UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE CRATEÚS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HÊNIO TIERRA LIMA SAMPAIO

TRANSFERÊNCIA DE CONHECIMENTO COM APRENDIZAGEM PROFUNDA PARA CLASSIFICAÇÃO DE TEXTOS EM LÍNGUA PORTUGUESA

CRATEÚS

2022

HÊNIO TIERRA LIMA SAMPAIO

TRANSFERÊNCIA DE CONHECIMENTO COM APRENDIZAGEM PROFUNDA PARA CLASSIFICAÇÃO DE TEXTOS EM LÍNGUA PORTUGUESA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus de Crateús da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Msc. Lívio Antônio Melo Freire (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. José Wellington Franco da Silva
Universidade Federal do Ceará (UFC)

Prof. Msc. Marciel Barros Pereira
Universidade Federal do Ceará (UFC)

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S183t Sampaio, Hênio Tierra Lima.
Transferência de conhecimento com aprendizagem profunda para classificação de textos em língua portuguesa / Hênio Tierra Lima Sampaio. – 2022.
41 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Crateús, Curso de Ciência da Computação, Crateús, 2022.
Orientação: Prof. Me. Lívio Antônio Melo Freire..
Coorientação: Prof. Dr. José Wellington Franco da Silva..

1. Inteligência Artificial. 2. Aprendizagem de Máquina. 3. Transferência de Aprendizagem. 4. Redes Neurais Artificiais. 5. Classificação automática de textos. I. Título.

CDD 004

AGRADECIMENTOS

Ao Prof. Msc. Lívio Antônio Melo Freire por me orientar em minha monografia.

A Prof. Msc. Lisieux Marie Marinho dos Santos Andrade por me orientar na disciplina de Projeto de Pesquisa Científica e Tecnológica.

Aos Prof. Dr. José Wellington Franco da Silva e Prof. Msc. Marciel Barros Pereira, por participarem da minha banca de defesa de TCC1.

A todos os demais professores que participaram da minha formação educacional superior por sua paciência e crédito depositados em mim, aos quais tenho admiração.

As minhas mãe e irmã que sempre me ajudaram, confiaram em mim, e me incentivaram a seguir adiante.

As outras mães presentes na minha vida: Filomena Furtado Sampaio, Maria Leônia Furtado Sampaio.

Ao meu tio Sávio Furtado Sampaio que foi responsável por bancar financeiramente parte importante da minha formação educacional.

Ao meu tio Prof. Dr. Raimundo Furtado Sampaio por gentilmente revisar este trabalho.

A amiga Dr. Paula Aragão Lima por gentilmente revisar este trabalho.

A todos os amigos estudantes da UFC que de algum modo me ajudaram na jornada da graduação e período de permanência na cidade de Crateús.

A mim mesmo por insistir nas agruras do ensino superior.

RESUMO

Abordagens baseadas em Aprendizagem de Máquina apresentam soluções computacionais para vários problemas do mundo real difíceis de serem resolvidos por meio da programação tradicional. Entre as diversas técnicas dessa área, destacam-se os métodos do campo da Aprendizagem Supervisionada, que são capazes de gerar modelos com base em dados rotulados. Na tarefa de classificação, esses modelos são funções que mapeiam um objeto do domínio de interesse para uma classe. No processo de aprendizagem, as classes são conhecidas previamente e identificadas pelo rótulo do dado. Algumas abordagens dentro desse campo, com diferentes graus de complexidade, podem ser utilizados para Classificação de Textos. Nesse domínio, modelos baseados em Redes Neurais Profundas (RNP), quando comparados à outras técnicas, apresentam resultados que refletem o estado da arte. Uma limitação das RNP para Processamento de Linguagem Natural (PLN), para geração de modelos robustos, é a exigência de disponibilidade de uma grande quantidade de dados rotulados. Quando não se tem acesso a um conjunto de dados grande o suficiente para uma boa performance do modelo, uma forma de contornar este problema é utilizando técnicas de Transferência de Conhecimento (TC). Trata-se de uma abordagem proeminente da área de Inteligência Artificial, que tem como finalidade utilizar um modelo gerado para um problema fonte como ponto de partida para a aprendizagem de um modelo alvo. Neste trabalho, aplicou-se essa técnica para melhorar o rendimento de classificação de textos em língua portuguesa. Investigou-se, por meio de experimentos computacionais, diferentes modelos de Redes Neurais Artificiais (RNA) e caracterizou-se as condições que garantem a TC nesses modelos.

Palavras-chave: Inteligência Artificial. Aprendizagem de Máquina. Transferência de Aprendizagem. Redes Neurais Artificiais. Classificação automática de textos.

ABSTRACT

Approaches based on Machine Learning present computational solutions to various real-world problems that are difficult to solve through traditional programming. Among the several techniques of this area, we highlight the methods of the field of Supervised Learning, which are capable of generating models based on labeled data. In the classification task, these models are functions that map an object of the domain of interest to a class. In the learning process, classes are known in advance and identified by the data label. Some approaches within this field, with varying degrees of complexity, can be used for Classification of Texts. In this domain, models based on Deep Neural Networks (DNN), when compared to other techniques, present state-of-the-art results. A limitation of DNNs for generating robust models is the requirement for availability of a large amount of labeled data. When you do not have access to a data set large enough for a satisfactory learning, one way around this problem is to use Knowledge Transfer techniques. This is a prominent approach in the area of Artificial Intelligence, whose purpose is to use a generated model for a source problem as a starting point for learning a target model. In this work, we apply this technique to improve the classification performance of texts in Portuguese language. It investigates, through computational experiments, different models of DNNs and characterizes the conditions that guarantee the transfer of knowledge.

Keywords: Artifitial Intelligence. Machine Learning. Transfer Learning. Artifitial Neural Networks. Automatic Classification of Texts.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

IA Inteligência Artificial

AS Aprendizagem Supervisionada

RNP Redes Neurais Profundas

RNA Redes Neurais Artificiais

RNC Redes Neurais Convolucionais

TC Transferência de Conhecimento

PLN Processamento de Linguagem Natural

CAT Classificação Automática de Textos

CA Classificação Automática

NS Neurônio Sigmoid

NP Neurônio Perceptron

ELMo Embeddings from Language Model

biLM Bidirectional Language Model

BERT Bidirectional Encoder Representations from Transformers

MLM Masked Language Modeling

NSP Next Sentence Prediction

LSTM Long short-term memory

GRU Gated Recurring Unit

ReLi Resenhas de Livros

1 INTRODUÇÃO

A Inteligência Artificial (IA) é uma área em amplo crescimento dentro e fora da academia, com cada vez mais importância no dia a dia da sociedade. Encontra-se várias aplicações de IA, que vão desde soluções de reconhecimento espacial, como um "óculos"capaz ajudar pessoas com problemas de visão (CARRATO *et al.*, 2015), ou carros possam dirigir sem interferência humana (BOJARSKI *et al.*, 2016), passando por problemas lógicos, como um computador capaz de vencer um humano no jogo chinês ”Go” (SILVER *et al.*, 2016), até a geladeira que pode falar com você de forma natural (ZE *et al.*, 2013) através de técnicas de PLN.

O campo da IA que está por trás das aplicações mencionadas corresponde, predominantemente, a métodos de Aprendizagem Supervisionada (AS). Por supervisão, entende-se que o processo de aprendizagem automática é conduzido por meio de amostras de dados rotuladas. Por exemplo, para que um programa de computador seja capaz reconhecer objetos na rua, é necessário que se disponha de um conjunto de dados com imagens (exemplos) identificadas com o tipo de objeto que contêm (rótulo).

Dentre as abordagens de AS, destacam-se as RNP. Esses modelos são definidos mediante o encadeamento hierárquico de camadas de RNA, que têm como finalidade extrair atributos em diferentes níveis de detalhamento sobre o objeto modelado. As RNP requerem quantidade volumosa de dados para serem geradas adequadamente. Além disso, mesmo quando a geração ocorre com sucesso, esses modelos são restritos ao domínio no qual foram produzidos (WANG *et al.*, 2014).

Em aplicações reais, dados rotulados podem ser escassos para a geração de modelos robustos. Uma das formas de contornar esse problema é através de técnicas de TC. Nessas abordagens, aproveita-se o modelo gerado por meio de uma base com grandes quantidades de dados (domínio de origem) para melhorar o desempenho do aprendizado de outro modelo em que se dispõe de quantidade reduzida de dados (domínio alvo).

Modelos de RNA podem ser configurados para as mais diversas tarefas, através da escolha das bases de dados adequadas ao propósito do domínio a ser modelado e da configuração dos componentes do modelo. Este trabalho é sobre Classificação Automática de Textos (CAT). Para tal, deve-se encontrar uma função capaz de mapear uma sequência de palavras para uma única classe de um conjunto definido previamente.

Foram utilizados métodos que comprovadamente funcionam na TC para classificação de textos em língua inglesa, adaptando-os e aplicando-os para a classificação de textos em língua

portuguesa.

A pesquisa foi realizada com base em trabalhos publicados recentemente acerca de transferência automática de conhecimento em língua inglesa. Como o assunto é bastante novo, não existem livros bem estabelecidos que abordem as técnicas a serem aplicadas, especialmente para o processamento de linguagem natural em língua portuguesa.

1.1 Objetivo geral

Aplicar técnicas de TC no treinamento de modelos de RNA para CAT em língua portuguesa.

1.2 Objetivos específicos

- Utilizar conjuntos de dados adequados escritos em língua portuguesa para servir de base para o treinamento do modelo de origem;

- Avaliar diferentes arquiteturas de RNA para a classificação de textos;

- Avaliar diferentes arquiteturas de RNA para a transferência de conhecimento;

- Avaliar configurações diferentes de hiperparâmetros envolvidos nos modelos para tarefas de classificação;

- Avaliar diferentes configurações de transferência de conhecimento para a aprendizagem do classificador do conjunto de dados alvo e realizar experimentos na tarefa de Análise de Sentimentos em língua portuguesa.

- Comparar diferentes arquiteturas de RNA para transferência de conhecimento em tarefa de classificação.

