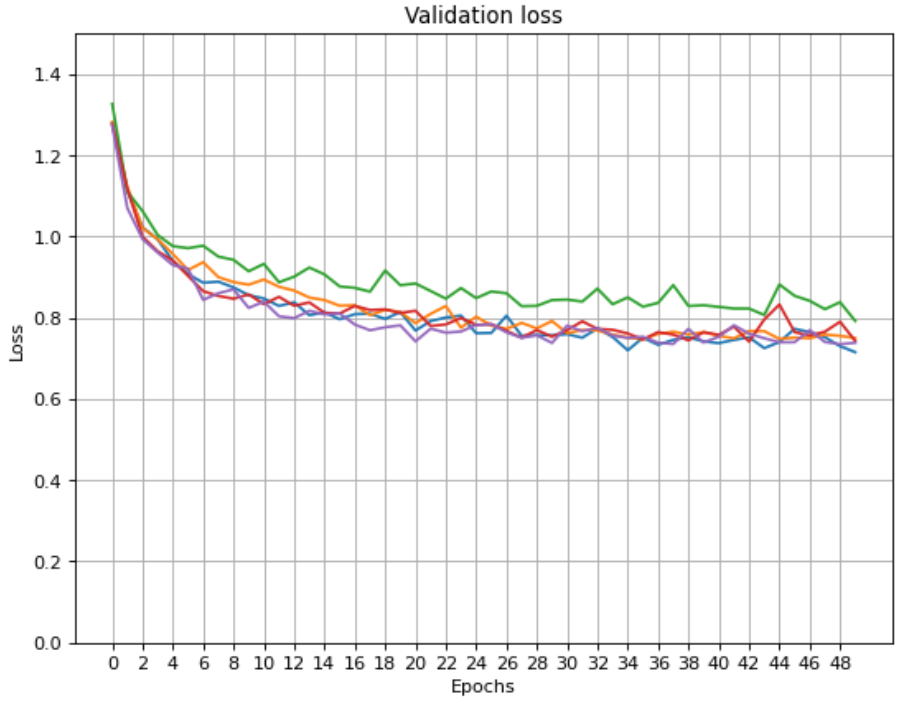

Figura 4.2: *Loss* de validação com última camada frases pré-processadas, LR= 1.00×10^{-4}. Fonte: De autoria própria.

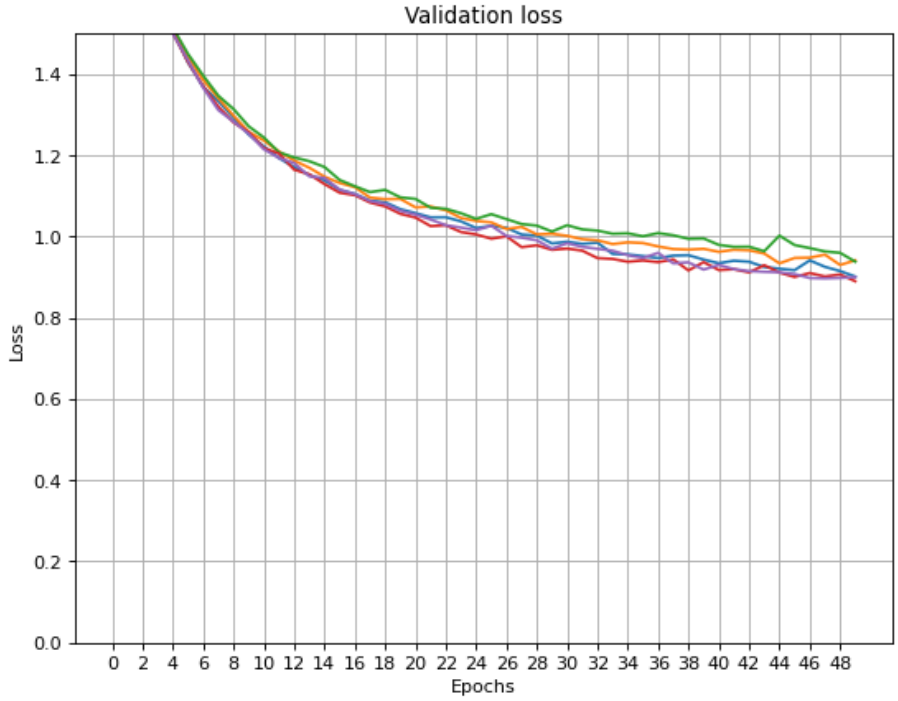

Figura 4.3: *Loss* de validação com última camada frases pré-processadas, LR= 1.00×10^{-5}. Fonte: De autoria própria.

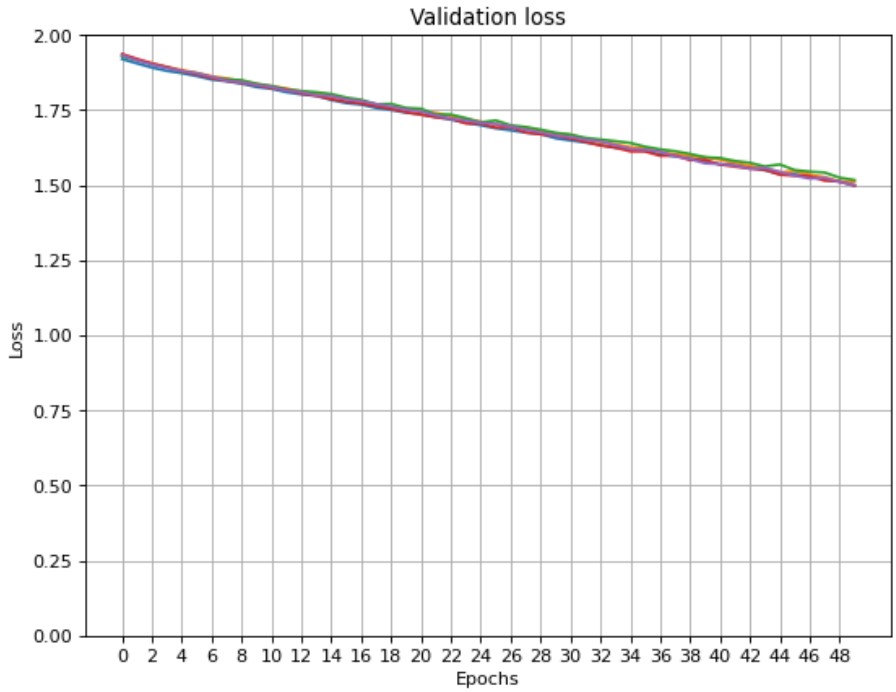

Figura 4.4: *Loss* de validação com última camada frases pré-processadas, LR= 1.00×10^{-6}. Fonte: De autoria própria.

4.1.2 Treinamento com liberação das últimas camadas do BERT-Base para retreino

Os resultados dos experimentos com o modelo BERT-Base acoplado ao classificador pode ser verificado nas tabelas de Precisão 4.1, Revocação 4.2 e medida F_1 4.3 nos experimentos de nome BERT-Base uma camada e BERT-Base duas camadas.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a última camada treinando com o modelo com LR=1.00×10^{-3}, é mostrada na imagem 4.5. A média dos valores de *Loss* foi de 0.300 registrada em média na época 47.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a última camada treinando com o modelo com TA =1.00×10^{-4}, é mostrada na imagem 4.6. A média das menores *Loss* foi de 0.058 registrada na época 48.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a última camada treinando com o modelo com TA =1.00×10^{-5}, é mostrada na imagem 4.7. A média das menores *Loss* foi de 0.098 registrada na época 47.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a última camada treinando com o modelo com TA =1.00×10^{-6}, é mostrada na imagem 4.8. A média das menores *Loss* foi de 0.651 registrada na época 49.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas duas camadas treinando com o modelo com TA =1.00×10^{-3}, é

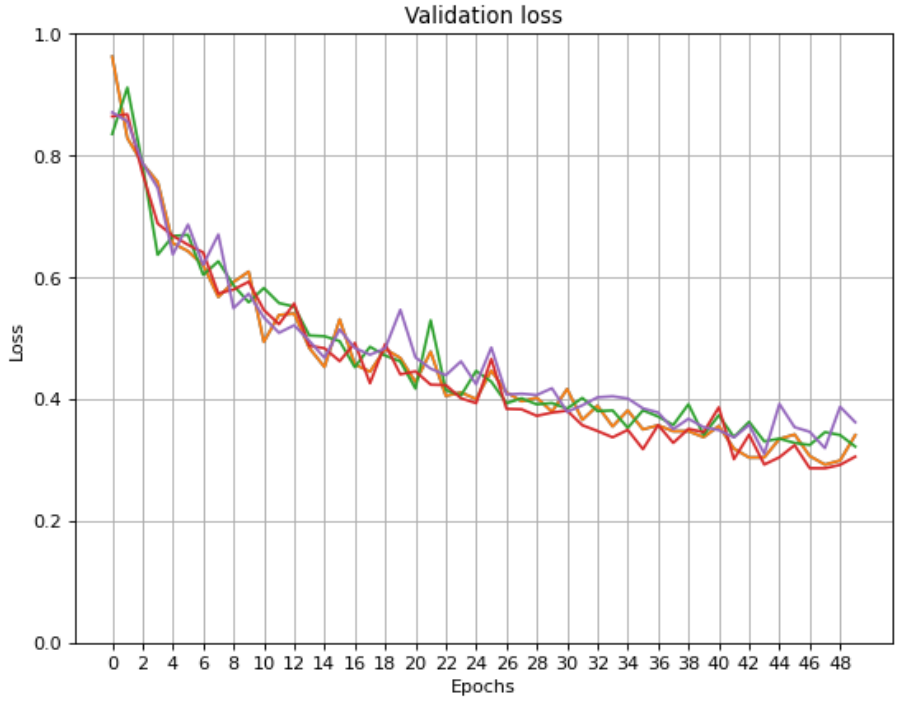

Figura 4.5: *Loss* de validação com última camada BERT liberada para treino, com LR=1.00×10^{-3}. Fonte: De autoria própria.

