Implementação de um Detector de Contas Bots em Redes Sociais

Mateus da Silveira Colissi[1], **Alexandre Zamberlan**[1], **Guilherme C. Kurtz**[1]

[1]Ciência da Computação – Universidade Franciscana (UFN)
Santa Maria – RS – Brasil

mateus.colissi@unifra.edu.br, {alexz,guilhermekurtz}@unifra.br

Abstract. *The growing popularity of social networks like Twitter has been accompanied by the rise of content polluters. False or manipulated news, disseminated by bots and shared by humans, lead to a fabrication of a population consensus, a subject to be taken into consideration during elections. Therefore, the object of this work is to implement a bot account detector on the Twitter social platform, in which relevant social account information was extracted based on specific heuristics in order to avoid manipulation of social network content. The technologies used are Python, Tweepy, MongoDB, JSON, etc. The results are solution modeling, bot detection techniques and validation of results.*

Resumo. *O alto crescimento da popularidade de redes sociais como Twitter tem sido acompanhado pelo aumento de poluidores de conteúdo. Notícias falsas ou manipuladas, disseminadas por bots e compartilhadas por humanos, levam a uma fabricação do consenso da população, assunto a ser levado em consideração em eleições. Portanto, o objetivo deste trabalho é implementar um detector de contas bots na plataforma social Twitter, em que foram extraídas informações relevantes de contas sociais, baseada em heurísticas determinadas, a fim de evitar a manipulação do conteúdo das redes sociais. As tecnologias utilizadas são Python, Tweepy, MongoDB, JSON, etc. Os resultados são a modelagem da solução, técnicas para detecção de bots e validação dos resultados.*

1. Introdução

Com a popularidade dos serviços das redes sociais tornou-se extremamente fácil e ágil a interação, o compartilhamento de informações ou opiniões e a comunicação com outros grupos de pessoas. A população está tão envolvida nessas plataformas que o conteúdo distribuído nessas conexões causam um grande impacto ao ponto de influenciar decisões no seu dia a dia. Bots ou sistemas autônomos são abundantes em uma rede social. Eles são criados com uma série de propósitos, como notícias, *marketing*, *spamming*, divulgação de conteúdo malicioso, e também campanhas políticas, conforme aponta [Chu et al. 2010]. Segundo o autor, estima-se que 51,8% de todo tráfego na internet é gerado por bots. Em redes sociais como Twitter percebeu-se um grande aumento na população de bots, que segundo informações do próprio site, em 2014, 5% do total de usuários vinham de contas falsas ou *spams* [Gilani et al. 2017]. Com isso, detectar esses bots pode ajudar a prevenir que a opinião e o consenso das pessoas sejam influenciados por esses sistemas autônomos, evitando e repudiando a divulgação de informações falsas.

O avanço na inteligência artificial permitiu aos programadores a produção de sistemas automatizados (socialbots) que são capazes de interagir com usuários legítimos e

compartilhar conteúdos baseados em um critério específico. Com essa tecnologia, esses bots atingiram um nível capaz de simular o comportamento humano, e com isso houve o acréscimo na dificuldade de sua detecção nas mídias sociais [Xie et al. 2012]. Usuários reais, ao se depararem com essas informações, tendem a contribuir com o bot, seja compartilhando ou mencionando sua divulgação, conferindo assim, uma maior credibilidade para o conteúdo falso [Itagiba 2017].

Dentro desse contexto, o objetivo deste trabalho é implementar um detector de contas bots em redes sociais utilizando a *API* (*Application Program Interface*) do Twitter.

2. Referencial Teórico

Nesta seção serão introduzidos os conceitos e métodos utilizados no desenvolvimento deste trabalho.

2.1. Bots

Bots são sistemas autônomos (robôs) criados para imitar ações desejadas pelo seu administrador, tais como bots da rede social Twitter, que são configurados para seguir pessoas (*follow*), postar e direcionar mensagens, enviar *URLs* ou *hashtags*. São capazes de compreender uma consulta (comando, ordem, questão) de um humano (ou bot mestre) e entregar uma resposta adequada [Itagiba 2017]. Os bots são utilizados há bastante tempo em aplicativos de mensagens mais famosos como Facebook, Skype e chatbots. Em 1966, Joseph Weizenbaum criou um "bot" chamado ELIZA que alcançou grande popularidade ao simular uma conversa com um terapeuta por meio da digitação [Martim 2017]. Os bots também são observados em diferentes situações, tais como: inteligência artificial em jogos; grupo de botnets[1]; assistentes virtuais, como no caso da Siri e Cortana; e finalmente como divulgador de informações/conteúdo (os socialbots) [Martim 2017].

2.1.1. Socialbots

Socialbots são softwares automatizados com objetivo de propagar conteúdo, possuem potencial malicioso para comprometer contas, espalhar *spams*, postar mensagens e enviar requisições de conexão. Eles são desenvolvidos para mimetizarem o ser humano, seja imitando um usuário de uma rede social, ou simulando seu comportamento com inteligência artificial [Boshmaf et al. 2013].

Segundo [Woolley and Guilbeault 2017], na eleição presidencial dos Estados Unidos em 2016, exércitos de bots permitiram que campanhas, candidatos e apoiadores alcançassem dois critérios chaves durante a eleição: (i) fabricar consenso; e (ii) democratizar a propaganda na rede. Os robôs de mídia fabricaram o consenso ampliando artificialmente o tráfego em torno de um candidato ou assunto político.

2.1.2. Comportamento dos Bots

Uma rede social possui três tipos de usuários: humanos, bots e ciborgues [Chu et al. 2010]. A categoria bots é, especificamente, dividida em: bots autodeclarados

[1]Grupo de computadores "sequestrados" utilizado para ataques, ou fins maliciosos.

