Construção de um modelo de análise de sentimentos em português, para avaliações de comentários em redes sociais.

William H. C. Becher[1], **Lucas B. Knaak**[1]

[1]Instituto de Ciências Exatas e Tecnológicas – Universidade Feevale (FEEVALE)
Novo Hamburgo – RS – Brazil

Abstract. *This paper will describe the metodology and tasks realized, in order to compare the performance between differents machine learning models. We will train this models with a dataset of app store reviews in portuguese, and validate the models in diferente datasets, also in portuguese.*

Resumo. *Este artigo descreve a metodologia e tarefas realizadas no trabalho de conclusão de curso da Pós Graduação Especialização em Ciência de Dados e Machine Learning. O objetivo deste artigo é comparar a performance de diferentes modelos de Machine Learning para realizar a análise de sentimento de tweets em português. Treinaremos os modelos com um dataset de reviews de aplicativos, extraídos da Google Play Store.*

1. Objetivo do Trabalho

As redes sociais evoluíram muito nos últimos anos, e se tornaram uma fonte variada de informações. Nessas plataformas, as marcas conseguem ter uma visão geral do que o público vem falando sobre os seus produtos, e analisar informações, discussões, reclamações e elogios sobre os produtos que os usuário adquiriram e utilizam no dia a dia. [Agarwal et al. 2011]

Hoje em dia, é muito simples encontrar um modelo pré-treinado em inglês, através da biblioteca *Transformers, do Hugging Face* [Wolf et al. 2020], por exemplo. Mas é muito complicado encontrar esses modelos específicos em português, ou outro idioma que não seja inglês. Neste trabalho utilizamos o *Twitter* como fonte de dados para a validação dos modelos que criamos, e os reviews de aplicativos da *Google Play Store* para criação e treinamento dos modelos.

Neste artigo, criamos diferentes modelos (SVM, RNN e CNN) e comparamos os resultados de treino/teste em diferentes *datasets*, para testar a eficiência de cada um. Utilizaremos este *dataset* para treinar um novo modelo, e através dele, realizar a análise de sentimento de tweets que fazem referência a algum produto ou marca, que coletamos e descrevemos na seção 3.1. Com essa análise, poderemos plotar alguns insights, através de nuvens de palavras, para as empresas conseguirem entender o que o público está falando de positivo ou negativo sobre a sua marca.

2. Introdução

Twitter é um serviço de micro-blogging que permite a postagem de mensagens de até 140 caracteres para que outras pessoas as visualizem também como uma rede social, permite que um usuário siga outros usuários e receba em sua página inicial ou aparelhos móveis as atualizações dessas pessoas na ordem em que foram postadas. [Santos et al. 2010]

Com a facilidade em postar e receber mensagens de seu grupo de seguidores, o Twitter é cada vez mais utilizado como meio de interação entre os consumidores e as empresas/marcas. Por causa disto, escolhemos utilizar esta rede social como fonte de coleta de dados, para a criação do dataset que será utilizado neste artigo.

Os modelos que criamos são tarefas de classificações binárias, para classificar o sentimento apenas em positivo e negativo. Não incluímos uma terceira classe 'neutra' como tentativa de mitigar possíveis classificações neutras, pois acreditamos que podem causar alguma confusão nos treinamentos, dificultando o aprendizado do modelo.

3. Coleta e Descrição dos Dados

Todos os arquivos utilizados neste artigo e podem ser encontrados no repositório git https://github.com/wbecher/analise-sentimentos-tweets-portugues.

3.1. Coleta dos Tweets

Como tarefa inicial, coletamos diversos tweets, para podermos realizar a rotulação dos mesmos e posteriormente treinar as redes neurais com este dataset rotulado. Para isso, utilizamos a biblioteca *tweepy*, capturando uma *stream* de tweets em tempo real, com base em palavras pré-determinadas. Coletamos mensagens de citaram algumas contas de grandes empresas, em setores de atuação diferentes como provedores de streaming, bancos digitais, refrigerantes, comércio online. As palavras chaves utilizadas para a coleta foram as seguintes: BancodoBrasil, Bancointer, Bradesco, C6Bank, CasasBahia, CocaCola_Br, DisneyPlusBR, NetflixBrasil, PepsiBr, PrimeVideoBR, amazonBR, americanascom, brkindle, globoplay, itau, lojasamericanas, nubank, pontofrio, santander_br, submarino, timeneon.