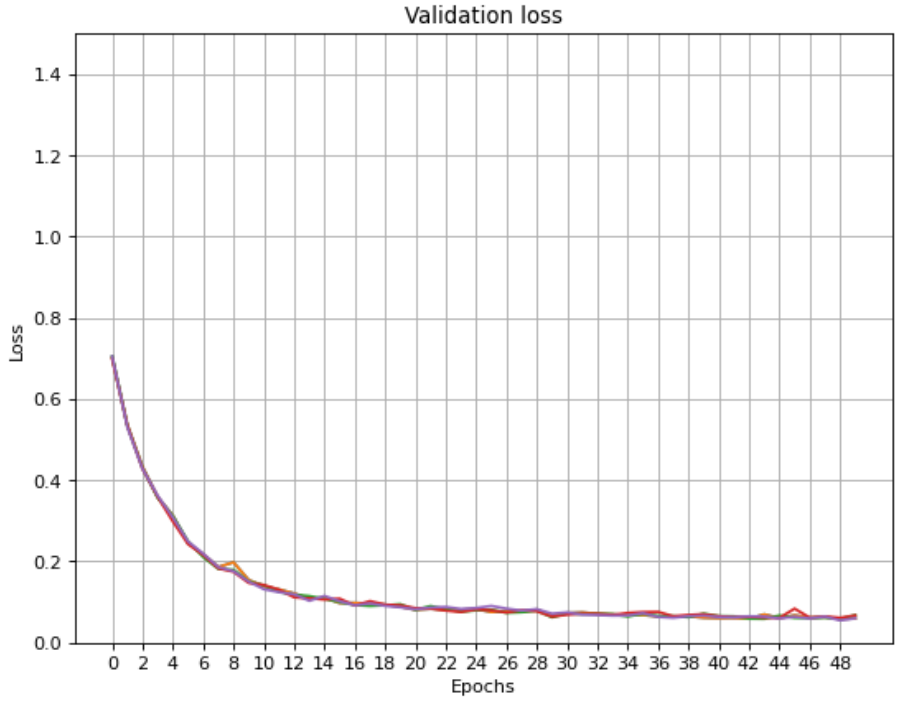

Figura 4.6: *Loss* de validação com última camada BERT liberada para treino com TA =1.00×10^{-4}. Fonte: De autoria própria.

mostrada na imagem 4.9. A média das menores *Loss* foi de 0.597 registrada na em média na época 45.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base,

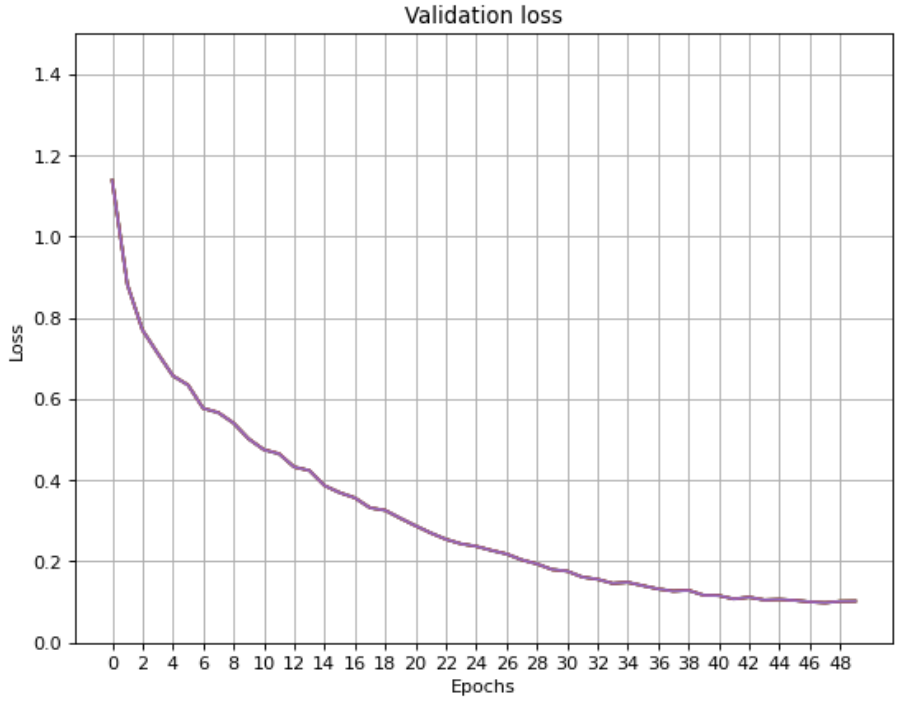

Figura 4.7: *Loss* de validação com última camada BERT liberada para treino com TA $=1.00 \times 10^{-5}$. Fonte: De autoria própria.

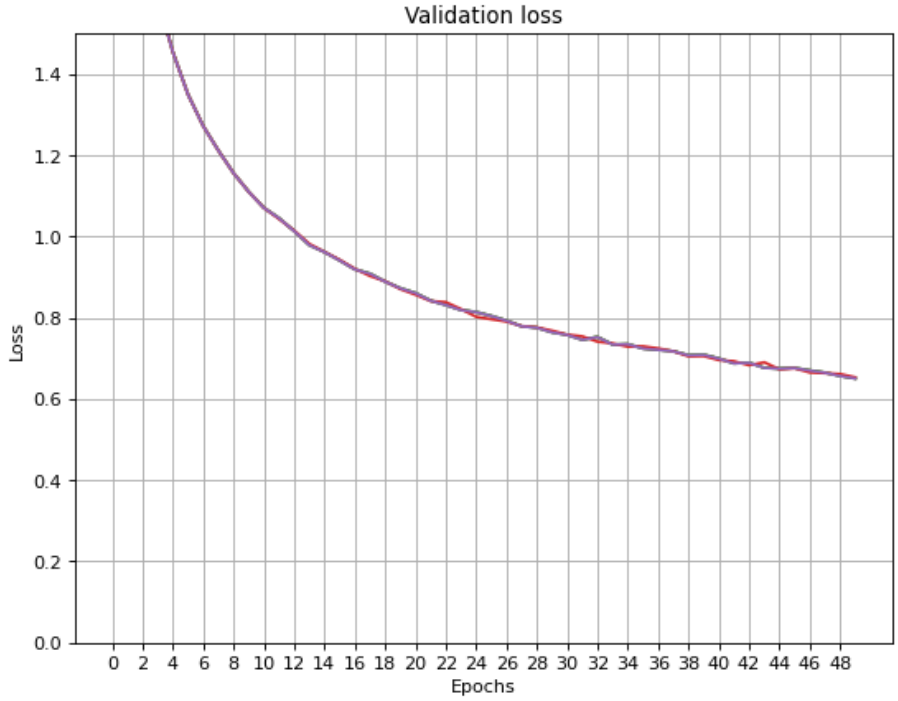

Figura 4.8: *Loss* de validação com última camada BERT liberada para treino com TA $=1.00 \times 10^{-6}$. Fonte: De autoria própria.

com a últimas duas camadas treinando com o modelo com TA $=1.00 \times 10^{-4}$, é mostrada na imagem 4.10. A média das menores *Loss* foi de 0.063 registrada em média na época 46.

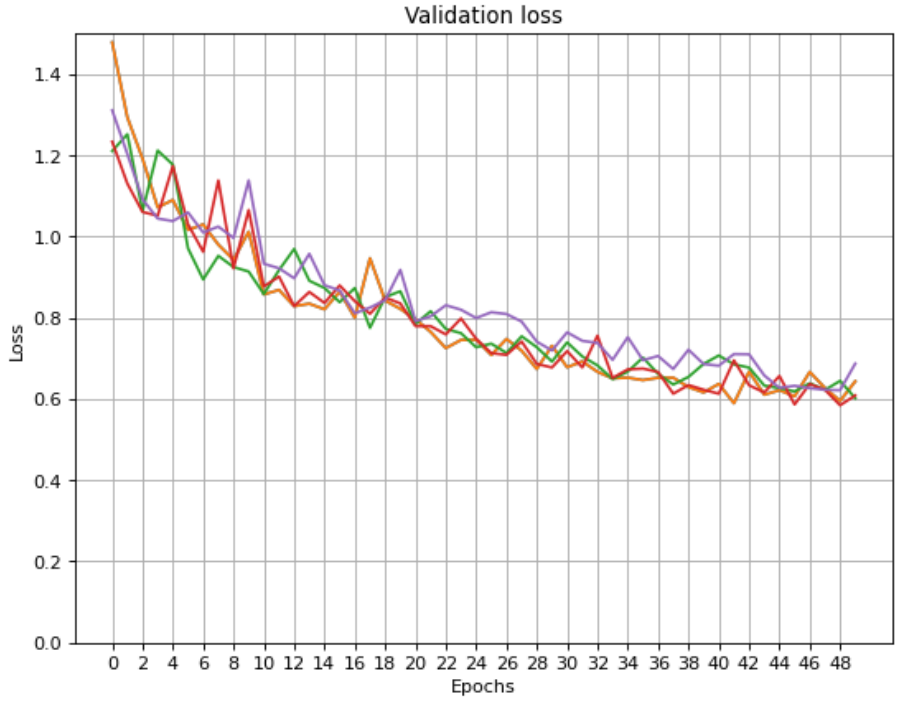

Figura 4.9: *Loss* de validação com última e penúltima camada BERT liberada para treino com TA $=1.00 \times 10^{-3}$. Fonte: De autoria própria.

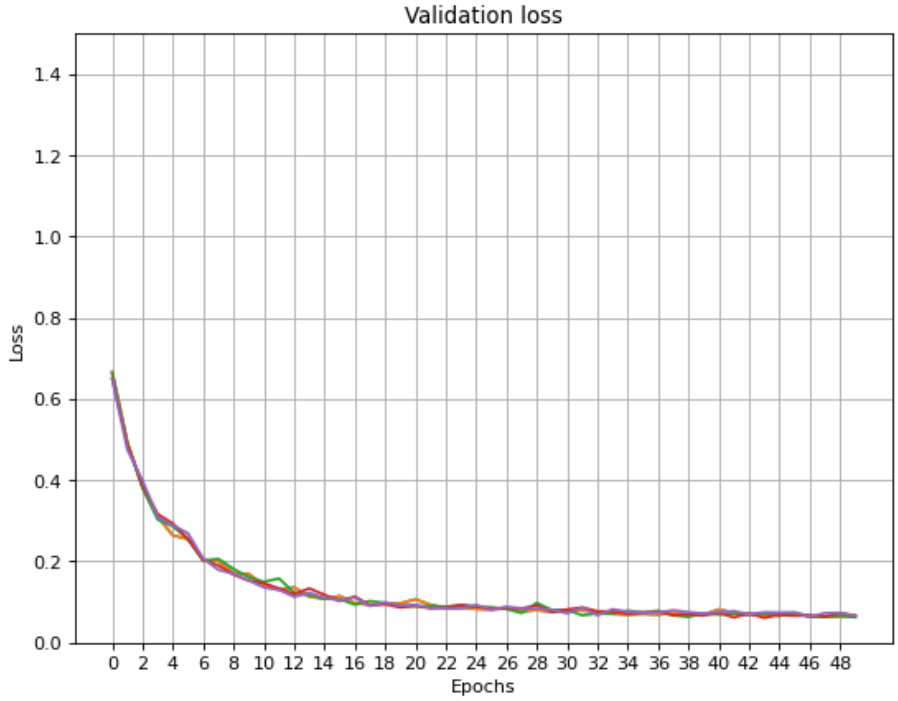

Figura 4.10: *Loss* de validação com última e penúltima camada BERT liberada para treino com TA $=1.00 \times 10^{-4}$. Fonte: De autoria própria.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas duas camadas treinando com o modelo com TA $=1.00 \times 10^{-5}$, é mostrada na imagem 4.11. A média das menores *Loss* foi de 0.076 registrada na

época 44.

