UNIVERSIDADE DE SÃO PAULO - USP

Instituto de Ciências Matemáticas e de Computação - ICMC

SCC0233 - Aplicações de Aprendizado de Máquina e Mineração de Dados

Estudo e Aplicação de Análise de Sentimento em Avaliações de Usuários

Guilherme de Pinho Montemovo	9779461
João Pedro Hannauer	9390486
João Pedro Sousa dos Reis	9293373
Pedro Avellar Machado	9779304
Rodrigo Anes Sena de Araújo	9763064

SÃO CARLOS

2020

Resumo

Com o objetivo de estudar e aplicar técnicas de Processamento de Linguagem Natural (PLN) para resolver a tarefa de Análise de Sentimento, que busca extrair e classificar sentenças de acordo com seu conteúdo, o projeto consiste em desenvolver modelos preditivos que consigam classificar corretamente as *reviews*, avaliações de usuários.

Foi utilizado um conjunto de dados de avaliações de filmes do site IMDb para desenvolver o projeto. Para isso, os textos foram manipulados de diferentes formas, usando técnicas de PLN como tokenização, *stemming* e vetorização, com a intenção de preparar estes dados para treinamento em modelos classificadores de aprendizado de máquina. Também foi testada uma nova abordagem considerada o estado da arte atualmente nas tarefas desta área, o BERT.

Os modelos foram treinados e testados, obtendo métricas de desempenho sobre o conjunto de dados escolhido, permitindo uma comparação com diferentes abordagens. Os modelos treinados também foram testados em outros conjuntos de dados para observação do desempenho em textos desconhecidos e de diferentes naturezas, a fim de avaliar o nível de generalização de cada modelo e como as avaliações são diferentes dependendo do avaliador e assunto.

Sumário

1. Introdução

O trabalho a seguir foi desenvolvido para a disciplina Aplicações de Aprendizado de Máquina e Mineração de Dados e, teve o objetivo de fazer com que os membros do grupo tivessem contato com a área de processamento de linguagem natural (PLN).

Com base nisso, o tema "Análise de Sentimento" foi escolhido e o projeto foi feito sempre tentando aprender sobre as inúmeras etapas do processamento desse tipo de dados. De tal forma, foram utilizados diferentes modelos classificadores e de *encoding*, sendo possível observar o funcionamento de cada um e assim comparar as suas eficácias diante do problema proposto.

Análise de Sentimento é um tema relativamente novo e tem sido cada vez mais relevante devido às grandes quantidades de texto gerado e os avanços nas técnicas da área. Essa tarefa é interessante não apenas pelo fator curiosidade, mas empresas e instituições podem utilizar as informações obtidas por bons classificadores para entender melhor os seus clientes, usuários e as opiniões acerca do seu produto ou serviço.

2. Apresentando a proposta e o conjunto de dados

A proposta deste projeto é o estudo e a aplicação de técnicas de PLN (Processamento de Linguagem Natural), particularmente na tarefa de **Análise de Sentimentos**, para desenvolver modelos satisfatórios de classificação em ***reviews*** (avaliação de usuários).

O conjunto de dados utilizado para o desenvolvimento do trabalho foi o **IMDb Movie Reviews** disponibilizado no Kaggle[1]. Este conjunto é consideravelmente conhecido e bastante utilizado na área, sendo o original disponibilizado pela Universidade de Stanford. Conta com 50 mil comentários de usuários avaliando algum filme no site IMDb. Cada comentário é classificado como positivo ou negativo.

	review	sentiment
0	One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked....	positive
1	A wonderful little production. The filming technique is very unassuming- very old-time-BBC f...	positive
2	I thought this was a wonderful way to spend time on a too hot summer weekend, sitting in the air...	positive
3	Basically there's a family where a little boy (Jake) thinks there's a zombie in his closet & his...	negative
4	Petter Mattei's "Love in the Time of Money" is a visually stunning film to watch. Mr. Mattei off...	positive

Imagem 1: As cinco primeiras linhas do conjunto de dados

Usamos este conjunto como base para preparar os dados e treinar classificadores, com o objetivo de obter um resultado satisfatório de classificação neste conjunto de dados. Em

seguida, também testar os modelos treinados em outros conjuntos, para avaliar o desempenho em diferentes dados e domínios.