2 FUNDAMENTAÇÃO TEÓRICA E ANÁLISE BIBLIOGRÁFICA

Este capítulo apresenta a fundamentação teórica do presente trabalho de pesquisa, bem como a análise bibliográfica dos trabalhos sobre os quais ele apoia.

2.1 Fundamentação Teórica

Este trabalho é direcionado ao estudo de RNA aplicadas ao PLN. No entanto, modelos de RNA para PLN são consideravelmente complexos, de modo que, para que se possa melhor compreender o funcionamento de uma rede neural, faz-se oportuno explicar o funcionamento básico de uma RNA através de um exemplo mais simples. O exemplo usado aqui, para ilustrar tal funcionamento básico, é a aplicação de uma RNA para o reconhecimento de dígitos (de 0 a 9) escritos a mão, que é um processo suficientemente simples para ser explicado.

2.1.1 Classificação Automática

Algumas das aplicações de IA mencionadas na introdução deste trabalho, como a do óculos que ajuda uma pessoa cega a se guiar sozinha (BAI *et al.*, 2017), são do tipo Classificação Automática (CA). Trata-se da geração de um modelo capaz de classificar amostras de dados em categorias pré-determinadas. A classificação de um dígito escrito à mão entre 10 possibilidades (de 0 a 9), o reconhecimento de um objeto ou mais objetos contidos numa imagem (pessoa, gato, carro) e a categorização de texto como sendo de sentimento com polaridade *positiva* ou *negativa* são exemplos de problemas desse tipo.

O problema pode ser formalizado do seguinte modo:

- seja $x \in X$ a descrição de uma instância, em que X é o *espaço de instâncias*;
- e um conjunto C de categorias;
- deve-se determinar a categoria $c(x) \in C$ onde c é a função de categorização cujo domínio é X e a imagem é C.

O espaço de instâncias define uma maneira única de representar os objetos do domínio. Por exemplo, no caso de imagens, o espaço pode ser definido como um vetor de dimensão d, com valores indicando a presença dos *pixels*.

Um *exemplo de treinamento* é uma instância $x \in X$, cuja a categoria $c(x)$ é conhecida. A função de categorização c que produziu a categoria é desconhecida e o objetivo do problema é estimá-la. Dado um conjunto de exemplos de treinamento D, deve-se encontrar uma função de

categorização, conhecida como hipótese, h, tal que:

$$\forall [x, c(x)] \in D : h(x) = c(x) \tag{2.1}$$

O processo de encontrar a função h é conhecido como *treinamento*, em que a estimativa dessa função é feita com base nos exemplos de treinamento.

Na CAT, objeto deste trabalho, o programa recebe como entrada um texto, e é esperado que seja capaz de categorizar esse texto em determinada classe entre as classes pré-definidas (GARG; SESHADRI, 2017). Assim, cada $x \in X$ corresponde a uma sequência de termos $x_1, x_2, \ldots, x_n$ (vide Figura 1).

Figura 1 – Consideradas três classes de sentimentos, Ruim, Neutro e Positivo – respectivamente representados por 1, 2 e 3 – a RNA seria capaz de detectar o sentimento de uma dada frase. No caso da imagem, seria Positivo – ou 3.

Fonte: (KOPANAKIS, 2021)

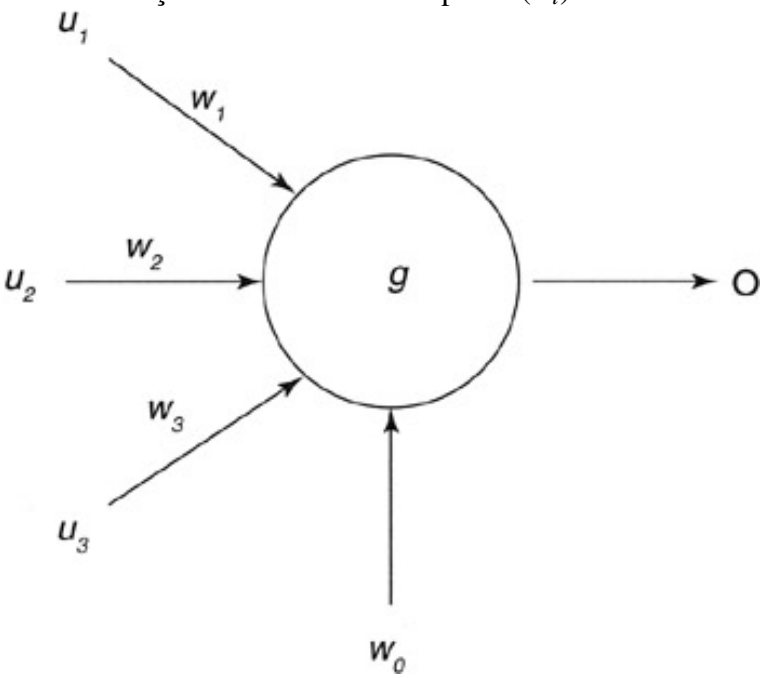
2.1.2 Redes Neurais Artificiais

O problema de reconhecimento (classificação) de dígitos escritos a mão, apesar de ser uma tarefa trivial se executada pela maioria das pessoas, é algo muito desafiador para ser resolvido de forma heurística[1] por um computador, pois tais abordagens requerem códigos bastante complexos, pela própria natureza do problema. Desta forma, há a exigência de códigos "extras" para muitos casos especiais e exceções. Mesmo assim, em geral, tais heurísticas costumam falhar quando utilizadas em casos para os quais não foram previamente desenhadas.

[1] Grosso modo, heurística é uma técnica baseada em conhecimento humano sobre dado problema e como resolvê-lo.

Por isso, a melhor abordagem é fazer com que o programa aprenda sozinho a lidar com o problema Nielsen (2015).

Figura 2 – Perceptron: às entradas (u_i) são feitas multiplicações escalares com os pesos (w_i).



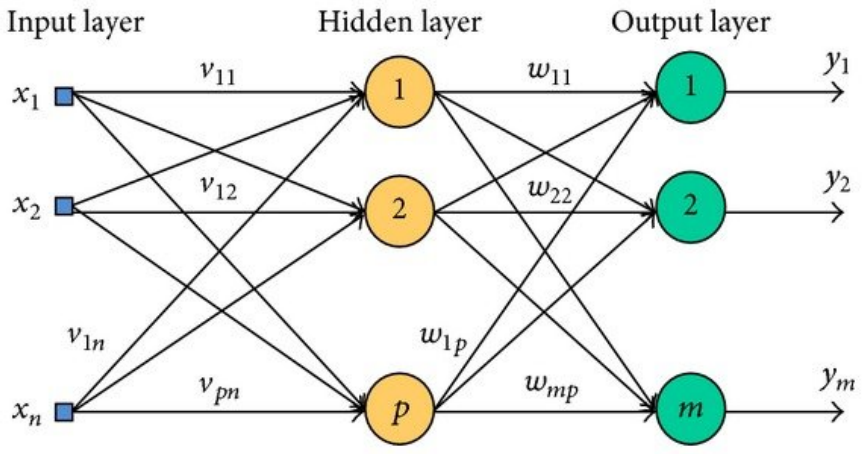
Fonte: (JONES; BY-KURATA, 1999).

Segundo (NIELSEN, 2015), o sistema visual do cérebro humano funciona como uma rede de camadas de neurônios, em que cada uma é especializada em fazer um ligeiro processamento de informação antes de repassá-la para a próxima, que também realizará outro ligeiro processamento sobre os dados e então os repassará para a próxima camada, e assim por diante. Inspirado neste princípio, Frank Rosenblatt, um psicólogo estadunidense, criou, ainda na década de 1950, um tipo de neurônio artificial chamado perceptron (Figura 2), uma abstração de um neurônio biológico que, em termos matemáticos, pode ser pensado como uma função que recebe como entrada uma determinada quantidade N de argumentos (x^1, x^2, x^3... x^N) e dá como saída 0 ou 1, que significam respectivamente uma resposta negativa e positiva do neurônio. No caso de reconhecimento de imagens, esses argumentos referem-se aos pixels – um ponto que compõe a imagem. Quando organizados em rede, esses neurônios podem realizar tarefas diversas, inclusive sendo possível criar redes neurais Turing-completas (SIEGELMANN; SONTAG, 1991).[2]

[2] Turing-completude significa que tal modelo é capaz de computar qualquer máquina de Turing, ou, colocando de outra forma, que é capaz de executar sequências de instruções do tipo condicional (i.e., se determinada condição é verdadeira, executar), ou jump (i.e., ir para uma dada instrução), por exemplo, baseado em valores armazenados em memória, memória esta que pode ser alocada e desalocada conforme a necessidade.

Assim como as redes neurais biológicas, uma RNA é constituída de camadas. As RNA mais simples costumam ter pelo menos 3 camadas de neurônios (vide Figura 4). A primeira camada é a chamada de “camada de entrada”, e não é realmente uma camada de neurônios perceptrons, mas sim é constituída de um simples vetor que descreve os dados a serem processados naquela instância. Por exemplo, no caso de uma imagem a ser processada em por uma RNA, esta camada de entrada seria na verdade um vetor cujos valores representam cada um dos *pixels* da imagem. A última camada da rede neural é chamada de “camada de saída”, que propriamente indica o resultado baseado no processamento das camadas anteriores, retornando valores numéricos, em que o neurônio com maior valor, ou *mais ativado*, indica a resposta. No caso de reconhecimento automático de dígitos de 0 a 9, essa camada de saída seria constituída de dez neurônios, cuja saída mais ativada, após o processamento da entrada, indica um dos dígitos. As demais camadas neurais, entre as camadas de entrada e de saída, ou seja, camadas internas, são chamadas de “camadas escondidas” pelo fato de os dados de treino não terem valores esperados para essas camadas. São essas camadas escondidas que fazem a maior parte do processamento de dados. A uma RNA com mais de uma camada escondida, é dado o nome de RNP (GOODFELLOW *et al.*, 2016).

Figura 3 – Rede Neural de Perceptrons com 3 camadas. Camada escondida em amarelo.