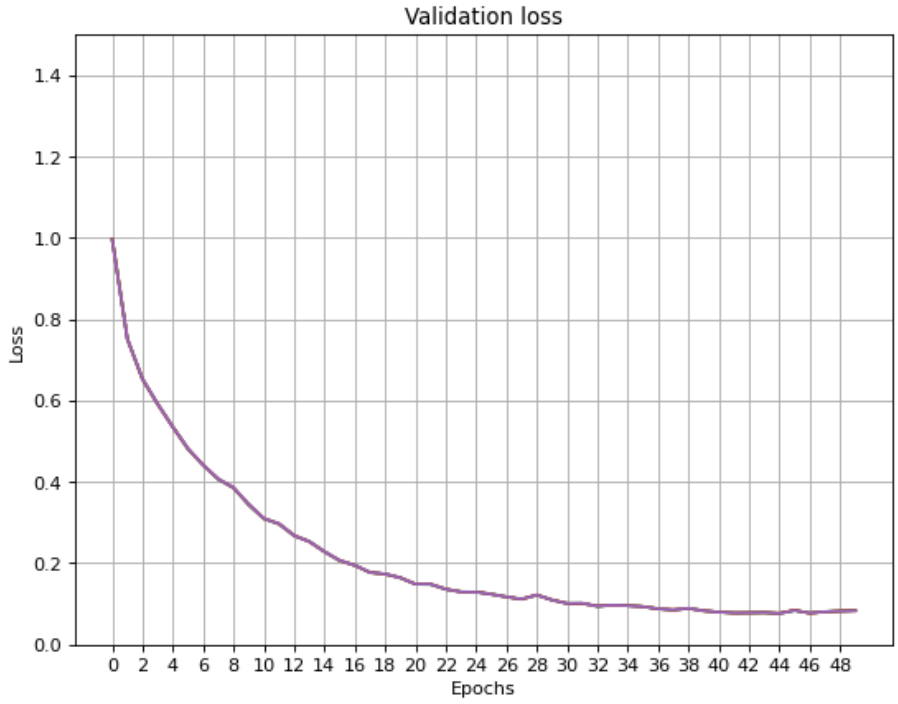

Figura 4.11: *Loss* de validação com última e penúltima camada BERT liberada para treino com TA $=1.00 \times 10^{-5}$. Fonte: De autoria própria.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas duas camadas treinando com o modelo com TA $=1.00 \times 10^{-6}$, é mostrada na imagem 4.12. A média das menores *Loss* foi de 0.504 registrada na época 49.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas duas camadas treinando com o modelo com TA $=1.00 \times 10^{-3}$, é mostrada na imagem 4.13. A média das menores *Loss* foi de 0. registrada na época 49.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas três camadas treinando com o modelo com TA $=1.00 \times 10^{-4}$, é mostrada na imagem 4.14. A média das menores *Losses* foi de 0.066 registrada na época 45.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas três camadas treinando com o modelo com TA $=1.00 \times 10^{-5}$, é mostrada na imagem 4.15. A média das menores *Loss* foi de 0.068 registrada na época 47.

A evolução da *Loss* de validação do treinamento do modelo com BERT-Base, com a últimas três camadas treinando com o modelo com TA $=1.00 \times 10^{-6}$, é mostrada na imagem 4.16. A média das menores *Loss* foi de 0.388 registrada na época 49.

As tabelas a seguir mostram os valores de de Precisão 4.1, Revocação 4.2 e medida F_1 4.3 para cada experimento. Os maiores valores estão destacados para cada classe.

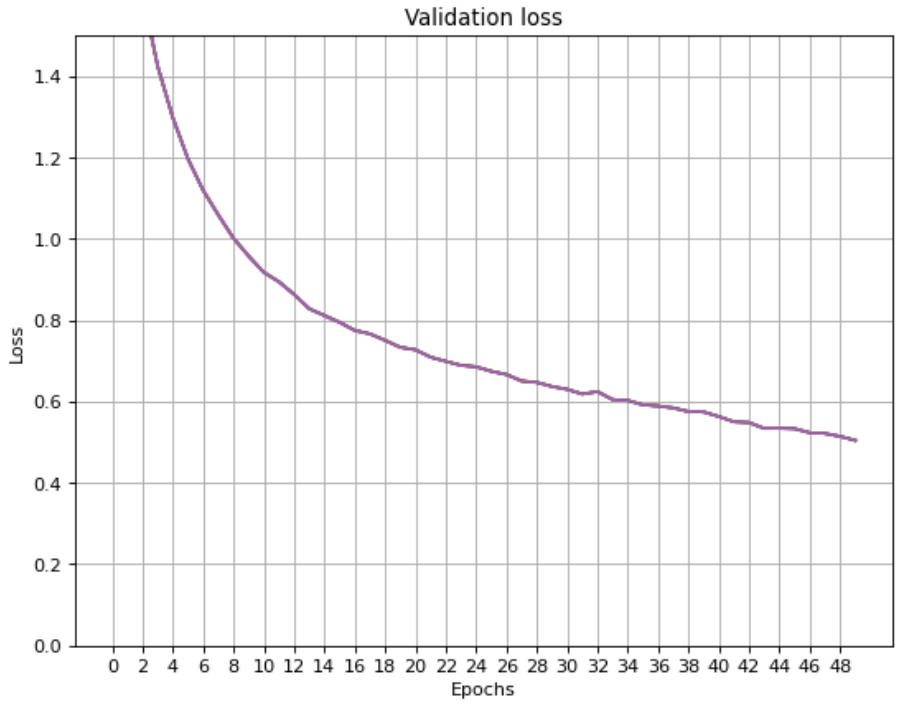

Figura 4.12: *Loss* de validação com última e penúltima camada BERT liberada para treino com TA $=1.00 \times 10^{-6}$. Fonte: De autoria própria.

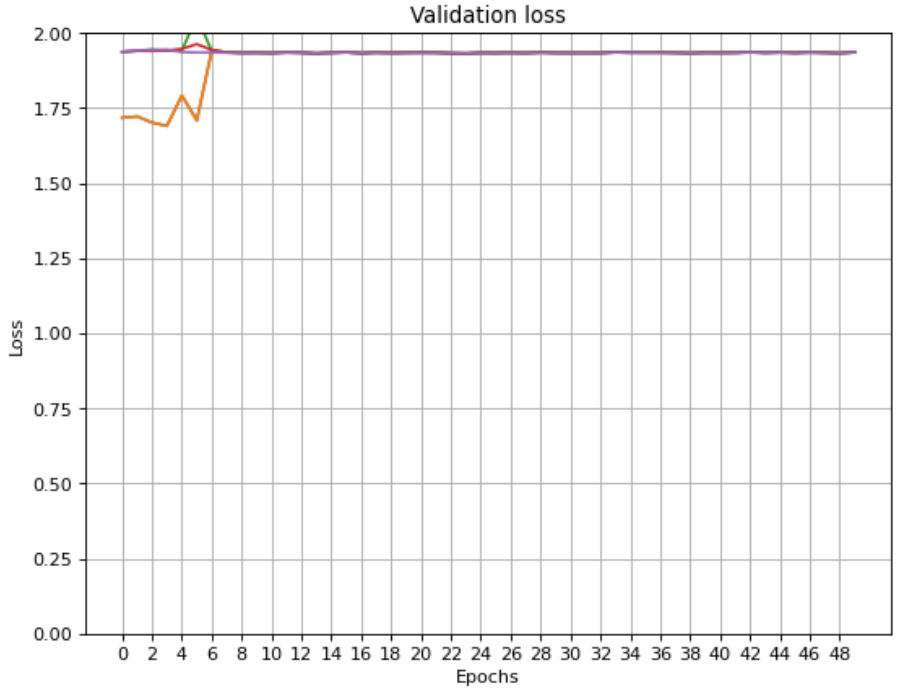

Figura 4.13: *Loss* de validação com três últimas camadas BERT liberadas para treino com TA $=1.00 \times 10^{-3}$. Fonte: De autoria própria.

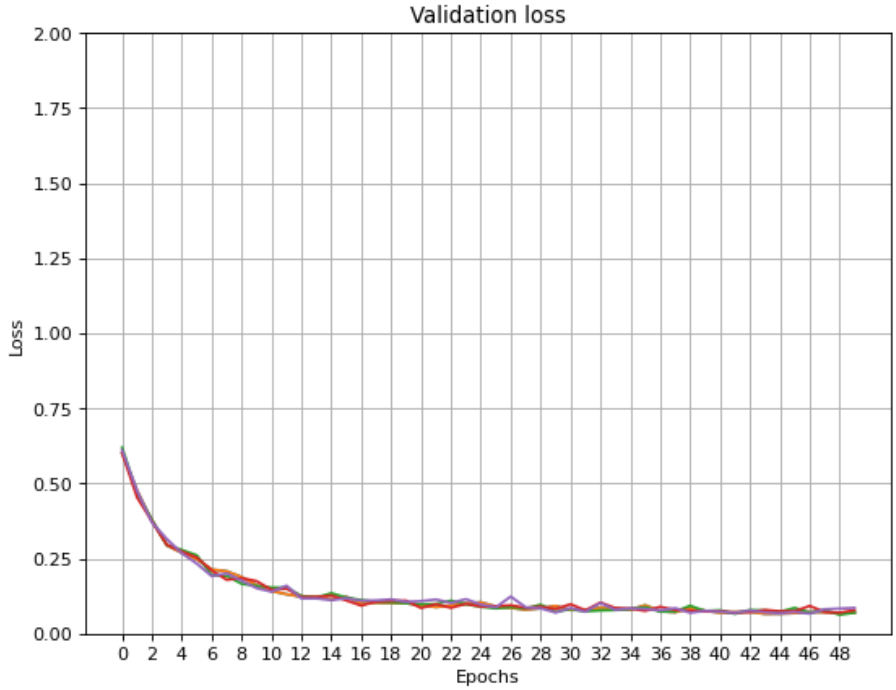

Figura 4.14: *Loss* de validação com três últimas camadas BERT liberadas para treino com TA $=1.00 \times 10^{-4}$. Fonte: De autoria própria.

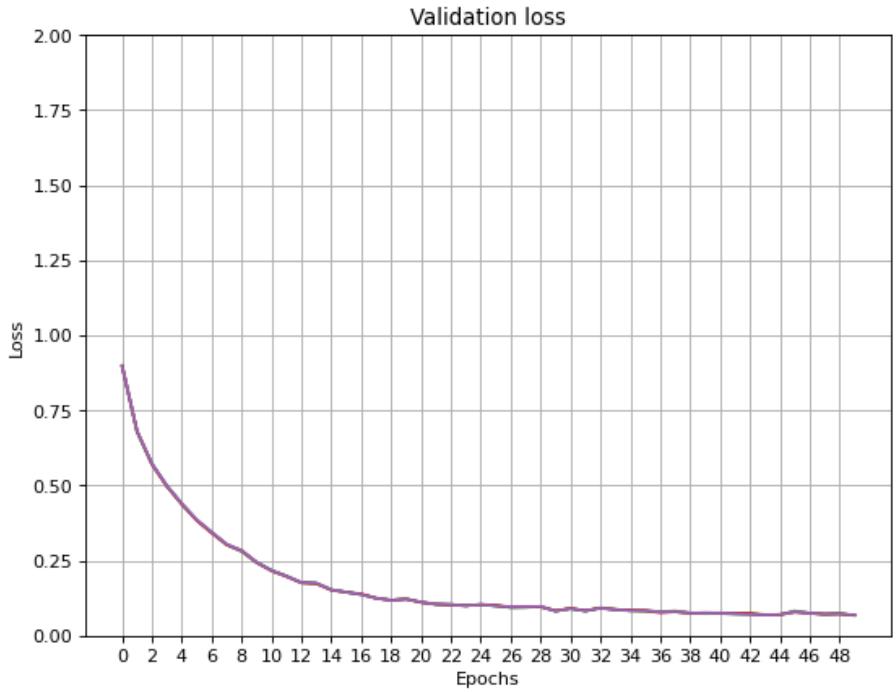

Figura 4.15: *Loss* de validação com três últimas camadas BERT liberadas para treino com TA $=1.00 \times 10^{-5}$. Fonte: De autoria própria.

