Gabriel Alves Miranda

Detecção de Bots no Twitter Através de Técnicas de Processamento de Linguagem Natural

Niterói - RJ, Brasil

22 de setembro de 2022

Universidade Federal Fluminense

Gabriel Alves Miranda

Detecção de Bots no Twitter Através de Técnicas de Processamento de Linguagem Natural

Trabalho de Conclusão de Curso

Monografia apresentada para obtenção do grau de Bacharel em Estatística pela Universidade Federal Fluminense.

Orientador(a): Prof. Dr. Douglas Rodrigues Pinto

Co-Orientador(a): Msc. Maiara Gripp de Souza

Niterói - RJ, Brasil

22 de setembro de 2022

Universidade Federal Fluminense

Gabriel Alves Miranda

Detecção de Bots no Twitter Através de Técnicas de Processamento de Linguagem Natural

Monografia de Projeto Final de Graduação sob o título *"Detecção de Bots no Twitter Através de Técnicas de Processamento de Linguagem Natural"*, defendida por Gabriel Alves Miranda e aprovada em 22 de setembro de 2022, na cidade de Niterói, no Estado do Rio de Janeiro, pela banca examinadora constituída pelos professores:

Prof. Dr. Douglas Rodrigues Pinto
Departamento de Estatística – UFF

Msc. Maiara Gripp de Souza
Departamento de Estatística – UFF

Profa. Dr. Karina Yuriko Yaginuma
Instituicao do 1º membro da banca

Profa. Dr. Jessica Quintanilha Kubrusly
Instituicao do 2º membro da banca

Niterói, 22 de setembro de 2022

Ficha catalográfica automática - SDC/BIME
Gerada com informações fornecidas pelo autor

M672d Miranda, Gabriel Alves
Detecção de Bots no Twitter Através de Técnicas de Processamento de Linguagem Natural / Gabriel Alves Miranda ; Douglas Rodrigues Pinto, orientador ; Maiara Gripp de Souza, coorientador. Niterói, 2022.
48 f.

Trabalho de Conclusão de Curso (Graduação em Estatística)-Universidade Federal Fluminense, Instituto de Matemática e Estatística, Niterói, 2022.

1. Processamento de Linguagem Natural. 2. Análise Textual. 3. Detecção de Bot. 4. Python. 5. Produção intelectual. I. Rodrigues Pinto, Douglas, orientador. II. Gripp de Souza, Maiara, coorientador. III. Universidade Federal Fluminense. Instituto de Matemática e Estatística. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

Resumo

Bot é definido como um programa capaz de performar atividades de forma automática ou com mínima intervenção humana. Nas redes sociais, o uso dos *bots* como mecanismo para disseminação de *fakenews* tem se mostrado um fenômeno recorrente na atualidade, por exemplo, no campo da disputa política. Desse modo, buscar ferramentas para identificar esses programas se mostra de extrema importância. Nesta monografia apresentamos um algoritmo que buscar identificar esses *bots* em redes sociais, através da análise textual das postagens dos usuários. A Análise Textual nesse contexto visa identificar aspectos como a morfologia e semântica do texto analisado e categorizá-lo. Os métodos serão aplicados nas bases de dados extraídas da rede social Twitter, e que serão analisadas em conjunto com o programa PEGABOT do Instituto de Tecnologia e Sociedade do Rio de Janeiro (ITS Rio) e do Instituto Tecnologia & Equidade. Foram testados três conjuntos de variáveis pelo método *Naive Bayes* e o modelo de classificação com as melhores métricas foi o que não utilizou como variável os *tokens* publicados, atingindo acurácia de 77%, sensibilidade de 80% e especificidade de 63%.

Palavras-chave: Processamento de linguagem natural. Análise Textual. Análise de sentimentos. Detecção de Bot. Python.

Dedicatória

Dedico este trabalho aos meus familiares, namorada e amigos, em especial ao meu falecido avô Enesio Miranda, que muito se orgulhava de seus netos cursarem uma universidade federal.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Desenvolvido entre 1964 e 1966, o primeiro *chatbot* da história foi lançado em 1966. ELIZA, criada por Joseph Weizenbaum, do MIT, buscava emular o papel de uma terapeuta respondendo às perguntas do usuário. Os *chatbots* são *softwares* desenvolvidos para conversar com humanos, com a proposta de se aproximar ao máximo de uma conversa real. Desde então, a busca por uma inteligência artificial capaz de se passar por um humano não cessou. Com o avanço da tecnologia e dos estudos em Processamento de Linguagem Natural (PLN), *bots* cada vez mais avançados foram sendo desenvolvidos. O Processamento de Linguagem Natural compreende, basicamente, a união de técnicas computacionais e linguisticas para análise de textos (conceito esse que será melhor desenvolvido posteriormente neste trabalho). Um dos exemplos de *chatbox* mais conhecidos da atualidade é a Alexa, desenvolvida pela Amazon, que, com o acesso à *Internet*, é capaz de responder perguntas complexas e até contar piadas.

No entanto, os *chatbots* não são os únicos tipos de *bots* de uso corrente. Com a popularização das redes sociais, novos tipos foram desenvolvidos. Desde interação com clientes até responder dúvidas frequentes, os *bots* são muito comuns nas redes. Mas, para além de propósitos comerciais, eles também são muito utilizados com finalidades escusas, como aumentar falsamente o engajamento de perfis e marcas nas redes, aplicar golpes financeiros e até realizar ataques coordenados com inúmeras motivações.

No campo político essas ferramentas também são cada vez mais utilizadas. Ruediger (2017) aponta para um crescimento no uso de contas automatizadas para influenciar o debate público. Segundo seu trabalho, no dia de maior manisfestação registrada a favor do *impeachment* da presidente Dilma Rousseff, cerca de 21% das interações de quem era contrário ao processo foram impulsionadas por robôs. Já do lado favorável, cerca de 16%. Tais dados demonstram como os *bots* podem influenciar no debate público.

Um exemplo de fraude financeira aplicada por *bots* foi o caso das carteiras digitais[1].

[1]Link para uma matéria jornalistica do caso: https://www.tecmundo.com.br/seguranca/230089-bots-twitter-fingem-ajudar-so-querem-roubar-criptomoedas.htm

Os programas buscavam usuários que fizessem algum tipo de reclamação com relação ao serviço de carteiras conhecidas, como a Trust Wallet e a MetaMask, e ofereciam suporte técnico. Passando-se pelo suporte oficial, enviavam um link para uma página falsa, onde a pessoa preencheria seus dados e teria sua carteira digital roubada.

Tendo em vista o cenário apresentado acima, um desafio que se coloca nessa nova realidade social é a detecção desses programas que poluem o ambiente das redes. Detectá-los pode ser uma tarefa um tanto quanto complicada, uma vez que quem cria esses robôs está sempre se atualizando conforme as últimas tecnologias de detecção. Desse modo, o desenvolvimento de novos programas e algoritmos capazes de detectá-los torna-se cada vez mais necessário.

Dentre as diversas abordagens para a detecção, o Processamento de Linguagem Natural pode ser uma ótima ferramenta. Trabalhos como o de Pang et al. (2002) e Dickerson (2014) utilizam técnicas de PLN com técnicas de aprendizado de máquinas para classificação de textos.

A rede social escolhida para desenvolver este trabalho foi o *Twitter*. Por ser baseada em publicações textuais de seus usuários, o *Twitter* é uma boa rede para a aplicação das técnicas de PLN. Além disso, a plataforma oferece ótimas ferramentas para a extração dos seus dados através de um simples cadastro, a depender da finalidade do estudo.

Além disso, o fomento do processamento de linguagem natural na língua portuguesa é outro ponto desta pesquisa. Apesar de existirem boas ferramentas e bons grupos de estudo de PLN em português, a maioria dos trabalhos desenvolvidos nesse ramo é em inglês.

Inspirado por Dickerson (2014) e motivado pelo programa PEGABOT, do Instituto de Tecnologia e Sociedade do Rio de Janeiro (ITS Rio) e do Instituto Tecnologia & Equidade, o presente trabalho busca contribuir para o debate público e para que o ambiente das redes sociais seja cada vez mais democrático e salutar.

Portanto, unindo as técnicas de Processamento de Linguagem Natural e de Aprendizado de Máquinas, este trabalho busca criar um algoritmo capaz de predizer se um perfil no *Twitter* é um bot ou um humano.

1.1 Revisão Bibliográfica

Na literatura científica existem diferentes abordagens para a identificação de *bots* no Twitter. Nesta seção serão apresentados alguns procedimentos e seus resultados, além dos textos que serviram de bases para este trabalho.

Liddy, E.D. (2001) desenvolve a definição de Processamento de Linguagem Natural e apresenta sua história desde as primeiras pesquisas na década de 1940. Elizabeth Liddy também define e apresenta os níveis de análise linguística, sendo eles: o fonológico, que corresponde à interpretação de discursos falados; morfológico, referente à natureza componencial das palavras; lexical, que diz respeito à interpretação do significado das palavras individualmente; sintático, que se trata da análise da estrutura gramatical; semântico, relativo ao significado das sentenças com foco na interação das palavras; discursivo, relacionado à propriedade do texto como um todo pela conexão entre as sentenças; e, por fim, o pragmático, que busca extrair o real significado das sentenças. Além disso, são apresentadas as abordagens para o PLN, são elas: a simbólica, performando profundas análises de fenômenos linguísticos baseados em conhecidos esquemas de representação e algoritmos associativos; a estatística, que emprega técnicas matemáticas e frequentemente usa grandes corpus[2] para desenvolver modelos generalizados aproximados de fenômenos linguísticos, baseados em exemplos reais desses fenômenos fornecidos pelo corpus sem adicionar conhecimento linguístico e a *connectionist*, em suma a união da abordagem simbólica e da estatística.

Dickerson (2014) cria uma arquitetura completa para a detecção de *bots* no Twitter, denominada *SentiBot*, que utiliza quatro grupos de variáveis: sintaxe do tweet, semântica do tweet, comportamento do usuário e métricas da plataforma. Os resultados obtidos mostram que variáveis relacionadas ao sentimento melhoraram significativamente a acurácia do seu modelo. Por exemplo, o resultado obtido na variável *Sentiment flip-flop score*, que mede o número de vezes que um usuário mudou de sentimento sobre determinado tópico, mostra que *bots* raramente mudam de sentimento e humanos tendem a mudar muito mais. Tal trabalho influenciou diretamente na escolha das variáveis utilizadas no grupo B e C deste trabalho.

Wang (2014) foca na detecção de *spammers bots*. *Spammers bots* são ferramentas com a função de publicar *tweets* repetidamente, muitas vezes com o exato mesmo conteúdo, para diversas finalidades, como, por exemplo, influenciar os *trending topics*[3], publicar

[2] conjunto de documentos, ou base de dados, textual.
[3] Temas mais comentados pelos usuários da plataforma.

conteúdo malicioso, etc. Wang define e fundamenta métricas que auxiliam na detecção dos *bots*, que foram inspiração para as métricas escolhidas neste trabalho. Além disso, testa o desempenho de 4 algoritmos de aprendizado de máquinas e conclui que o Naive Bayes tem a melhor performance para a sua base de dados.

Manning (2009) oferece toda a fundamentação teórica, de uma perspectiva da ciência da computação, do classificador *Naive Bayes*, passando pela sua aplicação na classificação de textos até as propriedades do modelo. Além disso, apresenta as duas abordagens mais conhecidas: *Naive Bayes* Bernoulli e *Naive Bayes* Multinomial, e suas vantagens e desvantagens.