Armazenamos primeiramente os dados *as is*, como são extraídos pela biblioteca tweepy, em um banco NoSQL MongoDB, e posteriormente os dados foram tratados, refinados e limpos, e exportados para um banco SQL relacional, e então salvos como CSV para facilitar a manipulação com a biblioteca *pandas*.

3.2. Coleta das Avaliações de *Apps*

Para conseguirmos treinar um modelo de previsão de sentimentos, precisamos de uma base rotulada, mas conseguir uma base em português, de tweets e dentro do contexto, não é uma tarefa fácil. Assim decidimos criar a nossa própria base rotulada, em português e de certa forma contextualizada. Com esse objetivo, de realizar a rotulagem automatizada dos tweets, treinamos um modelo utilizando como base avaliações de aplicativos coletadas da Google Play Store. Foram capturadas opiniões de usuários com 1 e 5 estrelas respectivamente, rotulando os comentários de 1 estrela como sendo negativos, e 5 estrelas como positivos. Dessa maneira pretendemos evitar poluir o modelo com opiniões mais neutras ou ambíguas.

Nesta coleta, foi utilizada a biblioteca *google_play_scraper*, que permite realizar a captura desses comentários de forma rápida. Os resultados foram armazenados diretamente em um arquivo *csv*.

3.3. Pré-processamento

Na limpeza dos dados, foi criada uma função para realizar a remoção de todos os caracteres especiais e *emojis* contidos nos textos, e todos os caracteres acentuados foram

alterados para sua versão sem acento. Também foram removidas todas as opiniões que possuíam o conteúdo de texto duplicado. Os números de telefone, urls e links foram removidos, por não apresentar relevância na análise de sentimento.

E como medida para diminuir a quantidade de *tokens* diferentes no alfabeto, os textos foram transformados para *lowercase*. Após estas etapas de processamento, foram removidos quaisquer instâncias que possuíam o texto nulo, que podem ter sido ocasionados por causa das remoções realizadas. Estes procedimentos de limpeza de dados, foram aplicados em outros três datasets já rotulados que utilizaremos para as avaliações dos modelos gerados, e serão apresentados na seção 6.

Após estes processos de limpeza, verificamos os datasets que iremos utilizar, e balanceamos os dados para conter aproximadamente a mesma quantidade de instâncias positivas e negativas. Então já criamos dois arquivos *.csv* separados contendo dados para treino e teste, de maneira a replicarmos os treinamentos dos modelos sempre com os mesmos dados de forma mais ágil e constante.

4. Modelos Utilizados

Nesta seção descrevemos os modelos que utilizamos no artigo, e os parâmetros utilizados para cada tokenização e treinamento.

4.1. Support Vector Machine (SVM)

Os algorítmos de *Support Vector Machines* (SVM) são utilizados para tarefas de classificação, regressão e detecção de *outliers*. Neste artigo, utilizamos a implementação SVC (*C-Support Vector Classification*), do pacote *sklearn* para realizar a classificação dos tweets em duas classes distintas, positivo e negativo e assim rotularmos os *tweets* automaticamente. A SVC foi treinada utilizando o dataset das avaliações da *Google Play Store*.

4.1.1. Tokenização

Utilizamos o tokenizador *TfidfVectorizer*, também disponível no pacote *sklearn* para realizar a tokenização do nosso dataset, e utilizar como input na SVM.

O tokenizador tem diversos parâmetros que podem ser alterados, um deles é o *n_grams*, que define o tamanhos dos n-gramas que serão utilizados para gerar o vocabulário e a matriz de features. No nosso caso utilizamos (1, 4), o que define que utilizaremos unigramas, bigramas, trigramas e quadrigramas. Na Figura 1 podemos ver um exemplo de vocabulário para um dataset de exemplo, e as features gerada para uma das frases.