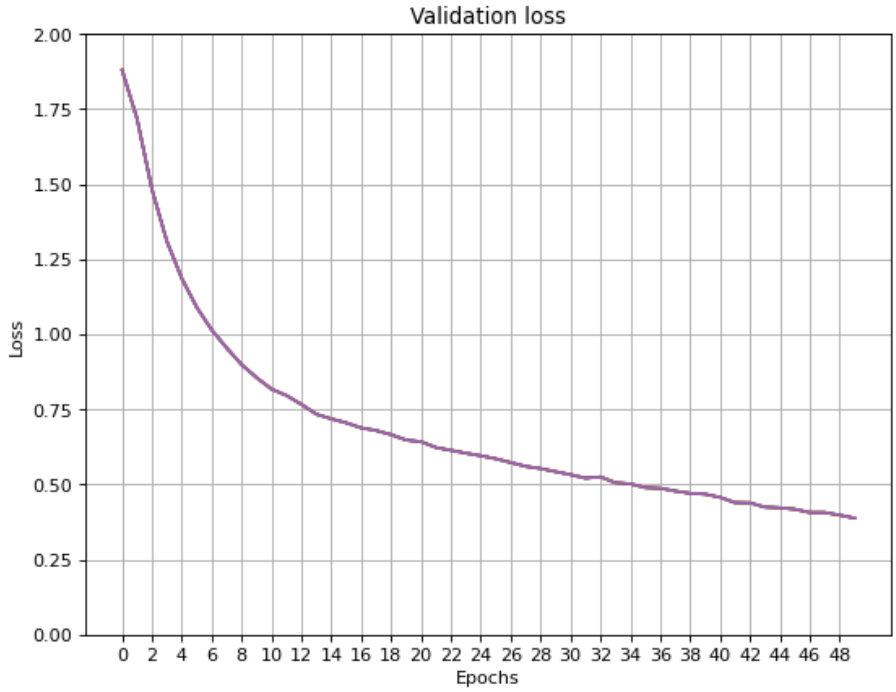

Figura 4.16: *Loss* de validação com três últimas camadas BERT liberadas para treino com TA $=1.00 \times 10^{-6}$. Fonte: De autoria própria.

Experimento	TA	Interlocutória	Decisão final	Marcam Data	Nada	Outro	Manifestações	Prisão
Pré-processado	1.00×10^{-3}	0.51	0.81	0.92	0.85	0.54	0.77	0.24
Pré-processado	1.00×10^{-4}	0.57	0.78	0.93	0.89	0.57	0.80	0.31
Pré-processado	1.00×10^{-5}	0.46	0.72	0.89	0.86	0.48	0.7	0.27
BERT-Base uma camada	1.00×10^{-3}	0.588	0.838	**0.99**	0.928	0.568	0.862	0.378
BERT-Base uma camada	1.00×10^{-4}	0.694	0.874	0.98	0.864	0.584	0.86	**0.78**
BERT-Base uma camada	1.00×10^{-5}	0.65	**0.892**	0.98	0.906	0.62	0.838	0.542
BERT-Base uma camada	1.00×10^{-6}	0.544	0.828	0.972	0.924	0.566	0.766	0.254
BERT-Base duas camadas	1.00×10^{-3}	0.634	0.802	0.98	**0.934**	0.444	**0.876**	0.224
BERT-Base duas camadas	1.00×10^{-4}	0.68	0.89	**0.99**	0.86	0.63	0.86	0.56
BERT-Base duas camadas	1.00×10^{-5}	0.684	0.882	0.988	0.866	0.634	0.858	0.5
BERT-Base duas camadas	1.00×10^{-6}	0.57	0.735	0.823	0.721	0.528	0.715	0.416
BERT-Base três camadas	1.00×10^{-3}	0.0	0.0	0.0	0.29	0.0	0.0	0.0
BERT-Base três camadas	1.00×10^{-4}	0.63	0.87	0.96	0.842	**0.656**	0.844	0.43
BERT-Base três camadas	1.00×10^{-5}	**0.71**	0.872	0.96	0.858	0.64	0.84	0.44
BERT-Base três camadas	1.00×10^{-6}	0.568	0.862	0.974	0.926	0.596	0.85	0.362

Tabela 4.1: Precisão por experimentos. Fonte: De autoria própria.

Experimento	TA	Interlocutória	Decisão final	Marcam Data	Nada	Outro	Manifestações	Prisão
Pré-processado	1.00×10^{-3}	0.51	0.81	0.92	0.85	0.54	0.77	0.24
Pré-processado	1.00×10^{-4}	0.57	0.78	0.93	**0.89**	0.57	0.80	0.31
Pré-processado	1.00×10^{-5}	0.46	0.72	0.89	0.86	0.48	0.7	0.27
BERT-Base uma camada	1.00×10^{-3}	0.702	0.914	**0.96**	0.798	0.646	0.806	0.53
BERT-Base uma camada	1.00×10^{-4}	0.674	0.93	**0.96**	0.864	0.612	0.828	0.41
BERT-Base uma camada	1.00×10^{-5}	**0.698**	**0.92**	**0.96**	0.862	0.638	0.84	0.53
BERT-Base uma camada	1.00×10^{-6}	0.676	0.864	**0.96**	0.758	0.604	0.786	0.41
BERT-Base duas camadas	1.00×10^{-3}	0.6	0.888	0.95	0.724	**0.756**	0.626	0.506
BERT-Base duas camadas	1.00×10^{-4}	0.66	**0.92**	0.95	**0.89**	0.64	0.814	0.53
BERT-Base duas camadas	1.00×10^{-5}	0.692	0.914	0.95	0.874	0.612	0.834	**0.59**
BERT-Base duas camadas	1.00×10^{-6}	0.576	0.761	0.791	0.728	0.51	0.695	0.491
BERT-Base três camadas	1.00×10^{-3}	0.0	0.0	0.0	1	0.0	0.0	0.0
BERT-Base três camadas	1.00×10^{-4}	0.654	**0.92**	0.94	0.874	0.53	**0.854**	0.53
BERT-Base três camadas	1.00×10^{-5}	0.69	0.912	**0.96**	0.862	0.598	0.85	0.47
BERT-Base três camadas	1.00×10^{-6}	0.712	0.906	0.96	0.816	0.644	0.812	0.53

Tabela 4.2: Revocação por experimentos. Fonte: De autoria própria.

Experimento	TA	Interlocutória	Decisão final	Marcam Data	Nada	Outro	Manifestações	Prisão
Pré-processado	1.00×10^{-3}	0.53	0.83	0.92	0.81	0.56	0.73	0.34
Pré-processado	1.00×10^{-4}	0.58	0.82	0.95	0.83	0.61	0.77	0.41
Pré-processado	1.00×10^{-5}	0.52	0.75	0.93	0.77	0.51	0.67	0.36
BERT-Base uma camada	1.00×10^{-3}	0.642	0.872	**0.97**	0.858	0.604	0.83	0.53
BERT-Base uma camada	1.00×10^{-4}	0.684	0.9	**0.97**	0.864	0.598	0.844	0.54
BERT-Base uma camada	1.00×10^{-5}	0.672	**0.904**	**0.97**	**0.882**	0.626	0.84	0.538
BERT-Base uma camada	1.00×10^{-6}	0.602	0.846	0.964	0.832	0.586	0.778	0.314
BERT-Base duas camadas	1.00×10^{-3}	0.614	0.842	0.962	0.816	0.562	0.728	0.31
BERT-Base duas camadas	1.00×10^{-4}	0.67	0.9	**0.97**	0.874	**0.634**	0.84	**0.55**
BERT-Base duas camadas	1.00×10^{-5}	0.69	0.896	0.968	0.87	0.622	0.844	0.54
BERT-Base duas camadas	1.00×10^{-6}	0.575	0.746	0.806	0.725	0.518	0.703	0.45
BERT-Base três camadas	1.00×10^{-3}	0.0	0.0	0.0	0.45	0.0	0.0	0.0
BERT-Base três camadas	1.00×10^{-4}	0.64	0.89	0.95	0.86	0.59	0.85	0.47
BERT-Base três camadas	1.00×10^{-5}	**0.702**	0.892	0.96	0.86	0.618	**0.846**	0.46
BERT-Base três camadas	1.00×10^{-6}	0.632	0.884	0.964	0.868	0.618	0.83	0.428

Tabela 4.3: Medida F1 por experimentos. Fonte: De autoria própria.

4.2 Discussão dos resultados

4.2.1 Resultado do treino dos classificadores

Configuração	Taxa de Aprendizado	Acurácia	Acurácia ponderada
Pré-processado	1.00×10^{-3}	0.73	0.74
Pré-processado	1.00×10^{-4}	0.76	0.77
Pré-processado	1.00×10^{-5}	0.69	0.69
BERT-Base uma camada	1.00×10^{-3}	0.8	0.806
BERT-Base uma camada	1.00×10^{-4}	0.818	0.816
BERT-Base uma camada	1.00×10^{-5}	**0.822**	**0.826**
BERT-Base uma camada	1.00×10^{-6}	0.766	0.772
BERT-Base duas camadas	1.00×10^{-3}	0.74	0.75
BERT-Base duas camadas	1.00×10^{-4}	0.82	0.82
BERT-Base duas camadas	1.00×10^{-5}	0.82	0.82
BERT-Base duas camadas	1.00×10^{-6}	0.683	0.683
BERT-Base três camadas	1.00×10^{-3}	0.29	0.13
BERT-Base três camadas	1.00×10^{-4}	0.81	0.81
BERT-Base três camadas	1.00×10^{-5}	0.82	0.818
BERT-Base três camadas	1.00×10^{-6}	0.804	0.81

Tabela 4.4: Acurácia e Acurácia ponderada por experimento. Fonte: De autoria própria.

Experimento	TA	Avg validation loss	SD validation loss	Avg epoch	SD epoch
BERT-Base uma camada	1.00×10^{-3}	0.300603	0.014882	46.6	2.190890
BERT-Base uma camada	1.00×10^{-4}	0.058264	0.002242	47	2.236067
BERT-Base uma camada	1.00×10^{-5}	0.098356	0.000031	47	0
BERT-Base uma camada	1.00×10^{-6}	0.651317	0.001195	49	0.001195
BERT-Base duas camadas	1.00×10^{-3}	0.597504	0.014989	45.4	4.037325
BERT-Base duas camadas	1.00×10^{-4}	0.063055	0.001173	46.4	2.190890
BERT-Base duas camadas	1.00×10^{-5}	0.076865	0.000305	44	0
BERT-Base duas camadas	1.00×10^{-6}	0.504650	0.000074	15	10
BERT-Base três camadas	1.00×10^{-3}	1.835331	0.132226	45.4	4.037325
BERT-Base três camadas	1.00×10^{-4}	0.066242	0.002542	45.6	2.509980
BERT-Base três camadas	1.00×10^{-5}	0.068147	0.000640	47.8	2.683281
BERT-Base três camadas	1.00×10^{-6}	0.388041	0.0000595	49	0

Tabela 4.5: Média e desvio padrão do menor valor de *loss* de cada experimento e média e desvio padrão da época que os valores de *loss* foram registrados. Fonte: De autoria própria.

Comparação entre Taxa de Aprendizado dos treinos com frases pré-processadas

Os experimentos com frases pré-processadas foram feitos com variações de Taxa de Aprendizado de 1.00×10^{-3}, 1.00×10^{-4} e 1.00×10^{-5}.

O modelo que se destacou mais em capacidade de classificação foi a versão com TA = 1.00×10^{-4} com Acurácia de 0.76 contra 0.73 do modelo com TA de 1.00×10^{-3} e 0.69 do modelo com TA de 1.00×10^{-5}.

Para a Acurácia ponderada os valores foram de 0.77 para o modelo com TA = 1.00×10^{-4}, de 0.74 para o modelo com TA de 1.00×10^{-3} e de 0.69 para o modelo com TA = 1.00×10^{-5}.

Comparando o medida F_1 por classe a tendência se mantém com as medidas do modelo com TA de 1.00×10^{-4} superando as medidas F_1 das outras variantes em seis das sete classes perdendo apenas na decisão final com um valor de 0.82 contra 0.83 do modelo com 1.00×10^{-3}. Para o rótulo prisão a varição foi maior e o modelo com TA de 1.00×10^{-4} obteve uma medida F_1 de 0.41 contra 0.36 e 0.34 dos modelos com TA de 1.00×10^{-3} e 1.00×10^{-5}.