Ruediger (2017) realizou 6 estudos de caso referentes a temas políticos entre os anos de 2014 e 2017 e observou que robôs motivam até 20% dos debates públicos no Twitter. Nas eleições presidenciais de 2014, de todas as interações nas horas analisadas, cerca de 11% foram motivadas por *tweets* ou *retweets* de robôs. Já na greve geral de 28 de abril de 2017, dentre os apoiadores, 22,39% das interações foram motivadas por tuítes automatizados. Apontando, assim, para um crescente uso dessas ferramentas para influenciar o debate público.

Alothali (2018) faz um *review* de diversos artigos que tratam de detecção de *bots* no Twitter. Passando pelas bases de dados utilizadas pelos estudos, métodos de detecção como os baseados em grafos, as técnicas de aprendizado de máquinas empregadas e as métricas para medir a performance dos modelos. Tal trabalho influenciou a escolha das variáveis relacionadas às funcionalidades da plataforma e da métrica para avaliar a acertividade do modelo.

1.2 Objetivos

O objetivo deste trabalho é criar um algoritmo capaz de detectar se um usuário do Twitter é bot ou humano, principalmente através do Processamento de Linguagem Natural, para textos em língua portuguesa.

Objetivos Específicos

1. Estudar técnicas de processamento de texto;
2. Estudar o algoritmo de aprendizado de máquinas *Naive Bayes*;
3. Aplicar os conhecimentos estudados neste trabalho em uma base de dados real de

tweets;

4. Criar um *script* em Python capaz de processar os dados e sugerir se o usuário é um *bot* ou um humano.

1.3 Organização

O Capítulo 1 apresenta um breve histórico desde a criação do primeiro *bot*, abordando casos de mal uso destes e as motivações para a criação de mecanismos de detecção. Além disso, neste Capítulo também será feita uma breve revisão bibliográfica a partir dos artigos julgados relevantes para este trabalho .

No Capítulo 2 deste trabalho serão apresentados os materiais utilizados, passando construção da base de dados, seleção de variáveis, técnicas de Processamento de Linguagem Natural, o método *Naive Bayes* com exemplos de aplicação, separação da base entre amostra treino e teste e pelas métricas para avaliação do modelo final.

No Capítulo 3 será feita uma análise descritiva da base de dados de modo a encontrar relação entre as variáveis e a classificação atribuida aos perfis. Além disso serão testadas três abordagens para o treinamento do modelo de predição para detecção de robôs no Twitter.

E, por fim, no Capítulo 4 serão apresentadas as conclusões alcançadas das análises exploratórias e o melhor ajuste, entre as três possibilidades testadas, para o modelo. Além de sugestões para os próximos passos deste trabalho.

2 Materiais e Métodos

Definido por Liddy (2001) como um conjunto de técnicas computacionais para analisar e representar textos que ocorrem naturalmente em um ou mais níveis de análise linguística com o objetivo de obter um processamento linguístico semelhante ao humano para uma diversidade de tarefas ou aplicações, o Processamento de Linguagem Natural é a principal ferramenta computacional para a análise de dados textuais. Algoritmos baseados em PLN capazes de realizar traduções, sumarizações de textos, responder perguntas de clientes em *chats* e analisar sentimentos são extremamente difundidos atualmente.

Observar o texto e extrair dele informações que, de certa forma, estão escondidas em um primeiro olhar pode contribuir para melhores predições. Informações essas como a morfologia e a semântica das palavras, elementos da análise textual. Neste trabalho, a Análise Textual será uma das principais ferramentas utilizadas para detecção de *bots* no Twitter, através de *tweets*, com técnicas de Aprendizado de Máquinas.

As bases de dados podem ser divididas em três categorias: estruturadas, isto é, organizadas segundo um padrão previamente estabelecido, como tabelas e planilhas; não-estruturadas, não organizadas, por exemplo um texto corrido; semi-estruturadas, que carregam características dos dois tipos citados anteriormente, como um arquivo no formato JSON. Os modelos de Aprendizado de Máquinas ditos supervisionados exigem bases de dados estruturadas, pois partem delas para treinar os modelos preditivos. Como textos são dados não-estruturados, algumas manipulações nos textos são necessárias antes desses dados serem utilizados. A seguir, apresentaremos todas as técnicas aplicadas para transformar os dados textuais em um banco de dados estruturado utilizadas neste trabalho.

2.1 Base de Dados

Os dados utilizados neste trabalho foram obtidos através da API do próprio Twitter via *script* desenvolvido na linguagem de programação Python (Van Rossum, 2009). O Python possui diversas bibliotecas que fazem a busca dos *tweets*. Para este trabalho foi

utilizada a biblioteca Tweepy (Roesslein, 2020).

A base de dados será criada a partir de perfis do Twitter que abordem temas políticos. Para a criação da base de dados, primeiro, através das palavras-chave "BolsonaroPresidente", "LulaPresidente", "bolsoladrao", "luladrao", "Lula", "Bolsonaro", foram obtidos os nomes de usuários de perfis que publicaram *tweets* sobre esses assuntos e as suas métricas relacionadas a funcionalidades da plataforma[1]. Dentre essa métricas existe uma indicadora[2] se o perfil é verificado[3] ou não. A verificação do perfil indica que se trata de uma conta de interesse público e autêntica, como governos, organizações jornalísticas ou de entretenimento. As contas com esse selo foram removidas do corpus analisado por não se tratarem de usuários comuns.

Além dos perfis coletados a partir das palavras-chave, alguns perfis foram coletados através de contas *honeypot* (Caverlle, 2011) criadas para atrair perfis automatizados no Twitter. Estas contas foram implementadas por um grupo de alunos da UFF e estão disponíveis no GitHub[4].

Em seguida, foram obtidos os duzentos últimos *tweets* desses usuários, excluindo-se os *retweets*, para comporem a base de dados junto com as métricas da plataforma. Durante a coleta da base foi observado que muitos usuários não tinham duzentos *tweets* escritos por eles mesmos. Desse modo, foram selecionados os usuários que possuiam pelo menos trinta *tweets* próprios. Como esse trabalho trata da análise textual, consideramos que é importante que sejam analisados textos escritos pelos próprios usuários.

Para métodos de aprendizado de máquinas supervisionado[5] se faz necessário bases de dados rotuladas, ou seja, previamente categorizada entre *bot* e não *bot*. Como não tivemos acesso a uma base rotulada em PT-BR, para fins experimentais os perfis foram submetidos ao PEGABOT para serem classificados como *bot* ou humano. O PEGABOT é uma ferramenta do Instituto do Tecnologia e Sociedade do Rio de Janeiro (ITS Rio) e do Instituto Equidade & Tecnologia, com o objetivo de atribuir uma pontuação de 0 a 100, sendo que quanto mais próximo de 100 maior a chance de ser *bot*. Trata-se de uma ferramenta gratuita, que pode ser acessada no https://pegabot.com.br/.

Para minimizar diagnósticos errados pelo PEGABOT, que venham a atrapalhar o modelo desenvolvido neste trabalho, foram usadas duas notas de corte para classificar

[1]Essas métricas são explicadas na seção a seguir.

[2]Variável binária, ou *dummy*, criada para representar uma variável com duas categorias.

[3]Mais informações sobre a verificação: https://help.twitter.com/pt/managing-your-account/twitter-verified-accounts

[4]Link para o GitHub: https://github.com/pmmp-tcc/Estudo-atual

[5]Vide seção 2.4

os usuários da base de dados. Foram considerados *bot* neste trabalho aqueles perfis que atingiram pontuação maior ou igual a 70 e foram classificados como humano os perfis com pontuação menor ou igual a 30.

Por fim, os *tweets* passaram pelas 4 etapas descritas na Seção 2.3. Na etapa de Tokenização(vide Seção 2.3), foram selecionados os 3000 primeiros *tokens* provenientes dos *tweets* para formar o corpus utilizado no treinamento do modelo.

Portanto, a base utilizada neste trabalho possui 7834 usuários do Twitter e 225 variáveis, 200 *tweets* e 24 outras variáveis que serão descritas na Seção 2.2, e pode ser consultada no repositório https://github.com/gamiranda/tcc.

2.2 Variáveis utilizadas

Além das próprias palavras dos *tweets*, outras variáveis podem ser importantes para a detecção dos bots. Nessa seção serão definidas as variáveis utilizadas no modelo preditivo. As variáveis estão divididas em 3 grupos:

1. Morfologia do *tweet*: Esse grupo corresponde à aspectos gramaticais dos *tweets*.
2. Semântica do *tweet*: Esse grupo trata de variáveis relacionadas à aspectos semânticos dos *tweets*.
3. Funcionalidades da plataforma: Esse grupo contém as variáveis relacionadas às funcionalidades do Twitter.

1. Morfologia do *tweet*

A morfologia das palavras diz respeito a sua classificação dentre as classes gramaticais. Foram utilizadas algumas classes gramaticais de modo a verificar as suas ocorrências nos textos analisados. Uma importante contribuição deste trabalho é a verificação da relevância de variáveis morfologias na detecção de *bots*, uma vez que nenhum dos trabalho anteriores de que tivemos ciência fez esse tipo de análise. Essa etapa foi construída utilizando o pacote Spacy do Python.

1. **Número médio de verbos:** é calculado obtendo o número de verbos num tweet e dividindo pelo número de tokens[6] no mesmo tweet.

[6]O conceito de Token será explicado na seção a seguir.

2. **Número médio de advérbios:** é calculado obtendo o número de advérbios num tweet e dividindo pelo número de tokens no mesmo tweet.

3. **Número médio de adjetivos:** é calculado obtendo o número de adjetivos num tweet e dividindo pelo número de tokens no mesmo tweet.

4. **Número médio de pronomes:** é calculado obtendo o número de pronomes num tweet e dividindo pelo número de tokens no mesmo tweet.

5. **Número médio de hashtags:** Esse é calculado pelo número de hashtags num tweet dividido pelo número de tokens no mesmo tweet.

6. **Número médio de emojis:** Esse é calculado pelo número de emojis num tweet dividido pelo número de tokens no mesmo tweet.

2. Semântica do *tweet*

O aspecto semântico de uma palavra está relacionado ao seu sentido e a interpretação desse sentido. Desse modo, foi utilizado, através do dicionário léxico LeIA, o cálculo do *score* de sentimento. O LeIA (Léxico para Inferência Adaptada) é uma versão em português do VADER (Valence Aware Dictionary and sEntiment Reasoner) com foco em análise de textos de redes sociais, contando com suporte para emoji e palavras abreviadas, que são estruturas comuns nesses ambientes. Este dicionário retorna 4 métricas ao analisar uma frase: porcentagem do texto com sentimento positivo, porcentagem do texto com sentimento negativo, porcentagem do texto com sentimento neutro e um valor de sentimento geral normalizado variando entre -1 (extremamente negativo) a +1 (extremamente positivo). Para determinar o sentimento do *tweet* analisado foi considerada a maior porcentagem. Por exemplo, a frase "Eu estou feliz" retorna negativo = 0,0, neutro = 0,297 e positivo = 0,703. Como a porcentagem de positivo foi maior, essa frase será considerada positiva. No entanto esse dicionário possui um problema, talvez por se tratar de uma adaptação: frase em que se esperaria um sentimento positivo maior nem sempre correspondem a expectativa. Enquanto, por exemplo, a frase "Eu estou feliz" retorna as métricas apresentadas, a frase "Eu estou muito feliz" retorna negativo = 0,0, neutro = 0,423 e positivo = 0,577. Ou seja, é considerada "menos positiva". Feita esta ressalva, com os *scores* obtidos foi possível calcular os seguintes indicadores:

1. **Número de *tweets* com score de sentimento positivo:** Frequência absoluta de *tweets* com *score* positivo para cada usuário.