Os códigos, gráficos e resultados deste projeto foram feitos utilizando a linguagem Python 3, em notebooks na ferramenta Google Colab. Algumas bibliotecas foram utilizadas para auxiliar no desenvolvimento, algumas delas são: pandas, MatPlotLib, NumPy, NLTK, TextBlob, sklearn, ktrain. Os *notebooks* e mais informações sobre as bibliotecas estão disponíveis nas referências.

3. Exploração e Pré-processamento

Nesta primeira etapa, o objetivo foi analisar o conjunto de dados para procurar inconsistências e determinar quais os passos a seguir para manipular os dados de forma a prepará-los para uso nos treinamentos.

Na análise exploratória não foram encontradas questões como valores nulos, duplicados, desbalanceados, etc. Então, a tarefa logo passou a ser encontrar formas de obter maior estruturação nos textos das avaliações, de forma a permitir os modelos encontrarem mais significado nas sentenças. Esta foi a sequência de manipulações utilizadas no texto bruto das *reviews* para obter os dados preparados:

a) **Retirando contrações**
   Palavras que são resultado da junção de duas palavras, são separadas
   `Exemplo: 'there's' -> 'there is'`
b) **Tokenização**
   Separando o texto em ‘tokens’,
   `Exemplo: 'One of the other' -> 'One', 'of', 'the', 'other'`
c) **Transformação para apenas letras minúsculas**
   `Exemplo: 'One', 'of', 'the' -> 'one', 'of', 'the'`
d) **Removendo pontuação**

e) **Removendo stopwords**
   Stopwords são palavras consideradas irrelevantes para algumas tarefas, já que não costumam agregar significado a um texto. Na maioria das vezes costumam ser artigos ou pronomes e aparecem em grande frequência dentro de um corpus.
   `Exemplo: Remoção de palavras como 'of', 'the', 'is'...`
f) **Aplicando stemming**
   Stemização é um processo que transforma as palavras no radical da mesma. É interessante pois muitas palavras podem ter diferentes formas, como os verbos, mas essencialmente são referentes a mesma coisa.
   `Exemplo: 'fighting' -> 'fight' ; 'wonderful' -> 'wonder'`

	review	sentiment	no_contract	review_description_str	tokenized	lower	no_punc	stopwords_removed	stemmed
0	One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked....	positive	[One, of, the, other, reviewers, has, mentioned, that, after, watching, just, 1, Oz, episode, yo...	One of the other reviewers has mentioned that after watching just 1 Oz episode you will be hooke...	[One, of, the, other, reviewers, has, mentioned, that, after, watching, just, 1, Oz, episode, yo...	[one, of, the, other, reviewers, has, mentioned, that, after, watching, just, 1, oz, episode, yo...	[one, of, the, other, reviewers, has, mentioned, that, after, watching, just, 1, oz, episode, yo...	[one, reviewers, mentioned, watching, 1, oz, episode, hooked, right, exactly, happened, first, t...	[one, review, mention, watch, 1, oz, episod, hook, right, exact, happen, first, thing, struck, o...

Imagem 2: Primeira linha do conjunto de dados com todos os passos da transformação

Após essas manipulações, foi feito um teste inicial gerando um coeficiente de sentimento usando a TextBlob (biblioteca para processamento de dados textuais) para observar se já seria obtido um resultado interessante.

	Dados realmente positivos	Dados realmente negativos
Dados com coeficiente positivo	22512	15124
Dados com coeficiente negativo	2488	9876

Ou seja, aproximadamente com uma acurácia de 60% e precisões de 60% e 80% para positivos e negativos respectivamente, pode se perceber que de fato o coeficiente consegue acertar mais do que errar, mas não consegue fazer uma separação muito boa, classificando muitos dados como positivos.

