Rede Neural Convolucional para Classificação de Sentimentos

Nomes: Vitor Zimbrunes Pamplona, Vladmir Yuri Farias de Lima Cavalcanti
Matrículas: 20190038570, 20190036575

Apresentação do Problema:

No cenário da análise de sentimentos em mídias sociais, um problema de destaque que abordamos é a classificação de postagens na plataforma Twitter. A crescente importância das redes sociais como uma fonte de informações e comunicação torna essencial a capacidade de identificar e analisar os sentimentos expressos nas postagens dos usuários.

Uma abordagem eficaz para lidar com esse desafio envolve o uso de Redes Neurais Convolucionais (CNN) - uma técnica de aprendizado profundo que se mostrou altamente eficaz na extração de recursos relevantes de dados textuais e na classificação de sentimentos. Neste contexto, exploramos como as CNNs podem ser aplicadas para analisar e classificar postagens em positivas e negativas.

Objetivos:

Coletar Dados representativos de postagens do twitter com abordagem a sentimentos positivos e negativos.

Realizar o pré-processamentos dos dados para possibilitar o modelo a lidar com as características específicas das postagens no Twitter, como emojis, hashtags assim como menções.

Projetar a arquitetura de CNN para a classificação de sentimentos, fazendo uso de camadas de convolução, pooling e camadas de saída.

Treinar o modelo com os dados de treinamento anotados, realizando ajustes de hiperparâmetros, como taxa de aprendizado e tamanho dos lotes.

Avaliação do desempenho da CNN com o conjunto de Teste e utilizando métricas como acurácia,precisão,recall e F1-score.

Analisar as previsões do modelo para encontrar quais palavras ou padrões influenciaram mais na classificação de sentimentos.

Criar a documentação detalhada do projeto, incluindo relatórios, resultados e problemas encontrados, assim como as soluções implementadas.

Dados utilizados e pré-processamento dos dados:

Os dados utilizados foram encontrados no site do kaggle: https://www.kaggle.com/code/paoloripamonti/twitter-sentiment-analysis/notebook sendo disponibilizados para este exato estudo, os dados disponíveis são mais de 1.6 milhões de Tweets formando o dataset com as labels do target, ids, data, flag, usuário e o texto da postagem. Os targets de texto são separados em 0 = Negativo, 2 = Neutro e 4 = Positivo, para nosso modelo nos adaptamos os dados e fizemos as classificações apenas em Negativas e Positivos, assim definindo os textos neutros no que for mais próximo.

O pré-processamento foi feito com a Limpeza dos dados, removendo as informações irrelevantes, como os números de ID, datas e os nomes dos usuários. Assim como a eliminação de boa parte dos dados disponíveis para se ter mais velocidade e eficiência no treinamento do modelo, limitando os dados disponíveis de 1.6 milhões para 100000 dados.

Foi feita a redução das Classes alterando os alvos originais de 3 dados alvos definidos no target para 2 dados alvo, assim o label 4 foi alterado para o valor 1, tendo agora os alvos 1 (positivo) e 0 (negativo).

Com o uso da biblioteca re, fizemos a limpeza com o Regex para limpar a poluição de símbolos do texto e links como o encaminhamento de imagens nas postagens, limitando os Textos do dataset para apenas as palavras.

Pelo dataset ser em inglês foi feito o download do vocabulário do spaCy de inglês para carregar as stopwords e limpar as stopwords do dados.

Após o carregamento das stopwords é feito a formatação dos textos para minúsculo junto da transformação da variável do modelo para a variável em inglês do spaCy para ser feito a tokenização dos dados.

É feita a limpeza de todos os dados de textos e realizado o processo de tokenização, fazendo uso de 2^{16} por frequentemente ser escolhido como um valor padrão pela boa capacidade de representação de texto para muitos casos comuns.

Também é realizado o Padding para definir que todos os vetores sejam do mesmo tamanho para impedir possíveis incoerências e erros nos dados.

Graças ao padding é definido o tamanho 61 para os vetores, esse sendo o valor do maior tamanho de grupo de palavra nos dados de texto.

Por fim é feita a divisão dos dados para o treinamento e o teste do modelo, foi definida a separação de 30% dos dados disponíveis para o teste e 70% para o treinamento do modelo, totalizando 70000 de treino e 30000 de teste.

Rede Neural:

Para a implementação da Rede Neural Convolucional nós puxamos a classe "DCNN" que é uma subclasse do "tf.keras.Model" já projetada para realizar tarefas de processamento de texto, nesse caso classificação. Fazemos a definição dos parâmetros de inicialização incluindo o tamanho do vocabulário, dimensão dos embeddings, o número de filtros convolucionais, número de unidades na camada de feedforward, o número de classes de saída, taxa de dropout e o indicador de treinamento.

