UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Leonardo Rodrigues da Mota

UMA APLICAÇÃO DE ANÁLISE DE SENTIMENTOS PARA MEDIR POPULARIDADE DOS CANDIDATOS À PRESIDÊNCIA

CAMPINA GRANDE - PB

2022

Leonardo Rodrigues da Mota

UMA APLICAÇÃO DE ANÁLISE DE SENTIMENTOS PARA MEDIR POPULARIDADE DOS CANDIDATOS À PRESIDÊNCIA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Herman Martins Gomes

CAMPINA GRANDE - PB

2022

Leonardo Rodrigues da Mota

Uma aplicação de análise de sentimentos para medir popularidade dos candidatos à presidência

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Herman Martins Gomes

Orientador – UASC/CEEI/UFCG

Eanes Torres Pereira

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 02 de Setembro de 2022.

CAMPINA GRANDE - PB

RESUMO

Este trabalho consistiu na criação de uma inteligência artificial para classificação de Tweets sobre os principais candidatos a presidência do Brasil no ano de 2022. O resultado foi uma inteligência artificial que para o conjunto de dados de teste atingiu um percentual de 93% de precisão, usando como modelo arvore de decisão. Também foi criada uma aplicação Web feita com as tecnologias Javascript, React, Python e Firebase, usada para exibir os resultados obtidos.

Uma aplicação de análise de sentimentos para medir popularidade dos candidatos à presidência

Leonardo Rodrigues da Mota
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
leonardo.mota@ccc.ufcg.edu.br

Herman Martins Gomes
(Orientador)
Universidade Federal de Campina Grande
Campina Grande, Paraíba, Brasil
hmg@computacao.ufcg.edu.br

ABSTRACT

This work consisted of creating an artificial intelligence to classify Tweets about the main candidates for the presidency of Brazil in the year 2022. The result was an artificial intelligence that for the test dataset reached a percentage of 93% accuracy, using as a decision tree model. A web application made with Javascript, React, Python and Firebase technologies was also created, used to display the results obtained.

Palavras-chave

Twitter,Classificador de sentimentos, eleições 2022, python,javascript.

Links úteis

Backend:
https://github.com/Leonardomotta/SentimentClassifierApi/tree/master
Frontend:
https://github.com/Leonardomotta/SentimentClassifierFrontend
Classificador:
https://github.com/Leonardomotta/notebookClassificador
Aplicação:
https://sentimentclassifierf.herokuapp.com/

1. Introdução

As redes sociais, ao longo da última década, tiveram um aumento expressivo em seu número de usuários. A constante evolução na telefonia móvel em conjunto com a popularização da internet, são fatores que justificam esse aumento [1]. Há não muito tempo, as pessoas precisavam estar diante de um computador para se conectar, o email era o principal meio de comunicação através da internet, uma vez que se encaixava bem com o ritmo assíncrono da época. Hoje em dia estimasse que os brasileiros, passam cerca de 91 horas por semana online [2]. Maior parte desse tempo é gasto em redes sociais, com isso as redes sociais tornaram-se uma ferramenta rica, tanto para a divulgação de ideias como também para coleta de informações.

Existe um grande volume de redes sociais espalhadas pela internet. Cada uma com sua proposta para atrair o usuário. Contudo, uma das melhores fontes para coleta de informação é o *Twitter*[3], que segue um modelo de microblog, onde através de textos curtos, os usuários enviam e recebem atualizações pessoais. Celebridades, políticos, cientistas e influenciadores religiosos estão no Twitter.

O Twitter tem estado em foco entre as mais variadas empresas, que utilizam técnicas de processamento de linguagem natural, para analisar a opinião dos usuários sobre seus produtos e direcionar propagandas ao público mais propenso a adquiri-lo.

Uma técnica muito utilizada é a análise de sentimentos que consiste em distinguir os sentimentos por trás de um texto [4]. O intuito deste trabalho de conclusão de curso é aplicar essa técnica em Tweets [1] de cunho politico, sobre os principais candidatos à presidência da república, e através disso, medir sua popularidade, assim como disponibilizar uma aplicação web onde os usuários possam ver os resultados obtidos.

2. Metodologia

A Concepção da aplicação ocorreu nas seguintes etapas:

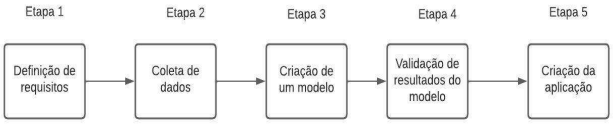


Figura 01: Etapas para a criação da aplicação

Etapa 1: Consistiu em um estudo, para entender as necessidades da aplicação e definição dos requisitos funcionais e não funcionais.

Etapa 2: O Twitter possui textos curtos e muitos deles incluem gírias, o que torna desafiador a criação de um modelo para avaliá-los usando uma base de dados preexistente. Foi necessário então a criação de um banco de dados personalizado, onde todos os Tweets foram classificados manualmente.