(informam o tempo, hora atualizada, etc), spambots (distribuem grandes quantidade de conteúdo não solicitados para os usuários) e socialbots (distribuem notícias, e interagem com usuários em redes sociais) [Boshmaf et al. 2013]. Para diferenciá-las, assume-se que humanos postam conteúdos pensativos e sentimentais, enquanto que os bots enviam conteúdos com tuítes automatizados, ou seja, conteúdo não original (tuítes copiados e abundância de retuítes - Figura 1), *spam* ou *URLs* maliciosos e links com conteúdo não relacionados. Ciborgues, por sua vez, postam diferentes tipos de tuítes, geralmente contendo um conteúdo original mais um agregador de conteúdo RSS (*Really Simple Syndication*) [Chu et al. 2010]. Utiliza-se métodos ou ferramentas para diferenciar as classes bots, que podem ser via trabalho manual, como por rotulagem, ou automática, por meio de contas armadilhas (*honeypots*) ou por aprendizado de máquina por meios de técnicas de reconhecimento de padrões. Sendo que o principal componente da detecção é feito através da comparação do conteúdo postado por uma conta já rotulada como bot, e uma conta suspeita [Lee et al. 2010].

Figura 1. Tuíte com conteúdo não original.

2.2. Detecção de Bots

A detecção de bots pode ser feita por diversas abordagens, sendo que a mais efetiva depende da rede social que deseja-se aplicar a detecção. Isso acontece devido aos diferentes dados providos pela *API* das redes sociais. Além da rede social, o tipo de bot que se deseja detectar também influencia no método escolhido para detecção, devido a diferentes abordagens nos níveis de comunicação [Haugen 2017]. Para detecção de socialbots no Twitter, frequentemente são utilizados métodos automatizados como *honeypots* e reconhecimento de padrões, que combinados com um conjunto de heurísticas e um *dataset* rotulado com dois tipos de contas (*spammers* ou legítimas), apresentam uma alta precisão na detecção [Lee et al. 2011].

2.2.1. *Honeypots*

Honeypots são contas armadilhas para automatizar a detecção de socialbots, com a finalidade de monitorar e armazenar o comportamento de ataques ou *spams*. Esse método assume que usuários não seguem bots, ou seja, a detecção dos bots é dada quando um bot segue a conta *honeypot*. Conforme a conta *honeypot* coleta evidências de *spams*, considerando um conjunto de heurísticas para precisão da detecção, as informações são extraídas do perfil suspeito, tais como: número de amigos, idade, entre outros [Lee et al. 2010].

2.2.2. Heurística para Detecção de Bots

Heurísticas são critérios, métodos ou princípios para decidir entre muitas alternativas, qual será a ação mais efetiva a fim de atingir um objetivo [Pearl 1984], ou seja, que não garante ser a mais otimizada ou perfeita, mas suficiente para os objetivos imediatos. As heurísticas para detecção de bots podem ser classificadas em quatro categorias: (i) demográfica, no qual é levada em consideração a idade da conta e o tamanho do nome na tela em relação com o nome da conta, em que é verificado se ambos possuem os mesmos caracteres, se não contém letras e números aleatórios ou se contém a palavra bot, como mostra a Figura 2; (ii) conectividade, analisando o percentual de amigos bidirecionais (*followers*/*following*), como é demonstrado na Equação 1:

$$\frac{|following \cap followers|}{|following|} e \frac{|following \cap followers|}{|followers|} \tag{1}$$

em que o módulo da intersecção de *followers* e *following* dividido pelo módulo dos *following*, e o módulo da intersecção de *followers* e *following* dividido pelo módulo de *followers*, demonstram esse percentual; (iii) conteúdo, em que são verificadas *URLs* e tuítes únicos; e (iv) histórico, calculando o desvio de *followers* durante o tempo, como é demonstrado na Equação 2:

$$\sqrt{\frac{1}{n-1}\sum_{i=1}^{n-1}(f_i + 1 - f_i)} \tag{2}$$

em que n é o número total de registros temporais e históricos registrados informados, e i significa o número de seguidores do usuário extraído em informações temporais e históricas [Lee et al. 2011].

Figura 2. Nome da tela.

2.2.3. Medida F_1

Medida F_1 é a média ponderada de precisão[2] e *recall*[3]

$$F_1(u, b) = 2\frac{PR}{P + R} \tag{3}$$

em que,

$$P = \frac{|u \cap b|}{|u|} \tag{4}$$

é a precisão do modelo, e

$$R = \frac{|u \cap b|}{|b|} \tag{5}$$

[2]Proporção dos casos positivos previstos que são corretamente positivos reais [Powers 2011].
[3]Proporção dos casos reais positivos, que são corretamente previstos como positivos [Powers 2011].

é o *recall*, em que dado um conjunto de todos os usuários U e um conjunto de bots rotulados $b \subset U$, no qual essa medida identifique um conjunto de usuários $u \subset U$ [Morstatter et al. 2016].

3. Trabalhos Correlatos

Segundo [Gilani et al. 2017] é possível estudar os efeitos dos bots no Twitter ao configurar uma conta bot e realizar análises em um conjunto de dados no servidor Web. Com esse estudo percebeu-se que utilizando duas características relevantes como frequência de cliques e *user agent string*[4] é possível montar um classificador de contas bots. Com isso, conclui-se que os bots eram apenas 4,08% dos visitantes, mas responsáveis por quase metade da frequência de cliques (44,91%). A partir disso, o autor realizou uma análise de séries temporais ao levar em conta a frequência de cliques em uma única conta do usuário do Twitter, percebendo assim sua correlação com o comportamento automatizado.

