SCRAPING NEWS SITES AND SOCIAL NETWORKS FOR PREJUDICE TERM ANALYSIS

Pedro Rangel Henriques, Cristiana Araújo, Isabel Ermida and Idalete Dias
Universidade do Minho, Portugal

ABSTRACT

Computer-Mediated Communication (CMC) has paved the way for new patterns of linguistic aggravation. Hidden behind the screen, anyone can comment on any other person's opinion using an offensive or injurious tone. Besides, types of prejudice such as homophobia, sexism, racism, xenophobia, anticlericalism, body/addiction shaming, among others, are easily found nowadays in social networks and other forms of interactive Web sites potentiated by Web 2.0. This increasing violence deserves further investigation from different academic perspectives, among which Sociolinguistics stands out. This paper is concerned with the design and development of a set of computer-based tools to collect articles and posts with the respective comment threads that can be used as sources to extract prejudice terms and allow different analyses to be conducted. These prejudice terms were devised using a sociolinguistic variable stratification approach. We will focus on the filters used to extract the relevant fields from the Web pages collected, and on the converters used to adapt formats to obtain a common format for information representation. We will also introduce the statistical analysis processor that explores the extracted data, in that format, to output a set of indicators.

KEYWORDS

Information Extraction, Social Web, Web Scraping, Computer-Mediated Communication

1. INTRODUCTION

Our team's project deals with hate speech online, i.e., the expression of prejudice and discrimination on the Internet. It aims to identify recurrent words, idioms and constructions, as well as communicative routines and patterns of pragmatic interaction, that may serve as a basis for abuse detection. The object of analysis is online social platforms, such as Facebook, YouTube, Twitter, and Web journal comment boards, where participants often engage in expressing opinions that are ideology-laden and biased against a range of social groups and subjects.

The project aims to build and annotate a comparable corpus of online texts, on a range of similar topics but in different languages, namely Portuguese and English. This contrastive analysis will hopefully allow us to assess the socio-cultural similarities and differences of the two linguistic contexts in an admittedly globalized society. Annotation of the corpus will begin by metadata, which will stratify the texts according to sociolinguistic variables such as age, gender, ethnicity, nationality and social class. This will pave the way for studies in language variation. Structural taggers will be inserted next, providing information about the morpho-syntactic organization of the texts, which may carry ideological intent. Finally, content taggers will help design a descriptive ontology of the corpora that will lend itself to various interpretive tasks. The results of this extended study will help refine informatics tools of control and detection on institutional levels, including automatic identification of situations of risk and abuse online.

The research question we intend to answer in this paper can be stated as follows: How to integrate into a single corpus posts with comments, containing prejudice terms, extracted from different social networks and news sites?

In this context, the paper focuses on the software paraphernalia that is under development to support the extraction from multiple sources of hate speech sentences.

The tools and techniques we have used may not be new, but the novelty and the strength of our work so far is that we have managed to integrate all available tools into a unified approach. Thus, we have succeeded in tackling the various problems which the different sources posed and can now choose the appropriate tool and extraction method for each different source. Besides, we have managed to integrate the partial results into a global analysis tool that is automated and available through a simple Web interface. Additionally - and crucially - we have also started to successfully collect the metadata of each extraction.

Section 2 presents a set of keywords for text extraction, devised according to a sociolinguistic variable stratification method. Section 3 introduces the approach followed to execute the tasks involved in the extraction. Then Sections 4 and 5 discuss in detail the extractors of material from sources such as social networks and newspaper sites. Section 6 presents the system of detection and analysis of discursive prejudices. Finally, Section 7 presents the conclusions and future work.

2. SOCIOLINGUISTIC STRATIFICATION OF ONLINE PREJUDICE AND DISCRIMINATION: KEYWORDS FOR DATA EXTRACTION

The project aims to analyze the linguistic forms and strategies of prejudice in Computer-Mediated Communication (CMC). As CMC is a phenomenon that brings together multiple manifestations, not only at the linguistic level (morphology, syntax, lexical semantics), but also (inter)actionally and pragmatically, it is important to cross it with the sociolinguistic variables it involves.