Etapa 3: Nesta etapa, ocorreu a criação de um modelo, utilizando a base de dados da etapa anterior, para treinar o classificador.

Etapa 4: Neste momento uma pequena amostra do banco de dados construído na etapa 2, foi utilizado para avaliar o modelo e sua precisão.

Etapa 5: Nesta etapa foi realizada a criação do Frontend[2] e Backend[3]. Assim como a implantação da aplicação

3. Coleta de requisitos

Nessa etapa foi necessário realizar um estudo para definir qual linguagem e modelo de aprendizagem se encaixava melhor com o

[1] Textos curtos feitos pelos usuários do Twitter
[2] Camada de interação do usuário na aplicação
[3] Camada de lógica de negocio da aplicação

objetivo e como treiná lo a fim de construí-lo com um bom grau de precisão. Bem como definir as tecnologias utilizadas na concepção aplicação web. A seguir uma lista com todos os requisitos utilizados na construção dessa aplicação:

Requisitos funcionais:

REQ 01: O usuário deve ser capaz de selecionar um candidato.

REQ 02: O usuário deve ser capaz de gerar um relatório de acordo com o ano e candidato.

REQ 03: O sistema deve apresentar os resultados em gráficos.

REQ 04: O usuário deve ser capaz de comparar dois candidatos.

REQ 05: O sistema deve exibir os candidatos com imagens e uma breve descrição sobre os mesmos.

REQ 06: O sistema deve armazenar as análises prévias em um banco de dados.

REQ 07: O sistema deve realizar operações de banco de dados através de uma API[4].

REQ 08: O modelo deve ter uma taxa de acerto de 70%.

Requisitos não funcionais

REQNF 01: O classificador de sentimentos deve ser construído na linguagem *Python*[5]

REQNF 02: A API deve ser construída em Python

REQNF 03: O Frontend deve ser construído em *Javascript*[6] utilizando *React*[7]

REQNF 04: O Firebase deve ser o banco de dados, utilizado.

4. Coleta de dados

A quantidade limitada de caracteres que o Twitter permite por publicação, a presença de gírias e termos regionais somadas a diversidade etária do seu público, faz com analisar os sentimentos deste tipo de texto se torne uma tarefa desafiadora. Com isso, a tarefa de treinar um modelo, usando alguma base de dados famosa como do *IMDB*[9] se torna inviável.

Dada a problemática citada acima se fez necessário a criação de uma base de dados construída a partir de Tweets em português. Essa seção visa descrever o processo seguido na criação dessa base.

4.1 Extração de Tweets

A primeira etapa da coleta de dados se fez pela extração de Tweets. Encontrar uma forma de extrair los dê maneira eficiente e garantir sua relevância foi um desafio, uma vez que a grande quantidade de textos repetidos e menções à Tweets poderiam gerar um desequilíbrio no conjunto de dados. A primeira tentativa ocorreu com o uso do *Tweepy*[10], uma biblioteca desenvolvida em Python que serve para facilitar a comunicação com a api do Twitter. O Tweepy exige uma chave fornecida pela propria API ao realizar o cadastro. Dentro da Api existem três níveis de acesso: Essential, que nada mais é que um nivel básico, ele permite cerca de 500 mil Tweets por mês; Elevated, similar ao anterior, porém, com 1 milhão de Tweets por mês; e, por último, o Academic research, que permite cerca de 10 milhões de Tweets por mês e libera para o usuário buscas nos arquivos históricos do Twitter.

Como o intuito deste trabalho é permitir que os usuários façam comparações de seus candidatos ao longo dos anos, apenas o Academic research atenderia a essa demanda. Ao entrar em contato com a equipe do Twitter, apenas o acesso ao plano Elevated foi concedido.

Com isso a utilização de um Web crawler[5] se tornou a opção mais viável para a conclusão deste. Na linguagem de programação Python existem varios crawlers, entretanto, o que mais atendeu as necessidades desse projeto foi o *Snscrape*[11]. Este crawler possui uma vasta gama de funcionalidades e acesso ao histórico de Tweets desde sua criação, não necessita de qualquer tipo de chave, contudo, sua maior desvantagem se dá pelo seu tempo de resposta quando comparado ao Tweepy. Apesar dessa desvantagem o Snscrape foi utilizado nessa implementação, uma vez que seu ônus de performance foi compensado pela forma que o sistema foi arquitetado, mais detalhes serão apresentados em uma seção posterior.

Com a ferramenta já selecionada, a próxima etapa foi a de definir como utiliza-la de maneira viável na criação de um banco de dados. O crawler foi modificado para ser chamado a cada dois dias extraindo uma amostra de cada Top threadings[6] e salvando esse retorno em um arquivo CSV[7] para serem avaliados posteriormente. Um dos problemas encontrados, foi a quantidade de Retweets[8] existentes, muitas vezes uma extração de um top threading, 50% dos Tweets extraídos são Retweets então foi necessário adicionar uma verificação antes de inserir o tweet no conjunto de dados.

