UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

MARCOS YURI ROSA JUNIOR

COMPARANDO BERT, ROBERTA E DISTILBERT PARA ANÁLISE DE SENTIMENTOS EM TEXTO

MEDIANEIRA

2O21

MARCOS YURI ROSA JUNIOR

COMPARANDO BERT, ROBERTA E DISTILBERT PARA ANÁLISE DE SENTIMENTOS EM TEXTO

COMPARING BERT, ROBERTA AND DISTILBERT FOR SENTIMENT ANALYSIS IN TEXT

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do título de Bacharel em Ciências da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).
Orientador(a): Prof. Arnaldo Candido Junior
Coorientador(a): Prof. Pedro Luiz de Paula Filho

MEDIANEIRA

2021

MARCOS YURI ROSA JUNIOR

COMPARANDO BERT, ROBERTA E DISTILBERT PARA ANÁLISE DE SENTIMENTOS EM TEXTO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciências da Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de aprovação: 10/dezembro/2021

Arnaldo Candido Junior
Doutor
Universidade Tecnológica Federal do Paraná

Pedro Luiz de Paula Filho
Doutor
Universidade Tecnológica Federal do Paraná

Jorge Aikes Junior
Mestre
Universidade Tecnológica Federal do Paraná

Ricardo Sobjak
Doutor
Universidade Tecnológica Federal do Paraná

MEDIANEIRA

2021

RESUMO

O desejo de melhorar a comunicação entre o homem e a máquina proporcionou vários avanços na área de Processamento de Linguagem Natural (PLN), uma das grandes sub-áreas da Inteligência Artificial (IA). Ao passar dos anos a criação e o aprimoramento das técnicas de PLN alteraram o conceito das melhores práticas para construção dos modelos de Redes Neurais Artificiais (RNAs) voltadas para esta área, surgindo a possibilidade de utilizar as RNAS para tarefas de PLN como a análise de sentimentos. Em 2018 um novo modelo de arquitetura chamado Bidirectional Encoder Representation from Transformer (BERT) alcançou um novo estado da arte, obtendo ótimos resultados em diferentes tipos de tarefas. Alguns novos modelos baseados em BERT surgiram com uma proposta de obter melhores resultados como o RoBERTa, e outros em abaixar o custo de treinamento e tentar manter a performance, como o DistilBERT. Neste Trabalho é comparado os modelos BERT, RoBERTa e DistilBERT com os seus parâmetros refinados para a tarefa de análise de sentimentos binária. Os resultados obtidos neste trabalho, indicam a superioridade dos modelos BERT e RoBERTa no quesito de acurácia, porém o DistilBERT necessita de cerca da metade do tempo necessário para finalizar o fine-tuning. A partir destes resultados, a escolha do melhor modelo para a tarefa de análise de sentimentos binária irá depender do cenário que será aplicado.

Palavras-chave: Redes Neurais, Processamento de Linguagem Natural, Inteligência Artificial.

ABSTRACT

The desire to improve communication between man and machine has provided several advances in the area of Natural Language Processing (PLN), one of the great sub-areas of Artificial Intelligence (AI). Over the years, the creation and improvement of PLN techniques have changed the concept of best practices for building models of Artificial Neural Networks (ANNs) aimed at this area, giving rise to the possibility of using RNAS for PLN tasks such as the analysis of feelings. In 2018, a new architectural model called Bidirectional Encoder Representationfrom Transformer (BERT) reached a new state of the art, obtaining excellent results in different types of tasks. Some new models based on BERT have emerged with a proposal to obtain better results such as RoBERTa, and others to lower the training cost and try to maintain performance, such as DistilBERT. In this work, the BERT, RoBERTa and DistilBERT models are compared with their refined parameters for the binary sentiment analysis task. The results obtained in this work indicate the superiority of the BERT and RoBERTa models in terms of accuracy, but the DistilBERT requires about half the time needed to complete the fine-tuning. From these results, the choice of the best model for a binary sentiment analysis task will depend on the scenario to be applied.

Keywords: Neural Networks, Natural Language Processing, Artificial Intelligence.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

O Processamento da Língua Natural (PLN) é uma das áreas de aplicação da Ciência Computação e da linguística que estuda a capacidade e as limitações da máquina em compreender e manipular a linguagem natural de texto ou fala (CHOWDHURY, 2003). A área é abrangente e alguns exemplos de aplicações incluem: mecanismo de buscas, assistentes pessoais virtuais, análise de sentimento, e *chatbots* (CHOWDHURY, 2003). É comum o uso de técnicas de Inteligência Artificial (IA) no PLN.

Uma das aplicações que ganhou destaque é a análise de sentimentos em texto, pois com ela é possível extrair de palavras, textos ou sentenças, varias informações como emoções, sentimentos e opiniões (LIU *et al.*, 2010). E com o grande volume de dados, fica cada vez mais necessário utilizar recursos de inteligencia artificial para conseguir extrair todas estas informações. Assim é possível a utilizar estar informações extraídas para vários casos de uso, como por exemplo, determinar se um comprador está satisfeito com produto a partir de seu comentário sobre o mesmo.

No ano de 2018, foi introduzida uma arquitetura de IA chamada Bidirectional Encoder Representations from Transformers (BERT), que alcançou um novo estado da arte na capacidade de processamento de linguagem natural (DEVLIN *et al.*, 2018). O modelo é baseado em Aprendizado Profundo (Deep Learning) e pode ser pensado como um passo na construção de algoritmos de IAs mais fortes, já que consegue realizar bem muitas tarefas distintas de PLN, como reconhecimento de entidades nomeadas, resposta de perguntas (KANEKO; KOMACHI, 2019). Inspirados no BERT, surgiram algumas variantes como o RoBERTa, que obteve resultados superiores aumentando o número de parâmetros e da base de dados. Outra variante é o DistilBERT, que é o resultado da destilação do BERT, para gerar vários benefícios como a redução do tempo de treinamento.

1.1 Objetivos Gerais e Específicos

O objetivo deste trabalho é realizar o fine-tuning dos modelos BERT, RoBERTa e DistilBERT para análise de sentimentos. Os objetivos específicos são:

- Selecionar um ou mais corpus (base de textos), para realizar o *fine-tuning* do modelo;
- Pré-processar os textos para adequá-los ao *fine-tuning*;
- Realizar o *fine-tuning* do BERT, RoBERTa e DistilBERT para análise de sentimentos;
- Avaliar a acurácia dos modelos para análise de sentimentos;
- Avaliar o tempo necessário para realizar o *fine-tuning* de cada modelo;
- Comparar os resultados.

1.2 Justificativa

O aprendizado de máquina tem revolucionado várias áreas de pesquisa, o que inclui o PLN (LECUN *et al.*, 2015) e todas as suas sub-áreas. A análise de sentimentos é um tópico que necessita de atenção, pois com ela é possível resolver diversos problemas do mundo real, e ao aumentar a sua acurácia, ou até mesmo ficar mais acessível a vários canais de comunicação, podem impactar positivamente na sociedade, como por exemplo identificar depressão ao analisar comentários com sentimentos negativos nas redes sociais.

E Diversos modelos e sistemas poderosos têm sido criados, tais como GPT (RADFORD *et al.*, 2018), BERT (DEVLIN *et al.*, 2018), Transformer (JADERBERG *et al.*, 2015), Watson (JONES, 2014), entre outros.

Com novos modelos surgindo recorrentemente, surgem dificuldades na escolha do modelo correto para a tarefa de análise de sentimentos binária, possibilitando a escolha de um modelo que não se adéqua bem a tarefa, ou que irá gerar desperdício por ser um modelo mais robusto que não apresenta ganhos de performance ao compara-lo à modelos menores.

A proposta deste trabalho consiste em realizar o *fine-tuning* dos modelos BERT, RoBERTa e DistilBERT para uma tarefa específica, que é a análise de sentimentos binária, e comparar os resultados. Com as comparações será possível entender as vantagens e desvantagem dos modelos apresentados para a tarefa de análise de sentimentos binária.

2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos sobre a elaboração de Redes Neurais Artificiais, Processamento de linguagem Natural e Análise de sentimentos. O foco é apresentar toda base teórica que é utilizada na construção dos modelo BERT, RoBERTa e DistilBERT.

2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são motivadas pelo entendimento que o cérebro humano processa as informações de uma forma distinta da arquitetura computacional tradicional (HAYKIN, 2007). O cérebro humano trabalha com um conjunto de componentes chamados neurônios, que quando conectados formam ligações denominadas sinapses. Esses neurônios são responsáveis por muitas tarefas como a visão. A visão faz uma representação do ambiente, como também, fornece toda informação necessária para interagir com o espaço em questão (CHURCHLAND; SEJNOWSKI, 2016). Outro fator importante, é que o cérebro realiza essas funções constantemente com facilidade, enquanto o sistema computacional tem uma grande dificuldade para resolver problemas dessa natureza.

De modo geral, a rede neural é projetada para se assimilar à forma em que o cérebro humano realiza determinadas tarefas ou funções em áreas de interesse (HAYKIN, 2007). As redes neurais são treinadas por meio de um algoritmo de aprendizagem, que tem como objetivo ajustar os pesos sinápticos da rede de forma organizada para uma certa finalidade. A modificação dos pesos sinápticos é a forma clássica utilizada para realizar o aprendizado, porém também é possível uma rede modificar sua topologia, alterando a forma em que são usados os seus neurônios.

De acordo com Haykin (2007), as Redes Neurais Artificiais dispõem de propriedades como:

1. Não-linearidade: um Neurônio artificial pode ser ou não linear;

2. Mapeamento de Entrada-Saída: o processo iterativo dos ajustes dos pesos podem ser aplicados de várias formas, eles podem ser classificados como (DOHME, 2003):
   (a) Aprendizado Supervisionado: quando é utilizado um agente externo que auxilia a rede indicando a saída desejada para determinada entrada;
   (b) Aprendizado Não Supervisionado (auto-regulação): quando ocorre a ausência de um agente externo;
   (c) Reforço: quando um "juiz" externo avalia a saída gerada pela rede.
3. Adaptabilidade: capacidade de regular seus pesos sinápticos em função de alterações do seu ambiente;
4. Resposta a Evidências: gera um índice de confiança de suas decisões e análises;
5. Informação Contextual: os neurônios são afetados pela atividade de outros neurônios;
6. Tolerância a Falhas: as redes neurais são robustas contra falhas de neurônios.