Comparação entre Taxas de Aprendizado dos treinos com uma camada BERT liberada para treino

Os experimentos com a última camada liberada para retreino foram feitos com variações de Taxa de Aprendizado de 1.00×10^{-3}, 1.00×10^{-4}, 1.00×10^{-5} e 1.00×10^{-6}.

O modelo que se destacou mais em capacidade de classificação foi a versão com TA de 1.00×10^{-5} com Acurácia de 0.822 contra 0.8 do modelo com TA = 1.00×10^{-3}, 0.818 do modelo com TA de 1.00×10^{-4} e 0.766 do modelo com TA de 1.00×10^{-6}.

Avaliando a Acurácia ponderada os valores foram de 0.826 para o modelo com TA = 1.00×10^{-5}, de 0.806 para o modelo com TA de 1.00×10^{-3}, de 0.816 para o modelo com TA = 1.00×10^{-4} e de 0.772 para o modelo com TA = 1.00×10^{-6}.

Comparando a medida F_1 por classe a tendência é confirmada no modelo com TA de 1.00×10^{-5} tendo quatro de sete dos valores maiores ou iguais aos outros modelos. O modelo com TA de 1.00×10^{-6} se destacou negativamente no rótulo prisão com valor medida F_1 de 0.314 comparado com 0.53, 0.54 e 0.538 dos modelos com TA de 1.00×10^{-3}, 1.00×10^{-4} e 1.00×10^{-5} respectivamente.

Comparação entre Taxas de Aprendizado dos treinos com duas camadas BERT liberadas para treino

Os experimentos com as duas últimas camada liberada para retreino foram feitos com variações de Taxas de Aprendizado de 1.00×10^{-3}, 1.00×10^{-4}, 1.00×10^{-5} e 1.00×10^{-6}.

Os modelos que se destacou mais em capacidade de classificação foi a versão com TA de 1.00×10^{-4} com Acurácia de 0.82 contra 0.74 do modelo com TA = 1.00×10^{-3}, 0.82 do modelo com TA de 1.00×10^{-5} e 0.683 do modelo com TA de 1.00×10^{-6}.

Avaliando a Acurácia ponderada os valores foram de 0.82 para o modelo com TA = 1.00×10^{-4}, de 0.75 para o modelo com TA de 1.00×10^{-3}, de 0.82 para o modelo

com TA = 1.00×10^{-5} e de 0.683 para o modelo com TA = 1.00×10^{-6}.

Comparando a medida F_1 por classe o modelo com TA de 1.00×10^{-4} supera os outros com cinco medidas de sete superando as outras variantes.

Comparação entre Taxa de Aprendizado dos treinos com três camadas BERT liberadas para treino

Os experimentos com as três últimas camada liberada para retreino foram feitos com variações de Taxa de Aprendizado de 1.00×10^{-3}, 1.00×10^{-4}, 1.00×10^{-5} e 1.00×10^{-6}.

Os modelos que se destacou mais em capacidade de classificação foi a versão com TA de 1.00×10^{-5} com Acurácia de 0.82 contra 0.29 do modelo com TA = 1.00×10^{-3}, 0.81 do modelo com TA de 1.00×10^{-4} e 0.804 do modelo com TA de 1.00×10^{-6}.

Avaliando a Acurácia ponderada os valores foram de 0.818 para o modelo com TA = 1.00×10^{-5}, de 0.13 para o modelo com TA de 1.00×10^{-3}, de 0.81 para o modelo com TA = 1.00×10^{-4} e de 0.81 para o modelo com TA = 1.00×10^{-6}.

Comparando a medida F_1 por classe o modelo com TA de 1.00×10^{-5} supera os outros com cinco medidas de sete superando as outras variantes.

Comparação entre experimentos com uma camada liberada para treino com TA = 1.00×10^{-3} e todos os experimentos com frases pré-processadas

O experimento de uma camada BERT liberada para treino na configuração de Taxa de Aprendizado de 1.00×10^{-3} mostrou ganho de Acurácia em relação aos experimentos com frases pré-processadas de 0.8 contra 0.73 do experimento utilizando frases pré-processadas com Taxa de Aprendizado 1.00×10^{-3}, 0.76 do experimento utilizando frases pré-processadas com Taxa de Aprendizado 1.00×10^{-4} e 0.69 do experimento utilizando frases pré-processadas com Taxa de Aprendizado 1.00×10^{-5}. Com o mesmo Taxa de Aprendizado a liberação de uma camada melhorou os indicadores de capacidade de classificação do modelo.

Comparação entre experimentos com uma camada, duas camadas e três camadas liberadas para treino com TA = 1.00×10^{-3}

Comparando os experimentos da tabela 4.3 BERT uma camada, BERT duas camadas com TA = 1.00×10^{-3} a Acurácia, Acurácia ponderada (tabela 4.4) e as medidas F_1 foram superiores no experimento de uma camada. O experimento BERT três camadas com TA = 1.00×10^{-3} não conseguiu aprender, ficando com desempenho pior que o experimento com frases pré-processadas de referência.

A redução da Acurácia do experimento com uma camada BERT retreinando foi de 0.8 para 0.74 do experimento com duas camada BERT retreinando. Na medida Acurácia ponderada a redução foi de 0.806 para 0.75 respectivamente.

A maior diferença foi para o rótulo Prisão em que a medida F_1 foi de 0.53 contra 0.312 para os classificadores respectivamente.

Comparação entre experimentos com uma camada, duas camadas e três camadas liberadas para treino com TA $= 1.00 \times 10^{-4}$

Comparando os experimentos BERT uma camada, BERT duas camadas e BERT três camadas com TA $= 1.00 \times 10^{-4}$ a Acurácia, Acurácia ponderada (tabela 4.4) e as medidas F_1 (tabela 4.3) foram melhores no experimento de duas camadas, seguido do experimento de uma camada e por último três camadas.

O aumento da Acurácia do experimento com uma camada BERT foi de 0.818 para 0.82 do experimento com duas camada BERT e reduziu para 0.81 com três camadas.

Na medida Acurácia ponderada o aumento foi de 0.816 do experimento de uma camada BERT para 0.82 no experimento de duas camadas BERT e reduziu para 0.81 no experimento de três camadas. A medida F_1 para as classes "Decisão final" com 0.9 e "Marcam data" com 0.97 foi igual entre os modelos.

A maior diferença foi para o rótulo "Outro" em que a medida F_1 foi de 0.598 do modelo com retreino de uma camada, 0.59 do modelo com retreino de três camadas e de 0.634 para o modelo com retreino de duas camadas.

Comparação entre experimentos com uma camada, duas camadas e três camadas liberadas para treino com TA $= 1.00 \times 10^{-5}$

Comparando os experimentos BERT uma camada, BERT duas camadas e BERT três camadas com TA $= 1.00 \times 10^{-5}$ a Acurácia, Acurácia ponderada (tabela 4.4) e as medidas F_1 (tabela 4.3) foram melhores no experimento de uma camada, seguido do experimento de duas camadas e por último três camadas.

A redução da Acurácia do experimento com uma camada BERT retreinando de 0.822 para 0.82 do experimento com duas camada BERT retreinando e se manteve em 0.82 no experimento com três camadas BERT.

Na medida Acurácia ponderada a redução foi de 0.826 no experimento de uma camada BERT para 0.82 no experimento de duas camadas BERT e novamente reduziu para 0.818 no experimento com três camadas BERT. O modelo com retreino de uma camada superou os modelos com retreino de duas camadas e três camadas nas classes "Decisão final" com medida F_1 de 0.904 contra 0.896 e 0.892 respectivamente. Para a classe "Marcam data" o experimento de uma camada obteve 0.97

contra 0.968 e 0,96 respectivamente. Para a classe "Nada" o experimento de uma camada obteve 0.882 contra 0.87 e 0.86 respectivamente. Para a classe "Outro" com 0.626 contra 0.622. No rótulo de "Prisão" houve um empate de 0.54.

O modelo com retreino de duas camadas superou o modelo com retreino de uma camada nos rótulos "Interlocutória" com 0.69 contra 0.672 e "Manifestações" com 0.844 contra 0.84.

Comparação entre experimentos com uma camada, duas camadas e três camadas liberadas para treino com TA $= 1.00 \times 10^{-6}$

Comparando o experimento BERT uma camada com TA $= 1.00 \times 10^{-6}$ com o experimento BERT duas camadas com TA $= 1.00 \times 10^{-6}$ a acurácia, acurácia ponderada (tabela 4.4) e as medidas F_1 (tabela 4.3) reduziram. A redução da acurácia do experimento com uma camada BERT retreinando de 0.766 para 0.683 do experimento com duas camada BERT retreinando. Na medida acurácia ponderada a redução foi de 0.772 para 0.683 respectivamente.

O modelo com retreino de uma camada superou o modelo com retreino de duas camadas em seis das sete classes: "Interlocutória" com 0.602 contra 0.575, "Decisão final" com 0.846 contra 0.746, "Marcam data" com 0.964 contra 0.806, "Nada" com 0.832 contra 0.725, "Outro" com 0.586 contra 0.518 e "Manifestações" com 0.778 contra 0.703. Para o rótulo "Prisão", o modelo com duas camadas foi superior, com 0.45 contra 0.314 do modelo com uma camada para retreino.

4.2.2 Classificação e separabilidade das classes

Avaliando a acurácia e acurácia ponderada os modelos com retreino de uma, duas e três camadas com TA $= 1.00 \times 10^{-5}$ mostraram as melhores medidas de classificação na média. O experimento inicial com frases pré-processadas com Taxa de Aprendizado 1.00×10^{-3} comparado com o experimento com melhor resultado teve um aumento de 0.73 de acurácia média para 0.822 e de acurácia ponderada média de 0.74 para 0.826.

Os modelos com duas e três camadas para retreino perderam capacidade de classificação com o aumento de parâmetros por conta da camada adicional retreinando. O modelo de duas camadas retreinando com o TA de 1.00×10^{-4} foi superior ao modelo de TA de 1.00×10^{-4} com uma camada liberada para retreino. Para o experimento com TA de 1.00×10^{-6} o modelo com retreino de três camadas superou os experimentos com duas e uma camada.

As imagens 4.17 e 4.18 mostram a saída antes da aplicação de *softmax* dos modelos com melhor desempenho utilizando a técnica t-SNE para reduzir a dimensionalidade de sete para duas dimensões. O desempenho superior das classificações

de elementos da classe “Marcam Data” pode ser observado pela uniformidade do agrupamento dos elementos e a distância dos outros agrupamentos. “Prisão” tem uma representatividade pequena no conjunto de dados consequentemente reduzindo a capacidade de aprendizado da classe pelos classificadores e pode ser a causa do pior desempenho dentre os rótulos classificados.

De acordo com a tabela 3.3 as classes “Nada” e “Outro” têm temas genéricos comparados aos outros rótulos, porém o classificador conseguiu separar a classe “Nada” com mais qualidade de acordo com a medida F_1 de 0,866 contra 0,602 da classe “Outro” na tabela 4.3 para o treino BERT-Base uma camada com TA = 1.00×10^{-5} e de 0,866 contra 0,62 para o treino BERT-Base duas camadas com TA = 1.00×10^{-5}. A proporção delas não é igual e indica vantagem do suporte mais alto de 33,3% e 10,7% respectivamente.

A Figura 4.19 é uma matriz de confusão média das 5 execuções do modelo BERT uma camada com TA = 1.00×10^{-5}. É possível observar a proximidade da classe “Outro” com “Decisão Interlocutória” e “Prazos e Manifestações”. Entre a classe “Interlocutória” é classificada com 71.8 elementos em média corretamente e 13 elementos são classificados em média como “Outro”. Para a classe “Outro” 119.2 elementos são classificados corretamente e 22.6 são classificados como “Interlocutória”. A classe “Manifestações” tem 37.4 elementos classificados como “Outro”. A classe “Outro” tem 31.8 elementos classificados como “Manifestações”. Na Figura 4.20 erros de classificação semelhante podem ser observados.