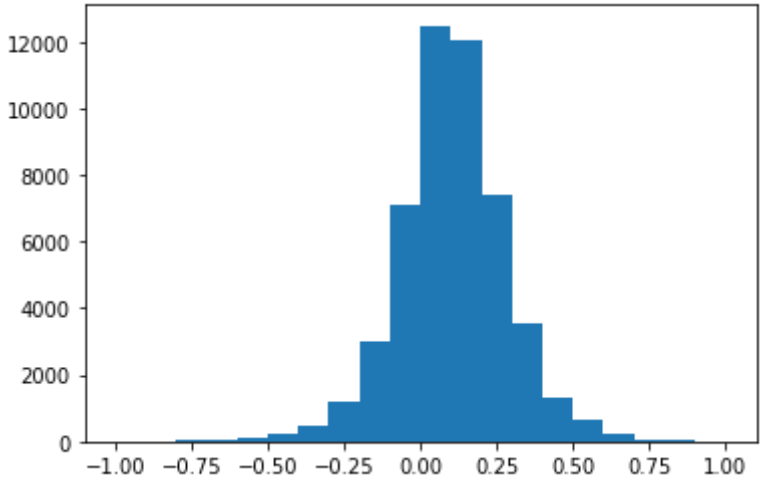
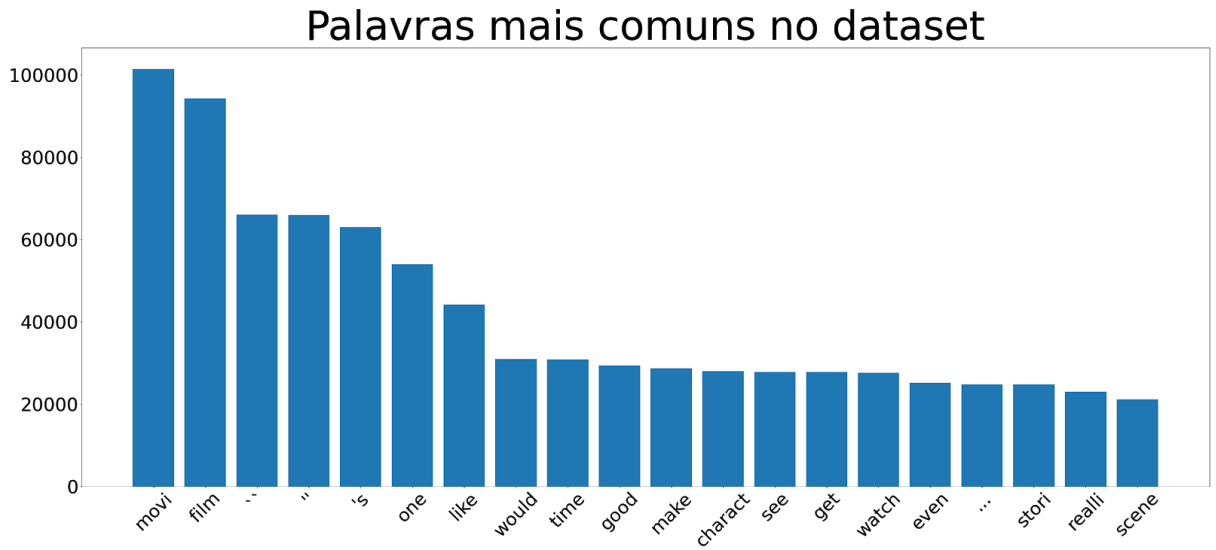


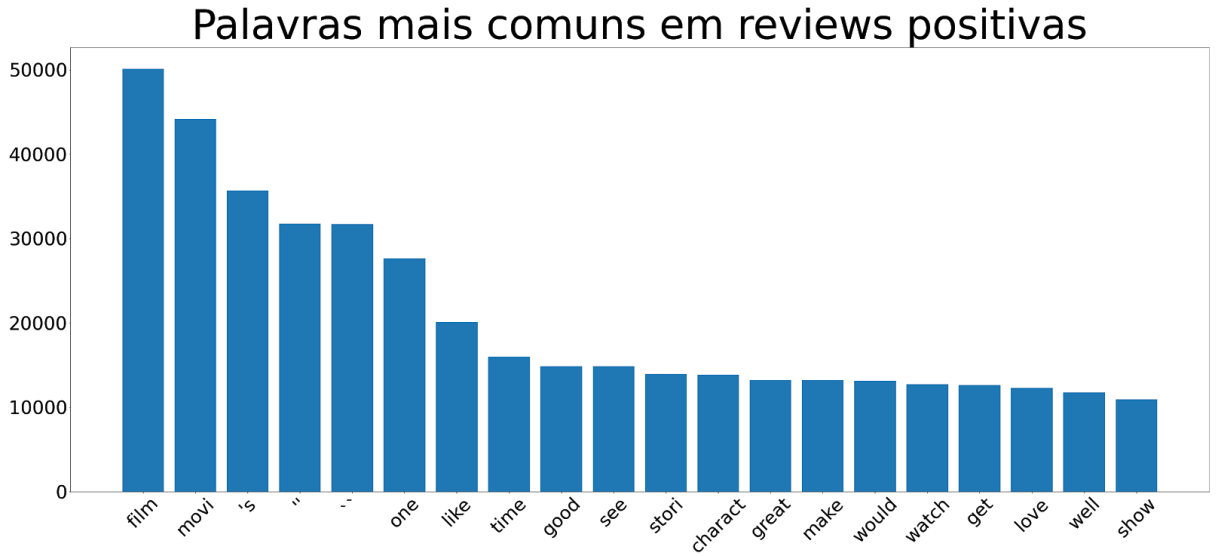
Gráfico 1: Distribuição dos coeficientes de sentimento da TextBlob nos dados.