Com os dados já selecionados e dispostos em um csv, foi necessária a classificação manual de cada um deles, para que posteriormente pudessem ser utilizados no treinamento do modelo. Nesse processo foram classificados 1200 Tweets, destes 1000 foram usados no treinamento do modelo e 200 usados no teste. Os Tweets são classificados em três categorias: positivos, negativos e neutros. Que são representados no dataset por 1,-1 e 0 respectivamente. Tweets positivos são aqueles que carregam mensagens de apoio ao tópico discutido, o contrário se aplica aos negativos, já Tweets neutros se encontram no meio-termo, geralmente são textos imparciais, como noticias e afins.

Essa primeira tentativa resultou em um desbalanceamento nos dados, uma vez que dentre eles apenas uma pequena parcela foi classificada como positivo ou neutro, devido à aleatoriedade com que foi feita a seleção. Ao aplicar o conjunto de dados descrito acima nos modelos, foi obtido um baixo grau de precisão, que será demonstrado na seção 5.1.

Para sanar esse problema, foi feita uma nova extração e classificação, resultando em uma base de dados com 4 mil Tweets, o resultado dessa base de dados foi combinado com uma já existente,*portuguese-sentiment-analysis*[12], essa combinação foi feita de modo a equilibrar o resultado final com uma quantidade próxima de Tweets, positivos negativos e neutros. Resultando num total de 8 mil Tweets, desses 5 mil foram utilizados para treinar o modelo. Os tres mil Tweets restantes foram utilizados para testar o modelo. Esses três mil possuem mil Tweets de cada label. Os resultados também estão descritos na seção 5.1.

Com os dados brutos já classificados foi necessário fazer uma limpeza antes de passá los para o modelo. Essa limpeza foi feita em três etapas. Primeiro foi necessário a remoção das stopwords do texto, que nada mais são que palavras irrelevantes ao sentindo. Para realizar essa remoção foi utilizado o modulo “stopwords” presente na biblioteca nltk, esse modulo devolve uma lista com

[4] Interface de programação da aplicação

[5] Código responsável por extrair dados de um site através da interface de usuários

[6] Ranking composto pelos 25 termos mais utilizados no Twitter

[7] Arquivo composto de valores separados por virgula

[8] Republicação de um Tweet

todas as stopwords na linguagem especificada, no caso deste projeto, português. Com essa lista disponível houve uma interação sobre cada tweet removendo dele essas palavras.

A etapa seguinte se deu pela remoção de termos comuns em Tweets, links, menções, pontuação, etc. Para essa etapa foi utilizado o módulo nativo de tratamento de expressões regulares da linguagem Python.

Na última etapa da limpeza de dados foi utilizada uma técnica chamada stemização que é o processo de reduzir a palavra ao seu radical. Este processo foi feito com o auxílio da biblioteca nltk.stem.

5. Criação e Avaliação de um modelo classificador

A análise de sentimentos em textos pode ser vista como um problema de classificação. Dentre um conjunto limitado de opções, o algoritmo deve ser capaz de definir em qual categoria melhor qualifica o sentimento geral apresentado pelo texto. Para tratar esta problemática existe um amplo conjunto de opções disponíveis, com isso foi necessário realizar um estudo para selecionar a que melhor se encaixa com este trabalho.

A linguagem de programação utilizada na criação do modelo foi Python, por possuir uma vasta gama de bibliotecas auxiliares para a implementação de algoritmos de aprendizado de máquina. Dentre elas, sklearn[13], foi escolhida por geralmente apresentar boa performance em suas implementações e possuir uma vasta gama de algoritmos. Diante da diversidade de opções foi necessário realizar uma avaliação comparativa entre os algoritmos para definir o que melhor se encaixava. Detalhes desse processo estão descritos na seção subsequente.

5.1 Avaliação comparativa

Para mensurar a eficácia dos modelos foi empregada a matriz de confusão, uma tabela onde pode se observar os valores preditos, em relação aos reais. Abaixo estão descritos os critérios utilizados nessa comparação:

Acurácia: É a razão entre a quantidade de acertos (positivos, TP, e negativos, TF) pela, a quantidade de predições total, as quais incluem os acertos e erros(falsos positivos, FP e falsos negativos, FN). A acurácia mede a taxa de sucesso do modelo sendo obtida pela fórmula abaixo.

$$acurácia = \frac{TP+TN}{TP+FP+TN+FN} = \frac{Predições\ corretas}{Todas\ as\ predições}$$

Recall: Mede a proporção de acertos positivos do modelo. O resultado é dado pela seguinte fórmula:

$$recall = \frac{TP}{TP+TN}$$

Precisão: É a razão entre o total de resultados classificados como positivos e quantos deles estão corretos, é dado pela seguinte fórmula:

$$precisão = \frac{TP}{TP+FP}$$

F-score: É dado pelo balanço entre a precisão e o recall.

$$FScore = 2 * \frac{precisão * recall}{precisão+recall}$$

A seguir teremos uma breve descrição dos algoritmos que serão utilizados nessa etapa de avaliação seguido dos resultados obtidos.