4.1.2. Resultados da SVM

A acurácia do modelo gerado foi de 94,87%. Com este resultado que consideramos satisfatório, realizamos a rotulagem do dataset de tweets.

```
# text = ["exemplo de frase tokenizada aqui.",
#         "outro exemplo.",
#         "exemplo bacana."]

vectorizer.vocabulary_
```

```
{'de': 2,
 'frase': 10,
 'tokenizada': 15,
 'aqui': 0,
 'exemplo de': 7,
 'de frase': 3,
 'frase tokenizada': 11,
 'tokenizada aqui': 16,
 'exemplo de frase': 8,
 'de frase tokenizada': 4,
 'frase tokenizada aqui': 12,
 'exemplo de frase tokenizada': 9,
 'de frase tokenizada aqui': 5,
 'outro': 13,
 'outro exemplo': 14,
 'bacana': 1,
 'exemplo bacana': 6}
```

```
# tokenizando a frase: exemplo de frase tokenizada aqui.
vector = vectorizer.transform([text[0]])

# Shape e Vetor de Features
print('Shape:', vector.shape)
print('Array:', vector.toarray())
```

```
Shape: (1, 17)
Array: [[0.2773501 0.        0.2773501 0.2773501 0.2773501 0.2773501 0.
  0.2773501 0.2773501 0.2773501 0.2773501 0.2773501 0.2773501 0.
  0.        0.2773501 0.2773501]]
```

Figura 1. Exemplo de tokenização, com vocabulário e features geradas

4.2. Convolutional Neural Network (CNN)

As Redes Neurais Convolucionais são redes neurais, onde uma das camadas consiste em uma convolução, e possuem a característica de serem *feed-forward*. Elas são muito utilizadas em problemas de análise e processamento de imagens, mas podem ser utilizadas para texto também.

4.2.1. Tokenizador

Para a execução da CNN, tokenizamos o corpus com o auxílio da classe *Tokenizer* disponibilizada pelo *keras*, que realiza a vetorização dos dados, e os transforma em features numéricas. Criamos a classe definindo o parâmetro *num_words* para 40000, que define quais palavras manter na análise, baseado na frequência com que elas aparecem no dataset.

Com a classe auxiliar criada, precisamos chamar o método *fit_on_texts* que irá realizar a análise e gerar algumas informações que podemos consultar nos atributos *word_counts*, *word_docs*, *word_index*, *document_count*.

Para finalizar o processo de tokenização, utilizaremos o método *texts_to_sequences*, que irá transformar as frases em uma sequência de inteiros, e concluímos com a aplicação do método *pad_sequences* que realiza a normalização do tamanho das entradas, para todos os vetores de entradas ficarem com o mesmo tamanho.

4.2.2. Resultados da CNN

Com os tokens gerados e a CNN treinada com o mesmo dataset da SVM, atingimos uma acurácia de 95,35%. Realizamos então a predição no dataset de tweets e armazenamos os dados para comparação posterior.

4.3. Recorrent Neural Network (RNN)

As redes neurais *feed-forward* são muito utilizadas para problemas de classificação e regressão. As CNNs são ótimas para complexos problemas de classificação de imagem. Mas as ativações destas redes fluem apenas em uma direção, partindo das camadas de *input*, até a camada de saída. Como esses sinais fluem em apenas uma direção, essas redes não são tão boas para problemas onde os dados são influenciados pelo tempo. Para trabalhar com dados em que a temporalidade é importante, necessitamos de uma arquitetura diferente. Para esses problemas as RNNs são uma opção.

Assim como uma série temporal de vendas não pode ser analisada fora de sua ordem natural, as palavras e caracteres de uma sentença perdem o sentido se forem rearranjadas. As RNNs conseguem levar a ordem dos dados em consideração, armazenando as informações dos dados anteriores em memória, salvando um 'estado' nos neurônios da rede. Na Figura 2 pode-se ver uma representação de uma rede recorrente, mantendo um estado 'h' durante a sua execução.