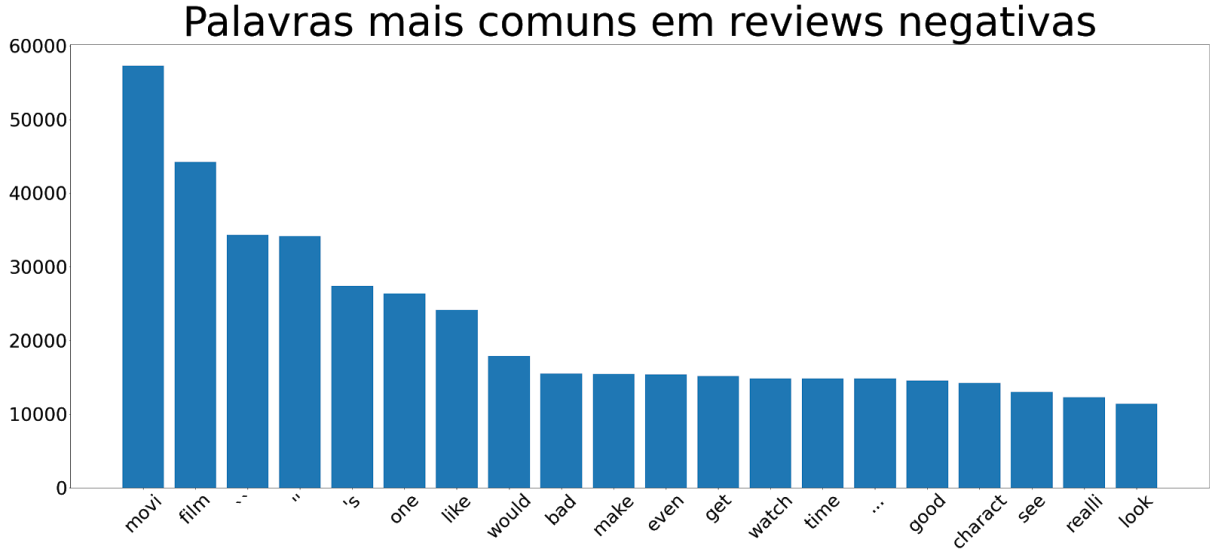
Através da análise do gráfico é possível notar que muitos dados têm coeficientes próximos de zero, que é o divisor entre positivo e negativo, logo causando alto grau de incerteza, impedindo uma separação mais definida.

Com os textos separados em *tokens*, foi feita uma análise para visualização de quais *tokens* mais aparecem no conjunto de dados, tanto no geral quanto em cada classificação de

sentimento. Os 20 tokens com mais ocorrências em todo o conjunto de dados, em seguida os com mais ocorrências apenas nas *reviews* positivas e negativas:

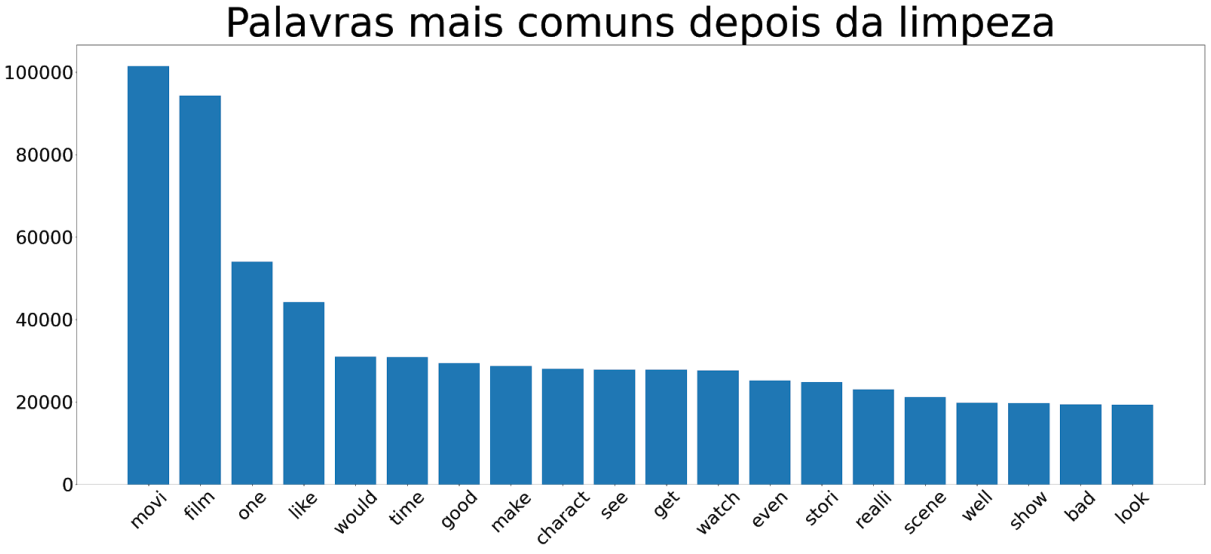






Utilizando os gráficos de ocorrência, podem ser feitas algumas observações: ainda existem *tokens* irrelevantes nos dados (aspas, ’s, ...), que não foram retirados das transformações anteriores. Existe ocorrência altíssima de alguns *tokens* (movi, film, one, ...), sendo estes não muito interessantes para nosso problema, já que aparecem com frequência no conjunto, tanto nas avaliações positivas, quanto nas negativas. E finalmente, comparando os gráficos de avaliações positivas e negativas, é possível encontrar palavras que indicam a presença desses sentimentos (*good, great, love*... vs *bad, even*...).

Para finalizar o pré-processamento, apenas os *tokens* compostos por caracteres alfabéticos foram mantidos, para remover os símbolos restantes. Desta forma, as maiores ocorrências do conjunto de dados após a limpeza:



No entanto, ainda é preciso converter todo esse texto para algum tipo de representação em que a máquina e os modelos classificadores consigam entender de alguma forma. Foi então necessário realizar uma vetorização, ou seja, uma transformação nos dados para que estes agora sejam representados como um grande vetor.

A vetorização escolhida foi a **TF-IDF**, que se trata de uma medida estatística que pontua a importância de uma palavra no documento pela frequência de aparecimento da mesma. Assim, se um *token* possui bastante recorrência no corpus significa que ele não possui tanto impacto para a classificação, então sua pontuação é menor. Este tipo de técnica visa amenizar o viés que palavras muito comuns em vocabulário possam causar, dando mais importância para palavras que realizam uma maior distinção.

4. Treinamento de classificadores

Para a escolha dos modelos de classificadores, foram utilizadas as implementações da biblioteca **sklearn**. Foram testados vários modelos, com diferentes parâmetros, os que obtiveram melhor desempenho foram: **Perceptron**, **Random Forest** (400 estimadores, critério entropia) e **Gradient Boosting** (400 estimadores, taxa de aprendizado 0.5). Com os modelos escolhidos, foram testados diferentes parâmetros no vetorizador TF-IDF. Os valores da tabela a seguir representam a média das acurácias obtidas por validação cruzada (5 folds) com diferentes vetorizadores TF-IDF nos três modelos.

	Perceptron	Random Forest	Gradient Boosting
TF-IDF padrão	0.854	**0.860**	0.853
TF-IDF max=0.9 min=100	0.837	0.848	**0.851**
TF-IDF max=0.9 min=100 bigramas	0.840	0.851	**0.856**
TF-IDF max=0.9 min=5 bigramas	**0.892**	0.870	0.865

Onde é usado max e min, apenas as entradas que têm ocorrência dentro do intervalo estabelecido (no nosso caso foi utilizado, no mínimo 100 ou 5 ocorrências e no máximo em 90% dos dados) são usadas. Os bigramas permitem que não apenas existam representações vetoriais para cada *token,* mas também combinação de dois *tokens* adjacentes. Os vetorizadores com bigramas levam mais tempo para treinar. Com relação aos modelos, o Perceptron é quase instantâneo, enquanto os outros demoram muito mais, como pode ser visto na tabela a seguir junto com outras métricas (média dos valores obtidos por *cross_validate* 5 *folds* usando a vetorização 'max=0.9 min=5 bigramas')