As RNAs têm como base a utilização dos neurônios artificiais, sendo o neurônio artificial a unidade fundamente da RNA. Em sua forma mais pura, o modelo neural contêm três elementos básicos:

1. Conjunto de sinapses: caracterizados por um peso, que diferente do cérebro-humano, estes pesos podem assumir valores negativos;
2. Somador: responsável por somar o sinais da entrada, ponderados pelos pesos sinápticos, formando uma combinação linear;
3. Função de ativação: define o intervalo permitido da amplitude do sinal, normalmente definido em [0,1] ou [-1,1].

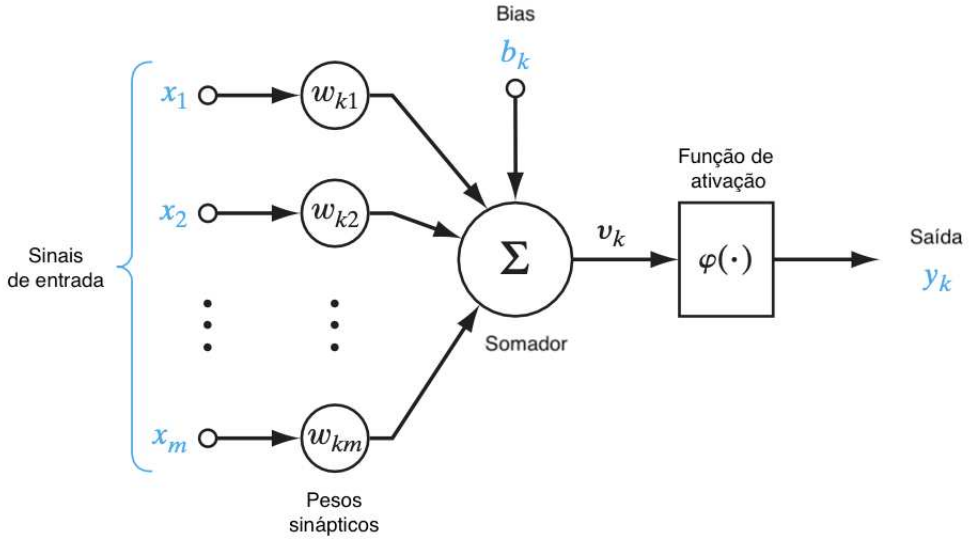
Em um modelo neural é comum a aplicação de um *bias* (ou viés), que pode alterar o sinal emitido pelo neurônio, sendo capaz de aumenta-lo ou diminui-lo. A Figura 1 faz uma representação de um modelo neural.

Os valores de entrada para o neurônio são representados pelo vetor x que contem n valores. Ao chegarem ao neurônio, os valores de entrada são multiplicados pelos seus respectivos pesos sinápticos, que são os elementos do vetor w, depois somados ao *bias* b para gerar o valor v, sendo v comumente denominado potencial de ativação. Este processo ocorre de acordo com a expressão apresentada na Equação 1.

$$v_k = \sum_{j=1}^{n} x_j w_{kj} + b \tag{1}$$

O valor v passa então por uma função matemática de ativação φ, responsável por limitar o valor a um certo intervalo, produzindo a saída y do neurônio. A Equação 2 apresenta o valor y que representa o sinal de saída do neurônio.

Figura 1 – Modelo de um Neurônio Artificial



Fonte: adaptado de Haykin (2007)

$$y_k = \varphi(v_k) \tag{2}$$

2.2 Funções de ativação

O processamento e a propagação do sinal de saída de cada neurônio se dá pela função de ativação (GOODFELLOW *et al.*, 2016). A escolha das funções de ativação de uma rede neural é muito importante, pois elas que definem como devem ser como será feito o tratamento do sinal. Algumas funções de ativação usadas são a Limiar (Degrau), Sigmoide Logística, Tangente Hiperbólica e ReLU (Rectified Linear Unit).

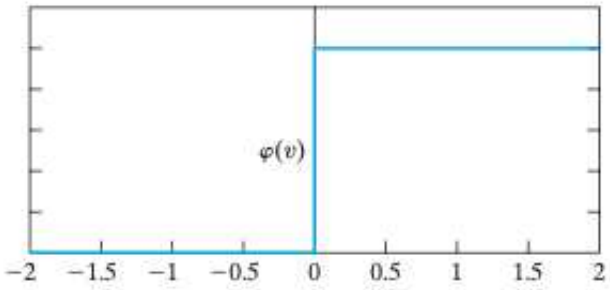
- Função de ativação Limiar

  Esta função de ativação é calculada como descrita na Equação 3.

$$y_k = \begin{cases} 1 & se \quad v_k \geq 0 \\ 0 & se \quad v_k < 0 \end{cases} \tag{3}$$

  A saída é 1 para valores de v positivos, e 0 para valores de v negativos. O comportamento da função é representada na Figura 2.

Figura 2 – Gráfico da função limiar



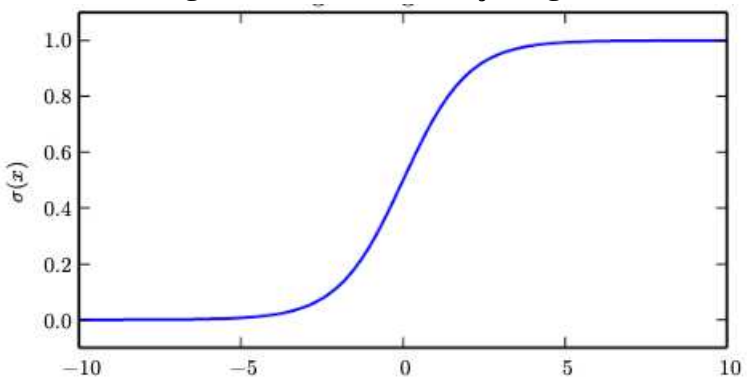
Fonte: (HAYKIN, 2007)

- Sigmoide Logística

$$\sigma(v) = \frac{1}{1+e^{-v}} \tag{4}$$

A função Sigmoide Logística é continua, ou seja, diferenciável em todos os pontos. Seu gráfico tem um formato de "s". É uma das formas mais comuns e tem um balanceamento interessante entre comportamento linear e não-linear (HAYKIN, 2007). Assume valores entre zero e um (GOODFELLOW *et al.*, 2016). Seu maior problema é a saturação quando a função aproxima dos valores de zero e um, pois os valores da derivada ficam muito perto de zero, o que compromete o aprendizado da rede em algoritmos como o de Retro-propagação. O gráfico da função Sigmoide Logística é representado na Figura 3.

Figura 3 – Gráfico da função logística



Fonte: (GOODFELLOW *et al.*, 2016)

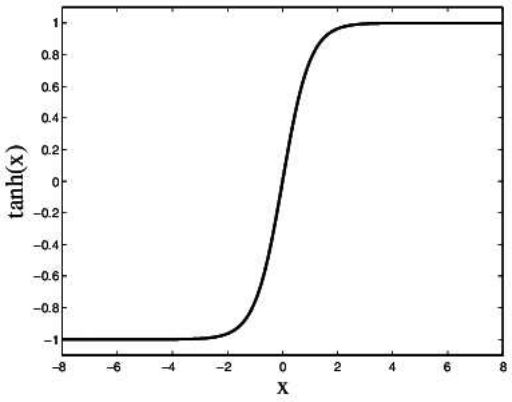
- Tangente Hiperbólica

A função de ativação da Tangente Hiperbólica é dada por:

$$tanh(v_k) = \frac{sinh(v_k)}{cosh(v_k)} \tag{5}$$

A função Tangente Hiperbólica (*tanH*) também é uma sigmoide, ou seja seu gráfico tem um formato de "s". Diferente da função sigmoide logística, a função *tanH* assume valores entre -1 e 1 (HAYKIN, 2007). O gráfico da função *tanH* é representado na Figura 4.

Figura 4 – Gráfico da função Tangente Hiperbólica



Fonte: (NAMIN *et al.*, 2009)

- ReLU

  A função ReLU é dada por:

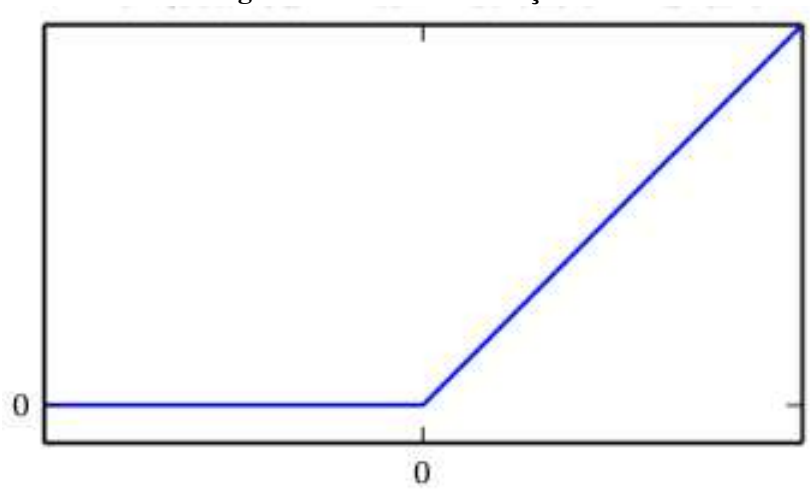
$$\varphi(v_k) = max(0, v_k) \tag{6}$$

  ReLU é uma das funções mais amplamente usadas pelo seu bom desempenho, e é recomendada para a maioria dos casos no uso em redes perceptron (GOODFELLOW *et al.*, 2016). A ReLU não é uma função linear, e apesar do seu desempenho, o aprendizado é prejudicado para valores de v devido a sua saturação nesse caso. O Gráfico de sua função é representado na Figura 5.

2.3 Retro-propagação

O Algoritmo de Retro-propagação, ou como comumente chamado *Backpropagation* do inglês, é o conceito de um sistema em que o valor de saída de um neurônio influencia em parte na entrada aplicada àquele nerônio (HAYKIN, 2007). A ideia consiste em aplicar a descida do gradiente, que origina o processo do Gradiente Descendente, que por sua vez, tenta minimizar os erros gerados pela rede comparando a saída desejada com a saída obtida. Portanto

Figura 5 – Gráfico da função ReLU



Fonte: (GOODFELLOW *et al.*, 2016)

este processo é um algorítimo de aprendizado supervisionado (HIROSE *et al.*, 1991).

O treinamento de uma Rede Neural Artificial utilizando o algorítimo de Retro-propagação pode ser divido em duas etapas. A primeira etapa é a fase *feedfoward*, que consiste no processo de inicialização dos neurônios da camada de entrada, e propagação do sinal de entrada para as camadas escondidas. A segunda etapa é a fase *backward*, na qual a saída obtida é comparada com a saída desejada, caso ocorra alguma divergência o erro será calculado e os pesos serão ajustados (HIROSE *et al.*, 1991). O ajuste de erro é propagado a partir da camada de saída, e percorre as camadas escondidas, até a camada de entrada. Nesse processo os pesos e os *biases* são ajustados conforme o erro.