Conforme [Morstatter et al. 2016], para detectar bots o autor propôs uma Medida F_1 como ilustra a Equação 3 na Seção 2.2.3 O autor usou um modelo de detecção no qual considera o *recall* em sua formulação em conjunto com heurísticas populares para detecção de contas bots. Para o uso desses métodos, foram utilizados dois *datasets* rotulados durante diferentes processos.

Segundo [Lee et al. 2011], adotar uma estratégia utilizando *honeypots* é vantajoso pois: (i) evidências são coletadas automaticamente; (ii) não necessitam interferência de usuários; (iii) identificam e filtram novas características que podem ser incorporadas ao conteúdo. Com base nas informações coletadas, uma ampla variedade de recursos é criada a partir de uma das quatro categorias a seguir: *User Demographics* (*UD*): recursos extraídos de informações descritivas sobre uma conta; *User Friendship Network* (*UFN*): recursos extraídos de informações de amizades; *User Content* (*UC*): recursos extraídos de tuítes postados; e *User History* (*UH*): recursos extraídos das informações de perfil temporal e histórico de usuários. Com esses identificadores, testou-se algoritmos de detecção usando o *Weka machine learning toolkit*[5]. O autor obteve 95% a 98% de precisão na detecção, em particular o método *Random Forest*[6] foi o que produziu a maior precisão. Para melhorar esse método, técnicas de *boosting* (algoritmos que aumentam a força de previsão) e *bagging* (algoritmos que minimizam a variância da previsão) foram aplicadas posteriormente, formando um classificador composto no qual resultou em um aumento da precisão da detecção.

4. Metodologia

Um estudo de técnicas de reconhecimento de padrões foi realizado para que seja possível aplicá-los na detecção de bots, e, a partir disso, desenvolver um detector completamente automatizado. O treinamento dos dados do *dataset* é constituído de informações extraídas pertencentes às categorias: *UD*, *UFN* e *UC*, definindo assim uma estratégia baseada nas heurísticas de detecção de bots.

[4]Relata à um site informações sobre o navegador e o sistema operacional.

[5]Coleção de algoritmos de aprendizado de máquina e ferramentas de pré processamento de dados [Witten et al. 2016].

[6]Constrói uma grande quantidade de árvores de decisão para fora do subconjunto de dados a partir de um treinamento único definido.

Neste trabalho é implementado um modelo baseado na Medida F_1 que deve considerar precisão e *recall*, através da aplicação de heurísticas comuns para detecção de bots. Essas heurísticas foram escolhidas com base em que bots: (i) não postam conteúdos originais; (ii) promovem conteúdo e *URLs*; (iii) geralmente são *spammers*; e (iv) não são seguidos por humanos. Um parser (é um analisador sintático) deverá ser implementado para analisar as informações extraídas do Twitter, em que o modelo é composto por: fração de retuítes; média do tamanho do tuíte; fração/quantidade de *URLs*; média de tempo entre *posts*; número de *followers* e *following*. Para o desenvolvimento deste trabalho foi utilizado a linguagem de programação Python, o sistema gerenciador de banco de dados MongoDB, e a integração com a *API* do Twitter via Tweepy, permitindo assim a extração das informações do usuário a fim de popular um *dataset* necessário para o desenvolvimento do classificador. Partindo de dados bases são utilizadas as informações contidas no *dataset* americano caverlee-2011[7] [Lee et al. 2011], com a finalidade de validar os resultados. É presente no *dataset* informações da conta (bot ou humana) pertencentes as categorias *UD*, *UFN*, *UC* e *UH*, como: quando foi criada, número de *followers* e *following*, tuítes, informações dos tuítes, etc. Também para validação dos resultados, utiliza-se métricas padrões apresentadas na Seção 2.2.3 como: (i) precisão; (ii) *recall*; (iii) Medida F_1; (iv) *fall-out*, como demonstra a Equação 6:

$$F = 1 - \frac{|U - G - B|}{|U - G|} \tag{6}$$

que calcula quantos "alarmes falsos" encontrados; (v) falsos positivos; e (vi) verdadeiros positivos [Morstatter et al. 2016], nos grupos classificados como bots, bots conhecidos e todos os usuários, representados na Figura 3. Precisão é uma proporção da observação corretamente prevista para o total observado. Quanto mais alta a precisão, melhor é o modelo utilizado. *Recall* é a proporção de observações positivas preditas corretamente. Medida F_1 é a média ponderada de precisão e *recall*, portanto essa medida deve levar em conta os falsos positivos e os falsos negativos [Morstatter et al. 2016]. Falso positivo acontece quando uma conta é prevista/classificada como bot, mas que na realidade é humana. Verdadeiro positivo é quando uma conta é prevista/classificada, por exemplo, como bot, e a conta era realmente um bot.

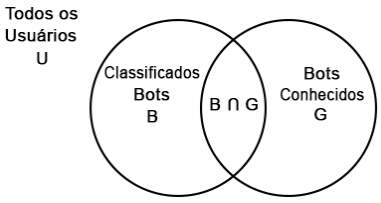

Figura 3. Modelo precisão - *recall*.

A Figura 4 ilustra, visão geral, o diagrama de classe com as relações entre as classes: *UserAccount* gera objeto de uma conta de usuário, apresenta métodos responsáveis por classificar nas categorias *UFN* (Figura 14) e *UD* (Figura 13), além de classificar o tipo de usuário; *TwitterAPI* é responsável por definir o usuário que será classificado, criar objetos do tipo *UserAccount* e *UserTweets* e ainda fazer requisições a *API* do Twitter

[7]https://botometer.iuni.iu.edu/bot-repository/datasets/caverlee-2011/caverlee-2011.zip.

via Tweepy; *UserTweets* gera objeto dos tuítes de uma conta de usuário, apresenta métodos *REF* (Figura 11) e *UT* (Figura 15), além do método responsável por classificar na categoria *UC* (Figura 12); *MongoDBManagment* é responsável pela conexão, inserção e busca de registros no MongoDB; *Main* é responsável por instanciar todas as classes; *File* é responsável por abrir e ler o arquivo (*dataset*) e retornar uma lista de IDs (identificadores) com *labels* (bot ou humano); *TwitterKeys* apresenta as chaves e *tokens* que permitem autenticar a conexão com o Twitter via Tweepy.