Naive Bayes

O algoritmo Naive Bayes, recebe seu nome devido uma combinação de palavras entre o sobrenome do seu criador, Thomas Bayes, e a palavra "naive" do inglês que significa ingênuo. O algoritmo é considerado ingênuo por não considerar correlação entre as variáveis de uma base de dados. O algoritmo foi utilizado por seu criador para tentar provar a existência de Deus, hoje é muito utilizado na área de aprendizado de máquina. Existem varias implementações do mesmo, nesse trabalho iremos considerar a distribuição multinomial e o complemento de bayes.

Distribuição multinomial: É muito utilizada devido a sua versatilidade, ela nada mais é que uma generalização do modelo binomial, que classifica a probabilidade de um evento acontecer ou não, contudo, a principal diferença entre o modelo binomial e o multinomial é a quantidade de categorias possíveis, neste trabalho onde um tweet pode ser classificado em três categorias, este modelo se encaixa muito bem.

Resultados obtidos com a distribuição multinomial:

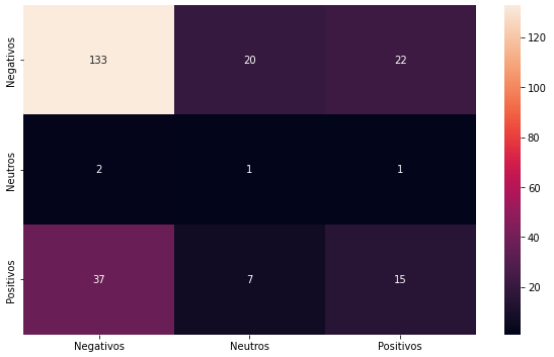


Figura 02: Matriz de confusão Naive Bayes multinomial

Acurácia media: 0.63

Polaridade	precision	recall	f1-score	support
-1	0.76	0.77	0.77	172
0	0.25	0.04	0.06	28
1	0.25	0.39	0.31	38

Figura 03: Métricas Naive Bayes multinomial

Complemento de Bayes: utiliza de pesos para equilibrar um conjunto de dados. Onde maior parte dos dados são negativos ou positivos.

Resultados obtidos com complemento de Bayes:

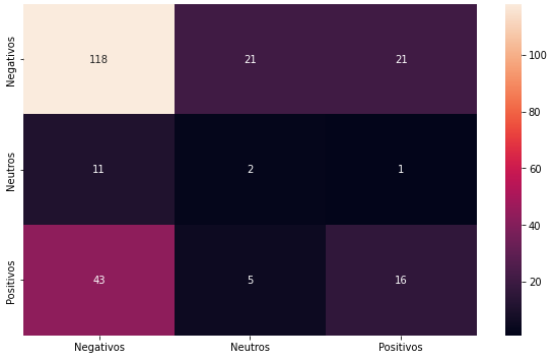


Figura 04: Matriz confusão complemento de Bayes

Acurácia media: 0.57

Polaridade	precision	recall	f1-score	support
-1	0.74	0.69	0.71	172
0	0.14	0.07	0.10	28
1	0.25	0.42	0.31	38

Figura 05: Métricas complemento de Bayes

Árvore de decisão

Este algoritmo é amplamente utilizado, devido sua simplicidade e apresentar bons resultados em suas previsões. São criados pontos de decisão os chamados "nós", os "ramos", são os caminhos possíveis que podem ser percorridos a partir de um nó. Após o dado passar por um dos dois caminhos ele cai em um novo nó, esse passo vai se repetindo formando assim uma estrutura similar a uma árvore.

Resultados obtidos com árvore de decisão:

Figura 06: Matriz confusão árvore de decisão

Acurácia media: 0.60

Polaridade	precision	recall	f1-score	support
-1	0.78	0.76	0.77	172
0	0.17	0.04	0.06	28
1	0.17	0.29	0.22	38

Figura 07: Métricas árvore de decisão

Gradiente descendente estocástico

Para entender o gradiente descendente estocástico é necessário, primeiramente entender como funciona o processo de atualização de pesos de uma rede neural. Nas redes neurais, temos uma função custo, essa função compara o resultado obtido com o resultado real, calculando o erro, através dele os pesos são ajustados de modo que nossa função custo esteja sempre diminuindo, o gradiente descendente é a derivada parcial, buscando sempre atingir o mínimo.