2. **Número de *tweets* com score de sentimento negativo:** Frequência absoluta de *tweets* com *score* negativo para cada usuário.

3. **Número de *tweets* com score de sentimento neutro:** Frequência absoluta de *tweets* com *score* neutro para cada usuário.

4. **Número de trocas de sentimento:** Quantas vezes o sentimento foi de positivo para negativo ou neutro, e vice-versa, entre os *tweets* analisados.

3. Funcionalidades da plataforma

Por fim, também foram utilizadas as informações disponíveis na base referentes a algumas funcionalidades do Twitter para obter os seguintes indicadores. As métricas escolhidas foram baseadas nos artigos de Dickerson (2014) e Alothali (2018).

1. **Número de seguidores:** Trata-se do número de seguidores que o usuário possui na sua conta.

2. **Número de seguidos:** Trata-se do número perfis seguidos pelo usuário na sua conta.

3. **Número de *tweets*:** Esse é o número de *tweets* do usuário desde que começou a utilizar o Twitter, contabilizando *tweets*, *retweets* e respostas.

4. **Se possui imagem de perfil:** Indicadora[7], se o perfil possui foto de perfil ou não.

5. **Ano de criação da conta:** Trata-se do ano em que a conta foi criada.

2.3 Processamentos dos Dados

O Processamento dos Dados corresponde à preparação dos dados para a criação do modelo preditor. O processamento adequado pode ser a diferença entre um modelo efetivo ou não. A seguir estão descritas as técnicas utilizadas nesse trabalho.

A **tokenização** é o processo de transformar uma frase em um vetor, onde cada posição desse vetor será ocupada por cada palavra ou pontuação presente na frase. Trata-se do primeiro passo no processamento dos dados textuais, de modo que esse vetor será utilizado para as etapas seguintes.

[7]Variável binária, ou *dummy*, criada para representar uma variável com duas categorias.

Exemplo 2.3.1. *Dada a frase:*

"Estou escrevendo um TCC para a conclusão do curso de Estatística, sobre Análise de Sentimentos. Meus amigos e amigas estão me ajudando."

Sua versão tokenizada será:

[Estou, escrevendo, um, TCC, para, a, conclusão, do, curso, de, Estatística, ",", sobre, Análise, de, Sentimentos, ".", Meus, amigos, e, amigas, estão, me, ajudando, "."]

Cada posição desse vetor é chamada *Token*. No exemplo acima, a frase foi separada em 25 *tokens*, considerando tanto as palavras quanto a pontuação.

Pode-se notar no **Exemplo 2.3.1** o uso de palavras de mesmo significado, ou semelhante, com diferentes declinações e gênero. As palavras "amigos", "amigas", "amigo", "amiga", por exemplo, carregam significado muito similar. Assim, reduzir esses quatro exemplos para uma única forma reduziria a complexidade da base sem perder informação. Para tanto, este trabalho utiliza a lemmatização.

Lemmatização é o processo de simplificação dos *tokens*, de modo que declinações, flexões, etc são substituídos por formas mais simples das respectivas palavras. Existem bibliotecas [8] em Python capazes de fazer esse processo de lemmatizar palavras. Neste trabalho foi utilizado o "pt_core_news_lg"da biblioteca *SPACY* (Honnibal, Montani, 2017), que possui uma acurácia de 77% nas lemmatizações [9].

Exemplo 2.3.2. *Dados os tokens:*

[Estou, escrevendo, um, TCC, para, a, conclusão, do, curso, de, Estatística, ",", sobre, Análise, de, Sentimentos, ".", Meus, amigos, e, amigas, estão, me, ajudando, "."]

Sua versão lemmatizada será:

[Estou, escrever, um, TCC, para, o, conclusão, do, cursar, de, Estatística, ",", sobre, Análise, de, Sentimentos, ".", Meus, amigo, e, amigo, estar, me, ajudar, "."]

[8] Bibliotecas no Python são conjuntos de códigos já desenvolvidos para realizar determinada tarefa.
[9] Link para a acurácia: https://spacy.io/models/pt

No **Exemplo 2.3.2** pode-se observar a ocorrência de palavras que não carregam consigo um real significado/sentimento, com "o", "de", "e", capazes de ajudar a criar melhores algoritmos de predição. Essas palavras são chamadas de *stop words*.

Stop words são palavras de uso comum a qualquer idioma, contribuem para a construção de sentido nas frases, mas por si só não constituem itens lexicais e, portanto, são opacas semanticamente. Preposições e artigos, são exemplos *stop words*. É uma boa prática remover esse conjunto de palavras para criar os modelos preditivos de textos, de modo que tais termos não interfiram nas predições.

Além de serem removidas as pontuações, também foram removidos caractéres especiais e números, estruturas muito comuns em *tweets* do *Twitter*, também foram removidas as palavras-chaves usadas para obtenção da base de dados, já que seriam palavras comuns entre os dois tipos de perfil classificados.

Exemplo 2.3.3. *Dados os tokens:*

[Estou, escrevendo, um, TCC, para, a, conclusão, do, curso, de, Estatística, ",", sobre, Análise, de, Sentimentos, ".", Meus, amigos, e, amigas, estão, me, ajudando, "."]

Sua versão após a remoção de stopwords será:

[escrever, TCC, conclusão, cursar, Estatística, Análise, Sentimentos, amigo, amigo, ajudar]

Para que o computador seja capaz de ler esses tokens, eles devem ser transformados em números. A técnica utilizada neste trabalho foi a criação de um *word index*.

Word Index é um dicionário criado a partir do conjunto de palavras da base de dados. Os tokens são separados e lhes é atribuido um número. Em seguida, todos os *tokens* do texto a serem analisados devem ser transformados conforme o número associado a palavra. Assim, quando o modelo preditivo for criado, as palavras dos *tweets* serão os números do dicionário.

Desse modo, um *tweet* será transformado em uma matriz de x colunas[10], correspondente ao número de palavras num *tweet*, e n linhas, correspondente ao número de observações na amostra. Cada posição dessa matriz é relativa ao *token* transformado em número pelo *word index*.

[10] O número de colunas resultantes é arbitrário, podendo ser maior ou menor a depender do problema observado.

Exemplo 2.3.4. *Dados os tokens:*

[escrever, TCC, conclusão, cursar, Estatística, Análise, Sentimentos, amigo, amigo, ajudar]

seu Word Index será:

Tabela 1: Word Index do Exemplo 2.4

Token	Word Index
escrever	1
TCC	2
conclusão	3
cursar	4
Estatística	5
Análise	6
Sentimentos	7
amigo	8
ajudar	9

e a frase, que tem 10 tokens com 9 tokens distintos, será uma matriz 1×10*:*

$$[1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 8\ 9]$$

de modo que se a frase tiver um número de tokens menor que o número de colunas preestabelecido, são preenchidos 0 *no início da linha para completar a matriz. Caso contrário, os primeiros tokens são removidos para que o tweet fique do tamanho escolhido. Vide exemplos 2.3.5 e 2.3.6*

Exemplo 2.3.5. *Dados os tokens:*

[escrever, TCC, conclusão, cursar, Estatística, Análise, Sentimentos]

seu Word Index será:

e a frase, que tem 7 tokens com 7 tokens distintos, será uma matriz 1×10*:*

$$[0\ 0\ 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7]$$

Exemplo 2.3.6. *Dados os tokens:*

[escrever, TCC, conclusão, cursar, Estatística, Análise, Sentimentos, amigo, amigo, ajudar, cursar, Estatística]

Tabela 2: Word Index do Exemplo 2.4

Token	Word Index
escrever	1
TCC	2
conclusão	3
cursar	4
Estatística	5
Análise	6
Sentimentos	7

Tabela 3: Word Index do Exemplo 2.4

Token	Word Index
escrever	1
TCC	2
conclusão	3
cursar	4
Estatística	5
Análise	6
Sentimentos	7
amigo	8
ajudar	9

seu Word Index será:

e a frase, que tem 12 tokens e 9 tokens distintos, será uma matriz 1×10*:*

$$[3\ 4\ 5\ 6\ 7\ 8\ 8\ 9\ 4\ 5]$$

2.4 Naive Bayes

Naive Bayes (NB) é um método supervisionado de Aprendizado de Máquinas do grupo dos classificadores probabilísticos (Medhat, 2014) baseado na aplicação do Teorema de Bayes e na presunção de independência condicional entre as variáveis explicativas. O termo *Naive*, ingênuo em inglês, decorre dessa hipótese de independência, que por vezes pode ser muito forte. Com ótimos resultados em Wang (2014), NB é um método de simples aplicação e com baixo custo computacional se comparado a outros modelos.

Os métodos de aprendizado supervisionado dependem da existência de bases de dados estruturadas previamente categorizadas para o treinamento do modelo preditivo (Medhat, 2014).

Seja C uma variável resposta, podendo assumir os valores 0 ou 1, sejam $X_1, \ldots, X_n$

variáveis explicativas e

$$P(C|X_1, \ldots, X_n) \,, \tag{2.1}$$

a probabilidade de C dado $X_1, \ldots, X_n$.

Pelo Teorema de Bayes, temos que

$$P(C|X_1, \ldots, X_n) = \frac{P(C)P(X_1, \ldots, X_n|C)}{P(X_1, \ldots, X_n)} \,, \tag{2.2}$$

sendo $P(C)$ é a probabilidade *a priori* de ocorrer a variável C, $P(X_1, \ldots, X_n|C)$ é a verossimilhança de $X_1, \ldots, X_n$ dado C e $P(X_1, \ldots, X_n)$ é a probabilidade conjunta das variáveis explicativas.

Veja que como o denominador não depende de C e os valores de X_i são sempre dados, então é suficiente considerar apenas o numerador (ROCHA, 2011). Logo, temos

$$P(C|X_1, \ldots, X_n) \propto P(C)P(X_1, \ldots X_n|C) \,, \tag{2.3}$$

i.e. a distribuição *a posteriori* de C dado $X_1, \ldots, X_n$ é proporcional à *priori* de C vezes a verossimilhança de $X_1, \ldots, X_n$ dado C.

Note que,

$$\begin{aligned} P(C)P(X_1, \ldots, X_n|C) &= P(C)P(X_1|C)P(X_2, \ldots, X_n|C, X_1) \\ &= P(C)P(X_2|C, X_1)P(X_3, \ldots, X_n|C, X_1, X_2) \\ &\ldots \\ &= P(C)P(X_1|C) \ldots P(X_n|C, X_1, \ldots, X_{n-1}) \,. \end{aligned}$$

Portanto,

$$P(C|X_1, \ldots, X_n) \propto P(C)P(X_1|C) \ldots P(X_n|C, X_1, \ldots, X_{n-1}) \,. \tag{2.4}$$

A aplicação do NB pressupõe que, dado C, cada variável X_i é condicionalmente independente de X_j, para $i \neq j$, então

$$P(X_i|C, X_j) = P(X_i|C) \,.$$

Substituindo na Equação 2.4, o modelo pode ser expresso como:

$$P(C|X_1, \ldots, X_n) \propto P(C) \prod_{i=1}^{n} P(X_i|C) \ . \tag{2.5}$$

Estimação dos Parâmetros

Para estimar $P(C)$ pode-se assumir que as classes são uniformemente distribuídas, de modo que $\hat{P}(C = 1) = \hat{P}(C = 0) = 1/2$, ou pode-se calcular a frequência relativa da classe na base de treino, onde $\hat{P}(C = 1) =$ (número de observações 1)/(número total de observações) e $\hat{P}(C = 0) =$ (número de observações 0)/(número total de observações). Já para estimar $P(X_i|C)$ existem duas formas: podem ser atribuidas distribuições de probabilidade, de forma que se X_i é uma variável discreta, comumente assume-se a distribuição multinomial; já se X_i é uma variável continua, assume-se a normalidade (Veja Exemplo 2.4.1); ou, utilizar a frequência relativa de ocorrência do evento na base de treinamento, vide Exemplo 2.4.2. Esse segundo caso é mais comum em problemas de classificação de textos.