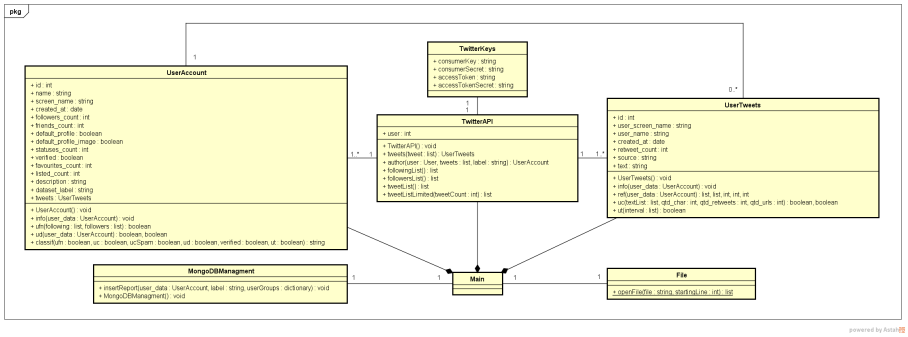

Figura 4. Diagrama de classe para fornecer uma visão de domínio do sistema.

Já as Figuras 5 e 6 expandem e detalham o diagrama de visão geral (Figura 4). É possível perceber a relação de dependência (composição) entre as classes *Main*, *UserAccount* e *UserTweets*. A classe *Main* instancia a *TwitterAPI* e seleciona o usuário, com isso a classe *TwitterAPI* extrai informações de uma conta do Twitter via Tweepy e instancia a *UserTweets* para criar um objeto do tipo *UserTweets*.

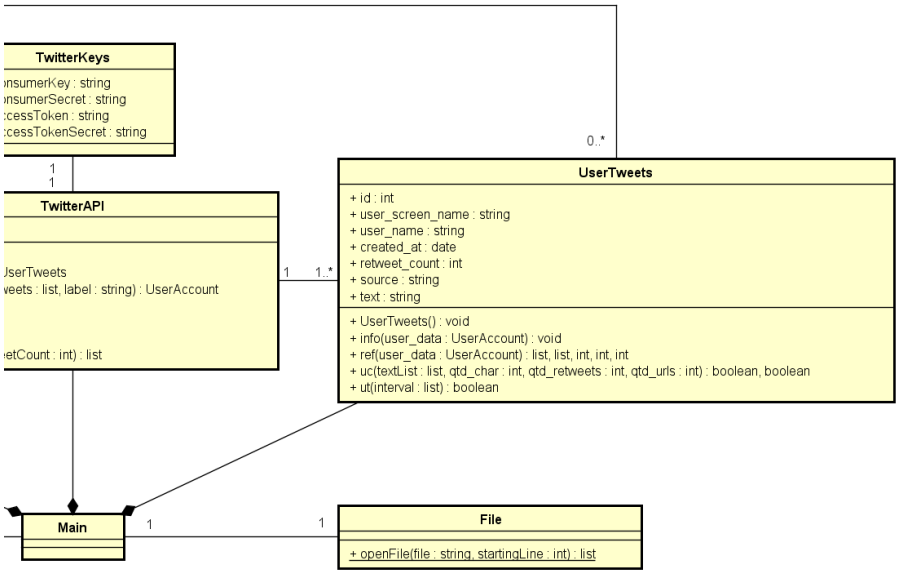

Figura 5. Diagrama de classe para fornecer uma visão de domínio do sistema.

Após criar um objeto do tipo *UserTweets* a *Main* acessa a *TwitterAPI* para criar um objeto do tipo *UserAccount* com as informações e tuítes (objeto *UserTweets*) do usuário.

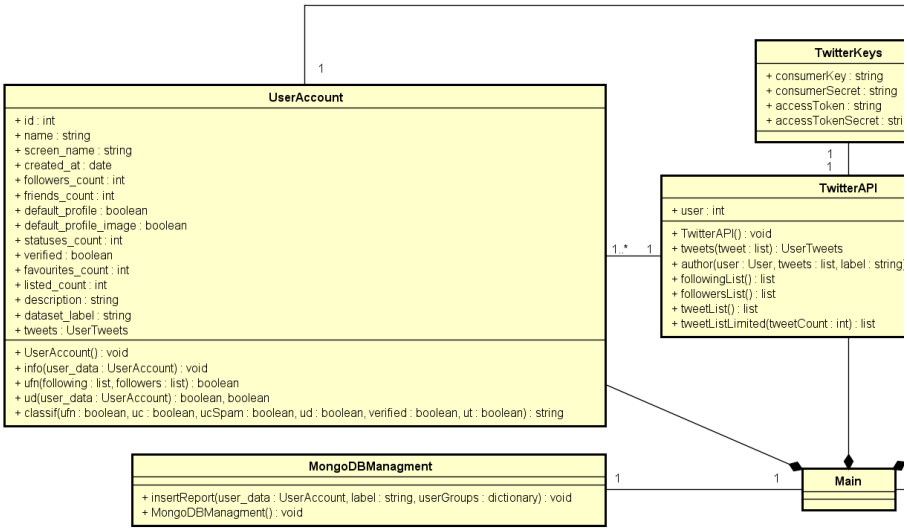

Figura 6. Diagrama de classe para fornecer uma visão de domínio do sistema.