With this caveat in mind, our team devised a keyword table for data extraction, at the very outset of the project. This table covers the following elements: (i) different types of prejudice; (ii) the corresponding sociolinguistic variables they target; and (iii) the various keywords and expressions which, in English and Portuguese, are deemed typical lexical signals of prejudice expression. Ten **types of prejudice** were considered: *sexism*, *ageism*, *racism*, *nationalism*, *classism*, *homophobia*, *anticlericalism*, *ideological shaming*, *body shaming*, and *addiction shaming*. The former five correspond to the following classic **sociolinguistic variables**, respectively: *gender, age, race (and ethnicity), nationality*, and *social class* (Labov, 2001). Prejudice types six to eight correspond to the following identity categories: *sexual identity*, *religious identity*, and *ideological/political identity* (Coupland, 2007). Finally, items nine and ten correspond to *physical identity* and *behavioral identity*. For each of these parameters a list of keywords, especially slurs, was developed, to help scrape the Web for texts that express prejudice.

This approach will cater for discursive and pragmatic analyses, with a view to identifying emerging ideological communities (Lambert, 2007). The aim is to know what actors - both active and passive, victimizers and victimized - populate the virtual world of marginalization and conflict, and what their recurrent identities and attitudes are in terms of linguistic expression.

The approach for extracting and analyzing comment boards is presented in Section 3.

3. EXTRACTION AND ANALYSIS - THE APPROACH

As previously mentioned, the project aims to analyze verbal interaction in online comment boards. At this stage, we are extracting data from two types of online platforms: newspaper sites and social networks (YouTube and Facebook), in the Portuguese language.

Before starting the extraction process, it is necessary to find news articles and publications that can trigger biased and discriminatory comments. Our approach to searching across platforms has been to use the keywords (see Section 2) in the platform search to find the publications that contain those keywords. After finding them we start the extraction. It is important to note that in addition to extracting comments we also extract the news articles, publications, audio-visual texts, etc., that triggered the thread of comments. As the goal of the project is to construct a corpus for linguists and other specialists to use in their research, we find it appropriate to keep the source text that gave rise to the verbal interaction. The news articles found in the newspaper sites will then be looked for in the respective newspaper Facebook page in order to extract the post and the comment thread from there. In general, we have discovered that the same news article has a greater number of comments on the newspaper's Facebook page, compared to the newspaper site, in the case of Portuguese newspapers. In the newspaper "Correio da Manhã" (a Portuguese tabloid), we are only going to extract the comment threads on the newspaper's Facebook page because the contents of the comment threads on the newspaper site have recently stopped being accessible. Our extraction and comment analysis schema is shown in Figure 1.

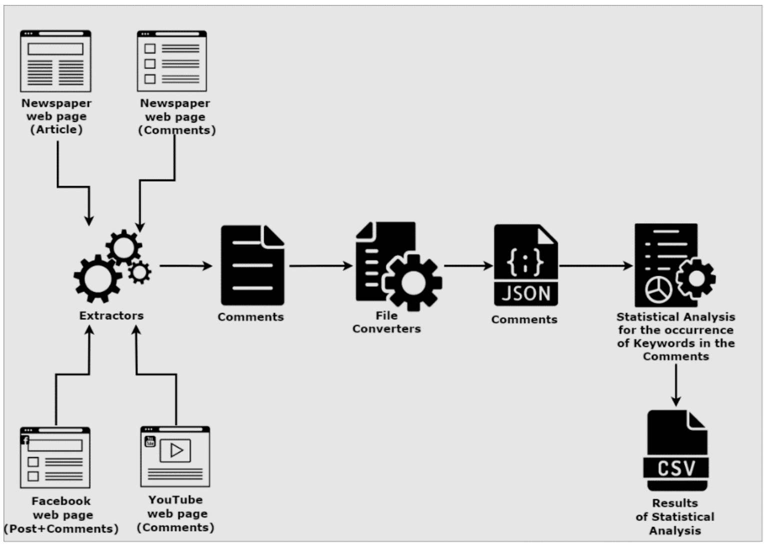

Newspaper web page (Article)
Newspaper web page (Comments)
Extractors
Comments
File Converters
JSON
Comments
Statistical Analysis for the occurrence of Keywords in the Comments
CSV
Results of Statistical Analysis
Facebook web page (Post+Comments)
YouTube web page (Comments)

Figure 1. General Extraction Schema[1]

Figure 1 is a simplified schema that contains all the steps from the extraction to the analysis of the comments. *Extractors* will scrape web pages (Newspaper, Facebook, YouTube) and as a result we obtain a file, which can be in a range of formats, such as HTML, JSON, CSV, etc., depending on each extractor. Subsequently the extracted files will be converted into a standard format resorting to the *File Converters*. In this case, we have chosen JSON as the standard format. Finally, comment threads in JSON format will be examined according to a *Statistical Analysis* to evaluate the *Frequency of occurrence of the keywords* which reveal prejudice and discrimination in the comments. The results of this analysis are saved in a *CSV file*. In the following sections we will explain all these processes in more detail.