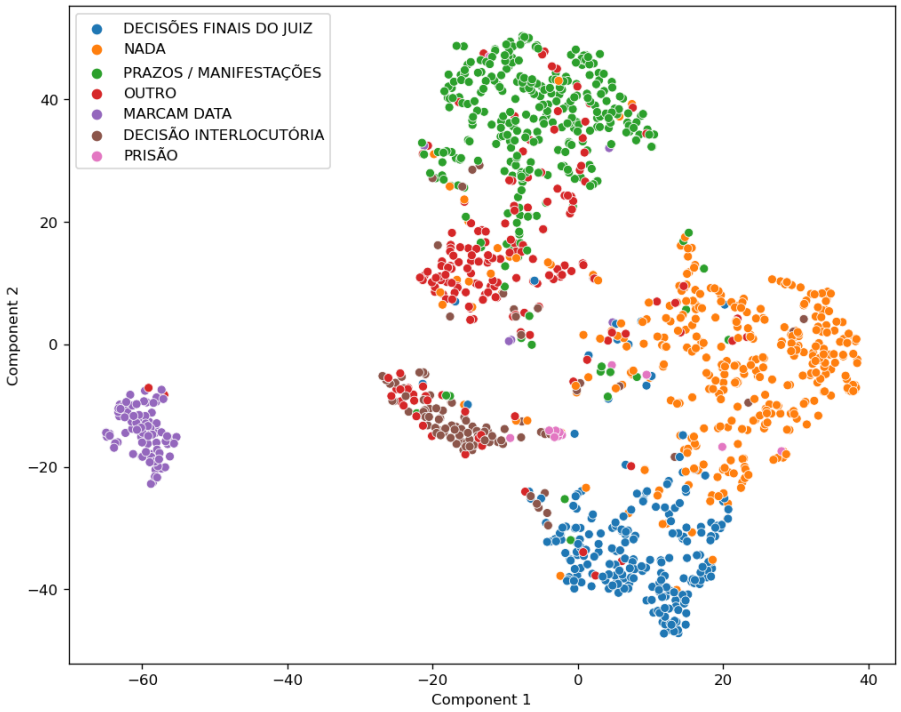

Figura 4.17: Agrupamentos gerados com elementos do conjunto de teste no classificador do experimento BERT-Base uma camada com TA $=1.00 \times 10^{-5}$. Fonte: De autoria própria.

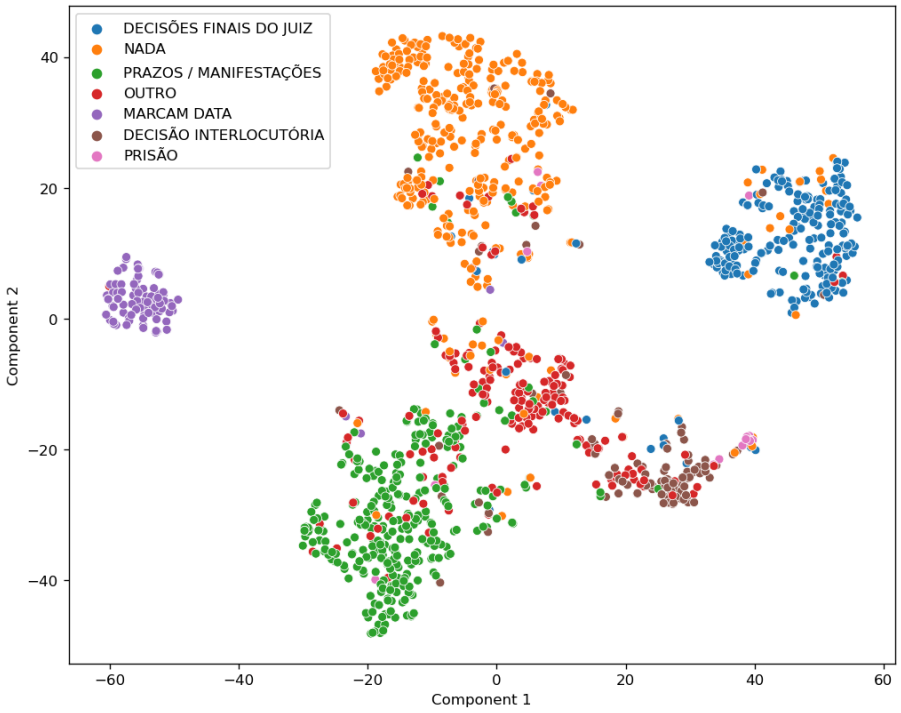

Figura 4.18: Agrupamentos gerados com elementos do conjunto de teste no classificador do experimento BERT-Base duas camadas com TA $=1.00 \times 10^{-5}$. Fonte: De autoria própria.

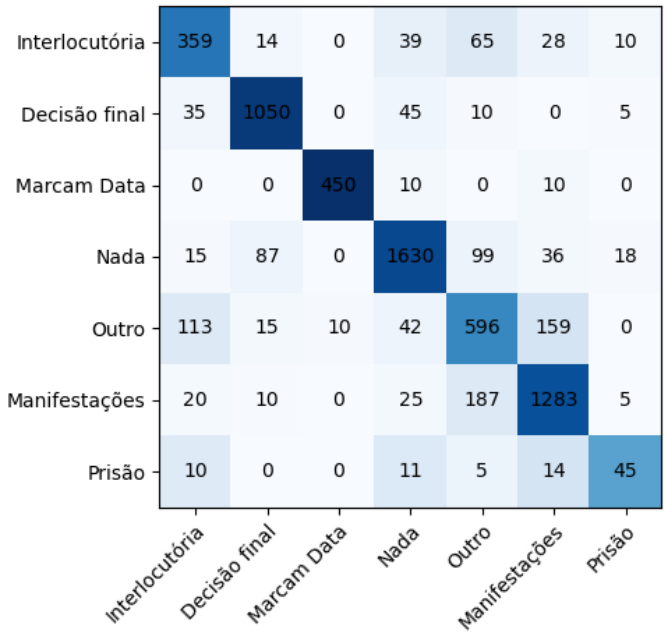

Figura 4.19: Soma das matrizes de confusão da validação cruzada do modelo de uma camada com TA $=1.00 \times 10^{-5}$. Linhas são o valor de referência do conjunto de teste e colunas são classificações. Fonte: De autoria própria.

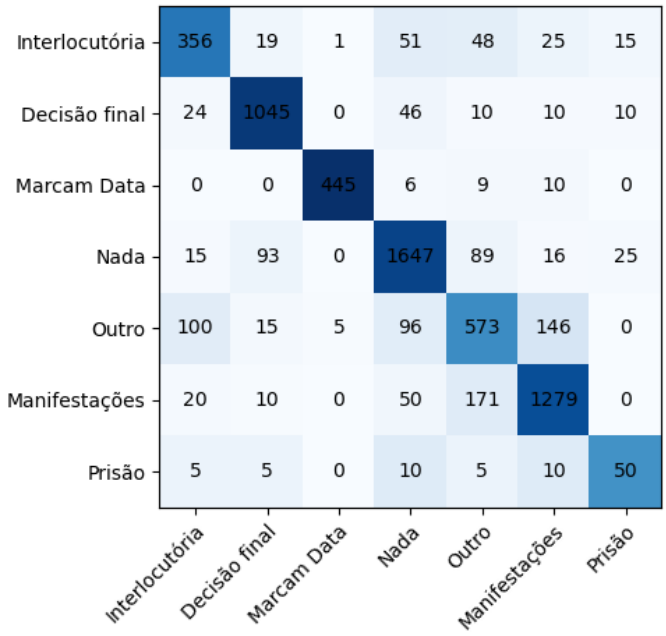

Figura 4.20: Soma das matrizes de confusão da validação cruzada do modelo de duas camadas com TA $= 1.00 \times 10^{-5}$. Linhas são o valor de referência do conjunto de teste e colunas são classificações. Fonte: De autoria própria.

4.2.3 Viabilidade e impacto na operação do CICIAR

O CICIAR opera em uma janela de de classificação das 19h as 7h para não sobrecarregar o Verde. Para verificar a viabilidade da implementação dessas alterações é necessário levar em conta se a carga de processamento da nova proposta de classificação é compatível com a disponibilidade atual de hardware e tempo de operação. A tabela 4.6 indica o tempo e o consumo de memória da placa de vídeo para atividades de treino e classificação.

	Tempo de treino	Tempo de teste	*Batch* de teste
Pré-processado	1min10s	1s	1313
BERT-Base uma camada	23h	34s	16
BERT-Base duas camada	30h	35s	16
BERT-Base três camada	33h	37s	16

Tabela 4.6: Tempo necessário para treinar e classificar. Fonte: De autoria própria.

Experimento	Parâmetros
Pré-processado	397.319
BERT-Base uma camada	7.678.464
BERT-Base duas camada	15.163.655
BERT-Base três camada	22.251.527

Tabela 4.7: Uso de recursos por tipo de treinamento e classificação. Fonte: De autoria própria.

A imagem 4.21 mostra a interação entre o CICIAR, o WebService do TJ e o Verde. Nessa imagem p1 representa o tempo da rotina de atualização dos avisos de intimações, p2 representa o tempo de recebimento de documentos do Verde, p3 representa a requisição de metadados ao WebService do TJ, p4 representa o envio de classificações ao Verde.

O maior tempo registrado de processamento dentro da janela de 720 minutos foi 481 minutos, 66,8% do tempo. A média atual de tempo para classificar documentos é de 14 segundos O tempo registrado para o classificador de frases pré-processadas foi de 1 segundo para 1313 frases.

Nos experimentos com modelo com BERT-Base acoplado ao classificador 34 segundos tempo de classificação para 1313 amostras divididas em *batchs* de 16, sendo necessárias 83 operações para as 1313 amostras. Cada operação de classificação dura 83/34 segundos = 2,44 segundos.

O tempo por amostra para cada experimento é de 1/1313 = 0,0007 segundos para a classificação de frases pré-processadas e 2,4/16 = 0,15 segundos para a classificação com utilizando a estrutura com BERT-Base acoplada ao classificador.

O tempo necessário para classificar uma frase por vez é de 0,228 segundos ou 4,37 frases por segundo para o classificador com BERT-Base duas camadas. A

classificação de uma frase não explora bem a capacidade de paralelismo da placa de vídeo, e pode ser uma referência de pior caso.

O tempo necessário para classificar uma frase com um classificador sem BERT acoplado, utilizando frases pré-processadas e classificando uma por vez é de 0,0019 segundos (510 por segundo).

O tempo total de operação (to) pode ser definido como to = P1 + P2 + P3 + P4. O impacto da alteração do classificador seria o aumento máximo de 0.228 segundos no passo p4 da operação. Segundo o relatório a média de tempo para o ciclo de operação é de 14,25 segundos com desvio padrão de 7,14 segundos. O tempo adicional do classificador com BERT-Base seria somado em 0,228 segundos, aumentando a média para 14,47 segundos e elevando em 1,54% no tempo de classificação. O tempo final necessário está dentro da janela de operação e é possível implementar essa alteração na operação do sistema.