Resultados obtidos considerando cem interações máximas:

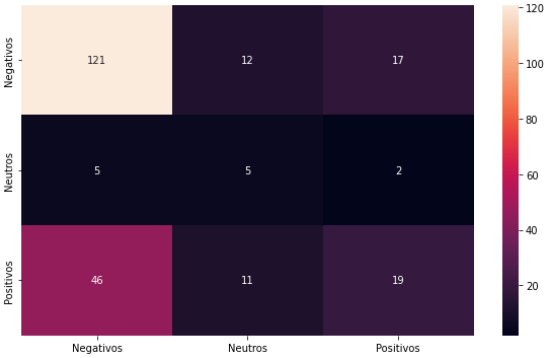


Figura 08: Matriz de confusão do gradiente estocástico

Acurácia media: 0.61

Polaridade	precision	recall	f1-score	support
-1	0.81	0.70	0.75	172
0	0.42	0.18	0.25	28
1	0.25	0.50	0.33	38

Figura 09: Métricas gradiente estocástico

Resultados obtidos considerando dez mil interações máximas:

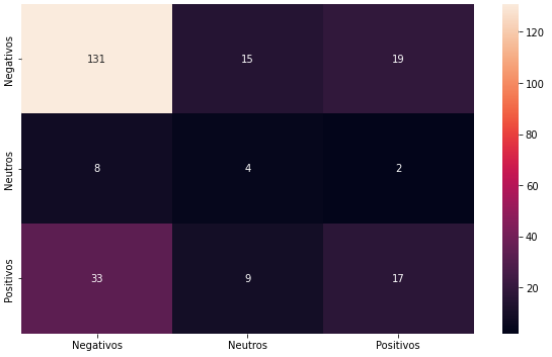


Figura 10: Matriz confusão gradiente estocástico (10000 interações)

Acurácia media: 0.64

Polaridade	precision	recall	f1-score	support
-1	0.79	0.76	0.78	172
0	0.29	0.14	0.19	28
1	0.29	0.45	0.35	38

Figura 11: Métricas gradiente estocástico (10000 interações)

K-nearest neighbors

Sempre que é preciso classificar um elemento, este algoritmo faz a escolha baseada em seus vizinhos, se a maior parte dos seus vizinhos mais próximos, foram classificados, como "A", o elemento atual também será classificado como "A".

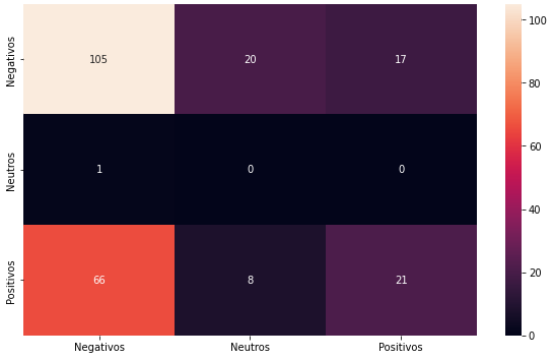


Figura 12: Matriz confusão KNN

Acurácia media: 0.53

Polaridade	precision	recall	f1-score	support
-1	0.74	0.61	0.67	172
0	0.00	0.00	0.00	28
1	0.22	0.55	0.32	38

Figura 13: Métricas KNN

Ao termino da análise nenhum dos modelos conseguiu atender, acurácia minima de 70% como foi especificado no REQ 08. Este problema ocorreu por dois motivos, a baixa quantidade de dados e sua má distribuição nos testes. Para sanar esse problema foi feita uma mesclagem com um conjunto de dados, já existente, usado para avaliar Tweets através de emojis. Foi realizado um tratamento de modo a manter um equilíbrio, entre os dois conjuntos de dados, Os resultados estão dispostos a seguir.

Naive Bayes Multinomial

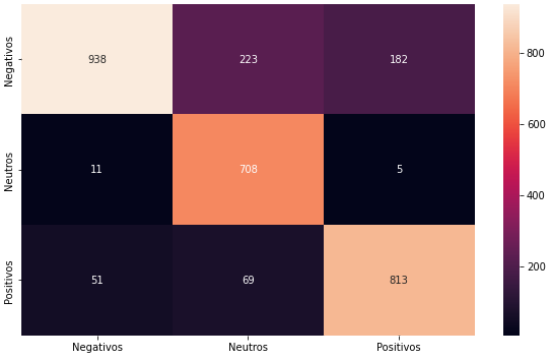


Figura 14: Matriz de confusão Naive Bayes multinomial

Acurácia media: 0.82

Polaridade	precision	recall	f1-score	support
-1	0.70	0.94	0.80	1000
0	0.98	0.71	0.82	1000
1	0.87	0.81	0.84	1000