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

2.1.3 Representação de Conhecimento

Cada perceptron possui um vetor de pesos que representam numericamente o conhecimento daquele neurônio específico, e assim, sua capacidade de processar dados. Internamente,

o neurônio utiliza esse vetor de pesos e faz simples operações matemáticas de produto escalar por elemento, multiplicando o vetor de entrada pelo seu vetor de pesos, e, se o resultado desta operação for maior que um certo valor determinado (viés ou *bias*), o neurônio irá disparar um sinal, propagando o processamento dos dados, juntamente com os demais neurônios de sua camada, para a próxima camada (ROSENBLATT, 1958).

Há várias formas de se representar conhecimento em sistemas computacionais, mas no caso de Redes Neurais de Perceptrons, essa representação de conhecimento é feita através do uso de vetores de pesos – e possivelmente também vetores de vieses. Os pesos são os dados que permitem definir o comportamento do neurônio em relação aos dados recebidos como entrada. Viés é uma forma de chamar o limite numérico que representa o valor mínimo desejado para se obter uma saída positiva do neurônio, sendo positivo o valor 1 em oposição ao valor 0, como mencionado. Esses dois vetores, que contêm o conhecimento do neurônio, podem ter seus valores ajustados – inclusive manualmente – para se obter o resultado desejado (GERSHENSON, 2003).

Alterar os valores dos vetores de um neurônio perceptron individualmente pode ser problemático e provavelmente gerará consequências drásticas no funcionamento dos demais neurônios da mesma camada e camadas posteriores, graças à natureza binária desse tipo de neurônio, o que torna o ajuste dificultoso, e o resultado da rede imprevisível. Isso fica ainda mais evidente quando se tenta fazer com que uma rede de perceptrons aprenda automaticamente, onde os valores dos vetores são atualizados durante a execução do treino (NIELSEN, 2015).

Para treinamento adequado de tais redes, é necessário que as amostras na base de dados estejam rotuladas, indicando qual é a saída esperada para cada entrada (HARRINGTON, 2012). Por exemplo, no caso de treinamento para reconhecimento de dígitos escritos a mão, ao treinar a rede para reconhecer, digamos, um ’8’, a rede deve receber uma indicação de qual é o resultado esperado para treinamento da rede para aquela entrada. Ainda no caso de um "8"como entrada para a RNA, a rede deve ser alimentada também com indicador de representação para 8, por exemplo, o vetor [0,0,0,0,0,0,0,0,1,0], onde 1 na nona posição do vetor indica que a saída esperada da classificação da rede para aquela entrada determinada é um 8. A esse tipo de aprendizagem, conforme visto anteriormente, é dado o nome de Aprendizagem Supervisionada.

2.1.4 Neurônios Sigmoid

Não são redes de neurônios *perceptrons* que interessam ao objetivo de reconhecimento de números escritos a mão, e sim uma outra classe de neurônios artificiais chamada Neurônio Sigmoid (NS) (HOFFMAN; BENSON, 1986). Também não interessa a este trabalho a Turing-completude de tais sistemas, e sim uma outra peculiaridade dos NS, que é a sua capacidade de aprender à medida que são expostos a novos dados. No entanto, é importante compreender como funciona um perceptron, pois este ainda é a base para um NS.

Um NS difere de um Neurônio Perceptron (NP) principalmente em um aspecto: o primeiro reajusta seu vetor de pesos (e vieses) gradualmente através da aplicação de uma regressão linear, este que é o mecanismo central que calcula os valores a se utilizar para atualização dos vetores, efetivamente aumentando o conhecimento do neurônio e do modelo como um todo. As atualizações graduais permitem a superação da imprevisibilidade de comportamento da rede ao ter os valores de pesos de um neurônio atualizados, que é a limitação dos Neurônios Perceptron. Ainda há outras diferenças no funcionamento de NS: todos os valores de entrada, o valor de saída, e os valores dos vetores de pesos e vieses são valores reais (decimais) entre 0 e 1 ao invés de inteiros 0 ou 1. Essa diferença do tipo de números inteiros para números reais idealmente permite que se possa realizar mudanças graduais (em pesos e vieses) nos vetores sem que haja comportamentos adversos inesperados ou de consequências dramáticas, o que possibilita que seja realizado o treinamento automatizado do modelo.

Explicados a estrutura de uma RNA e o funcionamento de um NS, podemos focar no reconhecimento automático de dígitos escritos à mão. No lugar de uma rede neural de Perceptrons, teremos uma rede neural de Sigmoids. Resumidamente, o processo ocorre da seguinte forma: para cada um dos dígitos, sua imagem, a camada de entrada – vetor representando *pixels* da imagem – alimenta a primeira camada escondida, da qual cada Neurônio Sigmoid é especializado em detectar partes bem definidas das imagens. Por exemplo, um neurônio da camada escondida contém dados em suas tabelas de pesos e vieses, que permitem identificar que a imagem analisada possui um círculo na parte superior da imagem (talvez uma indicação de que o número é um 8, ou talvez um 9?), antes de repassar os dados, junto dos demais neurônios de sua camada para a camada seguinte, digamos, a camada de saída, que por sua vez detecta finalmente qual é aquele número (NIELSEN, 2015).

Esse tipo de rede neural é chamado de rede neural *feedforward* ou adiante, expressão em língua inglesa que, no presente contexto, significa que os neurônios disparados na camada

atual sempre alimentam a próxima camada da hierarquia, sem formar ciclos. Existem outras classes de redes neurais como por exemplo as Redes Neurais Recorrentes, que permitem ciclos (BEBIS; GEORGIOPOULOS, 1994).

A execução de uma RNA exige alguns parâmetros para seu funcionamento. Alguns desses parâmetros precisam ser informados antes da execução do processo de aprendizagem, e esses são chamados de *hiperparâmetros*, que podem ser ajustados para se conseguir um melhor resultado da rede neural. Diferentes modelos e abordagens de treinamento exigem hiperparâmetros distintos. Exemplos de hiperparâmetros são: quantidade de camadas, quantidade de neurônios de cada camada, taxa de aprendizagem, épocas de treino. Demais parâmetros são definidos pelo processo de aprendizagem (BEBIS; GEORGIOPOULOS, 1994).

2.1.5 Função Objetivo

Função objetivo, também chamada de função de custo, ou função de erro, é um método utilizado para avaliar quão bem treinada está uma variável em relação ao valor que é esperado Nielsen (2015). O ajuste dos parâmetros da RNA (vetores de pesos e vieses) é feito visando à minimização dessa função. Para resolver um problema de otimização – no caso presente, conseguir a melhor aproximação do resultado esperado – precisa-se minimizar o erro, o que é feito como parte do processo iterativo de atualização de pesos.

Durante o processo de treinamento, calcula-se a derivada parcial da função objetivo em relação às variáveis envolvidas (pesos e vieses). Com base nessas derivadas, tem-se o vetor gradiente, que indica a direção e o sentido que, a partir de um determinado ponto, deve-se caminhar para minimizar a função.

São exemplos de funções que podem ser utilizadas como função de custo: função de erro quadrático médio, minimax, função L, etc (BERGER, 2013).

Abaixo, exemplo de função de custo, onde W são as matrizes de pesos dos neurônios, b são os vetores de vieses, n é o número total de amostras de treino, x é cada exemplo de treino, $c(x)$ é a saída esperada do neurônio, e $h(x)$ é o vetor de ativações da saída da rede quando x é a entrada.

$$C(w,b) \equiv \frac{1}{2n} \sum_{x} \|c(x) - h(x)\|^2 \tag{2.2}$$

Essa expressão é a função de Erro Quadrático Médio de uma RNP.

A escolha de uma função de custo para um projeto deve ser feita com base no objetivo que se deseja alcançar, visto que cada função de custo diferente possui características, vantagens e desvantagens.

2.1.6 Treinamento

O processo de treinamento de um modelo de RNP é realizado através de um processo iterativo, geralmente implementada fazendo-se uso do método de gradiente descendente estocástico. Nesse algoritmo, os parâmetros da RNA são atualizados com os seguintes passos:

1. Apresenta-se uma amostra de exemplos de treinamento à RNA;
2. Calcula-se o erro com base na função objetivo, tendo em vista que a classe correta é conhecida nessa etapa;
3. Calcula-se a derivada parcial de cada parâmetro da rede, utilizando o valor do erro;
4. Com base no gradiente, atualiza-se os parâmetros no sentido contrário ao informado por esse vetor;
5. Volta-se ao passo 1 ou encerra-se o procedimento caso se atinja um critério de parada.

Com o resultado desse processo, a função irá convergir, é esperado que o modelo esteja treinado satisfatoriamente para o conjunto de exemplos considerados. Quase sempre inicializamos todos os pesos no modelo para valores extraídos aleatoriamente de uma distribuição gaussiana ou uniforme. A escolha da distribuição gaussiana ou uniforme não parece importar muito, mas não foi exaustivamente estudada. A escala da distribuição inicial, no entanto, tem um grande efeito tanto no resultado do procedimento de otimização quanto na capacidade de generalização da rede (GOODFELLOW, 2016).