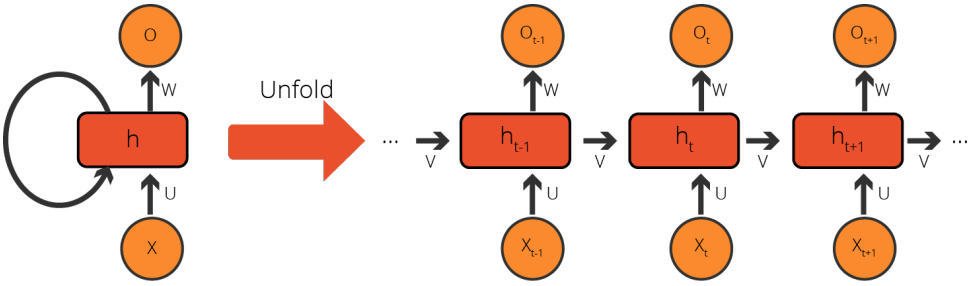

Figura 2. Estrutura de uma célula de RNN.

Existem tipos diferentes de Redes Recorrentes, tais como Vanilla RNN, GRU (*Gated Recurrent Unit*) e LSTM (*Long Short-Term Memory*). A principal razão pela qual as redes recorrentes são utilizadas nesses problemas, é pelo fato de operar sobre uma sequência de vetores: pode ser uma sequência na entrada, na saída, ou ainda em ambos. Na Figura 3 podemos ver exemplos mais concretos dessas estruturas [Karpathy 2020]. Na imagem, cada retângulo representa um vetor, e cada seta uma função. Os vetores de input são os vermelhos, os azuis são os vetores de saída e os verdes armazenam o status da RNN.

Cada tipo de estrutura é utilizado para um tipo de tarefa específica, a primeira, *one to one* é a maneira de processamento *Vanilla*, sem utilização de RNN, onde o tamanho da entrada e saída são fixos (exemplo: classificação de imagens). O segundo exemplo *one to many* é um exemplo onde a saída é uma sequência de vetores (exemplo: image captioning). No terceiro exemplo *many to one*, a entrada é uma sequência de vetores, e a saída é um vetor simples (exemplo: análise de sentimento). No quarto exemplo, *many to many*, tanto a entrada como a saída são sequências de vetores (exemplo: tradução

de sentenças de um idioma para outro). E no quinto exemplo, as entradas e saídas são sincronizadas (exemplo: classificação de vídeo, com o objetivo de catalogar cada frame do vídeo) [Karpathy 2020].

No código deste artigo, utilizamos a RNN, com a camada LSTM.

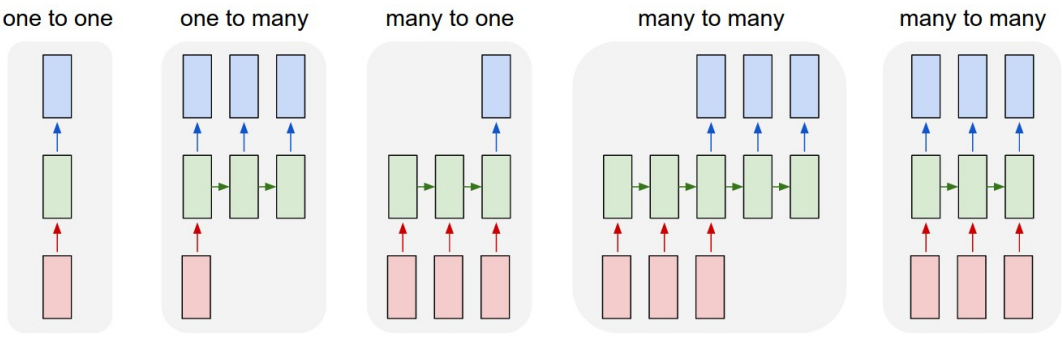

Figura 3. Possíveis estruturas de RNNs

4.3.1. Tokenizador

O tokenizador para a RNN foi o mesmo utilizado na etapa anterior, no treinamento da CNN, com os mesmos parâmetros.