Figura 15: Matriz de confusão Naive Bayes multinomial

Complemento de Naive Bayes

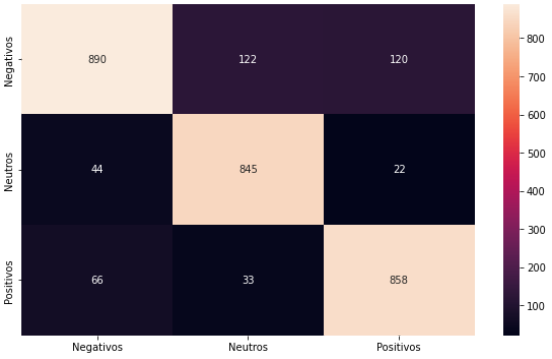


Figura 16: Matriz confusão complemento de Bayes

Acurácia media: 0.86

Polaridade	precision	recall	f1-score	support
-1	0.79	0.89	0.83	1000
0	0.93	0.84	0.88	1000
1	0.90	0.86	0.88	1000

Figura 17: Métricas complemento de Bayes

Árvore de decisão

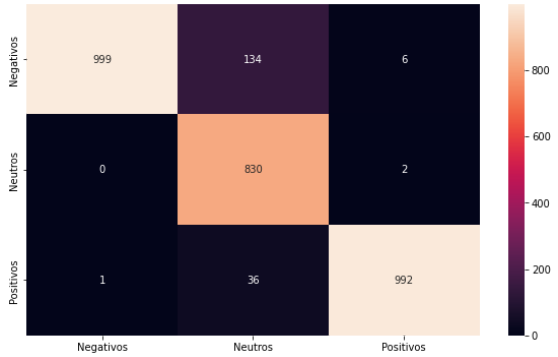


Figura 18: Matriz confusão árvore de decisão

Acurácia media: 0.93

Polaridade	precision	recall	f1-score	support
-1	0.88	1.00	0.93	1000
0	1.00	0.83	0.91	1000
1	0.96	0.99	0.98	1000

Figura 19: Métricas árvore de decisão

Gradiente descendente estocástico

Resultados obtidos considerando cem interações máximas:

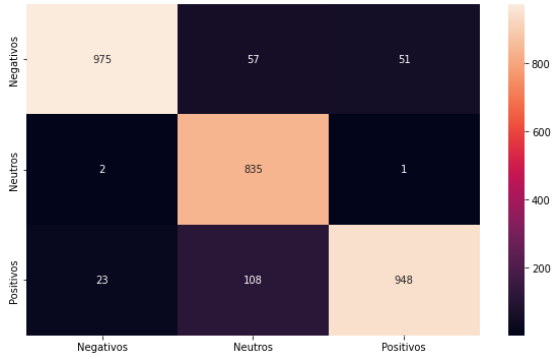


Figura 20: Matriz de confusão do gradiente estocástico

Acurácia media: 0.92

Polaridade	precision	recall	f1-score	support
-1	0.90	0.97	0.94	1000
0	1.00	0.83	0.91	1000
1	0.88	0.95	0.91	1000

Figura 21: Métricas gradiente estocástico

Resultados obtidos considerando dez mil interações máximas:

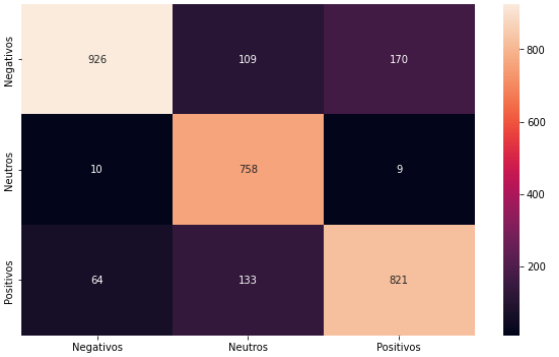


Figura 22: Matriz confusão gradiente estocástico (10000 interações)

Acurácia media: 0.83

K-nearest neighbors.

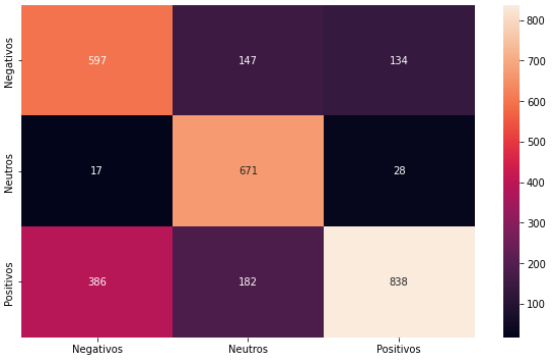


Figura 23: Matriz confusão KNN

Acurácia media: 0.70

Polaridade	precision	recall	f1-score	support
-1	0.68	0.60	0.64	1000
0	0.94	0.67	0.78	1000
1	0.60	0.84	0.70	1000

Figura 24: Métricas KNN

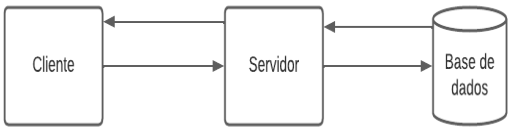
O gradiente descendente estocástico e o algoritmo de árvore de decisão, tiveram larga vantagem sobre os demais modelos, porém, ambos ficaram em uma faixa de precisão muito próxima. Entretanto, o modelo árvore de decisão, se mostrou mais interessante para este trabalho, pois além da leve vantagem de precisão, seu modelo "pickle" ocupa menos espaço no servidor, com isso foi selecionado integrado a aplicação.