Abaixo, tem-se o exemplo da saída de uma camada de neurônios, e ativação da camada seguinte, onde a variável l indica cada uma das camadas (NIELSEN, 2015).

$$a^l = \sigma(w^l a^{l-1} + b^l) \tag{2.3}$$

Esta expressão nos dá uma maneira global de pensar sobre como as ativações em uma camada se relacionam com ativações na camada anterior: apenas aplicamos a matriz de peso às ativações, e então adicionamos o vetor de vieses e finalmente aplicamos a função σ[3]. O erro

[3] σ (sigma) refere-se à função do NS.

da saída de um neurônio é dado pela expressão abaixo, onde $\sigma'(z_j^L)$ é a taxa de aprendizagem, L é o total de camadas da RNA, e j é um neurônio da camada:

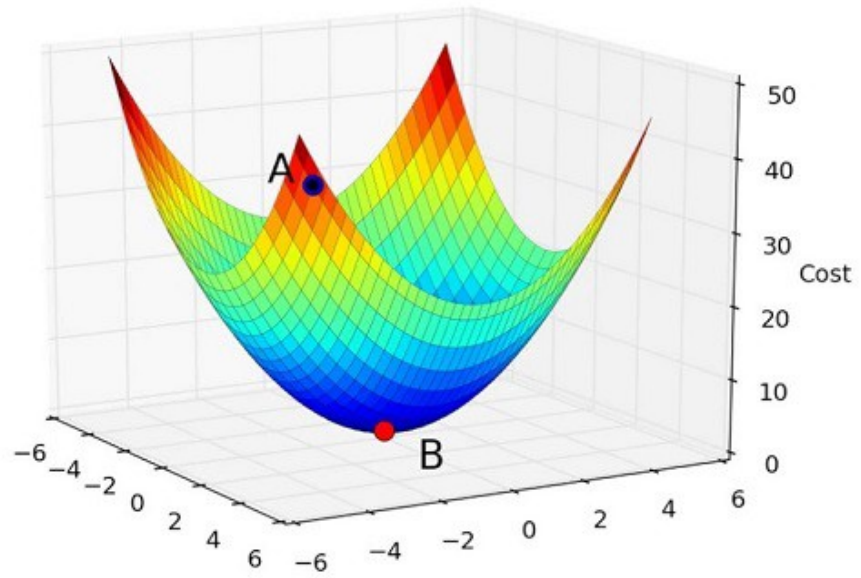
$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L) \tag{2.4}$$

O erro é usado no cálculo das derivadas parciais da função de custo pelos vetores de pesos e vieses, respectivamente:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \tag{2.5}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{2.6}$$

Figura 4 – Gradiente descendente estocástico. A é um ponto inicial, B é um ponto mínimo local.



Fonte: (IMRAM, 2017).

2.1.7 Transferência de Conhecimento

TC é um ramo da área de Aprendizagem Automática. Nessa abordagem, treina-se, inicialmente, o que se chama de *modelo de origem*, por meio de uma base de dados com quantidade significativa de exemplos. Posteriormente, utiliza-se o conhecimento gerado no pré-treino, ou seja, informações oriundas do *modelo de origem*, como entrada para a geração do *modelo alvo*, para o qual dispõe-se de poucos dados.

Para prática de transferência de conhecimento de um modelo de origem para um modelo alvo, primeiro é necessária a seleção correta da(s) base(s) de dados para o modelo de origem. É importante que os dados e a tarefa escolhida no pré-treino permitam gerar conhecimento que pode ser aproveitado. Após o pré-treino, deve-se aplicar os parâmetros gerados (ou parte deles), otimizados para a tarefa do modelo de origem, como ponto de partida no processo de geração do modelo para o problema alvo (vide Figura 5).

Após a TC, o modelo alvo é treinado aplicando-se o procedimento de aprendizagem convencional, como mencionado na Seção 2.1.6. Entretanto, os parâmetros reutilizados permitem que o modelo alvo seja capaz de reconhecer determinados padrões. É uma vantagem sobre o modelo completamente inicializado com valores aleatórios, que precisa de diversidade de dados para reconhecer padrões complexos.

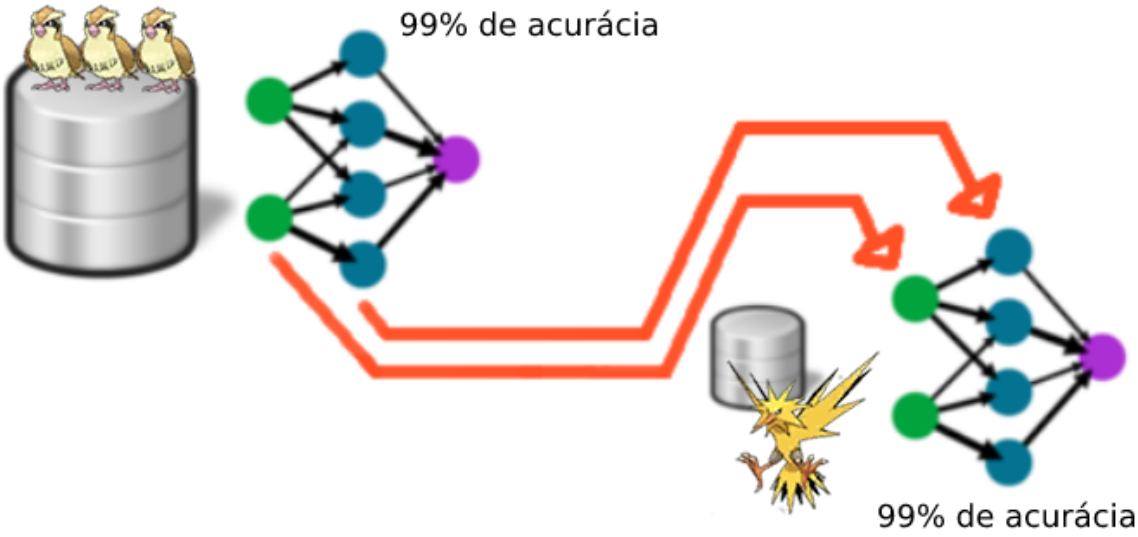
Na construção das arquiteturas de TC, o projetista pode definir quais parâmetros do modelo alvo serão reutilizados, assim como definir que valores podem ser evoluídos, até encontrar um modelo de transferência que lhe sirva melhor na dada situação. Normalmente, opta-se por reutilizar no modelo alvo as camadas próximas da entrada do modelo origem. Os dados presentes nessas camadas capturam padrões mais genéricos, podendo ser utilizados para uso em diferentes contextos, mesmo em tarefas pouco relacionadas. Como é de se esperar, quanto mais próximos os dados da(s) base(s) de origem e base alvo, em termos de contexto e vocabulário, no caso de problemas de PLN, maior tende a ser o ganho com a transferência de conhecimento (GARG; SESHADRI, 2017).

Exemplificando uma hipotética instância de TC à partir de uma base de dados treinada para tarefa diferente daquela em que o modelo de origem foi treinado originalmente, poderíamos querer treinar uma RNA para detecção de espécies de pokémons raros do tipo Voador, mas se as amostras de imagens desses pokémons estiverem disponíveis em pequena quantidade, poderíamos usar de TC à partir de um modelo treinado sobre uma base de dados com amplo número de exemplos de pokémons comuns do tipo Voador, uma vez que, sendo pokémons do mesmo tipo – Voador –, eles possuem características em comum – como bico, penas, asas –, que podem servir de base para o treinamento da RNA alvo da TC (vide Figura 5).

2.1.8 Arquitetura ELMo

Uma forma de realizar TC para RNA é através do uso de *word embeddings* extraídas à partir de um modelo de RNA pré-treinado em bancos de dados com grande quantidade de

Figura 5 – Transferência de Conhecimento entre RNAs para treinamento de modelo para detecção de pokémons raros (à direita) poderia usar como modelo de origem uma RNA treinada sobre imagens de pokémons comuns (à esquerda).



Fonte: (Autor, 2022)

amostras. Exemplos de *word embeddings* são as já mencionadas Word2Vec e GloVe. Uma técnica mais recente que essas é ELMo (PETERS *et al.*, 2018), uma representação de sentenças que considera o contexto das palavras que a constituem, ao contrário dos modelos mais antigos.

Embeddings ELMo são conseguidas usando uma Gated Recurring Unit (GRU) – tipo especial de Long short-term memory (LSTM) sem unidade de memória – bidirecional – Bidirectional Language Model (biLM) –, o que significa que este modelo gera relações entre as palavras da direita para a esquerda e da esquerda para a direita. O resultado pode ser usado para melhorar o desempenho de um modelo GRU pré-existente com poucas adaptações. O ELMo é uma combinação específica de tarefas das representações da camada intermediária no biLM (PETERS *et al.*, 2018).

Para fazer uso de *embeddings* ELMo, deve-se primeiramente implementar um método simples de combinações lineares, mostrado abaixo e descrito no artigo que introduz o modelo. O intuito deste método é extrair os dados das camadas do biLM e transformá-los nas *embeddings* propriamente ditas.

$$\text{y } \sum_{j=0}^{L} s_j h_j$$

Nesse método, L representa a quantidade de tarefas do modelo ELMo, s_j representa um vetor de pesos passível de aprendizado inicializado randomicamente que multiplica a representação h_j da camada biLM, e y representa um vetor de pesos passível de aprendizado inicializado randomicamente que permite que o modelo da tarefa escale para todo o vetor ELMo (PETERS *et al.*, 2018). A saída desse procedimento é o vetor ELMo referente à sentença de entrada, que então é processado normalmente pelo modelo alvo.

2.1.9 Arquitetura BERT

BERT (DEVLIN *et al.*, 2018) é um modelo de representação de linguagem da empresa Google. Ele usa duas etapas, pré-treinamento e ajuste fino, para criar modelos de última geração para uma ampla gama de tarefas. A arquitetura BERT consiste em vários codificadores Transformer empilhados juntos. Cada codificador Transformer encapsula duas subcamadas: uma camada de autoatenção e uma camada de alimentação direta (WINASTWAN, 2021).

Há pelo menos duas razões pelas quais o BERT é um modelo de linguagem poderoso: Ele é pré-treinado em dados não rotulados extraídos de grandes corpus que dispõem de milhões de exemplos. Como o nome sugere, ele é pré-treinado utilizando a natureza bidirecional das pilhas de codificador. Isso significa que BERT aprende informações de uma sequência de palavras não apenas da esquerda para a direita, mas também da direita para a esquerda, de forma semelhante ao modelo biLM (WINASTWAN, 2021).

Existem duas etapas na arquitetura BERT: pré-treinamento e ajuste fino. Durante o pré-treinamento, o modelo é treinado em dados não rotulados em diferentes tarefas de pré-treinamento através de métodos de aprendizagem não-supervisionada. Para ajuste fino, o modelo BERT é inicializado primeiro com os parâmetros pré-treinados e todos os parâmetros são ajustados usando dados rotulados das tarefas *downstream*. Cada tarefa de downstream tem modelos ajustados separados, mesmo que sejam inicializados com os mesmos parâmetros pré-treinados (DEVLIN *et al.*, 2018).