Para fazer os ajustes dos pesos em rede completamente conectadas do tipo *feedforward*, são usados uma taxa de aprendizagem (η), o valor da entrada (x) e o valor do erro (δ). A Equação 7 mostra como é feito o processo de ajuste dos pesos w.

$$w_{ij} = w_{ij} + \eta \delta_j x_i \tag{7}$$

O ajuste do erro se diferencia na última camada e nas camadas ocultas. Na ultima camada, os valores de δ são calculados com o valor de f' que é a derivada da função de ativação aplicada no potencial de ativação v, e o valor de g' que corresponde a derivada das função de custo aplicada na saída desejada e saída obtida, conforme a Equação 8.

$$\delta_i = f^{'}(v_i) g^{'}(d_i, y_i) \tag{8}$$

Nas camadas ocultas os valores de δ são calculados a partir do valor de f', a derivada da função de ativação aplicada sobre potencial de ativação v, e o valor somatório dos erros da

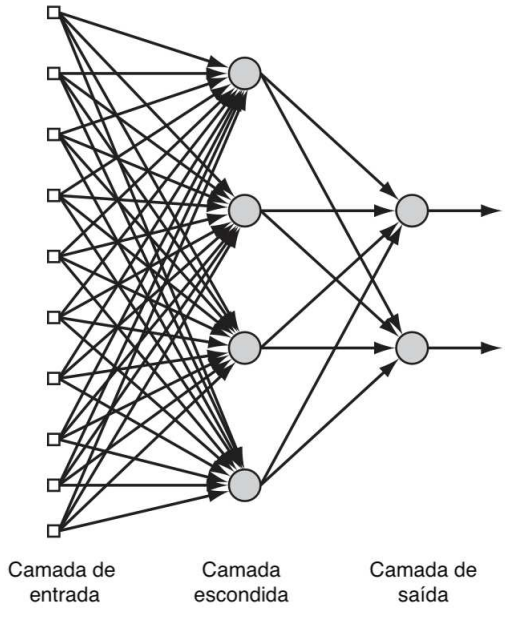
camada seguinte ponderado pelos respectivos pesos sinápticos que conectando o neurônio atual aos neurônios da camada seguinte. A formulação é descrita na Equação 9.

$$\delta_i = f^{'}(v_i) \sum w_{ij} \delta_j \tag{9}$$

2.4 MultiLayer Perceptron FeedFoward

A rede MultiLayer Perceptron (MLP) é uma evolução da rede Perceptron, e se distingue principalmente pela presença de camadas ocultas. Os papéis das camadas ocultas (ou intermediárias) podem ser diversos, mas com o aumento das camadas, a rede se torna capaz de extrair informações de ordem elevada (HAYKIN, 2007). A capacidade dos neurônios das camadas ocultas em extrair informações de ordem elevada é muito importante em casos que a camada de entrada é extensa.

Figura 6 – Gráfico da rede MLP



Fonte: adaptado de Haykin (2007)

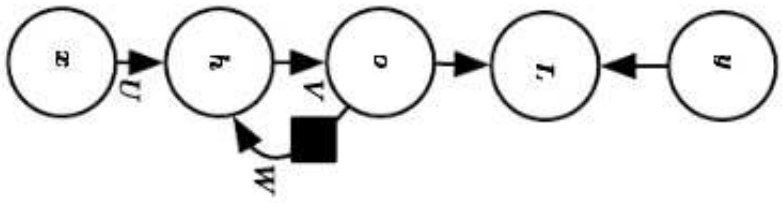
A MLP *FeedFoward* é uma versão simples da MLP. O sinal parte da camada de entrada, e consequentemente é propagado até as camadas ocultas, sendo que o sinal passa em cada uma das camadas ocultas na ordem de sua topologia até chegar na camada de saída. Ela

é considerada totalmente conectada quando todos os neurônios entre camadas vizinhas são conectados, e parcialmente conectada se houver a ausência de alguma conexão entre camadas vizinhas (HAYKIN, 2007). A Figura 6 representa o esboço de uma rede MLP.

2.5 Redes Recorrentes

A grande diferença da rede MLP Recorrente para uma rede *FeedFoward* é a presença de uma ou mais conexões de realimentação (HAYKIN, 2007), o que resulta em um grande impacto na capacidade de aprendizado da rede, e também no seu desempenho. Essa informação sequencial é preservada no estado oculto da rede recorrente, que consegue passar por muitas etapas de tempo à medida que ela avança para processar novas instâncias. O erro gerado pelos neurônios retornará por meio de retro-propagação e será usado para ajustar seus pesos até que o erro não diminua mais. A Figura 7 faz a representação de uma Rede Recorrente.

Figura 7 – Representação da Rede Recorrente



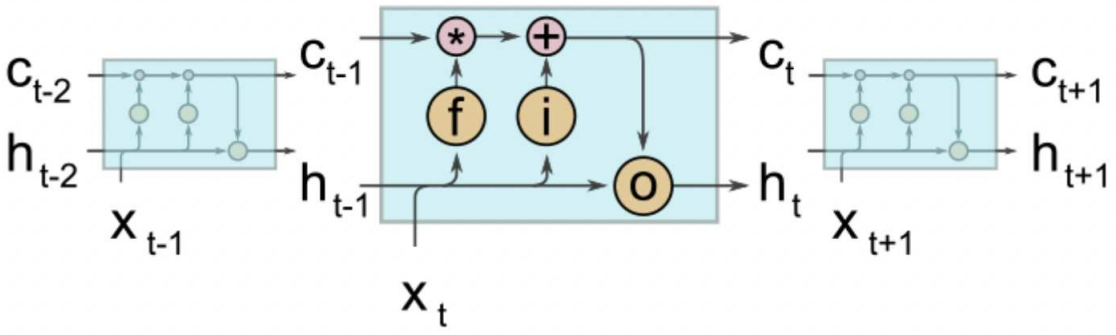
Fonte: (GOODFELLOW *et al.*, 2016)

2.6 Redes LSTM

A arquitetura *Long Short Term Memory* (LSTM) é inspirada nos circuitos digitais da eletrônica digital. Sua arquitetura consiste em um conjunto de ligações recorrentes conhecidos como blocos de memória. Cada bloco contêm uma ou mais células de memórias conectadas,

e também três unidades de multiplicação: entrada, saída e a porta de esquecimento (GRAVES, 2012). A Figura 8 é uma representação gráfica de uma rede LSTM.

Figura 8 – Representação da Rede LSTM



Fonte: adaptado de Kratzert *et al.* (2018)

Na entrada, é feito todo processamento de atribuição às células de memória. Alguns elementos necessários para calcular a saída da porta de entrada i_t são a entrada x_t, a saída do neurônio no passo anterior h_{t-1}, o valor da ativação da célula anterior c_{t-1}, a função de ativação Sigmoide Logística σ, a função de ativação tangente hiperbólica $tanh$, o *bias* b^i e matrizes de pesos w^{xi}, w^{hi} e w^{ci}, considerando t como o tempo. A formulação é descrita na Equação 10.

$$i_t = \sigma(w^{xi}x_t + w^{hi}h_{t-1} + w^{ci}c_{t-1} + b^i) \tag{10}$$

Na porta de esquecimento é feito o controle das informações que devem ser mantidas ou descartadas. A formulação é descrita na Equação 11, dados a matrizes de peso e o vetor *bias* desta porta.

$$f_t = \sigma(w^{xf}x_t + w^{hf}h_{t-1} + w^{cf}h_{t-1} + b^{nf}) \tag{11}$$

Na porta de sáida é feito o processamento de leitura e propagação para a rede recorrente. A Equação 12 apresenta a formulação para a saída do sinal, dados os respectivos pesos e a Equação 13 apresenta a saída da célula.

$$\phi_t = \sigma(w^{x\phi}x_t + w^{h\phi}h_{t-1} + w^{c\phi}h_{t-1} + b^{\phi}) \tag{12}$$

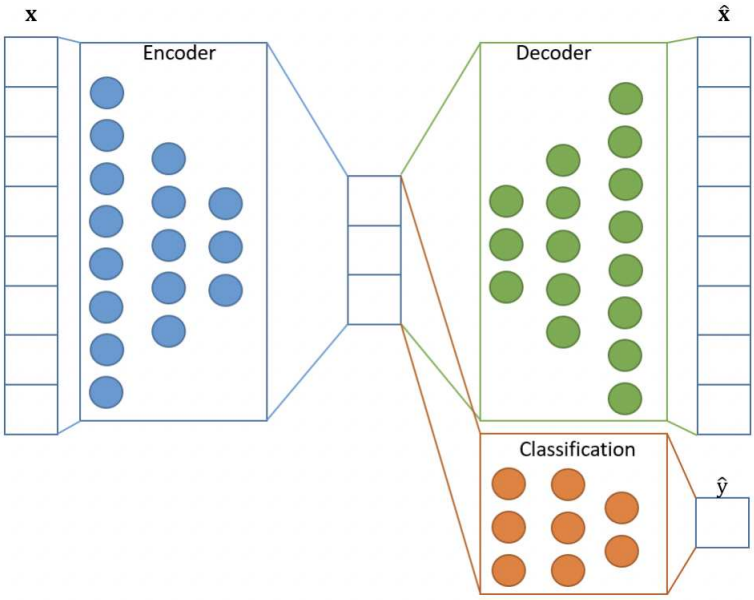
$$h_t = \phi_t \circ tanh(c_t) \tag{13}$$

2.7 Redes Auto-codificadoras

As Redes Auto-codificadoras são uma classe de rede neural não supervisionada que podem ser treinadas para aprender características de uma coleção de dados (BEZERRA, 2016). O modo que uma rede auto-codificadora é treinada, é de tentar reproduzir a sua entrada na saída.(GOODFELLOW *et al.*, 2016) As características obtidas são úteis para outros usos, como reconhecimento de objetos em imagens, e processamento de linguagem natural, devido a criação de representações mais concisas de um conjunto de dados.

Uma rede auto-codificadora contêm ao menos uma camada escondida, e as camadas de entrada e saída devem possuir o mesmo número de neurônios. Essa rede é treinada para criar o padrão mais equivalente possível da saída comparada a entrada (BEZERRA, 2016). O modelo gráfico com uma camada escondida é apresentado na Figura 9.

Figura 9 – Representação da Rede Auto-codificadora



Fonte: (BANK *et al.*, 2020)

Uma rede auto-codificadora realiza dois tipos de transformações utilizando funções. A primeira é a função de extração de características conhecida como *encoder*, que mapeia a entrada de dados em uma representação compactada. E a segunda função é a de reconstrução conhecida como *decoder*, que retorna a representação criada pelo *encoder* ao espaço original (BEZERRA, 2016).