6. Criação da aplicação

O processo de criação de da aplicação se deu em 3 partes. Definição de uma arquitetura, criação de um Backend e Frontend. Essa seção descreve em detalhes o passo a passo seguido.

6.1 Arquitetura

Nessa etapa foi utilizada a arquitetura Rest. Cliente-servidor. Onde o cliente envia requisições ao servidor, e a partir dos dados fornecidos o servidor envia uma resposta. Para tal foi criado um



Frontend e um Backend, hospedados em servidores distintos.

Figura 25: Arquitetura da aplicação

6.2 Frontend

O Frontend é a parte cliente dessa arquitetura, onde o usuário interage diretamente com a aplicação. Portanto, uma interface amigável e de fácil entendimento é essencial para o sucesso da mesma. Com isso em mente a ideia foi utilizar bibliotecas que facilitassem o desenvolvimento desta etapa, com o intuito de poder focar mais na usabilidade da aplicação.

Das diversas opções existentes no mercado, React foi selecionado para este trabalho. Sua arquitetura de componentes e facilidade de utilização, além do domínio prévio da tecnologia foram fatores essenciais para sua escolha.

A tela abaixo tem como intuito que o usuário selecione um dos candidatos apresentados ser feita a análise. Essa tela utiliza dois componentes, *NavbarComponent* e *CardComponent*, que representam respectivamente a barra de navegação e os *ca*[9]*rds* que guardam o conteúdo de cada candidato.

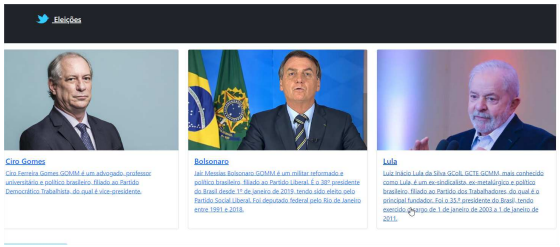


Figura 26: Tela inicial da aplicação

Quando o usuário seleciona um candidato ele é levado a tela abaixo. Essa tela é uma dashboard que apresenta a análise de Tweets de um candidato ao longo do ano, essa tela possui um componente para que o usuário coloque o ano desejado. Logo abaixo da barra de buscas, temos o conteúdo de cada ano separado por abas, cada gráfico mostra a análise de sentimentos para o candidato no ano especificado.

Todos os gráficos foram criados com um módulo chamado chart.js, nele existe uma imensa gama de gráficos para análise técnica. No momento que o usuário dispara a busca por ano, é feita uma solicitação ao servidor e as informações fornecidas são passadas a essa biblioteca. Onde são gerados gráficos em 3 formatos.

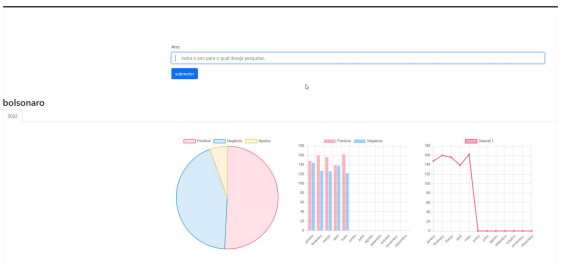


Figura 27: Apresentação de resultados

.

6.3 Backend

O Backend foi feito em Python usando Flask, uma lib para criação de *API 's rest*, em conjunto com isso foi utilizado Firebase para persistência de dados. A linguagem Python por sua vez foi escolhida devido a sua compatibilidade com o modelo do classificador, gerado via jupyter notebook. O Backend da aplicação recebe um ano e um candidato como query params[10] atraves disso para cada mês daquele ano, ele busca e classifica os sentimentos, contudo o processo de busca de Tweets pode ser um pouco custoso em termos de tempo de execução.

Para contornar esse problema foi criado um modulo auxiliar onde todas as requisições que chegam, são buscadas no banco de dados Firebase, onde cada candidato é representado por uma coleção. Cada coleção tem seus sub objetos referentes aos anos, com isso muitas das buscas podem ser resgatadas no banco, diminuindo consideravelmente o tempo de respostas. Em cenários onde não existem dados para o conjunto candidato x ano especificado, são realizadas buscas pelos Tweets e sua análise de sentimentos. Uma resposta é enviada para o usuário e os dados são persistidos para uso futuro.