4. EXTRACTING POSTS/COMMENTS FROM ONLINE SOCIAL NETWORKS

Extracting the posts and comment threads from Web social networks (YouTube and Facebook) is carried out using two freely available online tools. A detailed outline of the steps of this extraction is shown in Figure 2.

[1] All the icons of Figures 1, 2 and 5 were taken from the website: "The Noun Project" (*https://thenounproject.com/*). Accessed: 2019-04-10.

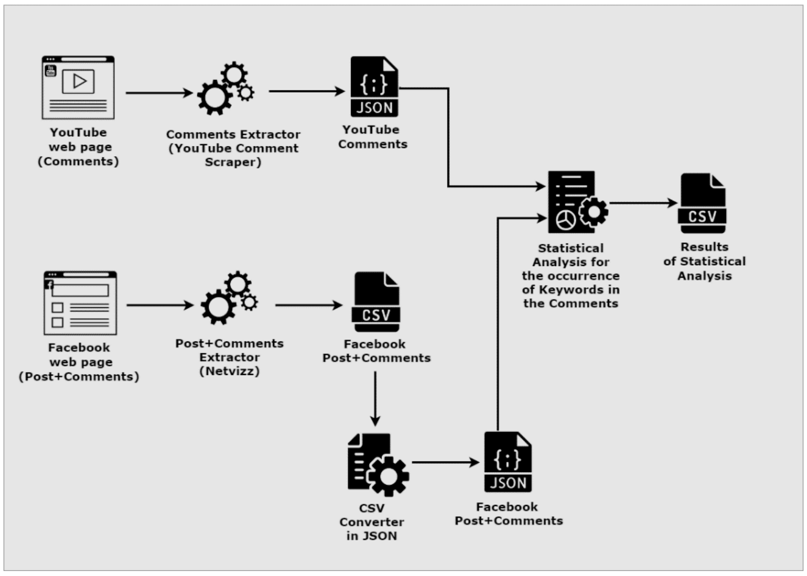
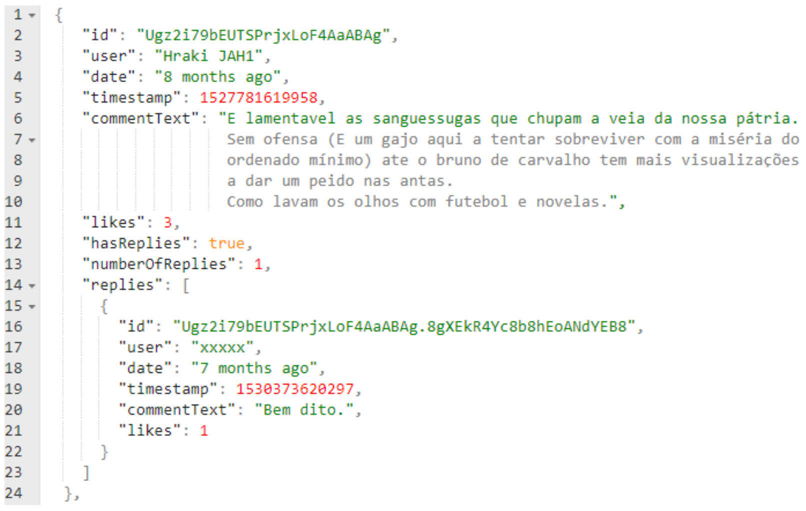

YouTube web page (Comments)
Comments Extractor (YouTube Comment Scraper)
JSON
YouTube Comments
Statistical Analysis for the occurrence of Keywords in the Comments
CSV
Results of Statistical Analysis
Facebook web page (Post+Comments)
Post+Comments Extractor (Netvizz)
CSV
Facebook Post+Comments
CSV Converter in JSON
JSON
Facebook Post+Comments