Durante o treinamento da rede, deve se criar um par de funções h e r em que o erro de

reconstrução da entrada para saída seja minimizado. O erro é representado pela função de custo L representada na equação 14. Deve se usar funções ativação com cautela pois a saída deve ser similar a entrada.

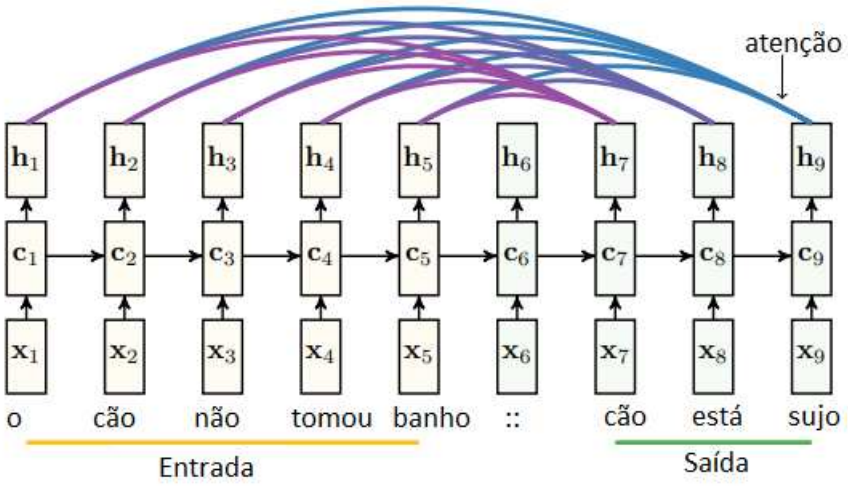
$$L(x) = \sum_{x^i \in x} \iota(x^i, r(h(x^i))) \tag{14}$$

2.8 Redes com Atenção

As Redes com Atenção demostraram um grande sucesso em várias áreas de interesse como a síntese de caligrafia (GRAVES, 2012), tradução (Bahdanaeu et al., 2015) e reconhecimento de voz (CHOROWSKI *et al.*, 2015).

As redes com atenção permitem que o modelo foque-se sobre os vetores de saída passados, reduzindo o problema da dissipação do gradiente (ROCKTÄSCHEL *et al.*, 2015). Diferente do LSTM, as redes com atenção não necessitam capturar todo o contexto semântico da entrada em seu estado celular. O processo é um pouco diferente, os vetores são gerados enquanto é feita a leitura da entrada, e as representações acumuladas no estado da células informam quais vetores de saída devem ser escolhidos com essas representações para se determinar a classe.

Figura 10 – Representação da Rede com atenção



Fonte: adaptado de Rocktäschel *et al.* (2015)

Em um contexto análise de texto, como apresentado na Figura 10, a formulação deve ser feita utilizando as Equações 15, 16 e 17 , considerando L o numero de palavras do voculário, e_l o primeiro neurônio de saída e h_n o último vetor de saída. O mecanismo de atenção irá produzir um vetor com os pesos de atenção α , e a representação ponderada r (ROCKTÄSCHEL *et al.*, 2015).

$$M = tanh(W^y y + W^h h_n \otimes e_L) \tag{15}$$

$$\alpha = softmax(w^t M) \tag{16}$$

$$r = Y\alpha^t \tag{17}$$

Os valores W^y e W^h são as projeções das matrizes treinadas, w e w^T é o vetor de parâmetros treinado e sua transposta respectivamente.

A representação final é obtida por uma combinação não linear de uma representação ponderada r da entrada, e o ultimo vetor da saída h_n conforme é apresentado na Formula 18, sendo W^p, W^x as matrizes de projeção treinadas.

$$h^* = tanh(W^p r + W^x h_N) \tag{18}$$

2.9 Processamento de Linguagem Natural

O Processamento de Linguagem Natural (PLN) consiste no desenvolvimento de modelos computacionais para a realização de tarefas que dependem de informações expressas em linguagem natural. O objetivo do PLN é fazer com que os computadores realizem tarefas úteis que envolvam a linguagem humana, como possibilitar a comunicação homem-máquina, melhorar a comunicação homem-homem ou realizar um processo útil envolvendo texto ou fala (MARTIN; JURAFSKY, 2009).

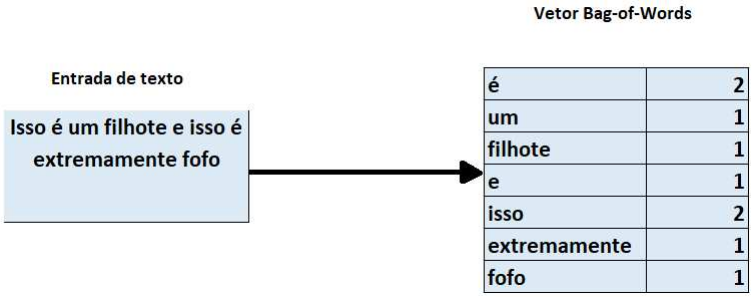
2.10 Técnicas de PLN

Essencialmente as redes neurais são algoritmos de aprendizado de máquina projetados para trabalharem com números, então algumas técnicas foram desenvolvidas para que seja possível incluir dados textuais.

2.10.1 Bag-of-Words

O *Bag-of-Words*, ou saco de palavras do português, é um modelo em que as palavras são transformadas em *tokens*, e para um certo contexto é verificada a frequência de cada um desses *tokens* (MIKOLOV *et al.*, 2013). Com o número de ocorrências de cada *token* é criado um vetor que cada posição representa uma palavra. A Figura 11 exemplifica o funcionamento do modelo.

Figura 11 – Representação da Bag-of-Words



Fonte: autoria própria (2019)

No exemplo da Figura 11, são identificados os seguintes *tokens*: "é", "um", "filhote", "e", "isso", "extremamente", "fofo". E então é verificada a frequência de cada *token*:

- “é” = 2
- “um” = 1
- “filhote” = 1
- “e” = 1
- “isso” = 2
- “extremamente” = 1
- “fofo” = 1

O conjunto desses dados constitui os elementos do vetor $v = [2,1,1,1,2,1,1]$.

Com a obtenção desse vetor, o modelo de Bag-of-Words pode ser utilizado para auxiliar as RNAs em tarefas como análise de sentimento em texto e classificação de assunto em texto.

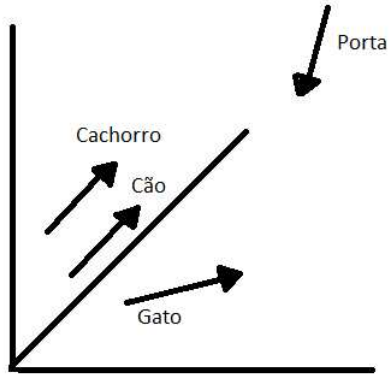
2.10.2 Word Embedding

Pode-se definir Word Embedding como um conjunto de técnicas que mapeiam a sintaxe e a semântica de uma linguagem em um espaço conhecido como *embedding space* (GHANNAY *et al.*, 2016).

Imaginando um espaço hipotético apresentado na Figura 12, em uma boa representação de *word embeddings* a palavra “cão” e “cachorro” devem ser mapeados em vetores muito próximos devido o grau de semelhança, e considerando que neste mesmo espaço, ao inserir

“gato” e “porta” , o vetor de “gato” deve estar mais próximo de “cachorro” ao invés de “porta”, considerando que gato e cachorro são animais domesticáveis, e porta um objeto inanimado.

Figura 12 – Representação de Word Embeddings



Fonte: autoria própria (2019)

O Word Embedding é processado utilizando técnicas de redução de dimensionalidade sobre a matriz de co-ocorrência que foi gerada a partir da entrada de texto. Existem vários modelos de tratamento como o *Common Bag of Words (CBOW)* e *Skip-Gram* .

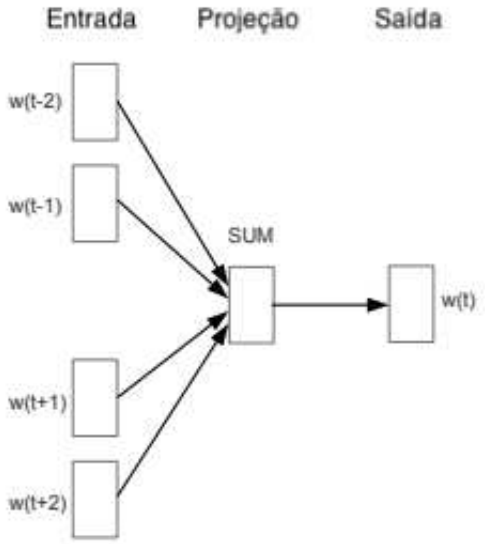
2.10.3 CBOW

Neste modelo, as entradas são vetores *One-Hot-Encoded*. Os vetores *One-Hot-Encoded* são vetores esparços, ou seja, para um contexto onde existem dez palavras, cada posição do vetor corresponde a uma palavra única, e o vetor *One-Hot-Encoded* de cada palavra é um vetor com o valor 1 na posição que representa esta palavra, e o valor 0 nas demais posições (MIKOLOV *et al.*, 2013). Exemplificando, em um contexto onde existem somente as palavras “gato” , “cachorro” e “passaro”, seus vetores *One-Hot-Encoded* seriam respectivamente : $v1 = (1,0,0)$, $v2 = (0,1,0)$, $v1 = (0,0,1)$.

Este modelo obtêm o contexto de cada palavra da entrada e tenta predizer a palavra corresponde para este contexto (GHANNAY *et al.*, 2016). Para tentar predizer a palavra alvo do contexto, as entradas são vetores *One-Hot-Encoded* de tamanho $C \times V$, onde V representa o

tamanho do vocabulário e C o tamanho do contexto. A camada escondida contêm N neurônios e a saída um vetor *One-Hot-Encoded* de tamanho V, como apresentado na Figura 13.