Correção da Amostra

No caso de se estimar $P(X_i|C)$ pela frequência relativa é possível que algum valor de C ou de alguma variável explicativa não ocorra na base de treinamento de modo que tenhamos $\hat{P}(X_i|C) = 0$. Tal situação excluiria toda informação das outras variáveis no momento da multiplicação das probabilidades associadas àquele caso. Para evitar esse tipo de perda, utiliza-se a correção de LaPlace, onde é somado o valor 1 em cada X_i de modo que $\hat{P}(X_i|C)$ nunca será zero.

Construção do Classificador

Com o modelo estimado, utiliza-se a regra da *a posteriori* máxima (MAP) para classificar uma observação. Portanto a classe escolhida para classificar uma observação será a de maior probabilidade:

$$\hat{y} = \underset{c \in \{0,1\}}{argmax} P(C = c) \prod_{i=1}^{n} P(X_i = x_i|C = c) \ .$$

Exemplo 2.4.1. *Considera uma variável contínua X_i. Primeiro a base é separada em relação a C e então são estimadas a média e a variância de X_i para cada $c \in \{0, 1\}$. Seja $\bar{x}_c$ a média dos valores de X_i classificados como c e S_c^2 a variância dos valores de X_i classificados como c. Então $P(X_i \leq x_i|c)$ pode ser calculada assumindo a distribuição*

Normal da seguinte forma:

$$P(X_i \leq x_i | C = c) = \int_{-\infty}^{x_i} \frac{1}{\sqrt{2\pi S_c^2}} \exp\{-\frac{1}{2S_c^2}(x_i - \bar{x}_c)^2\}\, dx$$

Exemplo 2.4.2. *Seja a seguinte base de dados categorizada*[11]*:*

Tabela 4: Base de dados de exemplo

	IDobservação	Tokens na observação	c = China = 1
amostra treino	1	Chinese Beijing Chinese	1
	2	Chinese Chinese Shanghai	1
	3	Chinese Macao	1
	4	Tokyo Japan Chinese	0
amostra teste	5	Chinese Chinese Chinese Tokyo Japan	?

Sejam as probabilidades *a priori* $\hat{P}(C = 1) = 3/4$ e $\hat{P}(C = 0) = 1/4$ e as seguintes probabilidades condicionais, calculadas pela frequência relativa dos termos na base de treinamento:

$$\begin{aligned}
\hat{P}(Chinese|C = 1) &= (5 + 1)/(8 + 6) = 3/7 \\
\hat{P}(Tokyo|C = 1) &= \hat{P}(Japan|C = 1) = (0 + 1)/(8 + 6) = 1/14 \\
\hat{P}(Chinese|C = 0) &= (1 + 1)/(3 + 6) = 2/9 \\
\hat{P}(Tokyo|C = 0) &= \hat{P}(Japan|C = 0) = (1 + 1)/(3 + 6) = 2/9\ .
\end{aligned}$$

Onde o numerador é o número de ocorrências da palavra mais a correção de LaPlace e o denominador é a quantidade de palavras na classe mais o tamanho do vocabulário da amostra treino. Logo, para a amostra teste temos:

$$\begin{aligned}
\hat{P}(C = 1|ID_5) &\propto 3/4 \times (3/7)^3 \times 1/14 \times 1/14 \approx 0.0003 \\
\hat{P}(C = 0|ID_5) &\propto 1/4 \times (2/9)^3 \times 2/9 \times 2/9 \approx 0.0001\ .
\end{aligned}$$

Como $1 = \underset{c \in \{0,1\}}{\arg\max} \hat{P}(C = c|ID5)$ então atribuimos a classe 1 à observação 5.

[11] Exemplo retirado de Manning(2008).

2.5 Amostra Treino, Teste e Validação

Para criação e avaliação do modelo desenvolvido, a base de dados foi dividida em três partes: uma amostra de treino para desenvolver o algoritmo de predição; uma amostra de teste para realizar o teste do modelo e de diferentes parâmetros para buscar o melhor ajuste; uma amostra de validação para obter as métricas de avalição final.

2.6 Avaliação do Modelo

2.6.1 Matriz de Confusão

A Matriz de Confusão é a matriz construída após as previsões para avaliar os resultados do modelo, as colunas correspondem aos valores reais e as linhas correspondem aos valores preditos.

Tabela 5: Matriz de Confusão

		Valores Preditos	
		Positivo	Negativo
Valores Reais	Positivo	Verdadeiro Positivo	Falso Negativo
	Negativo	Falso Positivo	Verdadeiro Negativo

Verdadeiro Positivo (VP): A quantidade de observações na amostra preditas como positivas que são verdadeiramente positivas;

Verdadeiro Negativo (VN): A quantidade de observações na amostra preditas como negativas que são verdadeiramente negativas;

Falso Positivo (FP): A quantidade de observações na amostra preditas como positivas que são negativas;

Falso Negativo (FN): A quantidade de observações na amostra preditas como negativas que são positivas.

Exemplo 2.6.1. *Exemplo de Matriz de Confusão para dados ficticios.*

Tabela 6: Matriz de Confusão para ocorrência de determinada doença

		Valores Preditos	
		Possui a Doença	Não Possui a Doença
Valores Reais	Possui a Doença	132	12
	Não Possui a Doença	32	50

Através da matriz, é possível verificar visualmente a acertividade do modelo. A partir dos dados da Matriz de Confusão, as seguintes métricas são calculadas:

2.6.2 Acurácia

A acurácia avalia a taxa de acerto geral do modelo, definida da seguinte forma:

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \, . \tag{2.6}$$

Exemplo 2.6.2. *Com base na Matriz de Confusão do Exemplo 7, a Acurácia será:*

$$Acurácia = \frac{132 + 50}{132 + 50 + 32 + 12} = 0,8053$$

2.6.3 Especificidade

A Especificidade é a taxa de predições negativas corretas dentre as observações preditas como negativas, definida pela seguinte fórmula:

$$Especificidade = \frac{VN}{VN + FN} \, . \tag{2.7}$$

Exemplo 2.6.3. *Com base na Matriz de Confusão do Exemplo 7, a Especificidade será:*

$$Especificidade = \frac{50}{50 + 12} = 0,8064$$

De 62 vezes que o modelo predisse que o paciente não tinha determinada doença, 50 vezes acertou. Assim, o modelo acerta a classe negativa 80,64% das vezes

2.6.4 Sensibilidade

A Sensibilidade, ou Recall, mede a acertividade dentre as observações positivas. Em outras palavras, quando a classe verdadeira era positiva, com que frequência o modelo predisse como positiva.

$$Sensibilidade = \frac{VP}{VP + FP} \, .$$

Exemplo 2.6.4. *Com base na Matriz de Confusão do Exemplo 7, a Sensibilidade será:*

$$Sensibilidade = \frac{132}{132 + 32} = 0,8049$$

Para os 144 casos de pacientes que possuem a doença, o modelo foi capaz de predizer 132 desses. Ou seja, o algoritmo detectou 80,49% dos casos positivos.

2.7 Princípio FAIR

Inspirado por Mondelli, Peterson e Gadelha (2019), este trabalho segue o princípio, por eles proposto, FAIR. Sigla para *Findable*, *Accessible*, *Interoperable*, *and Reusable* (Encontrável, Acessível, Interoperável e Reutilizável, respectivamente). Este princípio busca tornar os trabalhos reprodutíveis para quem os acessa no futuro.

Para tanto, toda a etapa de construção da base de dados e construção dos modelos estão descritos nas suas respectivas seções. Os *scripts* utilizados na criação da base de dados, das análises exploratórias e dos modelos podem ser consultados no repositório do *github* (MIRANDA, 2022).. Além disso, no Apêndice 2 estão descritas as especificações da linguagem de programação e bibliotecas utilizadas.

3 Análise dos Resultados

Neste capítulo serão apresentadas as análises descritivas da base de dados gerada para este trabalho, além de três diferentes ajustes do Naive Bayes para detecção de *bots* no Twitter.

3.1 Pré-Processamento

Para o pré-processamento da base de dados foram realizadas as etapas descritas na Seção 2.3. Primeiro, os 200 *tweets* mais recentes de cada usuário foram concatenados em uma única variável para compor o corpus deste trabalho. Nesse processo de concatenação, os *tweets* foram tokenizados e lemmatizados. Em seguida, foram removidas todas as *stop words*, caractéres especiais, pontuações e números (veja a Seção 2.3).

Nas redes sociais os emojis são ferramentas de comunicação muito difundidas e utilizadas, logo essas estruturas foram utilizadas para as análises. Assim, os emojis passaram por um processo de tradução[1] de modo que cada imagem recebeu um nome único. O emoji de coração vermelho[2], por exemplo, ficou traduzido no corpus como *red_heart*.

3.2 Análise Descritiva

A base de dados utilizada neste trabalho conta com 7.834 observações, isto é, contas distintas no Twitter, dos quais 2.301 foram considerados *bots* e 5.533 não-*bots* pelo critério de rotulação descrito na Seção 2.1. Note que a proporção entre bots e não bots na base de dados está bem desbalanceada, mas visto que a maior parte dos usuários do Twitter não são *bots*, esse resultado era esperado.

[1]Link para o dicionário de emojis: https://github.com/gamiranda/tcc/blob/main/Emoji_Dict.p
[2]Link para o emoji: https://emojipedia.org/red-heart/

3.2.1 Análise da Morfologia

Separando os dados com relação à variável resposta foi possível notar diferenças no comportamento dos dois tipos de usuários. Na Tabela 7 vemos que, em média, 9,05% dos *tokens* escritos por *bots* foram verbos, quando essa proporção entre os usuários humanos foi de cerca de 11,5% (veja a Figura 1 e Tabela 7).

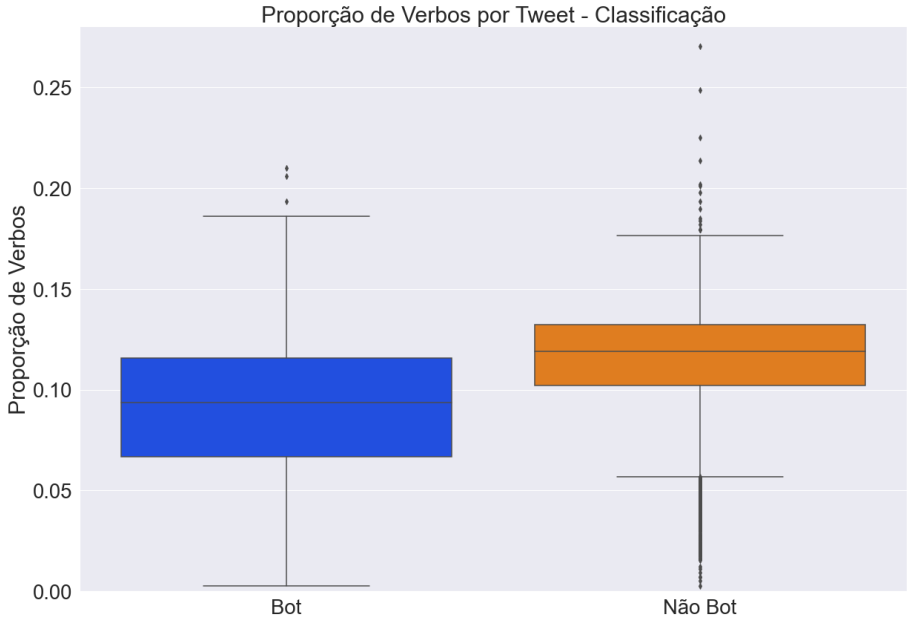

Figura 1: Box Plot da Proporção de Verbos por Tweet - Classificação.