Em relação ao diagrama de sequência, que representa a sequência dos processos e a colaboração entre as classes ao longo do tempo, há 4 figuras. A Figura 7 mostra a visão geral de todo o processo do sistema modelado e implementado.

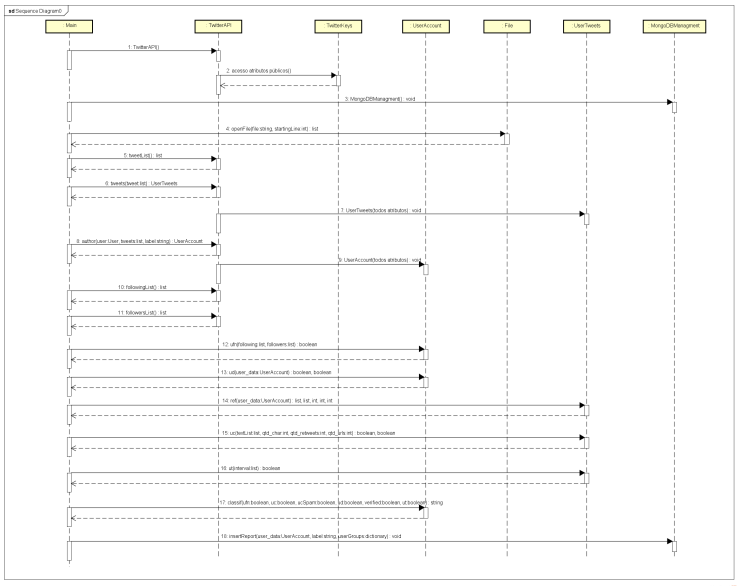

Figura 7. Diagrama de sequência para fornecer uma visão de interação entre as classes da proposta.

Já as Figuras 8, 9 e 10 ilustram de forma expandida e detalhada os processos da Figura 7. Destaca-se que o processo começa com a autenticação da *API* do Twitter, para isso é feito o acesso as chaves e *tokens*, em seguida é especificado a conexão com o MongoDB. É retornado uma lista de contas pela leitura do arquivo, com isso é requisitado a lista de tuítes de cada usuário e criado um objeto *UserTweets*.

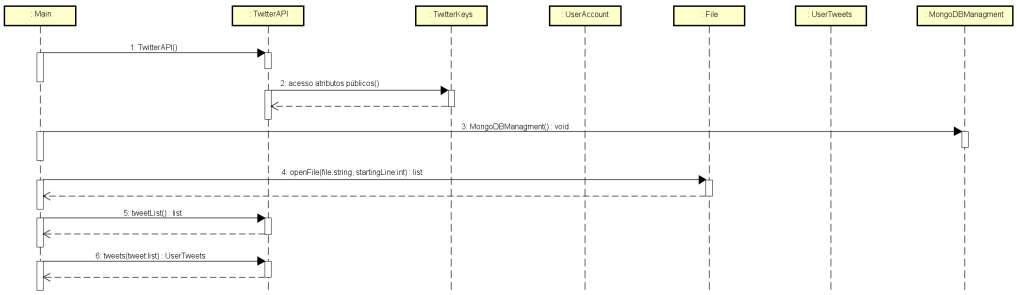

Figura 8. Diagrama de sequência para fornecer uma visão de interação entre as classes da proposta.

Em seguida, é criado um objeto *UserAccount*, é feita a requisição dos *following* e *followers* do usuário para classificar na categoria *UFN*.

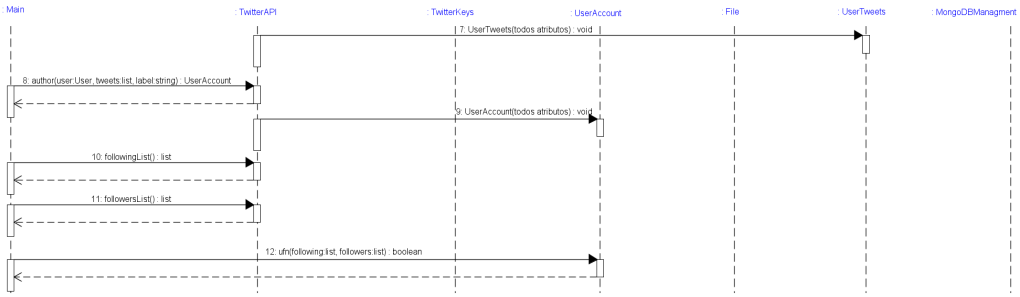

Figura 9. Diagrama de sequência para fornecer uma visão de interação entre as classes da proposta.

Também é classificado na categoria *UD*, em seguida começa o refinamento dos atributos do tuíte, com isso retorna o processo de classificar na categoria *UC* e é feito análises temporais (*UT*). Então a conta do usuário é classificada como *human* ou bot e informações relevantes do usuário são inseridas no banco de dados.

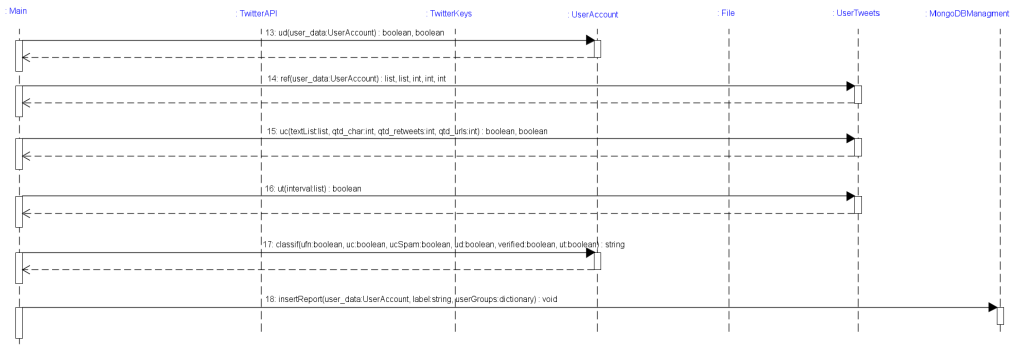

Figura 10. Diagrama de sequência para fornecer uma visão de interação entre as classes da proposta.