Figura 13 – Representação de CBOW



Fonte: (MIKOLOV *et al.*, 2013)

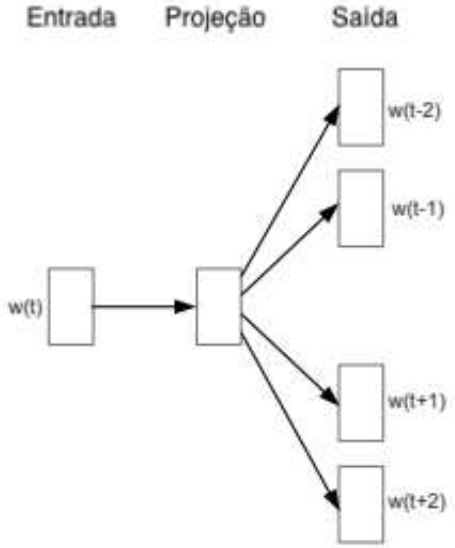
Por exemplo, para a frase "o céu é azul", se a palavra a ser predita for "céu", o contexto é formado pelas palavras "o", "é" e "azul". O modelo da Figura 13 obtêm C palavras de contexto, onde o valor de $W_{V\times N}$ é a matriz de peso que mapeia a entrada x_{Ck} para a camada escondida h_i de dimensão N , e o valor de $W'_{N\times V}$ é a matriz de peso que mapeia as saídas das camadas escondidas para a camada de saída y_j.

2.10.4 Skip-gram

O modelo de *Skip-gram* praticamente realiza o processo inverso do modelo de *CBOW*, ou seja, com uma palavra tenta-se predizer as demais palavras mais prováveis em um determinado contexto (MIKOLOV *et al.*, 2013).

O modelo da Figura 14 obtêm a palavra de contexto, onde o valor de $W_{V\times N}$ é a matriz de peso que mapeia a entrada $x \in \mathbb{R}^{V\times N}$ para a camada escondida $h \in \mathbb{R}^N$ e o valor de $W' \in \mathbb{R}^{N\times V}$ é a matriz de peso que mapeia as saídas das camadas escondidas para as camadas de saída $y_C \in \mathbb{R}^V$.

Figura 14 – Representação de Skip-gram



Fonte: (MIKOLOV *et al.*, 2013)

2.11 ElMo

O *Deep Contextualized Word Representations* mais conhecido como ELMo, é uma representação contextualizada de palavras que modela características complexas de uso de palavras (PETERS *et al.*, 2018).

O ELMo é uma evolução dos *Embeddings* convencionais, tendo em vista que ele prevê além de realizar o aprendizado de características sintáticas e semânticas das palavras, também tratar a polissemia que é um grande problema para os *Embeddings* convencionais. Resumindo, ele capaz de diferenciar sentidos diferentes que uma mesma palavra possa assumir dependendo do contexto em que ela está. (PETERS *et al.*, 2018).

O modelo de linguagem que é o ELMo se baseia para realizar o pré-treino é o *Bidirectional Language Model* (biLM). O biLM trabalha usando duas redes LSTM que tentam prever o próximo *token* que irá suceder a cada *token* do contexto, sendo que uma chamada de *forward* parte da origem da sequência de *tokens*, e a outra chamada *backward* parte do final da sequência de *tokens*. Para predizer uma palavra alvo, são analisados os contextos anterior e posterior da palavra alvo por duas subredes diferentes. O treinamento busca maximizar a probabilidade da palavra alvo ser predita dados os seus contextos a esquerda e a direita. Para isso, usa-se o logaritmo das probabilidades, conforme descrito na Equação 19.

$$\sum_{k=1}^{N} (\log p(t_k|t_1,.....,t_{k-1};\Theta_x,\Theta_{LSTM}^{\rightarrow},\Theta_s) + \log p(t_k|t_1,.....,t_{k-1};\Theta_N,\Theta_{LSTM}^{\leftarrow},\Theta_s)) \quad (19)$$

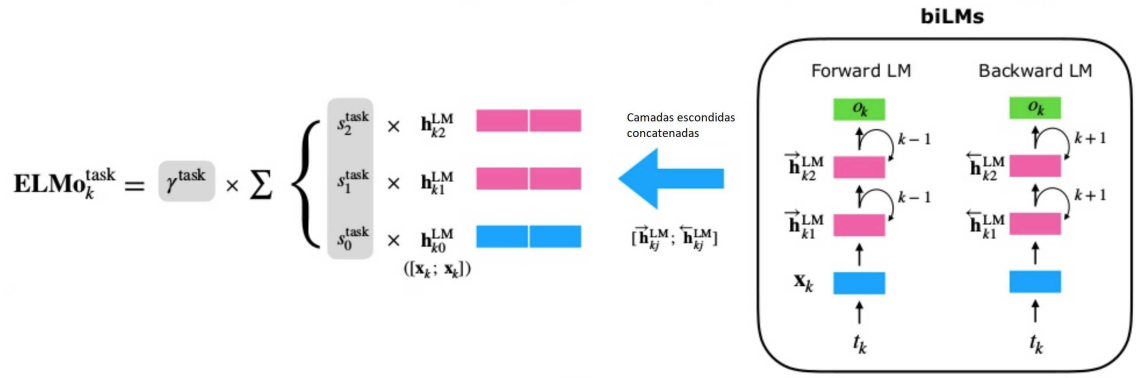
Na Equação 19, o valor de Θ_x é a representação de *embeddings* de cada *token*, e o valor de Θ_s representa as camadas escondidas de *forward* e *backward*, tendo seus parâmetros mantidos separadamente pelas redes LSTM.

A segunda parte do modelo consiste em usar os estados gerados pela rede LSTM para cada *token*, e criar um vetor de representação de cada palavra. No ELMo a representação de cada palavra é computada com a soma ponderada das ativações da camada rede *forward* concatenada com a soma ponderada das ativações da camada *backward*. O processo é ilustrado na Equação 20.

$$ELMo_k = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,j}^{KM} \quad (20)$$

O escalar γ^{task} permite que o modelo de tarefa dimensione todo o vetor ELMo. s_j^{task} representa os pesos normalizados por uma função *softmax*. Os indices k e j representam os índices das palavras e da camada escondida, respectivamente, dos valores extraídos.

Figura 15 – Representação de Elmo



Fonte: (PETERS *et al.*, 2018)

No exemplo da Figura 15, $h_{k,j}$ é a saída das redes LSTM para cada palavra k, e s_j é o peso de computado de $h_{k,j}$ computando a representação para k.

2.12 BERT

O Bidirectional Encoder Representations from Transformers (BERT) , foi projetado para pré-treinar representações bidirecionais de textos analisando simultaneamente uma sentença partindo do seu início ao fim e também do fim ao início (DEVLIN *et al.*, 2018). O resultado disso é que ele pode ser treinado para várias tarefas como responder perguntas e recuperação de informações em texto, com a adição de apenas uma camada de saída.

O BERT demonstrou-se eficiente na linha de recuperação de informações em texto quando comparado a tecnologias recentes (DEVLIN *et al.*, 2018). Um benchmark que prova o seu desempenho é o General Language Understanding Evaluation (GLUE) (WANG *et al.*, 2018), que possibilita avaliar o poder de compreensão de linguagem natural usando arquiteturas inteligentes. A tabela abaixo compara o BERT treinado no corpus *base* e o BERT treinado com o corpus **large**, com outras arquiteturas. QQP, CoLA e os demais itens listado na primeira coluna, são alguns datasets que o GLUE utiliza para realizar seus testes.

Tabela 1 – Tabela dos resultados de benchmark utilizando a ferramente GLUE

Sistema	Pre-OpenAI	BiLSTM+ELMo+attb	OpenAI	BERT (base)	BERT (large)
MNLI-(m/mm)	80,6/80,1	76,4/76,1	82,1/81,4	84,6/83,4	86,7/85,9
QQP	66,1	64,8	70,3	71,2	72,1
QNLI	82,3	79,8	87,4	90,5	92,7
SST-2	93,2	90,4	91,3	93,5	94,9
CoLA	35,0	36,0	45,4	52,1	60,5
STS-B	81,0	73,3	80,0	85,8	86,5
MRPC	86,0	84,9	82,3	88,9	89,3
RTE	61,7	56,8	56,0	66,4	78,1
Média	74,0	71,0	75,1	79,6	82,1

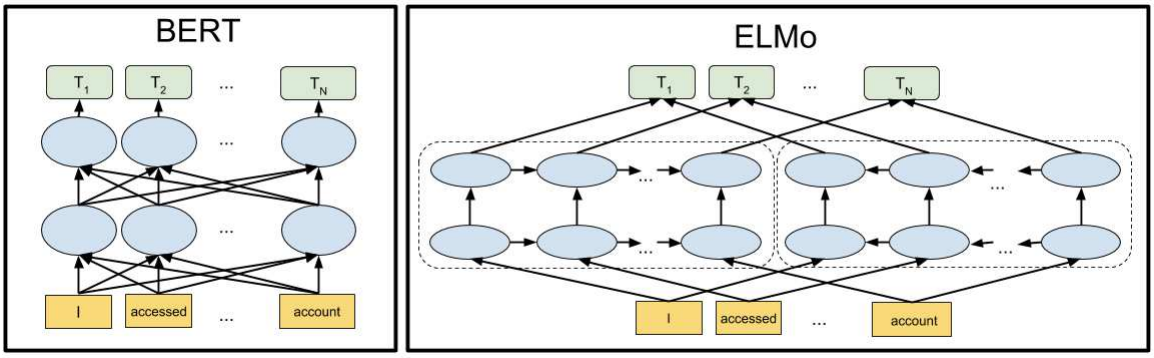
Fonte: (DEVLIN *et al.*, 2018)

BERT utiliza o Encoder Transformer multi-camadas bidirecional baseado no trabaho de (DEVLIN *et al.*, 2018). A sua entrada é representada por um texto ou por um par de sentenças. No processo de leitura do texto, diferente do ELMo, que usa modelos independentes para ler a sentença da esquerda para a direita e da direita para a esquerda, e faz a concatenação das representações somente nas últimas camadas. No BERT, a partir da primeira camada todas as representações são conectadas, como ilustrado na Figura 16.

No pré-treino, o BERT trabalha com o conceito de soma de *Embeddings*, sendo eles:

- *Embeddings* de *tokens*: representando o *embedding* da própria palavra.
- *Embeddings* de segmentação: representando o *Embedding* que os separa a primeira sentença da segunda sentença.

Figura 16 – Comparação do BERT e o ELMo



Fonte: Adaptado de Devlin *et al.* (2018)

- *Embeddings* posicionais: diferentemente das redes recorrentes, a arquitetura do Bert perde a informação do local em que uma palavra ocorreu na sentença. Os *embddings* posicionais são gerados por uma técnica para alterar os *embeddings* e codificar informação da posição em que a palavra original ocorreu na sentença. Assim, esses *embedding* são diferenciados pelas posições dos tokens na sentença.

Dessa forma, para cada palavra, é feita a soma dos três *Embeddings* para criar a representação que será inserida no modelo.