A construção das várias camadas do modelo foi feita em:

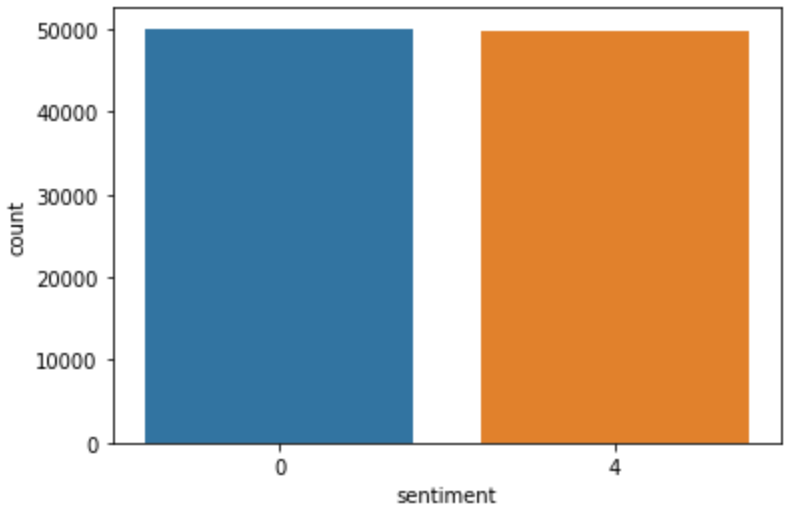
1. Camada de Embeddings: Camada que é inicializada com dimensões específicas para representar palavras como vetores densos.
2. Camadas de Convolução para Bigramas, Trigramas e Fourgramas: Três camadas convolucionais são definidas, cada uma com um kernel de tamanho diferente (2,3 e 4 respectivamente). Cada camada faz uso da ativação ReLU e tem o mesmo número de filtros (50). Essas camadas fazem a capturam de padrões de brigramas, trigramas e fourgramas no texto.
3. Camada de Pooling Global: Uma camada de pooling global é utilizada para extrair as características mais importantes das saídas convolucionais.
4. Camada Densa e Dropout: Uma camada densa com a quantidade de unidades especificada (512) é seguida por uma camada de dropout para evitar overfitting.
5. Camada de Saída: A última camada densa que foi configurada de acordo com o número de classes de saída. Onde se houver duas classes ("nb_classes == 2"), a ativação de sigmoid é utilizada; caso contrário o modelo fara uso da ativação softmax.

Com isso a função 'call' define a passagem direta da entrada pela rede. As palavras são primeiramente convertidas em embeddings, passam pelas camadas convolucionais e de poolings, e por fim são alimentadas através de camadas densas para a predição final.

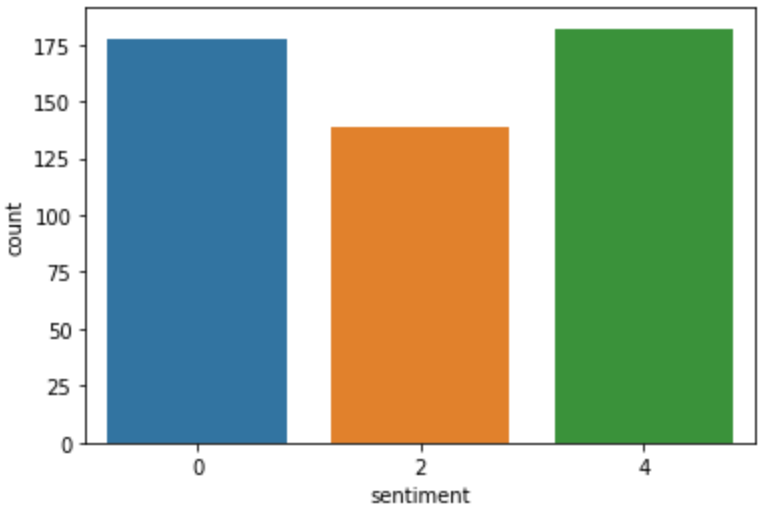
Resultados:

Com isso conseguimos chegar ao objetivo de criar uma Rede Neural Convolucional capaz de identificar e categorizar tweets positivos ou negativos, infelizmente não chegamos a uma classificação ótima obtendo apenas 73% de acurácia com o modelo como pode ser visto nos gráficos abaixos.

- Aqui fazemos a divisão dos dados em positivos e negativos:



- O gráfico abaixo representa a divisão dos dataset de teste antes de fazermos a alteração dos sentimentos neutros:

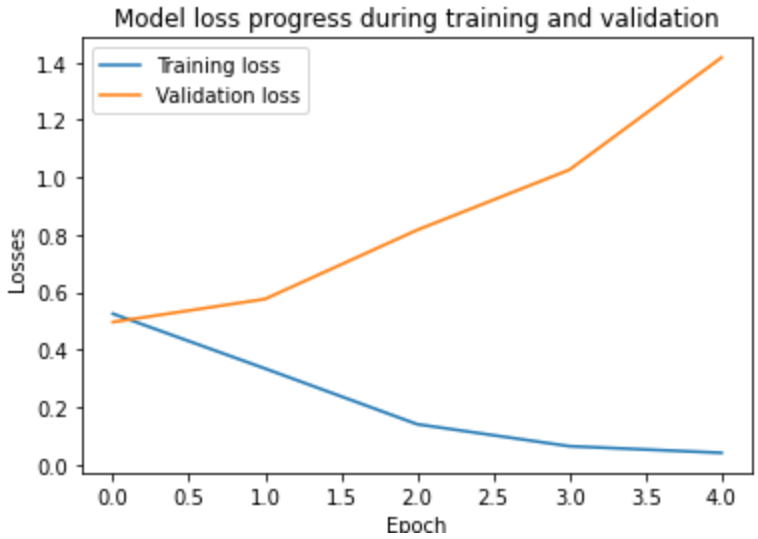


- Temos o relatório de classificação do modelo:

```
print(classification_report(test_labels, y_pred_test))
```

	precision	recall	f1-score	support
0	0.73	0.73	0.73	15030
1	0.73	0.72	0.72	14970
accuracy			0.73	30000
macro avg	0.73	0.73	0.73	30000
weighted avg	0.73	0.73	0.73	30000

- O gráfico representando o loss progession do treino e validação:



- O treino do modelo e sua acurácia comparada com a da validação: