Sistema de detecção de autoria textual por inteligência artificial

Raoni Leandro B. Nascimento[1], **Deylon Carlo**[2]

[1] Curso de Sistemas de Informação
Instituto Federal de Minas Gerais – Sabará, MG – Brasil

raonileandro@gmail.com, deyloncarlo@gmail.com

Resumo. *A inteligência artificial tem sido um dos principais temas de estudo, no que se refere a tecnologia. Com o aumento crescente na produção textual em nosso século, percebe-se a existência de um ambiente propicio para o desenvolvimento de ferramentas que verificam a autoria textual, capazes de aprender, identificar e reduzir o número de quantidade de fraudes que acontecem atualmente nas mais diversas áreas do conhecimento.*

1. Introdução

O trabalho desenvolvido consiste em criar um sistema capaz de identificar a autoria de um texto através do uso da inteligência artificial. Utilizando aprendizagem de máquina, o sistema, que através do fornecimento prévio de dados, possibilitará o reconhecimento de padrões e características que comprovam a autoria do texto produzido. Para isso, como fonte de teste, a primeira etapa consistirá na criação de crawlers através da linguagem de programação Python, que irão recolher notícias de economia identificando os autores e selecionando a notícia em uma pasta de seu respectivo autor. O que se pretende é reunir uma base de dados em um arquivo .arff consistente em que se possa aplicar variados testes, a fim de verificar se os processos de aprendizagem estão acontecendo conforme o esperado.

2. Metodologia

2.1. Coleta de dados

Com o objetivo de auxiliar os algoritmos que serão utilizados para identificar o autor de um determinado texto, foi criada uma base de dados, globo.arff, com reportagens de diversos autores. Nesta base de dados, arquivo no formato de entrada do programa WEKA, estão sendo armazenadas quatro informações referentes a cada reportagem coletada, sendo elas:

- **Título** - *Utilizado para diferenciar cada reportagem.*
- **Autor** - *Utilizado para identificar o responsável pela autoria do texto.*
- **Data de publicação** - *Data em que a reportagem foi publicada no portal G1.*
- **Conteúdo da reportagem** - *Utilizado para identificar características que singularizam cada autor.*

Em função do G1 se tratar de um portal web de notícias, e não de uma base de dados de notícia, tornou-se necessário criar dois Crawlers que pudessem, diariamente, coletar as últimas reportagens publicadas e, assim, coletar seus conteúdos.

2.2. Crawler RealCrawler

O crawler *realCrawler.py* ficou responsável por coletar os links das últimas reportagens publicadas no portal G1. Com o intuito de obter diferentes reportagens, porém ,frequentemente, reportagens de um mesmo autor, foi definido que apenas seriam coletas reportagens da sessão de economia do portal, ou seja, foram coletados apenas os links que estiverem no corpo da página *http://g1.globo.com/economia/*.

O crawler *realCrawler.py* foi programado para executar todos os dias às 20:00 horas durante 30 dias seguidos, sendo que a cada 2 dias eram obtidos 10 links diferentes dos já coletados anteriormente. Os links coletados foram armazenados em um arquivo, de nome *links.txt*, para gerenciar os links coletados e não permitir que um mesmo link fosse coletado mais de uma vez. Note que o *for* percorre todo o *array* verificando se o link já foi coletado. O resultado é definido dentro da estrutura condicional, como é possível ver abaixo.

```
print("\nEscrevendo no arquivo de links...\n")
            arquivo = open(self.filePath, 'a')

for link in self.linkList:
    if(link + '\n' in self.linksOnFile):
        print('Link ja coletado: ', link)
    elif ((link) + '\n' in self.linksOnFile) == False and
    ('/economia/' in link):
        arquivo.write(str(link))
        arquivo.write("\n")
        arquivo.close()
        print("\nFim escrita no arquivo de links...\n")
```

Construído em Python, o algoritmo utiliza a biblioteca Scrapy, para fazer leitura da página, utilizando das linguagens de formatação, HTML e CSS, para identificar os elementos HTML que continham os links das notícias.

```
def coletarLinks(self, response):
self.linkList = []
for v_div in response.css('div.feed-text-wrapper'):
v_link = v_div.css('a').xpath('@href').extract_first()
v_link = v_link.encode('ascii', 'ignore')
self.linkList.append(str(v_link))
escreverLinks()
```

No trecho destacado percebe-se como é realizada a coleta do link. O crawler verifica onde está a *¡div¿* com que possui a classe *'feed-text-wrapper'* presente em todos os links do site e obtém o link, de fato, do elemento HTML *¡a¿*. Logo, o link é armazenado em um array de *Strings*. Por último, é chamada a função *escreverLinks()*, que irá registrar o link no arquivo *links.txt*.

2.3. Crawler LinkColector

Este crawler é responsável coletar o conteúdo das notícias. Além disso, o *linkColector.py* verifica se o link, que está armazenado no arquivo *links.txt* já foi coletado anteriormente e

separa as informações contidas na notícia (autor, data, título e conteúdo) para, enfim, criar uma instância do conteúdo no arquivo globo.arff. Abaixo, tem-se um trecho do código que mostra como o conteúdo é instanciado no globo.arff.

```
 def createFileWithContent(p_title, p_author, p_date, p_content):
if(os.path.exists('content/' + p_author + '/') == False):
os.mkdir('content/' + p_author + '/')

v_file = open('content/' + p_author + '/' + p_title + '.arff', 'w')
v_file.write('@relation TEXTvsAUTHOR')
v_file.write('\n\n')
v_file.write('@attribute title string')
v_file.write('\n')
v_file.write('@attribute author string')
v_file.write('\n')
v_file.write('@attribute date string')
v_file.write('\n')
v_file.write('@attribute content string')
v_file.write('\n\n')
v_file.write('@data')
v_file.write('\n')
v_file.write( '"' + p_title + '"' + ',' +'"' +%
p_author + '"' + ',' + '"' + p_date +'"' + ',' +%
'"' + p_content + '"');
```

Depois deste processo, o link da notícia é armazenado no arquivo *linksCrawled.txt*. Neste arquivo estão armazenados todos os links cujas notícias foram coletadas. Os dados registrados totalizaram 13 linhas (links) da sessão de economia do G1, totalizando 1.7 KB.

Referências

[1] **Python documentation** Disponível em: https://www.python.org/doc/

[2] **Categorização de Textos utilizando algoritmos de aprendizagem de máquina com Weka** KULTZAK, Adriano Francisco. 74pg. Trabalho de conclusão de curso, 2016.