Além disso, perfis humanos tiveram uma proporção de pronomes entre seus *tokens* levemente maior do que dos *bots* (Figura 2). No entanto, a proporção de adjetivos entre os dois tipos de perfis foi bem similar, indicando que pode não ser uma variável informativa para o modelo (vide Apêndice 1).

Por fim, também foi observado que, em média, os *bots* publicam mais emojis do que os perfis humanos. Cerca de 4% dos *tokens* publicados por *bots* foram emoji, enquanto para não *bots* foi de 2% (vide a Tabela 7).

A Tabela 7 abaixo mostra a proporção média de tipos morfológicos encontrados nos *tweets* analisados para cada classificação.

Além disso, também foram analisados os *tokens*[3] utilizados por cada tipo de perfil. Veja, na Figuras 3, que o *token* mais frequente nos dois casos foi um emoji. O "*face_with_tears_of_joy*"[4] para os *bots* e "*rolling_on_the_floor_laughing*"[5] para os não *bots*.

[3]Chamo de *tokens* e não palavras porque existem estruturas como hashtags, arrobas, emoji, etc.
[4]Link para o emoji: https://emojipedia.org/face-with-tears-of-joy/
[5]Link para o emoji: https://emojipedia.org/rolling-on-the-floor-laughing/

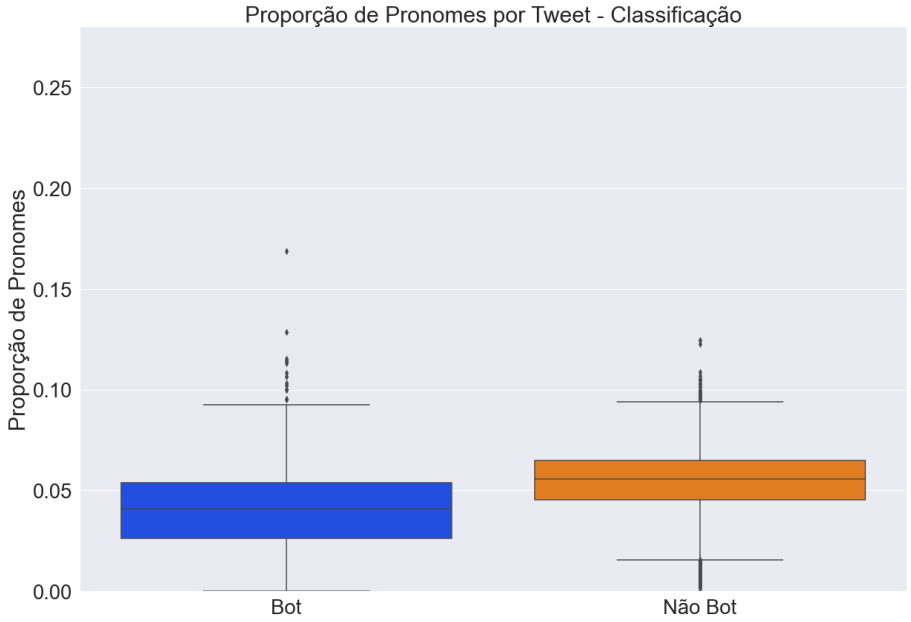

Figura 2: Box Plot da Proporção de Pronomes por Tweet - Classificação.

Tabela 7: Comparação da Proporção de Tipos Morfológicos - Bot e Não Bot

Bot		
Tipo Morfológico	Média	Mediana
Verbo	9,05	9,36
Advérbio	3,27	3,10
Adjetivo	5,22	4,83
Pronome	4,06	4,08
Hashtag	1,77	0,27
Emoji	4,21	1,70

Não Bot		
Tipo Morfológico	Média	Mediana
Verbo	11,47	11,88
Advérbio	4,69	4,71
Adjetivo	5,27	5,14
Pronome	5,42	5,57
Hashtag	0,47	0,12
Emoji	2,39	0,88

Outra coisa observada foi que o emoji mais comum entre os não *bots* não está entre os 30 *tokens* mais usados por *bots*, podendo ser um bom diferenciador entre as duas classificações. Por fim, dentre os 30 *tokens* mais frequentes entre os *bots*, 8 são emojis. Enquanto para humanos 3 dos 30 mais frequentes são emojis.

Por fim, uma ocorrência comum entre os *bots* e não muito vista entre humanos é a sigla "sdv", que significa "sigo de volta". Os *bots* seguem muitos perfis e buscam seguidores para conseguir maior engajamento nas suas publicações. Estratégia essa que parece funcionar, vide Tabela 10, onde pode-se ver que *bots* possuem, em média, 4 vezes mais seguidores do que os humanos.

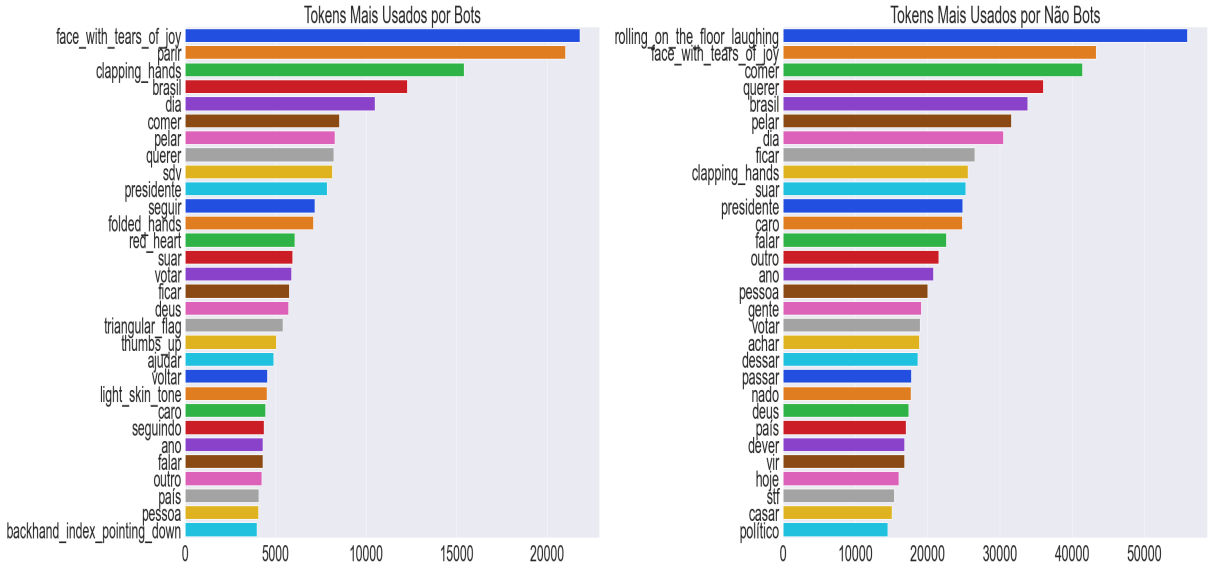

Figura 3: Gráfico de Barras das Palavras mais Usadas por Bots.

3.2.2 Análise da Semântica

Com relação ao sentimento transmitido pelo *tweet*, em ambos os casos o predominante foi o neutro. Além disso, em média, os *bots* tiveram mais *tweets* negativos do que os humanos e a média de textos com sentimento positivo foi similar nos grupos como pode ser visto na Tabela 8.

Onde foi possível observar maior diferença entre *bots* e não *bots* foi com relação a troca de sentimentos. Como pode ser visto na Figura 4 e na Tabela 8, *bots* tendem a trocar menos de sentimento, o que também foi observado por Dickerson (2014) com relação a sua variável *Sentiment flip-flop*.

As Tabelas 8 e 9 mostram, respectivamente, a média e mediana dos sentimentos considerados para este trabalho e da troca de sentimento nos últimos 200 *tweets*.

Tabela 8: Média dos sentimentos

Classificação	Positivo	Negativo	Neutro	Troca de Sentimento
Bot	5	11	184	15
Humano	6	8	186	21

Tabela 9: Mediana dos sentimentos

Classificação	Positivo	Negativo	Neutro	Troca de Sentimento
Bot	2	3	193	10
Humano	4	5	190	19

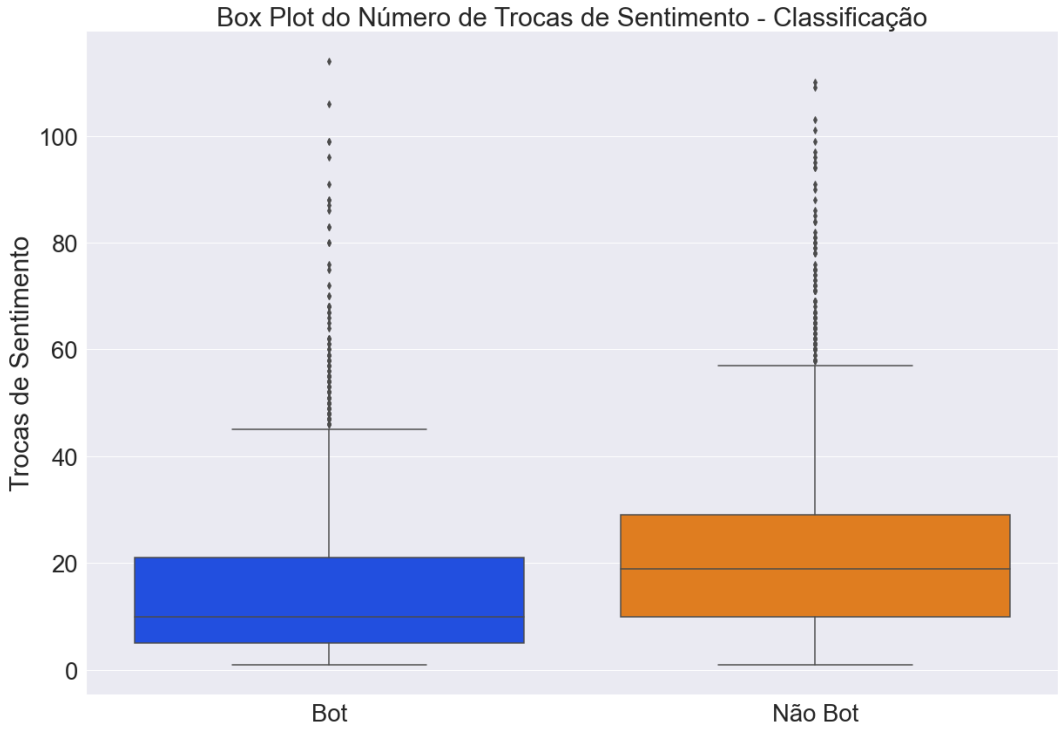

Figura 4: Box Plot do Número de Trocas de Sentimentos por Classificação.

3.2.3 Análise das Métricas da Plataforma

Com relação as métricas da plataforma as diferenças entre *bots* e perfis reais são mais visíveis. Em média, os *bots* possuem 4 vezes mais seguidores do que os humanos e são seguidos por, também, 4 vezes mais contas (vide Tabela 10). Além disso, a mediana tanto para o número de seguidores quanto para o número de seguidos é maior no grupo dos *bots*, indicando uma maior prevalência de seguidores e seguidos para os assim classificados. Além disso, os perfis classificados como não *bots* apresentaram mais *outliers*.