4.3.2. Resultados da RNN

Após o treinamento da RNN, em 5 épocas, obtivemos uma acurácia de 94,86%. Foi realizada então a predição dos tweets com base nesse modelo, e os resultados salvos para a análise posterior.

5. Datasets utilizados para validações dos modelos finais

Após os treinamentos dos diferentes modelos, podemos observar que a acurácia de todos os modelos ficou alta, acima dos 94% em todos os casos. Esta observação nos fez adicionar alguns experimentos, para avaliarmos as possíveis causas de um valor de acurácia tão alto, logo nos primeiros treinamentos.

Com isto em mente, iremos aplicar o modelo treinado em três datasets distintos já rotulados e difundidos pela internet, e avaliar como se comportam. Para estes experimentos, foram escolhidos os datasets de avaliações de filmes do IMDB, um com avaliações de produtos do site Americanas, e outro com tweets aleatórios.

O modelo escolhido para o teste foi a RNN, por apresentar uma boa acurácia e também por causa do tempo de treinamento que ficou menor.

5.1. Avaliações IMDB

O dataset IMDB é um dos mais famosos benchmarks para modelos de classificação de textos que existe. Dificilmente não será citado em papers que proponham trabalhos envolvendo este tipo de tarefa. [Fred 2018]

Este dataset consiste em um arquivo *.csv* com 3 colunas, sendo elas *text_en* contendo o texto em inglês, *text_pt* com o texto em portugês, e a coluna *sentiment* com a avaliação do texto em *neg* para negativo, ou *pos* para positivo. No nosso artigo utilizaremos apenas as colunas com o texto em português e a coluna de sentimento, onde alteramos os valores textuais para inteiros, sendo 0 (negativo) e 1 (positivo).

5.2. Avaliações Americanas

O outro dataset, foi o de reviews de produtos das lojas americanas. Ele contém mais de 130 mil reviews de usuários, coletados do site das Americanas.com, coletados entre Janeiro e Mayo de 2018. [b2wdigital 2018]

O dataset possui diversas colunas com o perfil bem detalhado de cada usuário que fez o review, mas para o nosso artigo, selecionamos apenas as colunas *review_text* contendo o texto da avaliação em português e a *overall_rating*, contendo a avaliação de estrelas dada pelo usuário.

5.3. Tweets Aleatórios

Este dataset contém tweets aleatórios coletados por um usuário e disponibilizado no Kaggle. Os textos possuem temas diversos, e estão rotulados em Positivos e Negativos. Utilizamos apenas as colunas de texto, e a de sentimento. [Cleves 2020]

5.4. Avaliação das Acurácias

Avaliamos o desempenho da RNN, utilizando os datasets descritos acima, e medindo a acurácia do modelo em cada um deles. O resultado pode ser visto na tabela 1.

Tabela 1. Desempenho do modelo RNN utilizado para rotulagem automatizada

	Acurácia
IMDB	65,00%
Americanas	91,00%
Tweets	52,82%

6. Experimentos e Resultados

Após a rotulagem automatizada do nosso dataset de tweets por três modelos distintos, realizamos um *ensemble* manual, para gerar um dataset com apenas um rótulo, levando em conta uma 'votação' para cada instância. Cada tweet foi analisado, e rotulado com a classe que foi escolhida por pelo menos 2 dos modelos.

Com este dataset novo, criamos dois modelos novos. Um onde usamos apenas este dataset para o treinamento, e outro onde utilizamos uma união deste dataset, com o de avaliações retiradas da Google Play Store.

Para termos um aprendizado mais consistente, foi realizado um balanceamento do dataset para possuirmos quantidades semelhantes de classes positivas e negativas. E por fim, removemos os tweets das empresas, Netflix, Nubank e Ponto Frio, para serem utilizados posteriormente nas avaliações.

Para a análise final, utilizamos a RNN, visto que obteve uma boa acurácia nos treinamentos anteriores e o seu tempo de treinamento foi mais rápido em comparação aos

outros algoritmos. Criamos diversos modelos utilizando modelos de rede e parâmetros diferentes. Estes resultados podem ser vistos no notebook número 12, disponível no repositório de código deste artigo.