Figure 2. YouTube and Facebook Extraction Schema

On YouTube the extraction takes place using the application *YouTube Comment Scraper*[2]. To start the extraction, it is necessary to enter the publication link and the application automatically extracts the thread with all comments in JSON and CSV format. In our case, the thread was extracted in JSON format, to standardize all extractions. In Figure 3, a fragment of a comment thread extracted from YouTube is exhibited.



```
{
    "id": "Ugz2i79bEUTSPrjxLoF4AaABAg",
    "user": "Hraki JAH1",
    "date": "8 months ago",
    "timestamp": 1527781619958,
    "commentText": "E lamentavel as sanguessugas que chupam a veia da nossa pátria.
                    Sem ofensa (E um gajo aqui a tentar sobreviver com a miséria do
                    ordenado mínimo) ate o bruno de carvalho tem mais visualizações
                    a dar um peido nas antas.
                    Como lavam os olhos com futebol e novelas.",
    "likes": 3,
    "hasReplies": true,
    "numberOfReplies": 1,
    "replies": [
      {
        "id": "Ugz2i79bEUTSPrjxLoF4AaABAg.8gXEkR4Yc8b8hEoANdYEB8",
        "user": "xxxxx",
        "date": "7 months ago",
        "timestamp": 1530373620297,
        "commentText": "Bem dito.",
        "likes": 1
      }
    ]
  },
```

Figure 3. Comment thread extracted from YouTube (fragment)

According to Klostermann (2015), the fields extracted in each comment thread are: *id* - ID of the comment; *user* - name or nickname of the author of the comment; *date* - publishing date of the comment; *timestamp* - time an event is logged by a computer, not the time of the event itself; *commentText* - text of the comment; *likes* - number of likes on the comment; *hasReplies* - whether the comment has replies; *numberOfReplies* - number of replies; *replies* - whether the comment is a reply to another comment (in threaded conversations).

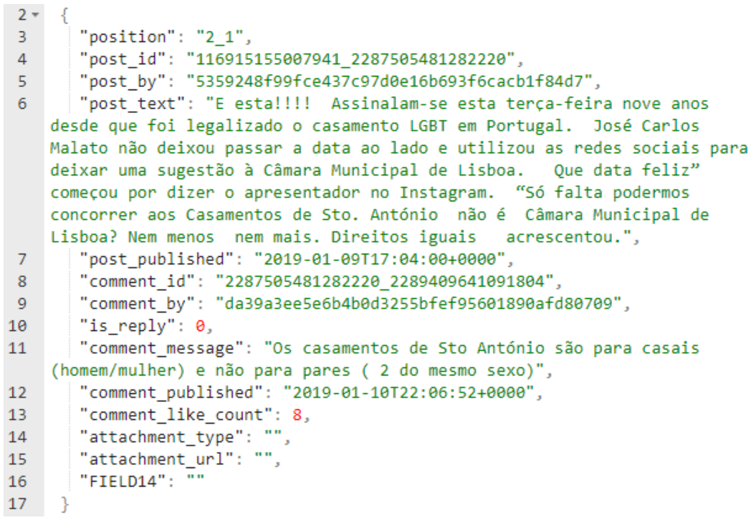
[2] YouTube Comment Scraper: *http://ytcomments.klostermann.ca/*.

It is important to note that this application only extracts the comment thread and does not extract any information regarding the publication. Extraction on Facebook is done with the tool *Netvizz*[3], a Facebook application that can be found by typing the name into the main Facebook search box. To extract the data from a page, it is only necessary to enter the page ID and date scope. The result of the extraction is a CSV file that in this case contains the comment thread and publication information (Rieder, 2013).

According to Rieder (2013), the fields extracted by Netvizz are: *position* - postnumber_commentnumber (e.g. 0_0 for the first comment on the first post); *post_id* - ID of the post; *post_by* - author of the post; *post_text* - text of the post; *post_published* - publishing date of the post; *comment_id* - ID of the comment; *comment_by* - author of the comment (column no longer in use, remains for compatibility); *is_reply* - whether the comment is a reply to another comment (in threaded conversations); *comment_message* - text of the comment; *comment_published* - publishing date of the comment; *comment_like_count* - number of likes on the comment; *attachment_type* - attachment type (eg: animated_image_share); *attachment_url* - attachment URL.