No entanto, a maior diferença entre os dois tipos de usuários foi com relação ao número de *tweets* publicados. Os *bots* possuíram, em média, 10 vezes mais *tweets* do que os perfis reais. Pode-se observar na Figura 7 que a mediana do número de *tweets* dos *bots* é maior que o terceiro quartil dos não *bots*, apontando para, de fato, um maior número de publicações[6] pelos *bots*.

As informações de média e mediana com relação às métricas de funcionalidades da plataforma estão dispostas nas Tabelas 10 e 11.

Já, com relação ao ano de criação dos perfis, a Figura 8 deixa clara uma diferença

[6]Chamo de publicações porque, como definido na seção 2.2 essa variável leva em consideração *tweets*, *retweets* e respostas.

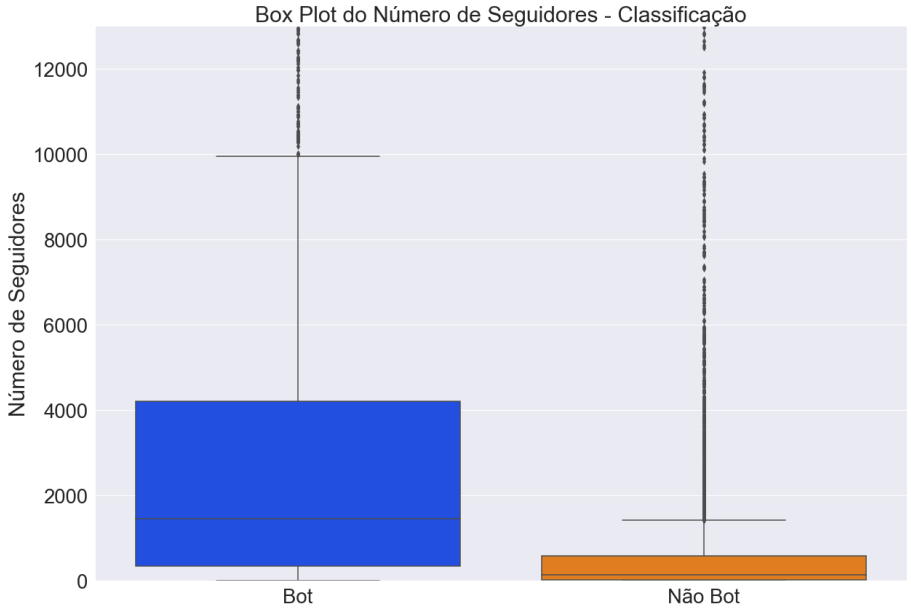

Figura 5: Box Plots do Número de Seguidores e Seguidos por Classificação.

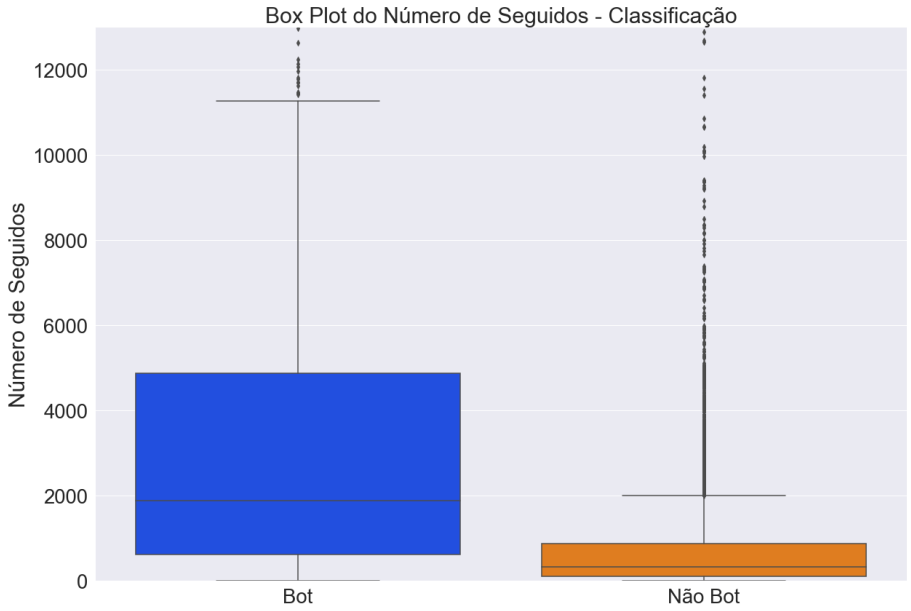

Figura 6: Box Plots do Número de Seguidos por Classificação.

Tabela 10: Média das métricas de plataforma

Classificação	Nº de Seguidores	Nº de Seguidos	Nº de Tweets
Bot	4748	4173	73625
Humano	1157	890	7046

importante entre os dois tipos. A maior parte das contas criadas por bots são novas. Na Figura 8 podemos verificar que mais de 60% dos perfis de bots analisados foram criados depois de 2020, enquanto que o percentual de perfis de humanos analisados criados depois de 2020 foi de apenas 35%.

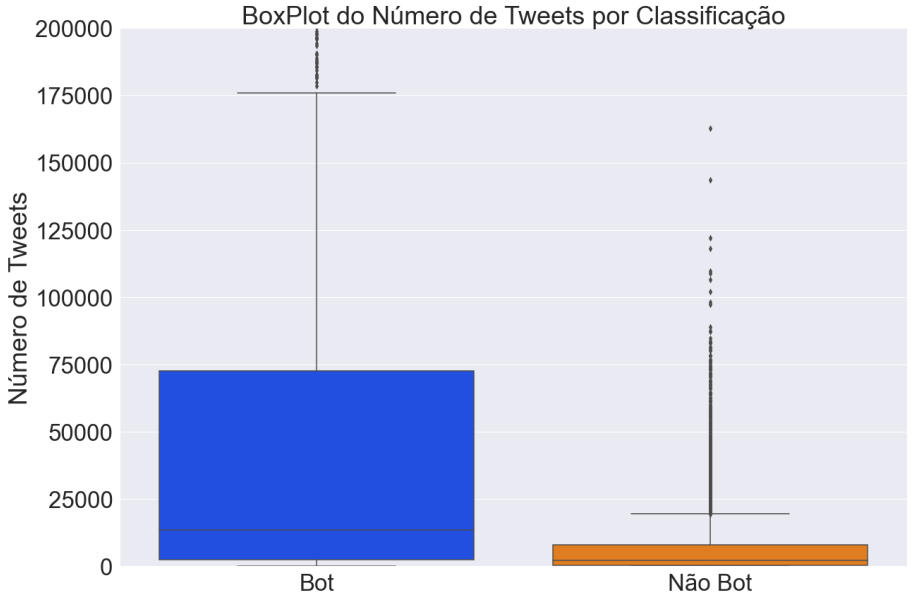

Figura 7: Box Plot do Número de Tweets por Classificação.

Tabela 11: Mediana das métricas de plataforma

Classificação	Nº de Seguidores	Nº de Seguidos	Nº de Tweets
Bot	1457	1878	13435
Humano	143	331	2147

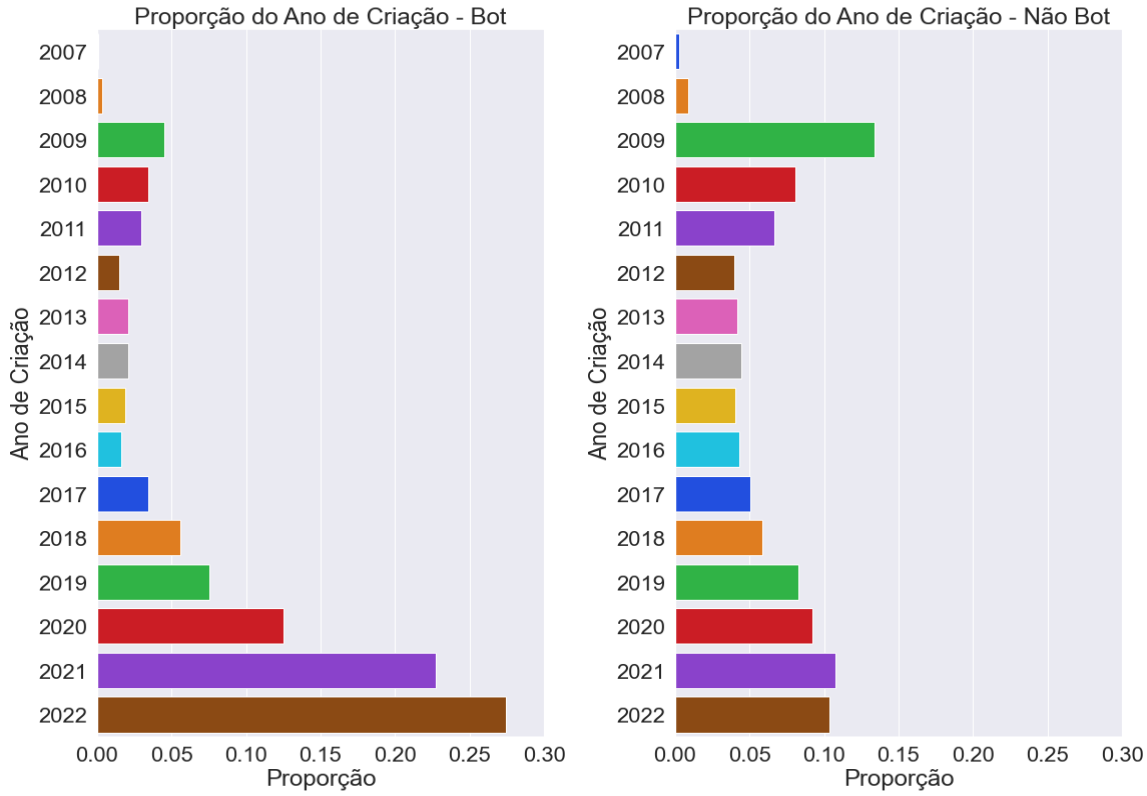

Figura 8: Gráfico de Barras do Ano de Criação das Contas Bot.

3.3 Modelagem

A partir da base utilizada na análise exploratória, os *tweets* concatenados foram separados em 3.015 variáveis, contando com a variável resposta, levando em consideração os 3.000 *tokens* mais recentes de cada perfil. Junto desse processo foi criado o *Word Index*. As etapas descritas neste paragráfo foram realizadas utilizando a biblioteca Keras.

Após todas as etapas do pré-processamento, a base de dados ficou com 3.015 variáveis.

Como definido no Capítulo de Materiais e Métodos, o algoritmo de aprendizado de máquinas utilizado neste trabalho foi o *Naive Bayes*. A biblioteca *Scikit Learn* oferece diversas funções para implementação deste algoritmo[7], além de oferecer as outras funções utilizadas no processo de separação de amostra e validação. Para este trabalho foi utilizada a função MultinomialNB() por abranger tanto dados de contagem quanto variáveis continuas (veja Exemplos 2.4.1 e 2.4.2).

Para gerar o modelo preditor, primeiro, a base foi dividida em amostra treino, teste e validação, na proporção de 80%, 10% e 10%. Assim, a amostra treino foi de 6.267 perfis, a amostra teste foi de 784 perfis e, validação, 784 perfis. Para realizar essa divisão entre treino e teste foi utilizada a função train_test_split() com os argumentos train_size = 0.8, random_state = 0. E em seguida, para dividir a amostra teste entre teste e validação foi utilizada a mesma função com os argumentos train_size = 0.5, random_state = 0.