	Perceptron	Random Forest	Gradient Boost
Acurácia	**0.892**	0.870	0.865
Precisão	**0.888**	0.859	0.852
Recall	**0.897**	0.889	0.883
F1	**0.892**	0.874	0.867
Tempo de *fit* (*s*)	**0.3**	1002	1030

5. Utilizando a técnica BERT

Lançado em 2018 pela Google, o **BERT** (*Bidirectional Encoder Representations from Transformers*) é atualmente considerado o estado da arte em linguagem de processamento natural. Trata-se de uma rede neural profunda que é capaz de entender e estabelecer relações entre todas as palavras de uma frase, ao invés de uma por uma de forma ordenada.

Para usar o BERT no projeto, utilizamos o **ktrain**, um *wrapper* leve da biblioteca *TensorFlow Keras*, que permite utilizar de forma simples com poucas linhas de código, abordagens de deep learning. O modelo recebe como entrada o texto bruto, já que faz suas próprias transformações e representações.

Usando esta abordagem, como esperado, o tempo gasto para treinar e validar é grande. Mas o resultado faz valer a pena. Desta vez sendo até bem melhor que o esperado, o resultado obtido utilizando o treinamento da própria biblioteca, com taxa de aprendizado de $2 * 10^{-5}$ e 3 épocas, foram:

	Acurácia no treino	Acurácia na validação
Época 1	0.911	0.939
Época 2	0.948	0.943
Época 3	**0.985**	**0.951**

6. Resultados finais

Analisando as acurácias obtidas nos diferentes testes, foi possível perceber que as mudanças nos parâmetros do vetorizador TF-IDF impactaram no resultado de cada modelo, fazendo com que os diferentes classificadores se alterassem em possuir o melhor resultado para cada teste. Além disso, é notável a velocidade de execução do Perceptron em relação aos outros classificadores. O Perceptron é um modelo mais simples que os outros utilizados e geralmente considerado menos poderoso, já que o mesmo só tem a capacidade de fazer apenas uma separação linear no espaço das entradas, mas nosso problema de fato, pode ser resolvido com esta abordagem, já que temos apenas duas classes. O bom desempenho do Perceptron mostra que possivelmente a vetorização resultou em uma boa separação das classes no espaço.

Por outro lado, a utilização da BERT correspondeu a toda euforia da comunidade de NLP, obtendo resultados extremamente ótimos que destoam totalmente dos classificadores comuns na literatura.

7. Aplicando os modelos em outros conjuntos de dados

Os resultados obtidos no conjunto de dados escolhido foram muito bons, mas com para testar se os modelos são realmente classificadores confiáveis e têm um bom nível de generalização, é interessante testar o desempenho do modelo com outros dados, não apresentados no treinamento, com diferentes características e de variadas fontes.

OBS: Para os modelos do *sklearn*, foram testadas duas formas de vetorização TF-IDF. Os dados apresentados foram obtidos com TF-IDF max=0.9 min=5 bigramas, que performou melhor.

Rotten Tomatoes

Rotten Tomatoes é um outro website agregador de críticas de cinema e televisão. Este porém, diferentemente do IMDb, só contém *reviews* feitas por críticos especializados de algum veículo de mídia. O conjunto de dados utilizado, também disponível no Kaggle[2], reúne mais de um milhão de registros com a crítica, informações do avaliador, data e classificação (chamado no site de '*Fresh*' ou *'Rotten*', que foram interpretados como positivo e negativo, respectivamente). Para testar nos modelos treinados, os dados foram manipulados para retirar inconsistências, balancear as classes e uma amostra de 10000 registros foi utilizada.

Portanto, pela similaridade do tema e estrutura, os modelos treinados anteriormente foram testados com parte desse conjunto de dados para observação do desempenho.

	Perceptron	Random Forest	Gradient Boost	BERT
Acurácia	0.702	0.633	0.620	**0.823**
Precisão	0.690	0.587	0.578	**0.816**
Recall	0.739	**0.910**	0.909	0.843

Os resultados mostram que o desempenho obtido no conjunto original não se manteve, mas ainda acertando mais do que o oposto. O BERT também obteve o melhor desempenho no geral. A diferença dos resultados neste conjunto podem ser relativos a algumas questões como o fato das avaliações nesse site serem feitas exclusivamente pela crítica especializada, que pode usar termos e estruturas diferentes; o conjunto de dados não está bem preparado para esta tarefa; ou os nossos modelos não obtiveram muita capacidade de generalização fora do conjunto previamente apresentado.