Um ponto negativo dos modelos bidirecionais é que, por ter acesso a todas as palavras da sentença de uma vez só, ele pode acessar a palavra que deve ser predito pelo modelo de língua. Para resolver este problema foi criado o conceito de *Masked Language Model* (MLM). No MLM 15% dos tokens são selecionados, e utilizando esses *tokens*, 80% são substituídos por "[MASK]", 10% são substituídos por uma palavra aleatória e 10% são mantidos no seu estado original. Essa técnica provê que o modelo tente predizer quais *tokens* devem estar nas posições que foram substituídas por [MASK], e que seja verificado constantemente a semântica do contexto, impedindo que o modelo decore a palavra a ser predita.

Com a técnica de MLM, o BERT é capaz de entender uma frase ou seção do contexto, mas ainda não entende como se comunicam. A técnica de *Next Sentence Prediction* (NSP) faz um teste binário em que a rede deve saber se uma sentença é sequência da outra. Na etapa de NSP, metade das vezes uma sentença B é sequência de A, e a outra metade é selecionado uma sentença aleatória do corpus. O objetivo do NSP é forçar a rede entender o relacionamento das sentenças(DEVLIN *et al.*, 2018).

O seu pré-treino são definidos em alguns parâmetros:

- 256 sequências * 512 tokens;
- 1 milhão de iterações por época;
- 40 épocas;

- Função de ativação GELU (ReLU modificada, usada no OpenAI GPT).

Os desenvolvedores aconselham que os *datasets* para o pré-treino sejam grandes como o do *BookCorpus*[1] que contem aproximadamente 800 milhões de palavras escritas em inglês (DEVLIN *et al.*, 2018). O *BookCorpus* é um conjunto de dados com uma semântica de alto nível, que pode conter livros e filmes, com uma grande variedade de entidades, assim sendo considerado um *dataset* amplamente rico ao ser comparado a outras bases disponíveis (ZHU *et al.*, 2015), e isso pode auxiliar no treinamento de redes neurais para as tarefas de PLN.

2.13 RoBERTa

O Robustly Optimized BERT Pretraining Approach (RoBERTa) é uma proposta provinda de uma avaliação minuciosa do BERT, e de acordo com Liu *et al.* (2019) acredita-se que o BERT foi sub-treinado. A proposta é fazer algumas modificações como:

- Treinar o modelo por mais tempo, com batches maiores e mais dados;
- Remover do treinamento a predição da próxima palavra;
- Treinamento em sequências mais longas;
- Alterar dinamicamente o padrão de mascaramento aplicado aos dados de treinamento.

É observado uma grande melhora no resultado quando comparado do BERT large, como é apresentado na Tabela 2 abaixo os testes SQuAD, MNLI-m e SST-2.

Tabela 2 – Teste comparativo entre RoBERTa e BERT

Modelo	Dados	Batch Size	Passos	SQuAD (v1.1/v2.0)	MNLI-m	SST-2
RoBERTa	160GB	8K	500K	94,6/89,4	90,2	96,4
BERT	13GB	256	1M	90,9/81,8	86,6	93,7

Fonte: (LIU *et al.*, 2019)

De acordo com Liu *et al.* (2019), é ressaltada a importância dessas decisões do design da rede, pois foi possível melhorar a acurácia do modelo sem a necessidade de realizar o *fine-tuning* para a tarefa especifica.

[1] https://huggingface.co/datasets/bookcorpus

2.14 DistilBERT

O DistilBERT é uma versão do BERT destilada, um modelo menor, mais rápido e menos computacionalmente custoso (SANH *et al.*, 2019).

A destilação em RNAs compreende em um conjunto de técnicas que têm como objetivo comprimir os modelos, para que seja possível utilizar os modelos de RNAs com menos recursos (GOU *et al.*, 2020). Uma das técnicas efetivas é fazer com que um modelo menor tente imitar os resultados de um modelo maior, no estilo aluno e professor.

De forma geral o DistilBERT tem mesma estrutura do BERT, porém as *embeddings* e o *pooler* são removidos, enquanto o número de camadas é reduzido pela metade. Foi verificado que comparado ao BERT, é possível reduzir o tamanho do modelo em 40%, o treinamento ser 60% mais rápido e reter 97% da performance, como é observados na Tabela 3.

Tabela 3 – Comparação de desempenho dos modelos BERT, DistilBERT e ELMo

Model	Score	CoLA	MNLI	MRPC	QNLI	QQP	RTE
ELMo	68,7	44,1	68,6	76,6	71,1	86,2	53,4
BERT-base	79,5	56,3	86,7	88,6	91,8	89,6	69,3
DistilBERT	77,0	51,3	82,2	87,5	89,2	88,5	59,9

Fonte: (SANH *et al.*, 2019)

De acordo com Sanh *et al.* (2019) o DistilBERT é uma prova que modelos de propósito geral como o BERT, podem ser treinados com sucesso a partir da destilação de RNAs.

2.15 Análise de Sentimentos

A Análise de sentimentos é um campo de estudo que analisa informações como opiniões, sentimento, emoções das pessoas sobre entidades como produtos, serviços, empresa entre outros (LIU *et al.*, 2010). Com o grande volume de informações disponíveis e a evolução do número de usuários da Internet, houve um grande esforço em classificar informações como comentários, resenhas, opiniões, pois essas são informações valiosas para várias áreas como o comércio, para alavancar a venda de produtos que são bem avaliados, e saúde, para identificar sintomas de depressão ou risco de suicídio como apresentado no trabalho de Madhu (2018), que

analisa a tendência de uma pessoa ser um potencial suicida analisando mensagens no Twitter ou blogs. Com o tempo surgiram técnicas de análise para a classificação dessas informações, e essas análises podem ser categorizadas como (LIU *et al.*, 2010):

- Documento: analisa todo o documento, e classifica se o mesmo expressa sentimento positivo ou negativo (PANG *et al.*, 2002);
- Sentença: analisa cada sentença, e classifica se a mesma expressa sentimento positivo, negativo ou neutro (WIEBE *et al.*, 1999);
- Entidade e aspecto: analisa todo documento, e determina o grau de afinidade do autor com as entidades apresentadas (HU; LIU, 2004).

Com o foco a análise do documento, existem vários métodos de aprendizado de máquina para extrair estas informações. Dois desses métodos, segundo Pang *et al.* (2002), são:

- Naive Bayes: consiste em encontrar uma probabilidade de determinada condição ser positiva , multiplicando o fator de condição com a probabilidade de ser positiva, a formula pode ser descrita como:
$$P(c|d) = \frac{P(c)P(d|c)}{P(d)} \tag{21}$$
Sendo “d” um documento e “c” classe a qual ele pertence.
- Maquina de Vetores de Suporte: as Máquinas de Vetores de Suporte (SVM) é um conjunto de ferramentas de classificação e regressão. Uma SVM constrói hiperplanos no espaço para classificação ou regressão dos dados. Utilizando este método, é possível criar espaços dimensionais sobre os textos, separando-os em positivo e negativo. A formula pode ser descrita como:
$$\overrightarrow{w} := \sum_j a_j c_j \overrightarrow{d_j}, a_j \geq 0 \tag{22}$$
Sendo “w” o vetor que separa os espaços, “d” o documento, e “c” correspondendo o valor positivo ou negativo de classe, e “a” sendo obtido ao resolver o problema de otimização.

Será apresentado a seguir dois trabalhos que utilizaram a análise de sentimentos para resolverem tarefas distintas:

- Educação: no trabalho de autoria de Altrabsheh *et al.* (2013), é proposto uma solução para auxiliar o ensino, utilizando os *feedbacks* dos alunos em redes sociais. segundo Altrabsheh *et al.* (2013) os alunos se sentem mais à vontade em expressar as suas opiniões nas redes sociais, à falar diretamente ao professor. Essas mensagens são capturadas das redes sociais, e então processadas e categorizadas. Assim o professor pode acessar a informação gerada pela análise, e aprimorar a forma de ensino;
- Saúde: no trabalho de autoria de Edara *et al.* (2019), é proposto a criação de uma rede LSTM e a utilização dos conceitos de análise de sentimentos, para extrair informações do

humor dos portadores de cancêr nas redes sociais, pois muitas pessoas compartilham as suas experiências com o câncer. Segundo Edara *et al.* (2019), o monitoramento do humor dessas pessoas afetadas, desempenham um papel crucial no tratamento dos pacientes.

3 MATERIAIS E MÉTODOS

Neste capítulo será descrita a metodologia que será aplicada para realizar o fine-tuning dos modelos BERT, RoBERTa e DistilBERT para análise de sentimentos binária.

3.1 Equipamentos

O treinamento de modelos baseados em BERT costuma ser bastante custoso computacionalmente, devido isso, é necessário sistemas com alto poder computacional para que seja possível realizar o treinamento em tempo hábil. A solução foi utilizar o Google Colab Pro. O Google Colab Pro é um serviço em nuvem da Google que oferece máquinas de alta performance por um custo acessível. O hardware que será utilizado para realizar o fine-tuning dos modelos podem ser visualizados na Tabela 4.

3.2 Ferramentas

A seguir serão apresentadas as ferramentas que serão utilizadas neste trabalho:

Tabela 4 – Especificações técnicas do Hardware utilizado

Espeficicações	Workstation
CPU	2vCPU @ 2.2GHz
Memória RAM	24GB
Placa de Vídeo	Nvidia T4 Tesla

Fonte: autoria própria (2021)

- Python: é uma linguagem de programação de alto nível, interpretada, muito utilizada na área de IA. Ela tem um grande suporte de bibliotecas e *frameworks* que auxiliam em diversas tarefas de IA, desde o pré-processamento até o treinamento dos modelos;
- Google Colab Pro: o Google Colab Pro é uma *integrated development environment* (IDE) de Notebooks, que integra várias bibliotécas de cunho ciêntifico, como a NumPy, ScyPy e Pandas;
- NumPy: é uma biblioteca do Python de código aberto, que adiciona ao Python o suporte de trabalhar com tensores de ordem *n* com uma grande facilidade;
- Pandas: Pandas é uma biblioteca do Python que provê funções auxiliares para tratamento de dados.
- TensorFlow: O TensorFlow é uma plataforma de código aberto para machine learning.
- Transformers: esta biblioteca contém implementações do PyTorch, pesos de modelo pré-treinados, scripts de uso e utilitários de conversão para modelos como BERT e GPT-2;
- HuggingFace: é uma biblioteca que tem como objetivo construir, treinar e implantar modelos de última geração com base no código aberto de referência em processamento de linguagem natural.

3.3 Métodos

As etapas do desenvolvimento são apresentadas na Figura 17, e detalhadas nos parágrafos a seguir.