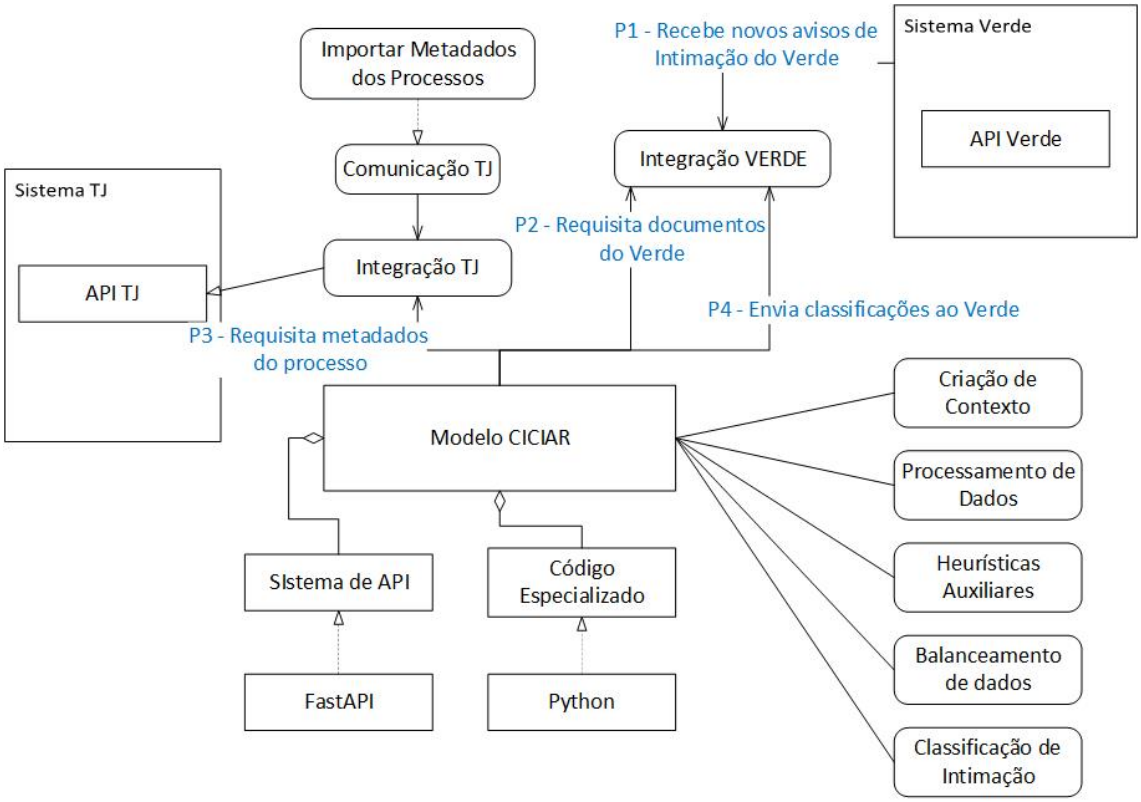

Figura 4.21: Ilustração da operação diária do CICIAR. Fonte: Adaptado de (PARREIRAS *et al.*, 2022)

Classificador	*Batch* 16 em segundos	unitário em segundos
Classificador sem BERT-Base	0,012	0,0019 (510/s)
Classificador com BERT-Base	0,15	0,228 (4,37/s)

Tabela 4.8: Tempo necessário para classificação. Fonte: De autoria própria.

Classificador	Tempo de classificação unitária	Tempo total
Classificador pré-processado	0,0019s	14,25s
Classificador BERT-Base	0,228s	14,47s

Tabela 4.9: Tempo necessário para operações. Fonte: De autoria própria.

Capítulo 5

Conclusão

O CICIAR é um sistema que está sendo implantado em um número crescente de órgãos da Defensoria Pública do Estado do Rio de Janeiro. Operando dentro do Verde, o objetivo é melhorar processos e auxiliar as atribuições dos Defensores Públicos. Toda melhoria desenvolvida que pode ser incorporada ao sistema melhora diretamente os usuários do CICIAR e consequentemente podem impactar positivamente o atendimento ao público.

A arquitetura de Transformadores traz implementações de modelos de linguagem que têm potencial para superar soluções anteriores e ao mesmo tempo conservar a capacidade de ser flexível em dimensões para atender diversos cenários.

Para o Português Brasileiro temos um modelo BERT-Base e um BERT-Large, porém comparado com o língua inglesa existe espaço para criar modelos com diferentes dimensões, configurações ou focados tarefas específicas.

Dos experimentos desenvolvidos foi possível obter incremento na capacidade de classificação utilizando o método de refino parcial do modelo ao longo do treinamento utilizando computadores de baixo custo.

A aplicação é viável no sistema e não impactará severamente os ciclos de operação.

5.1 Ferramentas de classificação e aplicações

O BERT e outras ferramentas baseadas em Transformadores mostram capacidade de transferência de conhecimento ao executar o pré-treino e posterior aplicação em outras tarefas com diferentes conjuntos de dados. A arquitetura de Transformadores deu um salto de desempenho ao superar ferramentas de modelo de linguagem como redes LSTM e redes RNN com melhorias como a redução da sensibilidade a distância das dependências entre *tokens*. A forma como são calculados os pesos ao longo das camadas possibilita grande potencial de paralelismos. No entanto são modelos

grandes e exigem grande quantidade de dados para treinar corretamente e grande poder computacional.

No idioma português brasileiro temos modelos disponíveis de BERT-Base e BERT-Large. Para o idioma Inglês há variantes como o RoBERTa com foco em superar o desempenho em tarefas genéricas se comparado a outros modelos, BERT-Tiny treinado de forma semelhante ao BERT-Base e Large mas com redução nas dimensões das camadas e parâmetros, além de outros gerados via técnica de destilação de conhecimento como o TinyBERT. As variantes menores tem mais chances de ser refinadas completamente ou até pré-treinadas do zero em hardwares mais modestos.

A exemplo do CICIAR temos possibilidade de aplicar cada vez mais em aplicações sejam elas do segmento público ou privado e há uma lacuna na variedade de modelos para aplicações em Português Brasileiro.

5.2 Ferramentas computacionais

Dentro do contexto de dificuldade de acesso a computadores de alto desempenho para desenvolver as atividades de criação e refinamento de modelos BERT, foram pesquisadas alternativas para melhoria das ferramentas de classificação com meios disponíveis como placas de vídeo mais acessíveis de 8GiB ou 12GiB de memória de vídeo por exemplo. Nessa faixa de recursos podemos treinar os classificadores em conjunto com a liberação de camadas do modelo BERT para o resultado do treinamento.

Os serviços oferecidos pelo Google Colaboratory (Colab) têm bom custo benefício comparado a outras alternativas de serviços de hardware ou plataforma, porém são limitados quanto ao uso como tempo limite de execução e desconexão ao detectar interface ociosa. Essas características desfavorecem as atividades de treinamento de grandes conjuntos de dados e de múltiplas experimentações.

5.3 Resultado do treinamento

A configuração proposta aumenta o número de parâmetros do modelo, o risco de sobreajuste aumenta, demanda mais épocas de treino e mais amostras e ajustes de hiperparâmetros. Nas referências foi verificado que para cada aplicação ou tarefa pode exigir uma taxa de aprendizado diferente por elementos como complexidade dos dados, da quantidade de dados e quantidade de parâmetros treinando. Essas variações demonstradas nas tabelas de resultados impactaram muito o resultado da capacidade de classificação.

No caso dos experimentos com congelamento em 10 camadas (BERT-Base duas camadas) os resultados de acurácia mudam de 76,6% para cerca de 82%. A configuração que se destacou mais foi com 11 camadas congeladas (BERT-Base uma camada) com taxa de aprendizagem de 1.00×10^{-5}. Os modelos com duas camadas retreinando tiveram mais dificuldade de treinar comparado as configurações semelhantes com uma camada.

O modelo de referência, com as frases pré-processadas pode ser superado na medida F1 por algum modelo com retreino de camadas. O rótulo de "Prisão" obteve aumento expressivo comparado a outros rótulos, mas com a pequena quantidade de amostras tem dificuldade para atingir níveis mínimos de acurácia e poder ser implantado em produção.

Os resultados estão alinhados com as conclusões de LEE *et al.* (2019) que observou melhorias com as camadas sendo liberadas para treinar junto aos classificadores. LIU *et al.* (2021), apesar de focar em aumento de desempenho, se fundamentou no congelamento de camadas para reduzir a carga de processamento mantendo a qualidade do treinamento. A melhoria decai nos experimentos feitos com a liberação da segunda camada em diante para o conjunto de dados utilizados.

De acordo com os experimentos propostos por LEE *et al.* (2019) para atingir 90% da melhoria com um quarto das camadas retreinando em relação ao refino de todas as camadas. Apesar desse resultado, a maioria dos gráficos dos experimentos mostram incremento na qualidade da classificação com uma camada refinando com o treino do classificador. Futuramente com mais dados o refinamento pode ser melhor aproveitado com mais camadas.

Foi verificada relação entre a quantidade de elementos com o desempenho de classificação no caso do rótulo "Prisão" que têm 1,8% das amostras totais. Entretanto, para os outros rótulos com menos elementos em ordem crescente após "Prisão", "Marcam Data" e "Decisão Interlocutória" e "Outro", com 5,5%, 6% e 10,7% respectivamente, obtiveram as melhores pontuações F1 no experimento com segunda melhor acurácia, com duas camadas refinando e Taxa de Aprendizado de 1.00×10^{-5}.

5.4 Aplicação no CICIAR

O modelo com representações vetoriais das frases, utilizado atualmente consome menos recursos computacionais, tanto de tempo de processamento quanto ocupação de memória, que o modelo proposto BERT-Base com as camadas de classificação. Porém, para atividade de classificação que é executada no dia-a-dia, o impacto não é impeditivo, e de acordo com a análise do quanto acrescentaria no tempo de operação, cerca de 1,54%, um aumento de menos de um segundo para a classificação.

5.5 Contribuições

As contribuições deste trabalho incluem:

- Avaliação de desempenho do refino parcial do modelo BERT pré-treinado;
- Oportunidade de melhoria do CICIAR e da rotina dos Defensores Públicos;
- Aprimorar o desempenho do classificador do CICIAR, com dados extraídos de processos reais;
- Aplicação do refino do modelo em computador com baixo poder de processamento.

Em relação à publicações, o autor participou da publicação do Artigos Completos "Inteligência artificial aplicada para o aumento da produtividade no atendimento de intimações", publicado em 31/07/2022.

5.6 Trabalhos futuros

A melhoria dos experimentos pode ser feita em diferentes camadas. Como experimento, pode-se investigar a viabilidade de pré-treino de modelo BERT reduzido, com menos de 12 camadas e menos de 768 de dimensão de saída, como o BERTTiny, em placas de vídeo com até 12GiB de memória e pode-se fazer uma comparação com o uso de modelo pré-treinado com retreino de camada.

Gerar modelo BERT com dimensões reduzidas capaz de ser treinado e refinado em computadores de baixo desempenho, além de refinar para utilização em aplicações específicas como o CICIAR.

Referências Bibliográficas

ALAMMAR, J. "The illustrated word2vec". 2017. Disponível em: <https://jalammar.github.io/illustrated-word2vec/>.

CHUNG, J., GULCEHRE, C., CHO, K., et al. "Gated Feedback Recurrent Neural Networks". In: Bach, F., Blei, D. (Eds.), *Proceedings of the 32nd International Conference on Machine Learning*, v. 37, *Proceedings of Machine Learning Research*, pp. 2067–2075, Lille, France, 07–09 Jul 2015. PMLR. Disponível em: <https://proceedings.mlr.press/v37/chung15.html>.

OLAH, C. "Understanding LSTM networks". Aug 2018. Disponível em: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

ALAMMAR, J. "The illustrated transformer". 2018a. Disponível em: <https://jalammar.github.io/illustrated-transformer/>.