7. Conclusão

Este trabalho teve como objetivo, criar uma aplicação simples e de fácil acesso que permitisse o internauta realizar consultas sobre a popularidade dos principais candidatos a presidência do Brasil. Utilizando de uma inteligência artificial, implementada com o algoritmo de árvore de decisão e alimentada pelos dados vindos de buscas no Twitter a aplicação conseguiu atingir um grau de 93% de acurácia media, para Tweets cujo nome dos candidatos estejam presentes.

Com a sua finalização, os usuários podem ter acesso às estatísticas de popularidade dos seus candidatos. A Importância deste se dá pela criação de uma nova via para obtenção desses resultados. Permitindo, que o mesmo seja usada no combate a manipulação de pesquisas.

Contudo, a ferramenta também não é livre de manipulações. Devido à popularidade do Twitter, é muito comum o uso de contas falsas, por milícias virtuais para manipular a opinião pública a

[9] container de conteúdo flexível

[10] Parâmetros de buscas passados atraves da url

respeito de um certo tema. Esse problema pode impactar diretamente a validade de alguns resultados obtidos. É importante que o usuário saiba dosar o uso da aplicação, com outros tipos de pesquisa, para garantir uma validade maior das informações obtidas.

Com isto, conclui-se que apesar de apresentar um grau de precisão satisfatório e atender os requisitos estipulados no início do desenvolvimento. O fato da plataforma não apresentar meios de defesa a coleta de dados manipulados, prejudica parcialmente sua validade, o que pode afastar usuários.

7.1 Desafios e limitações

Como mencionado anteriormente, não ter uma forma de lidar com o bots é um dos pontos fracos da inteligência artificial. Contudo, esse não foi unico fator limitante a qualidade do mesmo. O armazenamento de classificações previas teve que ser limitado a 8000 tweets por ano, por candidato, uma vez que qualquer candidato que ultrapasse essa quantidade retira a gratuidade do plano do firebase.
O desempenho da aplicação também foi prejudicado, uma vez que o Heroku, pode oscilar consideravelmente seu tempo de resposta.

7.2 Trabalhos futuros

Devido às limitações apresentadas anteriormente, pretendo otimizar o código para melhorar seu tempo de resposta e desenvolver uma inteligência artificial capaz de identificar Tweets publicados por contas falsas. Aumentando assim a validade e usabilidade da aplicação.

8. Referências

[1] Ablas,Barbara. Relembre a evolução e as mudanças das redes sociais na última década.TechTudo, 13/12/2020. Disponível em:<https://www.techtudo.com.br/noticias/23>.

[2] Ramos,Guilherme.Brasileiros passam mais da metade de suas vidas na Internet, estima pesquisa.TechTudo, 06/05/2022. Disponível em:<https://www.techtudo.com.br/noticias/2022/05/brasileiros-passam-mais-da-metade-de-suas-vidas-na-internet-estima-pesquisa.ghtml> .

[3] Twitter. Disponivel em:<https://twitter.com/>

[4] Abhijit,Roy. Relembre a evolução e as mudanças das redes sociais na última década.Towards Datascience, 13/12/2020. Disponível em:<https://towardsdatascience.com/a-guide-to-text-classification-and-sentiment-analysis-2ab021796317>

[5] Python. Disponível em:< https://www.python.org/>

[6] Javascript. Disponível em: <https://www.javascript.com/>

[7] React. Disponível em: <https://pt-br.reactjs.org/>

[8] Firebase.Disponível em <https://firebase.google.com/?hl=pt>

[9] IMDB. Disponível em <https://www.imdb.com/>

[10] Tweepy. Disponível em <https://www.tweepy.org/>

[11] SnsScrape. Disponível em <https://github.com/JustAnotherArchivist/snscrape>

[12] portuguese-sentiment-analysis. Disponível em <https://www.kaggle.com/datasets/faelk8/portuguese-sentiment-analysis?resource=download>

[13] Sklearn. Disponível em: <https://scikit-learn.org/stable/>

[14] Flask Disponível em <https://flask.palletsprojects.com/en/2.2.x/>

[15] Desconhecido. O que é e como funciona o algoritmo KNN?.Didática Tech, 13/12/2020. Disponível em:https://towardsdatascience.com/a-guide-to-text-classification-and-sentiment-analysis-2ab021796317 . Acesso em: 20/07/2022

[16] Desconhecido.Metrics and scoring: quantifying the quality of predictions scikit-learn., 13/12/2020. Disponível em:https://scikit-learn.org/stable/modules/model_evaluation.htm l.

Sobre o autor:

Leonardo Rodrigues Mota é um estudante do curso de ciência da computação na universidade federal de Campina Grande. Entusiasta de tecnologias web e Consultor na IBM Brasil.