A base de dados escolhida para o treinamento foi o corpus IMDb[1]. O corpus IMDb contêm 50 mil resenhas de filmes em inglês para processamento de linguagem natural ou análise de texto (MAAS *et al.*, 2011), sendo 12.500 resenhas positivas, e 12500 resenhas negativas. Este é um conjunto de dados bastante indicado para classificação de sentimento binário. Cada sentença do corpus contem duas informações, a classe descrita como “label” com os possíveis valores 0 para negativo e 1 para positivo, e o texto que é descrito como “text”.

Na lista a seguir será apresentado um exemplo de sentença com sentimento positivo e uma com sentimento negativo:

- Positivo: "Enjoyable and watchable. Tim Meadows at his best. A big boost from Billy Dee Williams. He and a very funny John Witherspoon provide a solid foundation for Mr.

[1] https://huggingface.co/datasets/imdb

Meadows' riffing. Have fun with this one.";

- Negativo: "This was the only time I ever walked out on a movie. Years later, I saw it in the cable listings and thought, "Maybe I should give it another try."Suffice to say that I was right the first time. This ranks second only to Godzilla 1998 as the worst movie I've ever seen."

Para utilizar o Corpus IMDb, é necessário realizar um pré-processamento, pois as informações estão dispostas no formato *HyperText Markup Language* (HTML), então será utilizada a ferramenta *pandas* para tratar os dados e gerar um formato *Comma-Separated Values* (CSV), que separa os *tokens* usando vírgulas.

Para que o BERT e seus derivados consigam realizar o fine-tuning, será necessário gerar um modelo de *dataset* do TensorFlow à partir do corpus pré-processado do IMBb.

Após o processamento e adequação do *dataset*, o mesmo será dividido em três partes, sendo:

- Treinamento: corresponde a 80% do *dataset*, será utilizada para realizar o fine-tuning do modelo;
- Validação: corresponde a 10% do *dataset*, será utilizada para validar o treinamento;
- Teste: corresponde a 10% do *dataset*, será utilizada para avaliar o modelo.

Os modelos pré-treinados do BERT, RoBERTa e DistilBERT serão obtidos a partir da biblioteca HuggingFace. A versão *base* foi escolhida para todos os modelos.

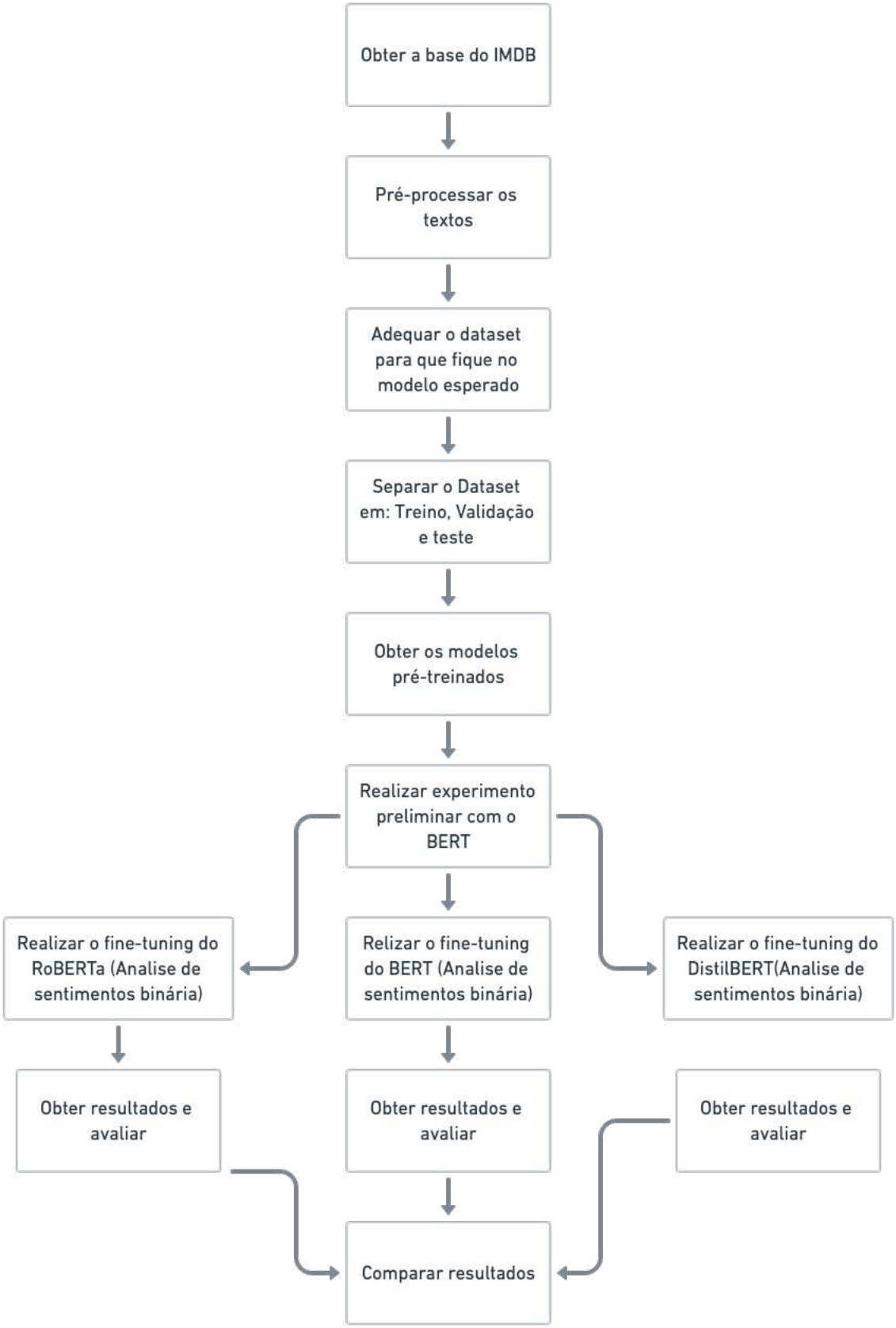
Antes de iniciar o *fine-tuning* dos três modelos, será realizado um experimento preliminar utilizando somente o BERT, para definir quais parâmetros serão utilizados nos três modelos.

Na etapa de *fine-tuning* os três modelos pré-treinados terão seus parâmetros atualizados a partir do fine-tuning, para que fiquem especialistas em análise de sentimentos binária. As partes do *dataset* de Treinamento e Validação serão utilizadas, para treinar e validar o momento de parada do treinamento respectivamente.

Na etapa de avaliação da análise de sentimentos binária será utilizada a parte de Teste do *dataset*. O procedimento dessa etapa consiste em utilizar os modelos e testar a acurácia ao predizer se as frases contidas no *dataset* expressam sentimento positivo ou negativo, com os seus resultados representados por 1 ou 0 respectivamente.

O resultado das avaliações retornará a acurácia do modelo, avaliando a quantidade de acertos e erros na forma percentual. Também será avaliado o tempo de treinamento dos modelos.

Figura 17 – Fluxograma de desenvolvimento



Fonte: autoria própria (2021)

4 RESULTADOS OBTIDOS

Neste capítulo, os três modelos treinados e avaliados são descritos. Os métodos de avaliação levaram em conta a acurácia dos modelos na etapa de testes e o tempo necessário para realizar o *fine-tuning*.

4.1 Experimento preliminar

O objetivo deste experimento preliminar é definir os parâmetros de *batch-size* e o número de épocas necessárias para o *fine-tuning*, utilizando o modelo BERT como base para depois replicar para os outros modelos. Em um primeiro momento o *batch-size* escolhido será de tamanho 32, e quatro o número de épocas.

Ao realizar o *fine-tuning* do modelo BERT com o corpus IMDb, foi constatado um comportamento de *overfitting* a partir da terceira época, pois a acurácia na validação foi superior a 0.99. Para contornar este efeito o número de épocas foi reduzido para dois.

Após a redução no número de épocas, o modelo BERT ao encerrar o seu *fine-tuning* apresentou uma acurácia na validação de 0.9750, sendo uma margem aceitável para validar o treinamento. Então foi definido os parâmetros para o *fine-tuning* em duas épocas e *batch-size* de 32 para todos os modelos.

4.2 Comparação entre modelos

Nesta etapas os três modelos BERT, RoBERTa e DistilBERT passaram pelo *fine-tuning* utilizando o mesmo conjunto de dados, e os mesmos parâmetros de treinamento. Os resultados

obtidos levando em consideração a acurácia são apresentados na Tabela 5.

Tabela 5 – Avaliação de acurácia dos modelos

Modelo	Acurácia de validação	Perda de validação	Acurácia de teste
BERT	0,9750	0,0767	0,8844
RoBERTa	0,9628	0,1110	0,8844
DistilBERT	0,9673	0,09626	0,8638

Fonte: autoria própria (2021)

Ao Analisar a Tabela 5, verifica-se que a acurácia nos testes dos modelos BERT, RoBERTa e DistilBERT foi de 88,44%, 88,44% e 86,38% respectivamente. O tempo de treinamento necessários para treinar os modelos serão medidos em segundos e apresentados na Tabela 6.

Tabela 6 – Avaliação de tempo de treinamento dos modelos

Modelo	Primeira época(s)	Segunda época(s)	Total(s)
BERT	645	622	1267
RoBERTa	702	636	1338
DistilBERT	330	318	648

Fonte: autoria própria (2021)

Ao Analisar a Tabela 6, verifica-se que o tempo total de treinamento do modelos BERT, RoBERTa e DistilBERT foi de 1267 segundos, 1338 segundos e 648 segundos respectivamente.

4.3 Discussão dos resultados

Primeiramente analisando os resultados da acurácia de validação, é verificado que o resultado dos três modelos ficou muito semelhante, sendo uma variação menor que 2%.

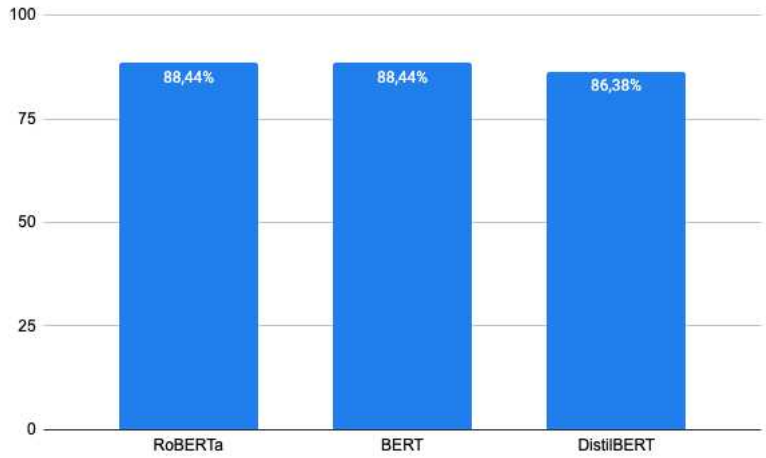
Os modelos BERT e RoBERTa obtiveram os melhores resultados neste quesito, mesmo que o modelo RoBERTa seja consideravelmente maior, não houve qualquer impacto na acurácia do modelo nas condições de *fine-tuning* aplicadas, possivelmente utilizando-se um dataset maior e testando-se mais parâmetros, poderia-se extrair um pouco mais o potencial do RoBERTa.