BERT é pré-treinado usando duas tarefas não supervisionadas: *Masked Language Modeling (MLM)*, que mascara uma porcentagem dos *tokens* de entrada aleatoriamente e, em

seguida, prevê esses *tokens* mascarados. Muitas tarefas importantes de downstream, como Resposta a Perguntas e Inferência de Linguagem Natural são baseadas na compreensão da relação entre duas sentenças, que não é capturada diretamente pela modelagem de linguagem. A fim de treinar um modelo que entenda as relações de sentença, BERT é pré-treinado para uma tarefa binarizada de previsão da próxima sentença que pode ser gerada trivialmente a partir de qualquer corpus monolíngue (DEVLIN *et al.*, 2018).

2.2 Análise Bibliográfica

Os trabalhos apresentados nesta seção abordam detalhadamente diferentes aspectos do funcionamento de Redes Neurais Artificiais, incluindo aspectos conceituais, estruturais, lógicos e matemáticos, bem como de metodologias de Transferência de Conhecimento.

2.2.1 Transfer Learning: The Impact of Test Set Word Vectors, with Applications to Political Tweets

O trabalho de Garg e Seshadri (2017) trata-se uma de pesquisa comparativa entre métodos de transferência de conhecimento de conjuntos de dados clássicos em PLN para o problema de classificar *tweets*. Há duas tarefas alvo, ambas sobre *tweets* de políticos do congresso norte-americano: a primeira corresponde à análise de sentimento (classificação do texto como positivo ou negativo) e a segunda é sobre o caráter subjetivo ou objetivo do texto.

Para o treinamento satisfatório de redes neurais profundas em PLN, em geral são necessários conjuntos de dados com dezenas de milhares de textos. No caso desse conjunto de dados, há apenas 1300 exemplos rotulados, de um universo de 400 mil coletados. Por outro lado, dispõe-se de bases de dados volumosas sobre domínios diferentes, que são usadas como pré-treino.

Os *tweets* automaticamente classificados como contendo sentimento totalizaram cerca de 10000 (dez mil) *tweets*, 10 porcento da quantidade total. Desse modo, o autor afirma ser possível a aplicação dos dados do treinamento baseado nas grandes bases – que contém enorme quantidade de dados rotulados com sentimentos – para melhorar o desempenho do aprendizado automático dos *tweets*.

Os resultados obtidos com o experimento não alcançaram o desempenho dos métodos de estado-da-arte usados para aprendizagem automática, no entanto, o autor julgou que seus

resultados ainda alcançaram ótimos índices, considerando o fato de que os dados de resultados existentes usados para comparação eram todos para avaliação de treinamento com dados de domínio próprio (*in-domain*), enquanto o experimento foi feito para domínio externo (out-of-domain).

2.2.2 Convolutional Neural Networks for Sentence Classification

No trabalho de (KIM, 2014), treinam-se algumas variações de uma simples RNA com apenas uma camada convolucional sobre vetores pré-treinados usando algoritmo Word2Vec (CHURCH, 2017), para CAT. Os autores também apresentam um modelo de RNA com dois canais de entrada, isto é, dois vetores de entrada, sendo um canal estático, isto é, que não é auto-ajustado, e o outro canal dinâmico, que permite o auto-ajuste.

No início do artigo, os autores explicam o funcionamento de Redes Neurais Convolucionais (RNC), e depois mostram a diferença em relação a este modelo de dois canais, diferença esta que está apenas no processamento das convoluções. No caso de uma RNA de canal único, faz-se uma aplicação simples dos filtros em cada vetor, e no caso da RNA de dois canais, faz-se isso para os dois canais e une-se os resultados das convoluções. Este método permite analisar comparativamente um processo de treinamento em que os vetores de entrada são auto-ajustados e quando os mesmos não são auto-ajustados.

O autor afirma que os modelos utilizados nos seus experimentos podem ter seus hiperparâmetros levemente ajustados para obter performance de estado da arte. Uma seção do artigo fala sobre regularização, que é uma forma de evitar sobre-ajuste (*overfitting*), que é quando a rede memoriza os dados com conjunto de treinamento e não consegue generalizar para novos dados. Para este fim, de regularização, o autor aplica uma operação de produto escalar entre o vetor de pesos de um neurônio da penúltima camada e um vetor de distribuição de Bernoulli, o que é feito no processo de treinamento.

No caso do experimento com o modelo de dois canais, sendo um dinâmico, com a entrada inicializada com Word2Vec, o autor compara os resultado dos canais após o treinamento do modelo da RNA. Apesar de poucos ajustes de hiperparâmetros, este modelo simples alcançou excelentes resultados em vários *benchmarks*, sugerindo que os vetores pré-treinados são extratores de recursos ‘universais’ que podem ser utilizados para várias tarefas de classificação.

2.2.3 Deep Contextualized Word Representations

Os autores apresentam um tipo de representação de palavra profundamente contextualizada, que modela características complexas de uso de palavra e como esses usos de palavras variam entre contextos linguísticos. Nesse modelo, os vetores de palavras são funções de estados internos de um modelo de linguagem bidirecional (biLM), treinado em um grande corpus. O trabalho mostra que tais representações podem ser facilmente adicionadas a modelos existentes e significantemente melhorar sua performance em diferentes tipos de tarefas, como análise de sentimentos, entre outros, sendo tal melhoria comparável à conseguida através de outros métodos de word embeddings como Word2Vec (CHURCH, 2017) ou GloVe (PENNINGTON *et al.*, 2014).

Diferente dos *word embeddings* tradicionais, no ELMo a cada token é atribuída uma representação que é uma função de toda a sentença de entrada. ELMo aprende uma combinação linear de vetores empilhados sobre cada palavra de entrada para cada tarefa final, o que melhora a performance comparado a usar apenas a camada LSTM superior. Combinar os estados desta forma permite conseguir representações de palavras bastante ricas. Usando avaliações intrínsecas, os autores mostraram que os estados superiores da LSTM capturam aspectos dependentes de contexto do significado de palavras, enquanto estados inferiores modelam aspectos sintáticos.

Através de ablações e outros experimentos, os autores também confirmam que as camadas biLM codificam com eficiência diferentes tipos de informações sintáticas e semânticas sobre o contexto, e que o uso de todas as camadas melhora o desempenho na maioria dos tipos de tarefas, com diferentes níveis de melhora a depender do tipo de tarefa.

Para tarefas de detecção de sentimentos, os autores afirmam que para realizar a transferência de conhecimento, extrair as *embeddings* ELMo das camadas biLM e utilizá-las como entrada para um modelo GRU pré-existente é suficiente para se obter uma melhora.

2.2.4 BERT: Pre-training of deep bidirectional transformers for language understanding

No trabalho de (DEVLIN *et al.*, 2018), os autores argumentam que as técnicas atuais para aplicar representações de linguagem pré-treinadas a tarefas de *downstream*[4], seja com base em recursos - como ELMo - e ajuste fino - como o *Transformer* pré-treinado generativo OpenAI GPT [5] (RADFORD *et al.*, 2019) -, restringem o poder das representações pré-treinadas,

[4] Uma tarefa de downstream é aquela que é o objetivo do treinamento
[5] <https://openai.com/>. Acessado em 02/07/2022

especialmente para as abordagens de ajuste fino. A principal limitação é que os modelos de linguagem padrão são unidirecionais, e isso limita a escolha de arquiteturas que podem ser usadas durante o pré-treinamento. Por exemplo, no OpenAI GPT, os autores utilizam uma arquitetura da esquerda para a direita, onde cada token só pode atender aos tokens anteriores nas camadas de autoatenção do *Transformer*. Essas restrições são sub-ótimas para tarefas em nível de sentença e podem ser muito prejudiciais ao aplicar abordagens baseadas em ajuste fino para tarefas em nível de token, como resposta a perguntas, onde é crucial incorporar contexto de ambas as direções.

Os autores melhoram as abordagens baseadas em ajuste fino propondo BERT: Bidirectional Encoder Representations from Transformers. BERT alivia a restrição de unidirecionalidade mencionada anteriormente usando um objetivo de pré-treinamento de "modelo de linguagem mascarado" (MLM), inspirado na tarefa Cloze (TAYLOR, 1953). O modelo de linguagem mascarada mascara aleatoriamente alguns dos *tokens* da entrada e o objetivo é prever o identificador do vocábulo original da máscara com base apenas em seu contexto. Ao contrário do pré-treinamento do modelo de linguagem da esquerda para a direita, o objetivo do MLM permite que a representação fusione o contexto esquerdo e direito, o que nos permite pré-treinar um transformador bidirecional profundo. Além do modelo de linguagem mascarada, também usaram uma tarefa de "previsão da próxima frase" que pré-treina conjuntamente representações de pares de texto.

Neste capítulo foram apresentadas a fundamentação teórica do presente trabalho, e revisões bibliográficas de artigos nos quais ele é baseado. A seguir é apresentada a metodologia seguida no desenvolvimento da pesquisa.

3 METODOLOGIA E DESENVOLVIMENTO

Este capítulo apresenta a metodologia seguida nesta pesquisa, desde a definição das bases de dados a serem utilizadas, até a implementação dos modelos de RNA, e análise das métricas.

O primeiro passo para o treinamento de um modelo de RNA com alta precisão foi a seleção adequada de um conjunto de bases de dados. Como o objetivo deste trabalho é de transferência de aprendizagem, deve-se dividi-las em bases de dados para modelos origem e para modelos alvo. A escolha das bases de dados deve ser determinada de acordo com certas restrições: bases de dados com grandes quantidades de dados são candidatas a bases de dados de modelo origem.

Um bom mecanismo para escolha entre as bases de dados candidatas para os modelos origem é a seleção com base na tarefa de classificação, que pode utilizar amostras acompanhadas de rótulos referentes a parâmetros de saída que deem indicação sobre positividade/neutralidade/-negatividade, ou objetividade/subjetividade etc. Também deve-se considerar na escolha da base de dados do modelo origem a semelhança entre os vocabulários das bases de dados de modelos origem e das bases de dados de modelos alvo.