A função MultinomialNB() possui o argumento fit_prior, onde se fit_prior = True a probabilidade *a priori* é calculada a partir da amostra de treino. Desse modo, a probabilidade *a priori* de um usuário ser *bot* foi de $0,29$.

Além disso, foram testadas três possibilidades de seleção de variáveis:

- **Modelo 1**: utilizando-se todas as variáveis disponíveis;
- **Modelo 2**: utilizando-se as variáveis relacionadas a morfologia, semântica e funcionalidades, conforme definidas na seção 2.2;
- **Modelo 3**: utilizando-se somente os *tokens*.

Separando essas três possibilidades de modelo vai ser possível avaliar o efeito das variáveis na predição. A seguir estão dispostos os resultados de teste para os três modelos treinados.

[7]Link para a página das implementações do Naive Bayes https://scikit-learn.org/stable/modules/naive_bayes.html.

3.3.1 Modelo 1

Para treinar o Modelo 1 foram utilizadas todas as variáveis disponíveis na base de dados, totalizando 3.014 variáveis explicativas, sem considerar a variável resposta. Veja, na Tabela 12, que a Sensibilidade é muito maior que a Especificidade. Ou seja, o Modelo 1 é muito melhor em predizer se o perfil analisado é não *bot* do que se o usuário é *bot*. A sua matriz de confusão e suas métricas de avaliação podem ser vistas na Tabela 12.

Tabela 12: Matriz de Confusão e Métricas de Qualidade de Ajuste do Modelo 1

		Valores Preditos	
		Não Bot	Bot
Valores Reais	Não Bot	391	172
	Bot	95	125

Métricas	Valores
Acurácia	0,6590
Sensibilidade	0,8045
Especificidade	0,4208

3.3.2 Modelo 2

Já para treinar o Modelo 2 foram utilizadas somente as variáveis descritas na Seção 2.2, totalizando 14 variáveis explicativas. Observe, comparando as Tabelas 12 e 13, que o Modelo 2 supera o Modelo 1 em Acurácia e Especificidade. A sua matriz de confusão e suas métricas de avaliação podem ser vistas na Tabela 13.

Tabela 13: Matriz de Confusão e Métricas de Qualidade de Ajuste do Modelo 2

		Valores Preditos	
		Não Bot	Bot
Valores Reais	Não Bot	510	53
	Bot	126	94

Métricas	Valores
Acurácia	0,7713
Sensibilidade	0,8018
Especificidade	0,6394

3.3.3 Modelo 3

Por fim, para treinar o Modelo 3 foram utilizados somente os tokens publicados pelos perfis, totalizando 3.000 variáveis explicativas. O Modelo 3 foi inferior tanto na Acurácia quanto na Sensibilidade e na Especificidade se comparado aos Modelos 1 e 2, vide Tabela 14. A sua matriz de confusão e suas métricas de avaliação podem ser vistas na Tabela 14.

Tabela 14: Matriz de Confusão e Métricas de Qualidade de Ajuste do Modelo 3

		Valores Preditos	
		Não Bot	Bot
Valores Reais	Não Bot	391	172
	Bot	105	115

Métricas	Valores
Acurácia	0,6462
Sensibilidade	0,7883
Especificidade	0,4006

3.3.4 Melhor Ajuste

Veja, na Tabela 13, que o modelo com as melhores métricas de avaliação foi o Modelo 2. Sem utilizar os *tokens* como covariáveis no treinamento , o Modelo 2 atingiu a melhor acurácia e especificidade dos três modelos avaliados.

Nas Tabelas 15 e 16 podem ser vistas a matriz de confusão e as métricas de avaliação para a amostra de validação.

Tabela 15: Matriz de Confusão para o Modelo 2 para os Dados de Validação

		Valores Preditos	
		Não Bot	Bot
Valores Reais	Não Bot	470	72
	Bot	128	114

Tabela 16: Métricas de Qualidade de Ajuste do Modelo 2 para os Dados de Validação

Métricas	Valores
Acurácia	0,7448
Sensibilidade	0,7859
Especificidade	0,6129

Na Tabela 16, com a amostra de validação, temos um modo de simular o modelo funcionando em produção, já que são dados não vistos no treinamento e no teste. Assim, o modelo treinado é capaz de predizer corretamente, se um dado perfil é *bot*, em cerca de 60% das ocasiões.

4 Conclusões

Este trabalho teve por objetivo treinar um modelo de aprendizado de máquinas capaz de predizer se um determinado usuário do Twitter é *bot* ou não. Como foi necessário gerar uma própria base experimental, a investigação ficou restrita a apenas um tema: os líderes das pesquisas para as eleições presidenciais do Brasil de 2022[1]. Para atingir a finalidade proposta foram selecionadas algumas variáveis baseadas em trabalhos aqui revisados e variáveis inéditas, que não observadas nesses artigos.

Para a construção da base experimental utilizada foi necessária a criação de um *script* para acessar a API do Twitter e obter esses dados de forma bruta. Em seguida, com técnicas de Processamento de Linguagem Natural como tokenização, lemmatização, etc essa base foi tratada de modo que fosse possível realizar as análises exploratórias e construção dos modelos.

Na etapa de análise exploratória dos dados foi possível observar diferenças relevantes no comportamento dos dois tipos de usuário analisados. Os usuários classificados como *bot* apresentaram menor incidência de verbos, advérbios e pronomes. No entanto apresentaram maior ocorrência de hashtags e emojis. Este último com uma diferença significativa, de modo que dentre os 30 *tokens* mais utilizados por perfis classificados como *bot*, 8 são emojis. Essa proporção para não *bots* é de 3 para 30. Já com relação ao sentimentos dos *tweets* a principal constatação foi de que os *bots* tendem a mudar menos o sentimento, conforme a classificação descrita na seção 2.2, transmitido nos textos.

No entanto, a principal diferença observada entre os dois tipos de usuários foi com relação as métricas da plataforma. As duas principais diferenças foram a idade dos perfis e o número de publicação. Foi observado que mais de 60% dos usuários classificados como *bots* tiveram suas contas criadas a partir de 2020, enquanto essa proporção para usuários não *bots* foi de 35%. Além disso, os perfis apontados como *bots* possuem 10 vezes mais publicações do que os não *bots*.

[1] https://www.estadao.com.br/politica/eleicoes/agregador-pesquisa-eleitoral-2022/?cargo=presidencial&modalidade=todas

Na fase de testes dos modelos pode ser observado que a Sensibilidade dos três modelos foi bem similar, ficando em torno de 80%. No entanto, o Modelo 2 apresentou melhores valores para Acurácia (77,13%) e Especificidade (63,94%). Sendo assim, o melhor modelo, dentre as possibilidades observadas, foi o que utilizava como variáveis somente as descritas na seção 2.2. Na Tabela 16 podem ser vistas as métricas de avaliação do melhor ajuste para a amostra de validação, de modo a simular a aplicação do modelo em um ambiente de produção.

A partir do que foi desenvolvido neste trabalho pode-se ponderar alguns caminhos para melhor a capacidade de predição do modelo gerado. Primeiro, obter uma base de dados com mais usuários previamente classificados e com temas aleatórios, de modo que o trabalho não precise se restringir a apenas um tema. Obter usuários previamente classificados como *bot* tornaria as análises e o classificador mais confiáveis. Da forma que foi feito neste trabalho, para criar a base experimental, não podemos afirmar que os perfis classificados como *bot* realmente o são, podemos afirmar que são *bots* segundo o PEGABOT. Além disso, utilizar as variáveis relacionadas a sentimento descritas por Dickerson (2014) para textos em língua portuguesa, para tentar reproduzir os mesmos resultados que o seu trabalho obteve. Por fim, pode-se testar modelos de aprendizado de máquinas mais robustos, como os baseados em árvores ou redes neurais.

Referências

ALMEIDA, R. J. A. LeIA - Léxico para Inferência Adaptada. [S.l.]: GitHub, 2018. Https://github.com/rafjaa/LeIA.

ALOTHALI, E. et al. Detecting social bots on twitter: A literature review. In: 2018 International Conference on Innovations in Information Technology (IIT). [S.l.: s.n.], 2018. p. 175–180.

CHOLLET, F. et al. Keras. 2015. ⟨https://keras.io⟩.

DICKERSON, J.; KAGAN, V.; SUBRAHMANIAN, V. Using sentiment to detect bots on twitter: Are humans more opinionated than bots? In: . [S.l.: s.n.], 2014. p. 620–627.

GILBERT, C. H. E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf. [S.l.: s.n.], 2014.

HARRIS, C. R. et al. Array programming with NumPy. Nature, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: ⟨https://doi.org/10.1038/s41586-020-2649-2⟩.

HONNIBAL, M.; MONTANI, I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear. 2017.

KLUYVER, T. et al. Jupyter notebooks – a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). Positioning and Power in Academic Publishing: Players, Agents and Agendas. [S.l.], 2016. p. 87 – 90.

LEE, K.; EOFF, B.; CAVERLEE, J. Seven months with the devils: A long-term study of content polluters on twitter. Proceedings of the International AAAI Conference on Web and Social Media, v. 5, n. 1, p. 185–192, Aug. 2021. Disponível em: ⟨https://ojs.aaai.org/index.php/ICWSM/article/view/14106⟩.

LIDDY, E. Natural language processing. Encyclopedia of Library and Information Science, 2nd Ed. NY. Marcel Decker, Inc., 2001.

MANNING, P. R. C. D.; SCHüTZE, H. Introduction to Information Retrieval. [S.l.]: Cambridge University Press, 2009.

MEDHAT AHMED HASSAN, H. K. W. Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, 2014.

MIRANDA, G. tcc. https://github.com/gamiranda/tcc, Acesso em: 06 julho 2022.

MONDELLI, M. L.; PETERSON, A.; GADELHA, L. Exploring reproducibility and fair principles in data science using ecological niche modeling as a case study. In: _____. [S.l.: s.n.], 2019. p. 23–33. ISBN 978-3-030-34145-9.

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up? sentiment classification using machine learning techniques. In: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002). Association for Computational Linguistics, 2002. p. 79–86. Disponível em: ⟨https://aclanthology.org/W02-1011⟩.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011.

ROCHA, A. Naive Bayes Classifier. 2011. Https://www.ic.unicamp.br/ rocha/teaching/2011s2. Acesso em 15 de jan. 2022.

ROESSLEIN, J. Tweepy: Twitter for python! https://github.com/tweepy/tweepy, 2020.

ROSSUM, G. V.; DRAKE, F. L. Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697.

RUEDIGER, M. A. Robôs, redes sociais e política no brasil: estudo sobre interferências ilegítimas no debate público na web, riscos à democracia e processo eleitoral de 2018. FGV, DAPP, 2017.

TEAM, T. pandas development. pandas-dev/pandas: Pandas. Zenodo, fev. 2020. Disponível em: ⟨https://doi.org/10.5281/zenodo.3509134⟩.

WANG, A. H. Detecting spam bots in online social networking sites: A machine learning approach. 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Pri-vacy (DBSEC, 2014.