Na Figura 11, é apresentado o refinamento dos atributos recebidos de uma conta do Twitter. São refinados atributos do tuíte como: quantidade de retuítes e *URLs*, total de caracteres dos tuítes e intervalo de tempo entre *posts*.

```
def ref(self, user_data):
  textList = []
  qtd_retweets = 0
  qtd_urls = 0
  qtd_char = 0
  interval = []

  f = "%Y-%m-%d %H:%M:%S"
  i = 0
  for tweet in user_data.tweets:
    self.info(tweet)

    textList.append(tweet.text.lower())
    if bool(re.search('RT ?@\w+', tweet.text)):
      qtd_retweets += 1

    if 'http' in tweet.text:
      qtd_urls += 1
    qtd_char += len(tweet.text)

    if i == 0:
      interval.append(0)
    else:
      seconds = (datetime.strptime(str(tweet.created_at), f) - datetime.strptime(str(aux), f)).total_seconds()
      interval.append(seconds / 60)
    aux = str(tweet.created_at)
    i += 1

  return textList, interval, qtd_char, qtd_retweets, qtd_urls
```

Figura 11. Refinamento dos atributos do tuíte.

A Figura 12 apresenta os dados extraídos e calculados da conta relacionados a categoria *User Content* (*UC*), em que são verificados os tuítes da conta. São analisadas as médias: (i) tamanho do tuíte; (ii) retuítes; (iii) *URLs* presentes no tuíte; e (iv) se há um padrão na média de caracteres por *posts*, em que caracteriza um spambot.

```
def uc(self, textList, qtd_char, qtd_retweets, qtd_urls):
    size_text_list = len(textList)

    average_size_tweet = qtd_char / size_text_list
    average_retweet_count = (100 * qtd_retweets) / size_text_list
    average_urls_count = (100 * qtd_urls) / size_text_list

    range_size_tweet = 0
    spambot = 0

    for tweet in textList:
        if len(tweet) == average_size_tweet:
            spambot += 1

    perc_spambot = (100 * spambot) / size_text_list

    if perc_spambot > 60:
        ucSpam = True
    else:
        ucSpam = False

    if average_retweet_count > 60 or average_urls_count > 60:
        uc = True
    else:
        uc = False

    return uc, ucSpam
```

Figura 12. Extraindo dados da categoria *User Content* (*UC*).

Figura 13 mostra os dados extraídos e verificados da conta relacionados a categoria *User Demographics* (*UD*), em que são verificados os atributos da conta. São anali-

sados se: (i) a conta é verificada; (ii) possui a palavra bot no nome, nome da tela ou na descrição; e (iii) se o perfil e a foto do perfil é *default* (padrão).

```python
def ud(self, user_data):
  ud = False
  verified = False

  self.info(user_data)

  if user_data.verified:
    verified = True
  else:
    if bool(re.search('\bbot\b', user_data.name.lower())):
      ud = True
    elif bool(re.search('\bbot\b', user_data.screen_name.lower())):
      ud = True
    elif bool(re.search('\bbot\b', user_data.description.lower())):
      ud = True

    if user_data.default_profile:
      ud = True
    elif user_data.default_profile_image:
      ud = True

  return verified, ud
```

Figura 13. Extraindo dados da categoria *User Demographics* (*UD*).

A Figura 14 apresenta os dados extraídos e verificados da conta relacionados a categoria *User Friendship Network* (*UFN*), em que são analisadas as porcentagens de amigos bidirecionais calculadas com os *followers* e *following*.

```python
def ufn(self, following, followers):
    account_following = len((set(following) & set(followers)).symmetric_difference(followers))
    follow_account = len((set(following) & set(followers)).symmetric_difference(following))

    perc_following = (100 * account_following) / len(followers)
    perc_follow = (100 * follow_account) / len(following)

    if  perc_following > 60 and perc_follow > 50:
        ufn = True
    elif perc_following > 60:
        ufn = True
    else:
        ufn = False

    return ufn
```

Figura 14. Extraindo dados da categoria *User Friendship Network* (*UFN*).

A Figura 15 mostra os dados temporais extraídos e calculados da conta, em que é verificado a quantidade de tuítes postados ao mesmo tempo.

```python
def ut(self, interval):
    same_moment = -1

    for timeInterval in interval:
        if timeInterval == 0:
            same_moment += 1

    perc_same_moment = (100 * same_moment) / len(interval)

    if perc_same_moment > 20:
        ut = True
    else:
        ut = False

    return ut
```

Figura 15. Extraindo dados temporais do tuíte.

O valor de comparação das porcentagens é resultado do valor encontrado através do método de *manual labeling*, em que é realizada múltiplas execuções em busca de um padrão ou referência. Os critérios utilizados para classificar a conta como bot ou humano são através das categorias *UD*, *UFN* e *UC*, levando em consideração o atributo temporal da conta (*UT*). A Figura 16 apresenta como foram utilizadas as categorias para classificação.

```
def classif(self, ufn, uc, ucSpam, ud, verified, ut):
  if verified:
    label = "human"
  elif ufn and uc:
    label = "bot"
  elif ufn and ucSpam:
    label = "bot"
  elif ufn and ud:
    label = "bot"
  elif ucSpam and uc:
    label = "bot"
  elif ucSpam and ut:
    label = "bot"
  elif uc and ut:
    label = "bot"
  elif uc and ucSpam:
    label = "bot"
  elif uc:
    label = "bot"
  elif ucSpam:
    label = "bot"
  else:
    label = "human"

  return label
```

Figura 16. Critérios Classificador

Nas Figuras 17 e 18 estão representados alguns dados extraídos relevantes para classificação, além de como estão inseridos no MongoDB.