As the results of Facebook extraction are obtained in CSV format, it is necessary to convert them into JSON format. The transformation of files extracted from Facebook in CSV format to JSON format is performed by a script in Python. For each line of the CSV file, the Converter forms a name/value pair, where the *field name* corresponds to its *column name* and *value* is the *extraction result* of the respective field. For example, "*comment_message*": "*Os casamentos de Sto António são para casais (homem/mulher) e não para pares (2 do mesmo sexo)*" ["*Saint Anthony's weddings are for man/woman couples, not for same-sex pairs*"], where "comment_message" is a column name and "*Os casamentos de Sto António são para casais (homem/mulher) e não para pares (2 do mesmo sexo)*" corresponds to the extraction result. The JSON generated by the converter is listed in Figure 4.

In Section 5 we will explain the extractors of newspaper sites in detail.



```
{
  "position": "2_1",
  "post_id": "116915155007941_2287505481282220",
  "post_by": "5359248f99fce437c97d0e16b693f6cacb1f84d7",
  "post_text": "E esta!!!!  Assinalam-se esta terça-feira nove anos
desde que foi legalizado o casamento LGBT em Portugal.  José Carlos
Malato não deixou passar a data ao lado e utilizou as redes sociais para
deixar uma sugestão à Câmara Municipal de Lisboa.   Que data feliz”
começou por dizer o apresentador no Instagram.  “Só falta podermos
concorrer aos Casamentos de Sto. António  não é  Câmara Municipal de
Lisboa? Nem menos  nem mais. Direitos iguais   acrescentou.",
  "post_published": "2019-01-09T17:04:00+0000",
  "comment_id": "2287505481282220_2289409641091804",
  "comment_by": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
  "is_reply": 0,
  "comment_message": "Os casamentos de Sto António são para casais
(homem/mulher) e não para pares ( 2 do mesmo sexo)",
  "comment_published": "2019-01-10T22:06:52+0000",
  "comment_like_count": 8,
  "attachment_type": "",
  "attachment_url": "",
  "FIELD14": ""
}
```

Figure 4. Comment thread extracted from Facebook in JSON (fragment)

5. EXTRACTING ARTICLES/COMMENTS FROM NEWSPAPER WEB PAGES

The comment thread and article extraction from the newspaper web pages is performed by Python scripts, one to extract the article from the newspaper, and another script to extract comments related to the article. A detailed outline of the steps of this extraction is shown in Figure 5.

The newspaper sites are dynamic Web pages, that is, they assemble content that comes from different sources. They use scripts that perform functions such as: database information, article comments, links to other news, links to different topics/news categories, etc., and fill the page with content from different sources in the

[3] Netvizz: https://apps.facebook.com/netvizz

visualization moment (i.e. when the page is requested). This means that the page content is not fully included in the original static page.

An example of content that is integrated into the newspaper article page is the comment thread and, therefore, its extraction becomes more difficult. This is noticeable when we click on “inspect element”: on the newspaper page we can see we can see the HTML code with the comment thread, but when we click on “view page source” we cannot find the HTML code with the comment thread. The explanation for this omission is that “view page source” only shows the content that was found on the server. However, the final DOM (Document Object Model) that is rendered by the browser can be very different, because the JavaScript that is compiled when opening the page manipulates the DOM and integrates content that comes from the different sources. This leads us to see the HTML code with the comment thread when we click on “inspect element” (Prutsachainimmit and Nadee, 2018).

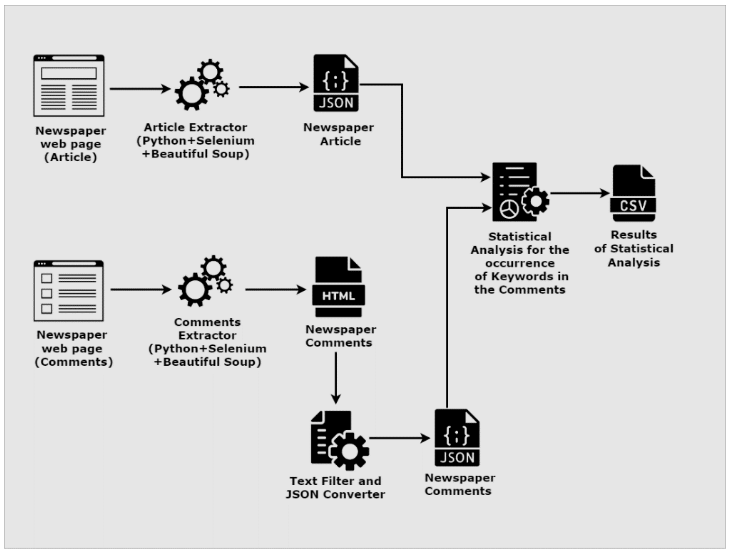

Newspaper web page (Article)
Article Extractor (Python+Selenium +Beautiful Soup)
JSON
Newspaper Article
Statistical Analysis for the occurrence of Keywords in the Comments
CSV
Results of Statistical Analysis
Newspaper web page (Comments)
Comments Extractor (Python+Selenium +Beautiful Soup)
HTML
Newspaper Comments
Text Filter and JSON Converter
JSON
Newspaper Comments