6.1. Modelo Final 1

Para geração do primeiro modelo, utilizamos apenas o dataset de tweets como input para a rede neural, separando 80% dos dados para treino, e 20% para teste. Utilizamos um limite de 33000 palavras no tokenizador.

Criamos uma RNN com duas camadas LSTM Bidirecionais, com tamanho 32. No treinamento com 5 épocas, obtivemos uma acurácia no treinamento de 96,6%, e atingimos um *loss* de 0,0934.

6.1.1. Acurácia Modelo Final 1

Nos dados de teste, este modelo atingiu uma acurácia de 88,9%.

6.2. Modelo Final 2

Para o treinamento do segundo modelo, utilizamos o mesmo dataset de tweets utilizado na seção 6.1, com os mesmos processamentos nos dados, mas unimos com o dataset de reviews da Google Play Store, afim de apresentar uma maior gama de estilos de texto, e gerar um modelo mais robusto. Utilizamos um limite de 61000 palavras no tokenizador.

Criamos uma RNN com a mesma estrutura de duas camadas LSTM Bidirecionais, com tamanho 32. No treinamento com 5 épocas, obtivemos uma acurácia de 97,34% e um *loss* de 0,0820.

6.2.1. Acurácia Modelo Final 2

Nos dados de teste, este modelo atingiu uma acurácia de 94,07%.

6.3. Validação dos Modelos Finais

Para validarmos os modelos gerados, realizamos uma rotulagem nos datasets de validação, descritos na seção 5.

6.3.1. Dataset IMDB

Este dataset possui um contexto diferente dos tweets que coletamos. Ele possui textos mais longos, em média 4x mais caracteres por mensagem, e um estilo de escrita diferente.

Neste dataset, com o Modelo 1, obtivemos 50,86% de acurácia. Com o Modelo 2, atingimos 69,86%.

6.3.2. Dataset Tweets Aleatórios

O segundo dataset também são tweets, conforme nosso dataset alvo, porém são de assuntos aleatórios, o tamanho dos textos são um pouco menor do que os que foram utilizados para treinar o modelo.

Neste dataset, com o Modelo 1, obtivemos 49,54% de acurácia. Com o Modelo 2, atingimos 53,60%.

6.3.3. Dataset Americanas

Por fim vamos utilizar um outro dataset de avaliação, onde os textos são em média no tamanho do modelo treinado, este dataset também utiliza avaliações de 1 a 5, e, da mesma forma que o dataset utilizado para o treino do modelo, também vamos considerar apenas as avaliações 1 como negativas e 5 como positivas.

Neste dataset, com o Modelo 1, obtivemos 59,92% de acurácia. Com o Modelo 2, atingimos 92,50%.

6.3.4. Validando NuBank, Ponto Frio e Netflix

Estes tweets são as instâncias que foram removidas do dataset para treinamento dos modelos finais. Neste dataset estamos dentro do mesmo contexto das mensagens e com um estilo de escrita parecido.

Aplicando os dois modelos nestes tweets das empresas, obtivemos os resultados que podem ser vistos na tabela 2.

Tabela 2. Acurácia dos modelos de rotulagem, e de validação

	Modelo Rotulagem	Modelo 1	Modelo 2
IMDB	65,65%	50,86%	69,86%
Tweets Aleatórios	52,82%	49,54%	53,60%
Americanas	91,01%	59,92%	92,50%
Tweets Coletados	-	53,62%	90,57%
Ponto Frio	-	58,56%	87,60%
Nubank	-	56,53%	84,99%
Netflix	-	52,67%	87,86%

6.4. Análise dos resultados

Obtivemos algumas melhoras no Modelo 2, com relação ao Modelo 1. No geral, a acurácia aumentou quando foi realizado o treinamento com o dataset mesclado e o aumento do limite de palavras no tokenizador. Utilizar um corpus com maior variedade de estilos de texto ajudou os algoritmos a apresentarem melhores resultados.