Figura 17. Dado inserido no MongoDB.

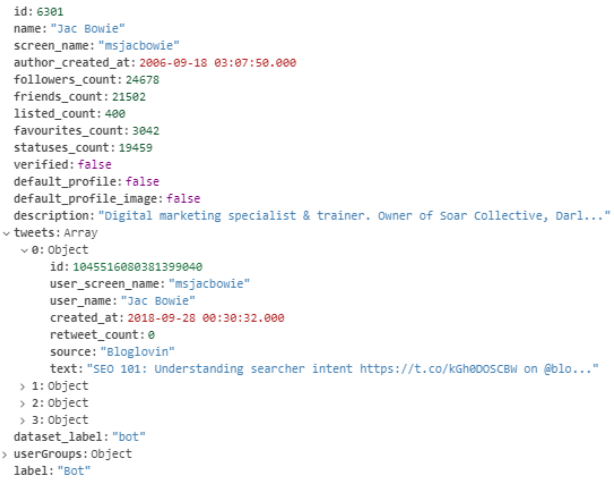

Figura 18. Dado inserido no MongoDB.

5. Resultados

Nesta seção, são apresentados os resultados que foram obtidos durante o desenvolvimento do trabalho. As Figuras 19 e 20 apresentam os resultados obtidos por meio das informações extraídas e calculadas da conta. Em que são expressos nas subpartes das figuras: (A) cálculo e percentuais de amigos bidirecionais; (B) informações relevantes da conta; (C) informações relevantes dos tuítes; (D) cálculos e percentuais dos atributos do tuíte; e (E) como foi classificado pelo algoritmo.

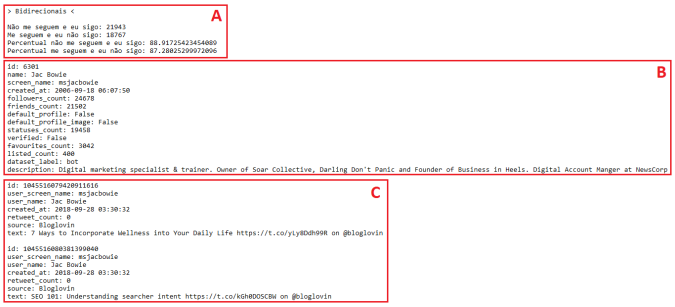

Figura 19. Informações Relevantes da conta e do tuíte.

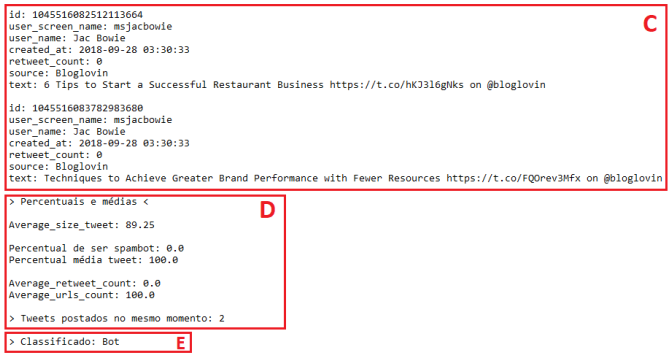

Figura 20. Informações Relevantes da conta e do tuíte.

Este trabalho analisou 100 contas do *dataset* caverlee-2011 [Lee et al. 2011], em que, 13 não foram analisadas devido ao *status* da conta (suspensa ou deletada). Dessas 87 contas são apresentados 41 bots e 46 humanos.

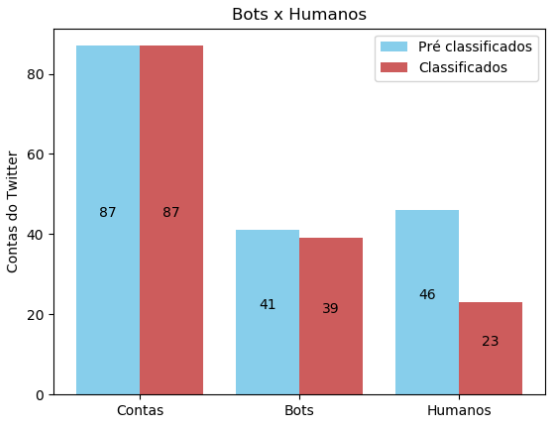

Figura 21. Resultado Gráfico do Classificador.

O classificador analisou corretamente 62 de 87 contas, especificamente 39 bots e 23 humanos (Figura 21), ou seja, o percentual de corretamente analisadas é 71,26%. Com isso, é validado os resultados dos dados analisados pelo classificador, em que as métricas utilizadas são apresentadas na Seção 4, e os percentuais encontrados estão presentes na Tabela 1.

Tabela 1. Validação dos Resultados

Precisão	*Recall*	Medida F_1	*Fall-Out*
50%	92,68%	64,96%	60,94%

6. Discussões

Nesta investigação, buscou-se construir um classificador com auxílio de tecnologias livres, *API* Tweepy, e a teoria de desenvolvimento orientada a objetos. E nesse ponto, nos trabalhos relacionados não é claro o uso dessa abordagem, sendo assim, um diferencial deste trabalho. Também, como ilustrado nas figuras dos diagramas de classes e sequências, procurou-se modelar e implementar o entorno do classificador com essa abordagem orientada a objetos, com foco no reuso, modularização, qualidade de serviços, entre outros.