Este trabalho aplica TC para CAT em português. Como AS é uma tarefa bastante comum em experimentos realizados em língua inglesa, optou-se por trabalhar nessa direção. Para tanto, buscou-se bases em português para permitir a realização do trabalho. Após extensa busca, não encontrou-se um número significativo de bases de dados para PLN este idioma disponíveis. Para a tarefa escolhida, encontrou-se o Corpus Resenhas de Livros (ReLi) (FREITAS *et al.*, 2012) [1], com 1600 resenhas de livros anotadas manualmente.

O Corpus ReLi contém um total de 259978 palavras e 12470 frases, das quais 3691 foram anotadas manualmente. Quanto à distribuição de polaridade de sentimento, há 2883 sentenças positivas, 596 negativas e 212 neutras. Os comentários foram retirados da rede social Skoob [2] e versam sobre 13 livros. O Corpus ReLi, além de altamente desbalanceado, possui um número significativo de exemplos cujo rótulo é ambíguo.

Os experimentos foram realizados utilizando duas arquiteturas bi-direcionais diferentes para comparação: ELMo e BERT. Para realizar a transferência de conhecimento, utilizamos como origem do conhecimento *embeddings* ELMo extraídos de um modelo GRU bidirecional

[1] <https://www.linguateca.pt/Repositorio/ReLi>. Acessado em 23/05/2022
[2] <https://www.skoob.com.br>. Acessado em 23/05/2022

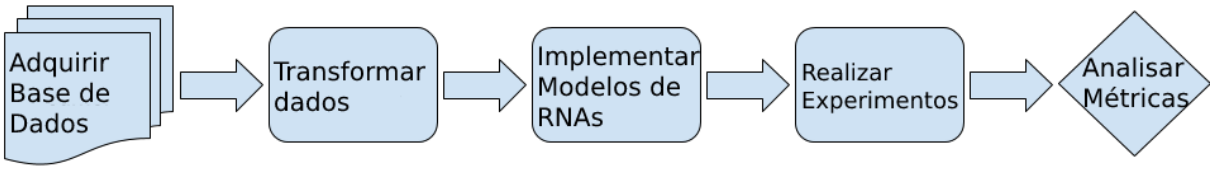
fornecido pela biblioteca para PLN AllenNLP, e treinado em uma base de dados contendo aproximadamente 1048500 sentenças e 2386973 palavras, originárias de fontes diversas de várias áreas de conhecimento, incluindo jornais, revistas e livros.

Para garantir boa performance, também é de crucial importância a determinação de hiperparâmetros adequados. Os hiperparâmetros utilizados para modelos de RNA para classificação de textos em conjuntos de dados diversos em outro idioma não necessariamente são úteis para modelos de RNA para a língua portuguesa, devido às diferenças estruturais e sintáticas entre as línguas.

3.1 Metodologia

Para a realização deste trabalho, foram planejadas e efetuadas as seguintes atividades:

1. Adquirir as bases de dados para o modelo base e alvo da TC
2. Transformar a base de dados original para o formato adequado
3. Selecionar os modelos para CAT e TC
4. Implementar os modelos para CAT e TC
5. Realizar experimentos de classificação com os modelos de RNA, com e sem TC
6. Analisar e comparar as métricas resultantes dos experimentos



3.2 Desenvolvimento

O desenvolvimento foi iniciado com a aquisição e preparação da base de dados do modelo alvo da transferência de conhecimento a partir do corpus ReLi, do qual foram utilizados os dados referentes às frases e polaridades, estes últimos que se constituem em 3 classes: positiva, neutra e negativa. O corpus ReLi ainda inclui dados referentes às notas atribuídas às resenhas de livros, mas tais dados foram ignorados nesta pesquisa.

A preparação dos dados consistiu em converter o formato da base de dados ReLi original para o fim de alimentar redes neurais, e exigiu a criação de um algoritmo para isso visto que não foi possível encontrar um algoritmo pronto. A biblioteca Pandas [3] disponível na linguagem Python foi a escolha natural para auxiliar em tal tarefa. O resultado da execução do algoritmo foi duas bases de dados com quantidades de amostras proporcionais para cada classe, sendo uma base de dados com 80% das amostras para treino, e o outra base de dados com 20% das amostras para testes.

O corpus ReLi é bastante desbalanceado, como ocorre com a maioria dos bancos de dados usados em aplicações de RNA no mundo real. Esse fator torna o modelo bastante enviesado após o treinamento, pois, nesses casos, a forma mais fácil de o modelo conseguir uma boa performance é ele considerar os exemplos como pertencendo a classe com maior número de amostras (KUBAT *et al.*, 1997). Para contornar essa limitação do banco ReLi utilizou-se a técnica de *oversampling* (KUBAT *et al.*, 1997) para reforçar a aprendizagem das classes com poucos exemplos: classes negativa e positiva. A técnica de *oversampling* para balanceamento de dados está entre as mais simples de se implementar, e consiste basicamente em replicar as amostras de classes pouco representadas até que o total de amostras daquela classe fique com uma quantidade próxima da classe com maior quantidade de exemplos. Outra limitação do banco de dados ReLi é que muitas de suas sentenças são dúbias em relação ao sentimento, potencialmente devido a incertezas das pessoas envolvidas no processo de rotulação manual dos exemplos ao avaliá-los.

Para desenvolvimento dos códigos de treinamento e testes da RNA, optou-se por utilização da biblioteca PyTorch [4] para Aprendizagem Profunda, escrita na linguagem de programação Python, pois, comparada à maioria das outras bibliotecas para tal finalidade, ela fornece níveis razoáveis de abstração e flexibilidade.

A primeira dentre as redes neurais implementadas na pesquisa foi uma GRU (CHO *et al.*, 2014), que é capaz de determinar matematicamente uma relação entre as palavras de uma frase ou documento antes de passar os dados para uma RNA. Esse é um modelo simples, que performa bem, e pode ser treinado rapidamente comparado a outros modelos de redes neurais. Este modelo foi utilizado como controle para comparação com a performance do modelo melhorado com transferência de conhecimento.

Os experimentos com o modelo GRU foi feito na minha máquina pessoal, com uma

[3] <https://pandas.pydata.org/>. Acessado em 24/05/2022
[4] <https://pytorch.org/>. Acessado em 24/05/2022

placa gráfica Nvidia Geforce GTX 1050Ti de 4GB de RAM, e o processo tomava cerca de 7 horas, executando validação de dez dobras.

A segunda RNA implementada foi feita no intuito de aplicar a transferência de conhecimento. O modelo de RNA utilizado como origem da transferência de conhecimento foi um modelo ELMo, disponibilizado publicamente pela biblioteca AllenNLP (PETERS *et al.*, 2018) [5], treinado sobre uma base dados contendo uma enorme quantidade de textos de língua portuguesa de diversas áreas de conhecimento e origens, do qual extraímos word-embeddings ELMo. O modelo de aprendizagem alvo da transferência de aprendizagem das *embeddings* ELMo pode ser criado a partir de um modelo GRU pré-existente, e essa foi a abordagem utilizada neste trabalho.

O ELMo é uma representação contextual profunda de palavras que modela características complexas do uso de palavras (por exemplo, sintaxe e semântica) e como esses usos variam em contextos linguísticos (ou seja, para modelar polissemia). Esses vetores de palavras são funções aprendidas dos estados internos de um modelo de linguagem bidirecional profunda biLM, que é pré-treinado em um corpus de texto grande. Eles podem ser facilmente adicionados aos modelos existentes e melhorar significativamente o estado da arte em uma ampla gama de problemas desafiadores da PLN, incluindo resposta a perguntas, vinculação textual e análise de sentimentos (PETERS *et al.*, 2018).

ELMo já está ultrapassada, e de acordo com o artigo original dos autores desse modelo, os resultados de melhoria de performance resultante da transferência de conhecimento de *embeddings* ELMo são comparáveis à transferência utilizando outros tipos de *word embeddings* simples.[6]

Os experimentos com o modelo ELMo foram feito na minha máquina pessoal, com uma placa gráfica Nvidia Geforce GTX 1050Ti de 4GB de RAM, e o processo tomava cerca de 11 horas, executando validação cruzada de dez dobras (*ten-fold cross-validation*).

A terceira RNA implementada foi feita no intuito de aplicar a transferência de conhecimento usando uma arquitetura BERT, mais avançada que GRU. O modelo BERT é estado da arte para transferência de conhecimento para tarefas de PLN, comparável a seu concorrente GPT-2, desenvolvido pela empresa OpenAI.

O modelo pré-treinado usado como modelo de origem para a arquitetura BERT foi BERTimbau (SOUZA *et al.*, 2020) *large*, que foi obtido a partir de um corpus de $2,7$ bilhões

[5] <https://allenai.org/allennlp/software/allennlp-library>. Acessado em 25/05/2022

[6] como Word2Vec ou GloVe

Hiperparâmetro	Valores testados
Tamanho de *embeddings*	50, 75, 100, 200, 300, 400, 500 1000
Neurônios nas camadas escondidas	50, 75, 100, 120, 150, 200, 250, 300, 400, 500
Camadas escondidas	1, 2, 3
Taxa de aprendizagem	$3e-6$, $4e-6$, $5e-6$, $6e-6$, $7e-6$
Dropout	0, 0.1, 0.2, 0.3
Paciência	3
Semente	5, 6

Tabela 1 – Hiperparâmetros utilizados nos experimentos com GRU.

de instâncias de palavras BrWaC (FILHO *et al.*, 2018), e pode ser encontrada no repositório Huggingface [7].

Devido a BERT ser um modelo cujo treinamento é muito mais custoso que o dos modelos anteriores, treinar na minha própria máquina seria proibitivo em relação ao tempo de processamento, e portanto, utilizou-se a plataforma Google Colab com uma placa gráfica Nvidia T4 de 16GB para fazer o ajuste fino (*fine-tunning*). O processo de cada instância de treino durava cerca de 3 horas, sem validação cruzada de dez dobras.