Dataset reviews Amazon, Yelp e IMDb

Este conjunto de dados, também disponível no Kaggle[3], reúne cerca de 1000 avaliações de usuários de cada um desses sites, de diferentes ramos (Amazon: *e-commerce,* logo avaliação de produtos; Yelp: Avaliação de estabelecimentos comerciais; IMDb: Avaliação de filmes). A estrutura deste conjunto é a mesma do conjunto original: texto da avaliação de um usuário, classificada como positiva ou negativa. O interessante da aplicação dos modelos nestes dados é exatamente observar como os modelos se comportam diante de avaliações de naturezas diferentes e consequentemente sentenças variadas. Os testes também foram realizados nesta nova amostra de *reviews* do IMDb para avaliar se o desempenho com relação ao conjunto original se mantém.

Amazon				
	Perceptron	Random Forest	Gradient Boost	BERT
Acurácia	0.738	0.647	0.630	**0.882**
Precisão	0.720	0.593	0.578	**0.917**
Recall	0.780	0.936	**0.964**	0.840

Yelp				
	Perceptron	Random Forest	Gradient Boost	BERT
Acurácia	0.717	0.673	0.648	**0.889**
Precisão	0.685	0.608	0.590	**0.916**
Recall	0.804	**0.976**	0.974	0.856

Novo IMDb				
	Perceptron	Random Forest	Gradient Boost	BERT
Acurácia	0.820	0.798	0.798	**0.945**
Precisão	0.852	0.737	0.732	**0.962**
Recall	0.788	0.946	**0.961**	0.930

As métricas no geral são boas, mas ainda revelam que os modelos treinados especificamente com o conjunto de dados original, não podem ser generalizadas e aplicadas a

outros domínios sem perda de desempenho, principalmente com relação aos modelos do *sklearn*. Interessante notar que os modelos *ensemble* (Random Forest e Gradient Boost) conseguem excelente *recall*, o que indica que o modelo tem identificado casos positivos corretamente, mas a julgar pelas outras métricas, pode estar classificando exageradamente como esta classe. O modelo baseado em BERT novamente apresentou os melhores desempenhos no geral, com ótimas métricas na classificação de todos os *reviews* de diferentes fontes testadas. No novo conjunto do IMDb, ficou bem próximo do valor obtido na validação do conjunto original.

8. Conclusão

O grupo acredita que mais do que os resultados obtidos e demonstrados acima, a conclusão que fica deste trabalho foi o aprendizado obtido. Durante todo o projeto diferentes técnicas, métodos e algoritmos foram estudados com o objetivo de se entender um pouco mais do universo do processamento de linguagem natural. O trabalho proporcionou uma experimentação prática e direcionada deste assunto que foi muito interessante para se criar uma base nova de conhecimento por parte dos alunos.

9. Referências

[1] IMDb Dataset of 50K Movie Reviews
https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

[2] Rotten Tomatoes movies and critic reviews dataset
https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset

[3] Dataset 'Sentiment Labelled Sentences'
https://www.kaggle.com/marklvl/sentiment-labelled-sentences-data-set

Github com os *notebooks* utilizados para desenvolvimento
https://github.com/pedroavellar/imdb_sentiment

Outros
NLP Part 2| Pre-Processing Text Data Using Python
https://towardsdatascience.com/preprocessing-text-data-using-python-576206753c28

NLP Part 3| Exploratory Data Analysis of Text Data
https://towardsdatascience.com/nlp-part-3-exploratory-data-analysis-of-text-data-1caa8ab3f79d

BERT Explained: State of the art language model for NLP
https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Sentiment Classification Using BERT
https://kgptalkie.com/sentiment-classification-using-bert/

Bibliotecas Utilizadas
Pandas: Manipulação e Análise de dados (https://pandas.pydata.org/)

Matplotlib: Geração de gráficos (https://matplotlib.org/)

NumPy: Funções matemáticas (https://numpy.org/)

NLTK: Funções relacionadas a processamento de linguagem natural (https://www.nltk.org/)

TextBlob: Processamento de dados textuais (https://textblob.readthedocs.io/en/dev/)

scikit-learn: Aprendizado de máquina (https://scikit-learn.org/stable/)

ktrain: Deep learning e AI (BERT) (https://github.com/amaiya/ktrain)