Levantamos algumas hipóteses ao analisar os resultados mais profundamente. Pudemos perceber que muitos dos resultados que haviam apresentado erros de predição, possuíam um conteúdo de texto mais neutro. Pudemos avaliar isto ao analisar a média

de acordo com o peso da classificação gerado pela RNN. Ao gerar a predição, a RNN provê resultado de acordo com um valor que pode ir de 0 a 1, quanto mais próximo de 1, mais positivo o sentimento. Na tabela 3 podemos ver uma sumarização destes valores. Nela conseguimos analisar que os Falsos Positivo, sempre obtiveram uma média menor se comparados com os Verdadeiros Positivos. Já o Falso Negativo obteve uma média maior, se comparados com os Verdadeiros Negativos, confirmando assim a hipótese de que são sentimentos mais neutros e que podem gerar alguma inconsistência.

Não podemos descartar também a hipótese de haver alguns erros de rotulagem nos dados, visto que ambas as estratégias utilizadas foram automatizadas. Mesmo ao utilizar os reviews da Google Play Store, ainda assim podem haver comentários positivos avaliados com 1 estrela apenas, ou o inverso, com um comentário negativo que foi avaliado com 5 estrelas.

Tabela 3. Avaliação dos pesos gerados pela RNN nas predições do Ponto Frio

Resultado	Média dos Pesos
Todos Positivos	0,8688
Todos Negativos	0,1476
Verdadeiro Positivo	0,9670
Falso Positivo	0,8353
Verdadeiro Negativo	0,0469
Falso Negativo	0,1644

7. Trabalhos Futuros

Pudemos perceber que a qualidade dos dados de input, influenciam bastante na acurácia do modelo. Acreditamos que como trabalho futuro podemos melhorar o processo de coleta e rotulagem dos dados. Uma das idéias, é criar uma ferramenta que possa ser utilizada para realizar essa classificação do dataset inicial de forma mais fácil, de forma a evitar rotulagens errôneas.

Outro processo que podemos aplicar em futuros trabalhos é o retreinamento/fine tunning de alguma rede já existente, de modo a ampliar a acurácia dos nossos modelos.

Podemos também tentar realizar a aplicação de Transformers, visto que é projetado para ser desenvolvido e extendido por pesquisadores, e auxilia bastante na prática de processos de aprendizado de máquina. [Wolf et al. 2020]

8. Conclusão

Obtivemos resultados satisfatórios nos modelos para rotulagem dos dados. Com algumas melhorias no processo de coleta e rotulagem inicial, pensamos ser possível melhorar ainda mais estes modelos. Os tweets e avaliações que coletamos foram rotulados de forma automática, o que não evita erros de avaliação.

As avaliações de acurácia realizadas em datasets distintos também obtiveram resultados modestos, mas que podem ser melhoradas futuramente, principalmente melhorando os dados de entrada dos modelos. Conseguimos melhores métricas ao utilizar um modelo treinado com o dataset mesclado, o que demonstra que a qualidade dos dados de entrada é um fator que influencia bastante no resultado final.

Referências

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., and Passonneau, R. J. (2011). Sentiment analysis of twitter data. In *Proceedings of the workshop on language in social media (LSM 2011)*, pages 30–38.

b2wdigital (2018). Reviews de produtos do site americanas.com - https://github.com/b2wdigital/b2w-reviews01. *github*.

Cleves, V. (2020). Sentiment analysis on tweets in portuguese - https://www.kaggle.com/viniciuscleves/sentiment-analysis-on-tweets-in-portuguese. *Kaggle*.

Fred, L. (2018). Tradução do dataset imdb para o português - https://www.kaggle.com/luisfredgs/imdb-ptbr. *Kaggle*.

Karpathy, A. (2020). The unreasonable effectiveness of recurrent neural networks - http://karpathy.github.io/2015/05/21/rnn-effectiveness/. *Blog Pessoal*.

Santos, L. M., Esmin, A. A. A., Zambalde, A. L., and Nobre, F. M. (2010). Twitter, análise de sentimento e desenvolvimento de produtos: Quanto os usuários estão expressando suas opiniões? *Prisma. com*, 1(13):159–170.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.