Para garantir a validade dos resultados dos modelos GRU testados, utilizou-se a técnica de validação cruzada de dez dobras. O mesmo não foi feito com o modelo BERT, devido ao tempo de treinamento proibitivo do modelo, mas também devido a esse modelo dispensar esse tipo de validação, visto que é uma arquitetura para modelos pré-treinados.

Ainda, sabe-se que a acurácia não é uma boa métrica para medir a performance de modelos de RNA com bases de dados desbalanceados (KUBAT *et al.*, 1997), portanto, utilizou-se a técnica de matriz de confusão para avaliação dos resultados obtidos nos testes, bem como algumas métricas derivadas a partir dela: precisão e *f-measure*.

Para o modelo GRU sem TC foram realizados testes variando os hiperparâmetros: tamanho de *embeddings*, quantidade de neurônios nas camadas escondidas, quantidade de camadas escondidas, e *dropout*. Os valores testados nos experimentos com GRU podem ser conferidos na Tabela 1.

Para o modelo GRU com TC foram realizados testes variando os hiperparâmetros de quantidade de neurônios nas camadas escondidas, quantidade de camadas escondidas, e *dropout*.

[7] <https://huggingface.co/liaad/ud_srl-pt_bertimbau-large>. Acessado em 24/05/2022

Hiperparâmetro	Valores testados
Tamanho de *embeddings*	1024
Neurônios nas camadas escondidas	25, 30, 35, 40, 50, 60, 70, 80, 90, 100, 120, 150, 200
Camadas escondidas	1, 2, 3
Taxa de aprendizagem	$3e-6$, $4e-6$, $5e-6$, $6e-6$, $7e-6$
Dropout	0, 0.1, 0.2, 0.3
Paciência	3
Semente	5, 6

Tabela 2 – Hiperparâmetros utilizados nos experimentos com ELMo.

Hiperparâmetro	Valores testados
Tamanho de *embeddings*	1024
Taxa de aprendizagem	$3e-6$, $4e-6$, $5e-6$, $6e-6$, $7e-6$
Dropout	0.1, 0.2, 0.3, 0.4, 0.5, 0.6
Paciência	3
Semente	1234, 4321

Tabela 3 – Hiperparâmetros utilizados nos experimentos com BERT.

A dimensão das *embeddings* foi determinada pela dimensão das *embeddings* ELMo, que eram de tamanho 1024. Os valores testados nos experimentos com ELMo podem ser conferidos na Tabela 2.

Para o modelo BERT foram realizados testes variando os hiperparâmetros de taxa de aprendizagem e *dropout*. A dimensão usada foi determinada pela dimensão das *embeddings* BERT usado como modelo de origem, que eram de tamanho 1024. Os valores testados nos experimentos com ELMo podem ser conferidos na Tabela 3.

Neste capítulo, foram apresentadas a metodologia e o desenvolvimento seguidos neste trabalho. A seguir são apresentados os resultados obtidos.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos apos a realização dos experimentos propostos, incluindo resultados e métricas de experimentos sem utilização de transferência de conhecimento, e com utilização de transferência de conhecimento.

Como já mencionado no capítulo de Metodologia, visando à validação dos resultados obtidos nos experimentos realizados, utilizou-se técnica de validação cruzada de dez dobras (ou *ten-fold cross-validation*).

Para os experimentos realizados sem transferência de conhecimento usando GRU, a melhor configuração de hiperparâmetros para a GRU que se conseguiu com o banco de dados ReLi é com tamanho de *embeddings* 128, quantidade de neurônios nas camadas escondidas 64, quantidade de camadas escondidas 2, taxa de aprendizagem 0.000005, e *dropout* 0.3. Usando tal configuração, conseguiu-se chegar às seguintes métricas como resultado da validação cruzada de dez dobras: acurácia 0.770400, precisão 0.762313 e f-*measure* 0.765974; com a matriz de confusão na Tabela 4.

Para os experimentos realizados com o modelo GRU e transferência de conhecimento à partir das *embeddings* ELMo, a melhor configuração de hiperparâmetros obtida para o banco de dados ReLi foi com os hiperparâmetros tamanho de *embeddings* 1024, quantidade de neurônios nas camadas escondidas 10, quantidade de camadas escondidas 1, taxa de aprendizagem $5e-6$, e *dropout* 0.1. Usando tal configuração, conseguiu-se chegar às seguintes métricas como resultado da validação cruzada de dez dobras: acurácia 0.782800, precisão 0.794947 e f-*measure* 0.787308; com a matriz de confusão na Tabela 5.

		Rótulo predito		
		Negativo	*Neutro*	*Positivo*
Rótulo verdadeiro	*Negativo*	29	67	22
	Neutro	40	1496	183
	Positivo	23	228	317

Tabela 4 – Matriz de confusão resultante de operação de validação cruzada de dez dobras do modelo GRU com a seguinte configuração de hiperparâmetros: tamanho de *embeddings* 128, quantidade de neurônios nas camadas escondidas 64, quantidade de camadas escondidas 2, taxa de aprendizagem $6e-6$, e dropout 0.3. Métricas resultantes: acurácia 0.770400, precisão 0.762313, e f-*measure* 0.765974.

		Rótulo predito		
		Negativo	*Neutro*	*Positivo*
Rótulo verdadeiro	*Negativo*	40	52	26
	Neutro	54	1437	229
	Positivo	21	146	400

Tabela 5 – Matriz de confusão resultante de operação de validação cruzada de dez dobras do modelo GRU + ELMo com a seguinte configuração de hiperparâmetros: tamanho de *embeddings* 1024, quantidade de neurônios nas camadas escondidas 16, taxa de aprendizagem $5e-6$, quantidade de camadas escondidas 2, e dropout 0.3. Métricas resultantes: acurácia 0.782800, precisão 0.774947, e f-*measure* 0.777308.

		Rótulo predito		
		Negativo	*Neutro*	*Positivo*
Rótulo verdadeiro	*Negativo*	67	39	11
	Neutro	19	1645	155
	Positivo	2	116	443

Tabela 6 – Matriz de confusão resultante validação do processo de treino, com dados de teste sobre o modelo BERT treinado sobre os dados de treino usando os seguintes hiperparâmetros: taxa de aprendizagem $6e-6$ e *dropout* 0.5 e semente 1234. Métricas resultantes: acurácia 0.863036, precisão 864849, e f-*measure* 0.863012.

Para os experimentos realizados com o modelo BERT, a melhor configuração de hiperparâmetros obtida para o banco de dados ReLi foi com os hiperparâmetros, taxa de aprendizagem $6e-6$, *dropout* 0.1, semente 1234 . Usando tal configuração, conseguiu-se chegar às seguintes métricas como resultado: acurácia 0.863036, precisão 0.864849 e f-*measure* 0.863012; com a matriz de confusão na Tabela 6.

A Tabela 7 mostra um comparativo entre as métricas obtidas nos experimentos com modelo GRU sem uso de TC e com uso de TC com ELMo, e através do modelo BERT. Podemos notar que a TC à partir de um modelo BERT resulta em uma melhora total de 9.26%, e 8% a mais de acurácia em relação ao modelo GRU com TC usando *embeddings* ELMo. Também podemos observar (Tabela 7 e Figuras 6, 7 e 8) que o modelo treinado com a arquitetura BERT supera os outros dois modelos em todas as métricas analisadas - precisão e f-*measure*.

Os experimentos realizados no presente trabalho usando ELMo obtiveram uma

	GRU	GRU + ELMo	BERT
Acurácia	0.770400	0.782800	0.863036
Precisão	0.762313	0.774947	0.864849
f-*measure*	0.765974	0.777308	0.863012

Tabela 7 – Comparativo das métricas obtidas nos experimentos realizados.

melhoria superior à obtida nos experimentos do artigo que introduz as *embbedings* ELMo, estes realizados na língua inglesa. O que pode ser indicativo de que técnicas de TC podem gerar melhores resultados quando aplicados em modelos de RNA treinados sobre textos em língua portuguesa que modelos de RNA treinados sobre textos em língua inglesa.