VASWANI, A., SHAZEER, N., PARMAR, N., et al. "Attention is all you need", *Advances in neural information processing systems*, v. 30, 2017.

ADALOGLOU, N. "Why multi-head self attention works: Math, intuitions and 10+1 hidden insights". Mar 2021. Disponível em: <https://theaisummer.com/self-attention/>.

DEVLIN, J., CHANG, M.-W., LEE, K., et al. "Bert: Pre-training of deep bidirectional transformers for language understanding", *arXiv preprint arXiv:1810.04805*, 2018.

SALAZAR, J., LIANG, D., NGUYEN, T. Q., et al. "Masked language model scoring", *arXiv preprint arXiv:1910.14659*, 2019.

SUN, Y., ZHENG, Y., HAO, C., et al. "NSP-BERT: A Prompt-based Zero-Shot Learner Through an Original Pre-training Task–Next Sentence Prediction", *arXiv preprint arXiv:2109.03564*, 2021.

ALAMMAR, J. "The illustrated Bert, Elmo, and Co. (how NLP cracked transfer learning)". 2018b. Disponível em: <https://jalammar.github.io/illustrated-bert/>.

JIAO, X., YIN, Y., SHANG, L., et al. “Tinybert: Distilling bert for natural language understanding”, *arXiv preprint arXiv:1909.10351*, 2019.

CICIAR, S. D. A. “Sistema de Avaliação CICIAR”. 2020.

CICIAR, R. “Relatório de Janela de execução da classificação - Onda 4”. 2022.

PARREIRAS, M., VASCONCELLOS, A., MANGELI, E., et al. “Inteligência artificial aplicada para o aumento da produtividade no atendimento de intimações”. In: *Anais do X Workshop de Computação Aplicada em Governo Eletrônico*, pp. 180–191. SBC, 2022.

CICIAR, R. “Relatório - Dimensionamento de Hardware para Pré-treinar e Treinar modelos BERT”. 2021a.

CICIAR, R. “Relatório Desempenho do Modelo CICIA”. 2021b.

KOWSARI, K., JAFARI MEIMANDI, K., HEIDARYSAFA, M., et al. “Text classification algorithms: A survey”, *Information*, v. 10, n. 4, pp. 150, 2019.

MASTELLA, J. O. *Uma metodologia usando ambientes paralelos para otimização da classificação de textos aplicada a documentos jurídicos*. Tese de Mestrado, Pontifícia Universidade Católica do Rio Grande do Sul, 2020.

PEDRUZZI, P. “REFLEXÕES SOBRE O IMPACTO DA INTELIGÊNCIA ARTIFICIAL NA PRÁTICA JURÍDICA”, *Anais dos Congressos Estaduais de Magistrados-RS*, v. 1, n. 1, 2020.

“google-research/bert - BERT”. `https://github.com/google-research/bert`, 2022. Acesso em: Acessado em: 02/04/2022.

LEE, J., TANG, R., LIN, J. “What would elsa do? freezing layers during transformer fine-tuning”, *arXiv preprint arXiv:1911.03090*, 2019.

RADFORD, A., NARASIMHAN, K., SALIMANS, T., et al. “Improving language understanding with unsupervised learning”, , 2018.

MOLLOY, D. “The great graphics card shortage of 2020 (and 2021)”. `https://www.bbc.com/news/technology-55755820`, 2021. Acesso em: 02-abr-2022.

GOLDBERG, Y. “A primer on neural network models for natural language processing”, *Journal of Artificial Intelligence Research*, v. 57, pp. 345–420, 2016.

ALMEIDA, F., XEXÉO, G. "Word embeddings: A survey", *arXiv preprint arXiv:1901.09069*, 2019.

BENGIO, Y. "Neural net language models", *Scholarpedia*, v. 3, n. 1, pp. 3881, 2008.

BENGIO, Y., DUCHARME, R., VINCENT, P. "A neural probabilistic language model", *Advances in neural information processing systems*, v. 13, 2000.

CHUNG, J., GULCEHRE, C., CHO, K., et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*, 2014.

BENGIO, Y., SIMARD, P., FRASCONI, P. "Learning long-term dependencies with gradient descent is difficult", *IEEE transactions on neural networks*, v. 5, n. 2, pp. 157–166, 1994.

MIKOLOV, T., KARAFIÁT, M., BURGET, L., et al. "Recurrent neural network based language model." In: *Interspeech*, v. 2, pp. 1045–1048. Makuhari, 2010.

HOCHREITER, S., SCHMIDHUBER, J. "Long short-term memory", *Neural computation*, v. 9, n. 8, pp. 1735–1780, 1997.

GERS, F. A., SCHMIDHUBER, J., CUMMINS, F. "Learning to forget: Continual prediction with LSTM", *Neural computation*, v. 12, n. 10, pp. 2451–2471, 2000.

GERS, F. A., SCHRAUDOLPH, N. N., SCHMIDHUBER, J. "Learning precise timing with LSTM recurrent networks", *Journal of machine learning research*, v. 3, n. Aug, pp. 115–143, 2002.

BOTTOU, L. "Stochastic gradient descent tricks". In: *Neural networks: Tricks of the trade*, Springer, pp. 421–436, , 2012.

RUMELHART, D. E., HINTON, G. E., WILLIAMS, R. J. *Learning internal representations by error propagation.* Relatório técnico, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

SINGH, S., MAHMOOD, A. "The NLP cookbook: Modern recipes for transformer based deep learning architectures", *IEEE Access*, v. 9, pp. 68675–68702, 2021.

SHIV, V., QUIRK, C. "Novel positional encodings to enable tree-based transformers". In: Wallach, H., Larochelle, H., Beygelzimer, A., et al. (Eds.),

Advances in Neural Information Processing Systems, v. 32. Curran Associates, Inc., 2019. Disponível em: <https://proceedings.neurips.cc/paper/2019/file/6e0917469214d8fbd8c517dcdc6b8dcf-Paper.pdf>.

YANG, Z., DAI, Z., YANG, Y., et al. "Xlnet: Generalized autoregressive pretraining for language understanding", *Advances in neural information processing systems*, v. 32, 2019.

TAYLOR, W. L. ""Cloze procedure": A new tool for measuring readability", *Journalism quarterly*, v. 30, n. 4, pp. 415–433, 1953.

ZHU, Y., KIROS, R., ZEMEL, R., et al. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books". In: *Proceedings of the IEEE international conference on computer vision*, pp. 19–27, 2015.

LIU, Y., OTT, M., GOYAL, N., et al. "Roberta: A robustly optimized bert pretraining approach", *arXiv preprint arXiv:1907.11692*, 2019.

LAN, Z., CHEN, M., GOODMAN, S., et al. "Albert: A lite bert for self-supervised learning of language representations", *arXiv preprint arXiv:1909.11942*, 2019.

LEE, J., YOON, W., KIM, S., et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining", *Bioinformatics*, v. 36, n. 4, pp. 1234–1240, 2020.

HINTON, G., VINYALS, O., DEAN, J. "Distilling the knowledge in a neural network (2015)", *arXiv preprint arXiv:1503.02531*, v. 2, 2015.

TURC, I., CHANG, M.-W., LEE, K., et al. "Well-read students learn better: The impact of student initialization on knowledge distillation", *arXiv preprint arXiv:1908.08962*, v. 13, 2019.

SOUZA, F., NOGUEIRA, R., LOTUFO, R. "BERTimbau: pretrained BERT models for Brazilian Portuguese". In: *9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear)*, 2020.

POLO, F. M., MENDONÇA, G. C. F., PARREIRA, K. C. J., et al. "LegalNLP–Natural Language Processing methods for the Brazilian Legal Language", *arXiv preprint arXiv:2110.15709*, 2021.

WARSTADT, A., SINGH, A., BOWMAN, S. R. "Neural network acceptability judgments", *Transactions of the Association for Computational Linguistics*, v. 7, pp. 625–641, 2019.

DOLAN, B., BROCKETT, C. "Automatically constructing a corpus of sentential paraphrases". In: *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

CER, D., DIAB, M., AGIRRE, E., et al. "Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation", *arXiv preprint arXiv:1708.00055*, 2017.

IYER, S., DANDEKAR, N., CSERNAI, K., et al. "First quora dataset release: Question pairs. data. quora. com", 2017.

SOCHER, R., PERELYGIN, A., WU, J., et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

BENTIVOGLI, L., CLARK, P., DAGAN, I., et al. "The Fifth PASCAL Recognizing Textual Entailment Challenge." In: *TAC*, 2009.

WILLIAMS, A., NANGIA, N., BOWMAN, S. R. "A broad-coverage challenge corpus for sentence understanding through inference", *arXiv preprint arXiv:1704.05426*, 2017.

ALISHAHI, A., CHRUPAŁA, G., LINZEN, T. "Analyzing and interpreting neural networks for NLP: A report on the first BlackboxNLP workshop", *Natural Language Engineering*, v. 25, n. 4, pp. 543–557, 2019.

ZAMPIERI, M., GEBRE, B. G. "Automatic identification of language varieties: The case of Portuguese". In: *KONVENS2012-The 11th Conference on Natural Language Processing*, pp. 233–237. Österreichischen Gesellschaft für Artificial Intelligende (ÖGAI), 2012.

TOLEDO, C. M., CUNHA, A., SCARTON, C., et al. "Automatic classification of written descriptions by healthy adults: an overview of the application of natural language processing and machine learning techniques to clinical discourse analysis", *Dementia & Neuropsychologia*, v. 8, pp. 227–235, 2014.

BRUM, H. B., NUNES, M. D. G. V. "Building a sentiment corpus of tweets in brazilian portuguese", *arXiv preprint arXiv:1712.08917*, 2017.

HARTMANN, N., FONSECA, E., SHULBY, C., et al. "Portuguese word embeddings: Evaluating on word analogies and natural language tasks", *arXiv preprint arXiv:1708.06025*, 2017.

SANTOS, W., FUNABASHI, A., PARABONI, I. "Searching Brazilian Twitter for signs of mental health issues". In: *Proceedings of The 12th Language Resources and Evaluation Conference*, pp. 6111–6117, 2020.

CABRAL, L., MONTEIRO, J. M., DA SILVA, J. W. F., et al. "Fakewhastapp. br: NLP and machine learning techniques for misinformation detection in brazilian portuguese whatsapp messages". In: *Proceedings of the 23rd International Conference on Enterprise Information Systems, ICEIS*, pp. 26–28, 2021.

LEME, B. *Classificação automática de documentos de características econômicas para defesa jurídica.* Tese de Doutorado, Universidade de São Paulo, 2021.

NGUYEN, M.-T., NGUYEN, V.-H., YUN, S.-J., et al. "Recurrent neural network for partial discharge diagnosis in gas-insulated switchgear", *Energies*, v. 11, n. 5, pp. 1202, 2018.

DA SILVA, N. C., BRAZ, F., DE CAMPOS, T., et al. "Document type classification for Brazil's supreme court using a Convolutional Neural Network". In: *10th International Conference on Forensic Computer Science and Cyber Law (ICoFCS), Sao Paulo, Brazil*, pp. 29–30, 2018.

SULEA, O.-M., ZAMPIERI, M., MALMASI, S., et al. "Exploring the use of text classification in the legal domain", *arXiv preprint arXiv:1710.09306*, 2017.

VAN ROSSUM, G., DRAKE JR, F. L. "The python language reference", *Python software foundation*, 2014.

LIU, Y., AGARWAL, S., VENKATARAMAN, S. "AutoFreeze: Automatically Freezing Model Blocks to Accelerate Fine-tuning", *arXiv preprint arXiv:2102.01386*, 2021.