APÊNDICE 1 – Figuras não Utilizadas no Texto

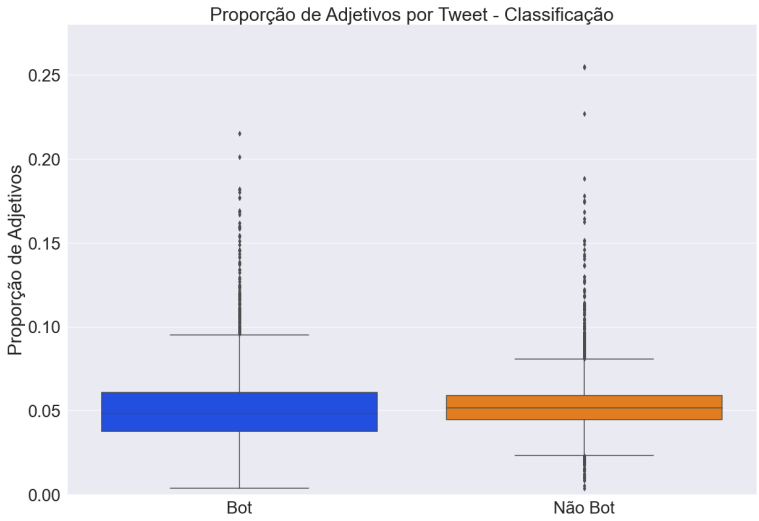

Figura 9: Box Plot da Proporção de Adjetivos por Tweet - Classificação

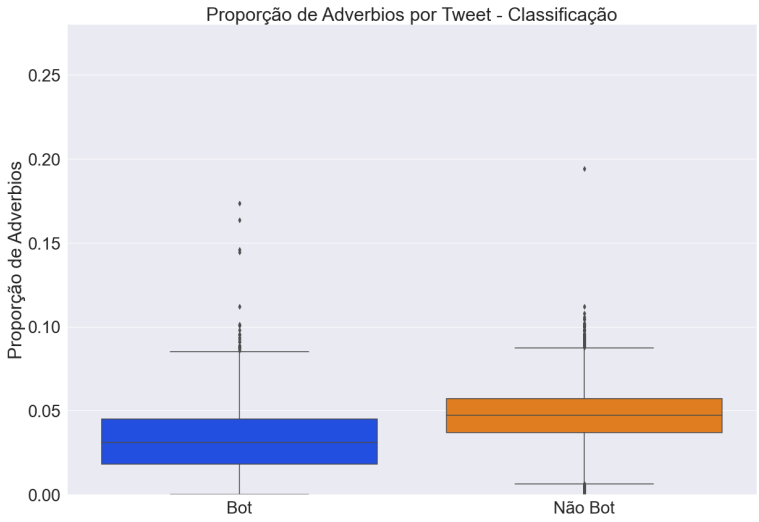

Figura 10: Box Plot da Proporção de Advérbios por Tweet - Classificação

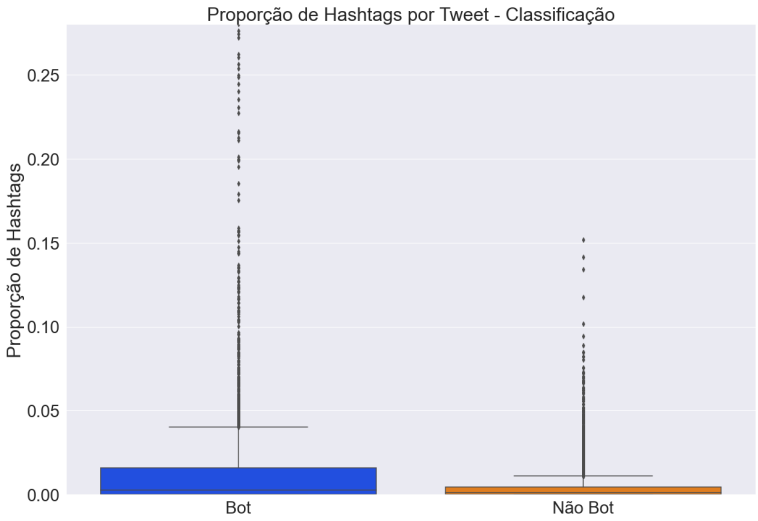

Figura 11: Box Plot da Proporção de Hashtags por Tweet - Classificação

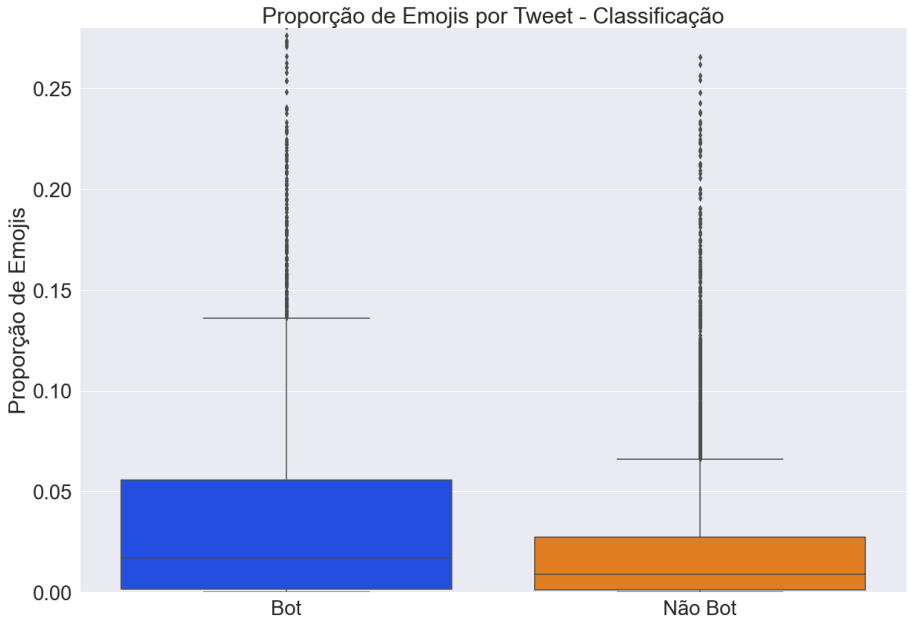

Figura 12: Box Plot da Proporção de Emojis por Tweet - Classificação.

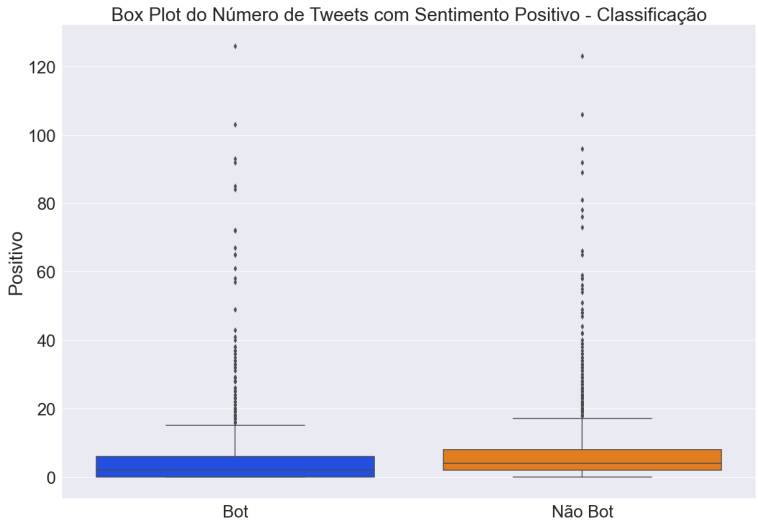

Figura 13: Box Plot do Número de Tweets com Sentimento Positivo - Classificação

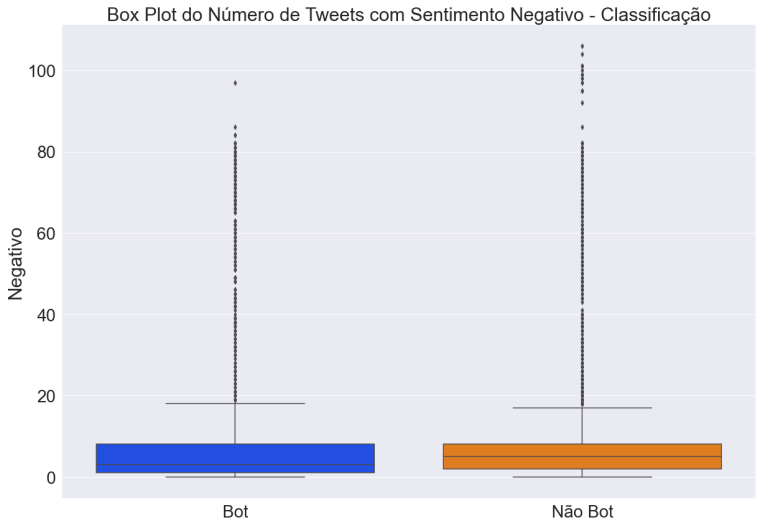

Figura 14: Box Plot do Número de Tweets com Sentimento Negativo - Classificação

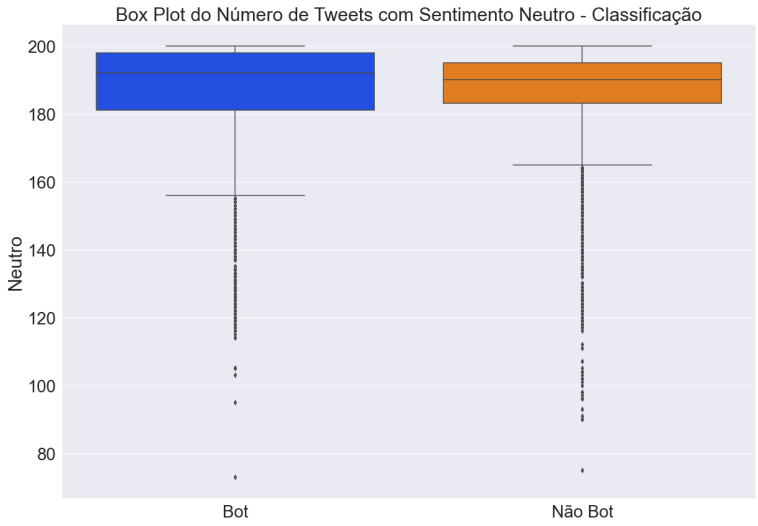

Figura 15: Box Plot do Número de Tweets com Sentimento Neutro - Classificação

APÊNDICE 2 – Especificações

Conforme Mondelli, Peterson e Gadelha (2019), este Apêndice contém as especificações dos materiais técnicos utilizados neste trabalho. Os *scripts* e bases de dados utilizados neste trabalho podem ser consultados no repositório do *github* (MIRANDA, 2022).

Os modelos foram treinados na IDE Jupyter Notebook 6.4.5 (KLUYVER et al., 2016) no Python 3.9.7 (ROSSUM, 2009) em um computador pessoal com as seguintes especificações técnicas:

- **Sistema Operacional:** Windows 10 Pro 21H1
- **Processador:** AMD Ryzen 5 3600 6-Core Processor 3.59 GHz
- **RAM instalada:** 16,0 GB
- **Tipo de sistema:** Sistema operacional de 64 bits, processador baseado em x64

Já com relação a versão das bibliotecas usadas:

- **pandas:** versão 1.4.3
- **keras:** versão 2.7
- **numpy:** versão 1.23.0
- **spacy:** versão 3.2
- **tweepy:** versão 4.10.0
- **sklearn:** versão 1.1

APÊNDICE 3 – Bibliotecas Utilizadas

Neste apêncide temos a lista das principais bibliotecas do Python utilizadas neste trabalho e suas aplicações.

- **pandas 1.4.3:** utilizada na manipulação das bases de dados;
- **keras 2.7:** utilizada no processo de criação do Word Index;
- **numpy 1.23.0:** utilizada para manipulação de variáveis;
- **spacy 3.2:** utilizada na lemmatização e na criação das variáveis de morfologia e semântica, além da lista de *stop words*;
- **tweepy 4.10.0:** utilizada para acessar a API do Twitter a baixar os dados públicos de usuários da base de dados;
- **sklearn 1.1:** utilizada na parte de aprendizado de máquinas, da separação entre treino, teste e validação à criação do *Naive Bayes*.