O Modelo DistilBERT coincidentemente neste teste, reteu cerca de 97% da performance dos outros dois modelos, que foi o mesmo valor de retenção apresentado pelos autores do DistilBERT. A representação é a apresentada na Figura 18.

Em um cenário em que o objetivo é maximizar os acertos, fica clara a superioridade dos modelos BERT e RoBERTa nos testes realizados, com uma vantagem para o modelo original BERT, em vez do RoBERTa, pois não houve mundaça na acurácia, e o modelo RoBERTa é consideravelmente maior.

Em um outro cenário que existe uma certa limitação de hardware, como alguns sistemas embarcados, existe a possibilidade de optar-se por um modelo menor como o DistilBERT.

Figura 18 – Gráfico de acurácia dos modelos BERT, RoBERTa e DistilBERT



Fonte: autoria própria (2021)

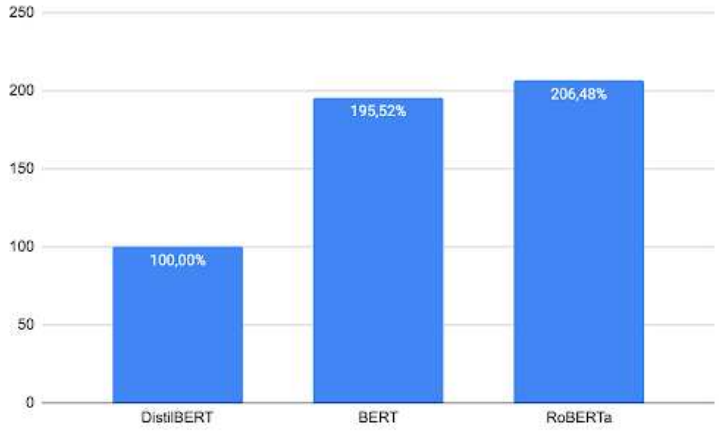
Ao avaliar o tempo de treinamento dos modelos, é verificado que quanto maior o modelo, mais tempo foi necessário para finalizar o *fine-tuning*, sendo o DistilBERT o mais rápido e o RoBERTa o mais demorado.

Os modelos BERT e RoBERTa ficaram com resultados muito próximos, enquanto o DistilBERT obteve resultados muito superiores, aproximadamente 50% do tempo dos outros modelos.

Em um cenário em que existe limitação de hardware e capital, e o prazo seja um fator importante, o DistilBERT é uma alternativa, pois ele poderá reduzir custos de treinamento e tempo necessário.

É apresentada na Figura 19 a comparação de tempo entre os modelos usando o tempo de treinamento do DistilBERT como base. O modelo BERT precisou de 195% do tempo, e o RoBERTa 206% do tempo.

Figura 19 – Gráfico comparando o tempo de *fine-tuning* dos modelos



Fonte: autoria própria (2021)

5 CONCLUSÕES

Neste trabalho foi abordado o tema de redes neurais artificiais e processamento de linguagem natural, com o foco na análise de sentimentos utilizando os modelos BERT, RoBERTa e DistilBERT.

Todos os objetivos gerais e específicos propostos foram cumpridos, compreendendo eles como as etapas para realizar o *fine-tuning* para análise de sentimentos dos modelos BERT e seus derivados, e entender as vantagens e desvantagens. Com estes modelos será possível extrair informações sobre a análise de sentimentos de diferentes fontes de textos, e utiliza-los para aplicações do mundo real, para aumentar o lucro sobre produtos, ou até mesmo salvar vidas.

Os resultados apresentados no Capítulo 4 são úteis a profissionais e pesquisadores que desejarem utilizar algum modelo BERT para esta tarefa específica, pois é apresentado que a escolha do modelo irá depender de fatores como hardware e tempo disponível, e se o objetivo será maximizar a acurácia utilizando os mesmo parâmetros e base de dados, o BERT será uma escolha mais eficiente, pois possivelmente irá evitar desperdícios de recursos e ficará com a melhor performance. Também será útil para servir de comparação para outros modelos neurais, que não necessariamente precisam ser derivados de BERT.

Uma análise que pode ser feita sobre o resultado do RoBERTa, é que possivelmente o modelo não conseguiu extrair todo o seu potencial utilizando os mesmos parâmetros que foi utilizado nos outros modelos, por se tratar te um modelo feito para trabalhar com *datasets* maiores e utilizar mais parâmetros como batch sizes maiores, conjunto de dados maiores, da mesma forma que é feito em seu pré-treino.

REFERÊNCIAS

ALTRABSHEH, N.; GABER, M. M.; COCEA, M. Sa-e: sentiment analysis for education. **Frontiers in Artificial Intelligence and Applications**, v. 255, p. 353–362, 2013.

BANK, D.; KOENIGSTEIN, N.; GIRYES, R. Autoencoders. **arXiv preprint arXiv:2003.05991**, 2020.

BEZERRA, E. Introdução à aprendizagem profunda. **Artigo–31º Simpósio Brasileiro de Banco de Dados–SBBD2016–Salvador**, 2016.

CHOROWSKI, J.; BAHDANAU, D.; SERDYUK, D.; CHO, K.; BENGIO, Y. Attention-based models for speech recognition. **arXiv preprint arXiv:1506.07503**, 2015.

CHOWDHURY, G. G. Natural language processing. **Annual review of information science and technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, 2003.

CHURCHLAND, P. S.; SEJNOWSKI, T. J. **The computational brain**. [S.l.]: MIT press, 2016.

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

DOHME, V. D. **Atividades lúdicas na educação: o caminho de tijolos amarelos do aprendizado**. [S.l.]: Vozes, 2003.

EDARA, D. C.; VANUKURI, L. P.; SISTLA, V.; KOLLI, V. K. K. Sentiment analysis and text categorization of cancer medical records with lstm. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–17, 2019.

GHANNAY, S.; FAVRE, B.; ESTEVE, Y.; CAMELIN, N. Word embedding evaluation and combination. In: **Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)**. [S.l.: s.n.], 2016. p. 300–305.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. [S.l.]: MIT press, 2016.

GOU, J.; YU, B.; MAYBANK, S. J.; TAO, D. Knowledge distillation: A survey. **CoRR**, abs/2006.05525, 2020.

GRAVES, A. Supervised sequence labelling. In: **Supervised sequence labelling with recurrent neural networks**. [S.l.]: Springer, 2012. p. 5–13.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007.

HIROSE, Y.; YAMASHITA, K.; SHIMPEI. Back-propagation algorithm which varies the number of hidden units. **Neural Networks**, v. 4, n. 1, p. 61–66, 1991. ISSN 0893-6080. Disponível em: <https://www.sciencedirect.com/science/article/pii/089360809190032Z>.

HU, M.; LIU, B. Mining and summarizing customer reviews. In: **Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2004. (KDD '04), p. 168–177. ISBN 1581138881. Disponível em: <https://doi.org/10.1145/1014052.1014073>.

JADERBERG, M.; SIMONYAN, K.; ZISSERMAN, A.; KAVUKCUOGLU, k. Spatial transformer networks. In: CORTES, C.; LAWRENCE, N.; LEE, D.; SUGIYAMA, M.; GARNETT, R. (Ed.). **Advances in Neural Information Processing Systems**. Curran Associates, Inc., 2015. v. 28. Disponível em: <https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf>.

JONES, N. Computer science: The learning machines. **Nature News**, v. 505, n. 7482, p. 146, 2014.

KANEKO, M.; KOMACHI, M. Multi-head multi-layer attention to deep language representations for grammatical error detection. **arXiv preprint arXiv:1904.07334**, 2019.

KRATZERT, F.; KLOTZ, D.; BRENNER, C.; SCHULZ, K.; HERRNEGGER, M. Rainfall–runoff modelling using long short-term memory (lstm) networks. **Hydrology and Earth System Sciences**, v. 22, n. 11, p. 6005–6022, 2018. Disponível em: <https://hess.copernicus.org/articles/22/6005/2018/>.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436, 2015.

LIU, B. *et al.* Sentiment analysis and subjectivity. **Handbook of natural language processing**, Oxfordshire, v. 2, n. 2010, p. 627–666, 2010.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. **arXiv preprint arXiv:1907.11692**, 2019.

MAAS, A. L.; DALY, R. E.; PHAM, P. T.; HUANG, D.; NG, A. Y.; POTTS, C. Learning word vectors for sentiment analysis. In: **Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies**. Portland, Oregon, USA: Association for Computational Linguistics, 2011. p. 142–150. Disponível em: <http://www.aclweb.org/anthology/P11-1015>.

MADHU, S. An approach to analyze suicidal tendency in blogs and tweets using sentiment analysis. **Int. J. Sci. Res. Comput. Sci. Eng.**, v. 6, n. 4, p. 34–36, 2018.

MARTIN, J. H.; JURAFSKY, D. **Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition**. [S.l.]: Pearson/Prentice Hall Upper Saddle River, 2009.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

NAMIN, A.; LEBOEUF, K.; MUSCEDERE, R.; WU, H.; AHMADI, M. Efficient hardware implementation of the hyperbolic tangent sigmoid function. In: **Proceedings - IEEE International Symposium on Circuits and Systems**. [S.l.: s.n.], 2009. p. 2117 – 2120.

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up? sentiment classification using machine learning techniques. In: **EMNLP**. [S.l.: s.n.], 2002.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. **arXiv preprint arXiv:1802.05365**, 2018.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding with unsupervised learning**. [S.l.], 2018.

ROCKTÄSCHEL, T.; GREFENSTETTE, E.; HERMANN, K. M.; KOČISKỲ, T.; BLUNSOM, P. Reasoning about entailment with neural attention. **arXiv preprint arXiv:1509.06664**, 2015.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2019.

WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O.; BOWMAN, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. **arXiv preprint arXiv:1804.07461**, 2018.

WIEBE, J. M.; BRUCE, R. F.; O'HARA, T. P. Development and use of a gold-standard data set for subjectivity classifications. In: **Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics**. College Park, Maryland, USA: Association for Computational Linguistics, 1999. p. 246–253. Disponível em: <https://aclanthology.org/P99-1032>.

ZHU, Y.; KIROS, R.; ZEMEL, R.; SALAKHUTDINOV, R.; URTASUN, R.; TORRALBA, A.; FIDLER, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In: **The IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015.