Figura 6 – Acurácia dos modelos

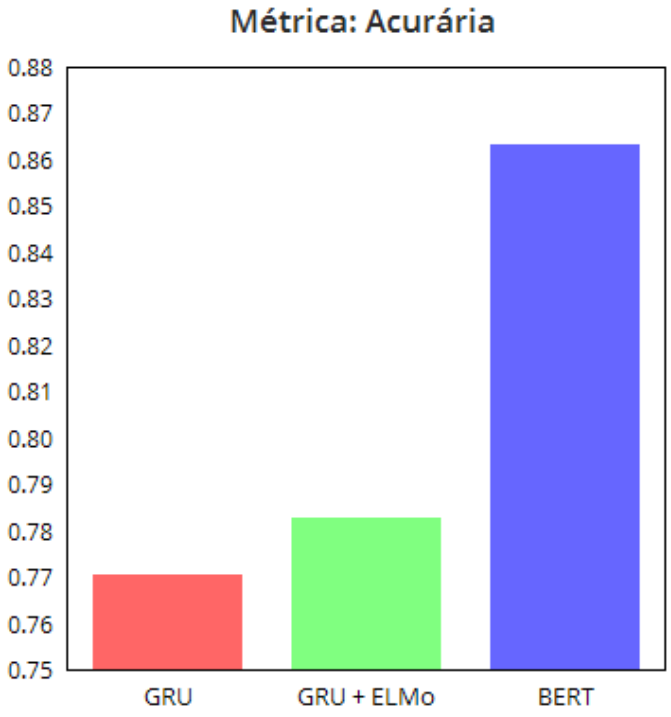


Figura 7 – Precisão dos modelos

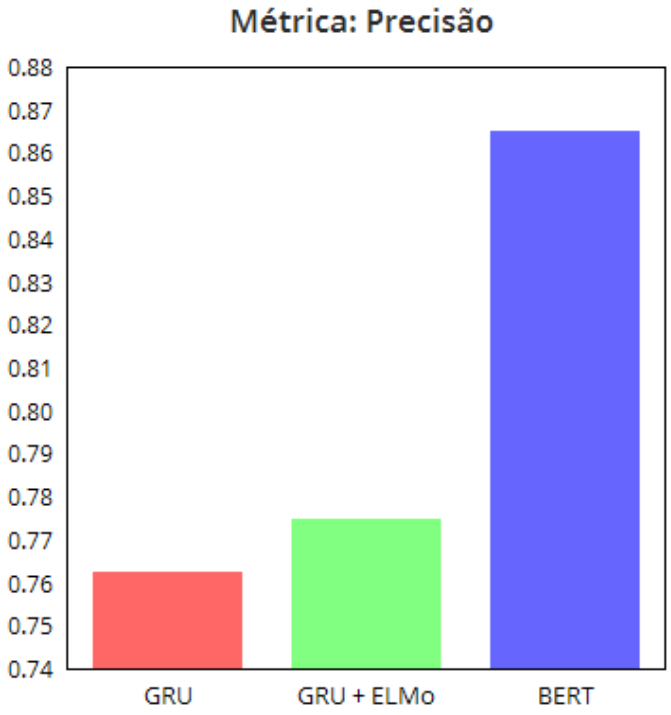
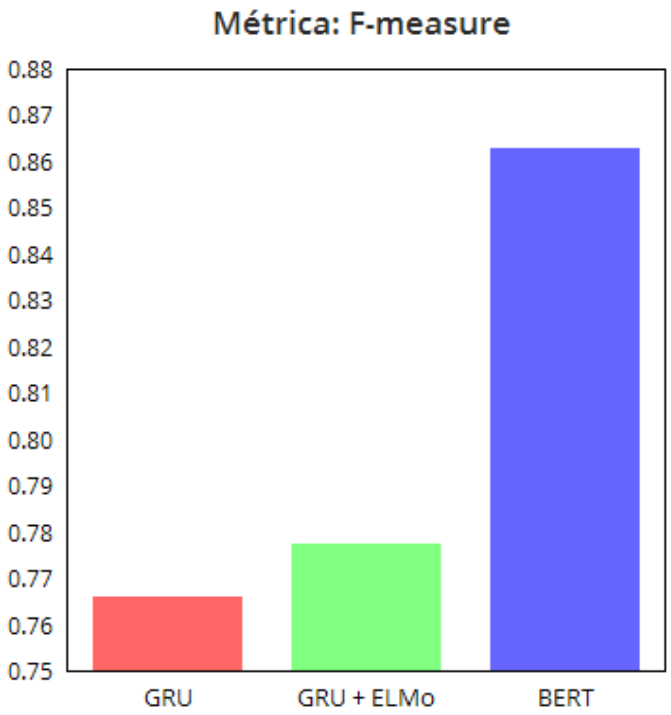


Figura 8 – f-*measure* dos modelos



5 CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo apresenta as conclusões do presente trabalho de pesquisa, bem como possíveis futuros trabalhos.

Com esta pesquisa, conclui-se que é possível tirar proveito de técnica de transferência de conhecimento entre RNA treinadas sobre textos na língua portuguesa utilizando como fonte da transferência *embeddings* ELMo extraídas a partir de um modelo de RNA biLM, mesmo utilizando bases de dados de domínios diferentes para os modelos fonte e alvo.

Nos experimentos com BERT, conseguimos uma melhora de mais de 8% em relação ao modelo GRU com ELMo, e 9.26% em relação ao modelo GRU sem TC, mas não foi possível alcançar resultados de estado-da-arte, possivelmente devido a problemas inerentes do banco de dados ReLi como exemplos ambíguos, o acentuado desbalanceamento de representação das classes dos exemplos, além do fato de o modelo BERT utilizado como ponto de partida ter sido pré-treinado em tarefas de domínio externo (*out-of-domain*) em relação ao modelo alvo.

Como proposta para trabalhos futuros, pretende-se melhorar a performance dos testes através de inspeção e melhoramento da base dados ReLi utilizada nos treinamentos, e testar outras arquiteturas de RNA para transferência de conhecimento.

REFERÊNCIAS

BAI, J.; LIAN, S.; LIU, Z.; WANG, K.; LIU, D. Smart guiding glasses for visually impaired people in indoor environment. **IEEE Transactions on Consumer Electronics**, v. 63, n. 3, p. 258–266, 2017.

BEBIS, G.; GEORGIOPOULOS, M. Feed-forward neural networks. **IEEE Potentials**, v. 13, n. 4, p. 27–31, Oct 1994. ISSN 0278-6648.

BERGER, J. O. **Statistical decision theory and Bayesian analysis**. [S.l.]: Springer Science & Business Media, 2013.

BOJARSKI, M.; TESTA, D. D.; DWORAKOWSKI, D.; FIRNER, B.; FLEPP, B.; GOYAL, P.; JACKEL, L. D.; MONFORT, M.; MULLER, U.; ZHANG, J. *et al.* End to end learning for self-driving cars. **arXiv preprint arXiv:1604.07316**, 2016.

CARRATO, S.; FENU, G.; MEDVET, E.; MUMOLO, E.; PELLEGRINO, F. A.; RAMPONI, G. Towards more natural social interactions of visually impaired persons. In: SPRINGER. **International Conference on Advanced Concepts for Intelligent Vision Systems**. [S.l.], 2015. p. 729–740.

CHO, K.; MERRIËNBOER, B. V.; GULCEHRE, C.; BAHDANAU, D.; BOUGARES, F.; SCHWENK, H.; BENGIO, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. **arXiv preprint arXiv:1406.1078**, 2014.

CHURCH, K. W. Word2vec. **Natural Language Engineering**, Cambridge University Press, v. 23, n. 1, p. 155–162, 2017.

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.

FILHO, J. A. W.; WILKENS, R.; IDIART, M.; VILLAVICENCIO, A. The brwac corpus: A new open resource for brazilian portuguese. In: **Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)**. [S.l.: s.n.], 2018.

FREITAS, C.; MOTTA, E.; MILIDIÚ, R.; CÉSAR, J. Vampiro que brilha... rá! desafios na anotaçao de opiniao em um corpus de resenhas de livros. **ENCONTRO DE LINGUÍSTICA DE CORPUS**, v. 11, p. 22, 2012.

GARG, N.; SESHADRI, A. Transfer learning: The impact of test set word vectors, with applications to political tweets. 2017.

GERSHENSON, C. Artificial neural networks for beginners. **arXiv preprint cs/0308031**, 2003.

GOODFELLOW, I. **Deep Learning**. [S.l.]: The MIT Press, 2016. 302 p.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HARRINGTON, P. Machine learning in action. **Shelter Island, NY: Manning Publications Co**, 2012.

HOFFMAN, G.; BENSON, M. Neurons with hysteresis form a network that can learn without any changes in synaptic connec- tion strengths. In: **Denker JS (ed) Proe. of AlP Conf. on Neural Networks for Computing. AIP**. [S.l.]: International Institute of Informatics and Systemics, 1986.

IMRAM, A. A. **Intuition of Gradient Descent for Machine Learning**. 2017. Disponível em: <https://medium.com/abdullah-al-imran/intuition-of-gradient-descent-for-machine-learning-49e1b6b89c8b>. Acesso em: 07 nov. 2018.

JONES, A. R.; BY-KURATA, D. F. **Visual Basic Developer's Guide to Asp and Iis (Developer's Handbook Series)**. [S.l.]: SYBEX Inc., 1999.

KIM, Y. Convolutional neural networks for sentence classification. **arXiv preprint arXiv:1408.5882**, 2014.

KOO, C. L.; LIEW, M. J.; MOHAMAD, M. S.; SALLEH, M.; HAKIM, A. A review for detecting gene-gene interactions using machine learning methods in genetic epidemiology. **BioMed research international**, Hindawi, v. 2013, 2013.

KOPANAKIS, J. **Sentiment Analysis**. 2021. Disponível em: <https://www.mentionlytics.com/blog/how-to-use-sentiment-analysis-for-brand-building/>.

KUBAT, M.; MATWIN, S. *et al.* Addressing the curse of imbalanced training sets: one-sided selection. In: NASHVILLE, USA. **Icml**. [S.l.], 1997. v. 97, p. 179–186.

NIELSEN, M. A. **Neural Networks and Deep Learning**. Determination Press, 2015. Disponível em: <http://neuralnetworksanddeeplearning.com/index.html>. Acesso em: 1 nov. 2018.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. [S.l.: s.n.], 2014. p. 1532–1543.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. In: **Proc. of NAACL**. [S.l.: s.n.], 2018.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. *et al.* Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, p. 9, 2019.

ROSENBLATT, F. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SIEGELMANN, H. T.; SONTAG, E. D. Turing computability with neural nets. **Applied Mathematics Letters**, Elsevier, v. 4, n. 6, p. 77–80, 1991.

SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; DRIESSCHE, G. V. D.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M. *et al.* Mastering the game of go with deep neural networks and tree search. **nature**, Nature Publishing Group, v. 529, n. 7587, p. 484, 2016.

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)**. [S.l.: s.n.], 2020.

TAYLOR, W. L. "cloze procedure": A new tool for measuring readability. **Journalism quarterly**, SAGE Publications Sage CA: Los Angeles, CA, v. 30, n. 4, p. 415–433, 1953.

WANG, J.; SONG, Y.; LEUNG, T.; ROSENBERG, C.; WANG, J.; PHILBIN, J.; CHEN, B.; WU, Y. Learning fine-grained image similarity with deep ranking. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition**. [S.l.: s.n.], 2014. p. 1386–1393.

WINASTWAN, R. **Text Classification with BERT in PyTorch**. 2021. Disponível em: <https://towardsdatascience.com/text-classification-with-bert-in-pytorch-887965e5820f>. Acesso em: 28 feb. 2022.

ZE, H.; SENIOR, A.; SCHUSTER, M. Statistical parametric speech synthesis using deep neural networks. In: IEEE. **Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on**. [S.l.], 2013. p. 7962–7966.