Outra discussão bastante pertinente, é a escolha das heurísticas apresentadas na Seção 4 e o processo de validação F_1, em que registra-se o sucesso na escolha das heurísticas relevantes na detecção de bots.

Em relação aos trabalhos correlatos apresentados, é possível destacar que esta pesquisa apenas utilizando as heurísticas relevantes, comparada com os outros trabalhos em que foi utilizado algoritmos de aprendizado de máquina, aumento de precisão e redução de variância, revelou resultados interessantes e satisfatórios.

Finalmente, também é importante registrar o desempenho da *API* Tweepy, uma vez que é necessário seu uso para cada execução do algoritmo deste trabalho, em que para classificar a conta como humano, a limitação de requisições aos tuítes do usuário via *API* tornou-se um problema, já que o número de tuítes que podiam ser analisados eram pequenos. Dessa forma, o resultado do classificador poderia ser induzido a um resultado equivocado.

Portanto, agregou resultado, mas foi limitante na questão do número de contas e o número de tuítes que conseguia se analisar, ou seja, requisições a *API* (Figura 22) para conseguir os dados da conta incluso os tuítes.

Endpoint	Resource family	Requests / window (user auth)	Requests / window (app auth)
GET followers/ids	followers	15	15
GET statuses/user_timeline	statuses	900	1500
GET friends/ids	friends	15	15

Figura 22. *Rate Limiting.*

7. Conclusões

O trabalho apresentou um embasamento teórico, pois se fez necessário para introduzir conceitos e métodos utilizados por autores correlatos, assim percebendo-se suas estratégias para a detecção de bots, bem como o motivo de suas escolhas. Realizou-se uma pesquisa da amplitude do problema a ser resolvido, o porquê do uso malicioso de bots em redes sociais ser um problema, definição de bots e seus comportamentos, técnicas de detecção de bots, maneiras de classificar um bot e como validar os resultados obtidos da classificação.

Percebeu-se na pesquisa do referencial teórico uma variedade nos resultados obtidos com base nos diferentes métodos utilizados para a classificação, como métodos manuais, ou seja, apenas com a própria retirada de informações, do que com métodos automatizados como o *Random Forest* com base na precisão, *recall* e na Medida F_1.

Os resultados apresentados foram satisfatórios, pois somente utilizando heurísticas comuns na detecção de bots apresentou resultados semelhantes a pesquisas iniciais dos trabalhos correlatos, em que inicialmente não utilizou-se aprendizado de máquina e algoritmos de *boost*.

Percebeu-se limitações no número de contas que poderiam ser analisadas durante um intervalo de tempo via requisições a *API* Tweepy. Essas restrições são provenientes da versão padrão de requisições de acesso ao Twitter, em que, para cada tipo de comunicação (*GET followers/ids*, *GET friends/ids* e *GET statuses/user_timeline*) entre *API* e algoritmo há um janela de tempo limite por requisição (Figura 22).

Para os trabalhos futuros é possível implementar um classificador utilizando métodos de aprendizado de máquina e algoritmos de *boost*, bem como utilizar heurísticas mais complexas na detecção. Além disso, há a possibilidade de implementar e validar o classificador em diferentes *datasets*, em que as contas de usuários dos mesmos estejam

dispersos ao redor do globo, como o ASONAM 2015 [Morstatter et al. 2016], que contém informações de usuários líbios (Norte da África). Com isso os idiomas desses países são irrelevantes para o processo de implementar um detector.

Referências

Boshmaf, Y., Muslukhov, I., Beznosov, K., and Ripeanu, M. (2013). Design and analysis of a social botnet. *Computer Networks*, 57(2):556–578.

Chu, Z., Gianvecchio, S., Wang, H., and Jajodia, S. (2010). Who is tweeting on twitter: human, bot, or cyborg? In *Proceedings of the 26th annual computer security applications conference*, pages 21–30. ACM.

Gilani, Z., Farahbakhsh, R., and Crowcroft, J. (2017). Do bots impact twitter activity? In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 781–782. International World Wide Web Conferences Steering Committee.

Haugen, G. M. S. (2017). Manipulation and deception with social bots: Strategies and indicators for minimizing impact. Master's thesis, NTNU.

Itagiba, G. (2017). Fake news e internet: esquemas, bots e a disputa pela atenção. Disponível em: <https://itsrio.org/wp-content/uploads/2017/04/v2_fake-news-e-internet-bots.pdf>. Acesso em: Março 2018.

Lee, K., Caverlee, J., and Webb, S. (2010). Uncovering social spammers: social honeypots+ machine learning. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 435–442. ACM.

Lee, K., Eoff, B. D., and Caverlee, J. (2011). Seven months with the devils: A long-term study of content polluters on twitter. In *ICWSM*.

Martim, J. (2017). What are bots? Disponível em: <https://www.techadvisor.co.uk/feature/software/what-are-bots-3638979/>. Acesso em: Março 2018.

Morstatter, F., Wu, L., Nazer, T. H., Carley, K. M., and Liu, H. (2016). A new approach to bot detection: striking the balance between precision and recall. In *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pages 533–540. IEEE.

Pearl, J. (1984). Heuristics: intelligent search strategies for computer problem solving.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Woolley, S. C. and Guilbeault, D. R. (2017). Computational propaganda in the united states of america: Manufacturing consensus online. *Computational Propgaganda Research Project*, page 22.

Xie, Y., Yu, F., Ke, Q., Abadi, M., Gillum, E., Vitaldevaria, K., Walter, J., Huang, J., and Mao, Z. M. (2012). Innocent by association: early recognition of legitimate users. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 353–364. ACM.