Figure 5. Newspaper Extraction Schema

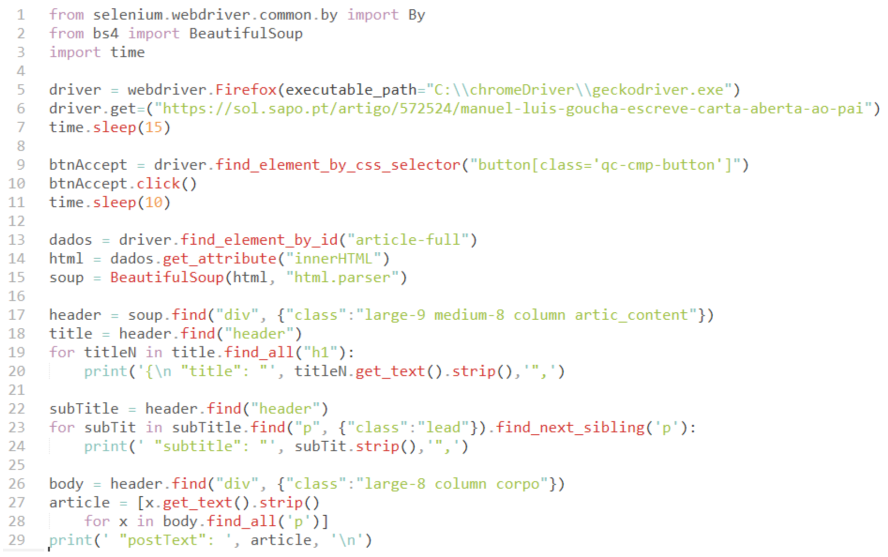
Faced with this difficulty, it was necessary to understand the different types of scraping techniques avaliable and how we would apply them to carry out the extraction. Scraping can be either static or dynamic. *Static scraping* bypasses JavaScript and searches the server's Web pages without the help of a browser. This type of scraping extracts exactly all the information we see when we click on “view page source”. Static scraping works using two Python packages: *Requests*[4] to find web pages and *Beautiful Soup*[5] to parse HTML pages. *Dynamic scraping* uses a real browser and allows the JavaScript on the page to run. Subsequently, it queries the DOM to extract the content we are looking for. There are also cases where it is necessary to automate the browser, that is, to simulate the steps a user takes to obtain the content needed. Dynamic scraping works by using the *Selenium*[6] Python package to automate the browser and interactively interact with the DOM. Selenium requires a *WebDriver*[7] (the browser is automated) for the Internet browser (Mitchell, 2015; Jarmul and Lawson, 2017).

In our case, the most adequate scraping technique is the dynamic one because the content of the comments comes from another source, as explained above. Dynamic scraping will allow us to find the information that comes from another source, but we also use static scraping to extract the information we want after finding it with dynamic scraping.Figure *6* shows a fragment of the Newspaper Article extractor (in this example we are extracting news from the Portuguese newspaper “Sol”), which, as already mentioned, uses dynamic scraping (Selenium and WebDriver) and static scraping (Beautiful Soup).

[4] Requests: https://2.python-requests.org/en/master/
[5] Beautiful Soup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[6] Selenium: https://www.seleniumhq.org
[7] WebDriver: https://www.seleniumhq.org/projects/webdriver/

All the data we want to extract from the newspaper article is on the source page, but we have decided to use both types of scraping, so to make it applicable to most types of content. In lines 5-14 we use dynamic scraping to open and load the page, and close cookie windows, etc. that appear when the page loads; and we use static scraping, lines 15-29, to search for the elements and extract the information.

The WebDriver we used was *Geckodriver* as it is the driver used for the connection to *Firefox* (line 5). The driver runs the browser and opens the link to the page we entered in the *driver.get* function (line 6). Subsequently, we look for the tag <*div*> whose *id* is *article-full* and store in the variable *dados* (lines 13 and 14). This tag contains all the HTML code with the information that we want to extract.

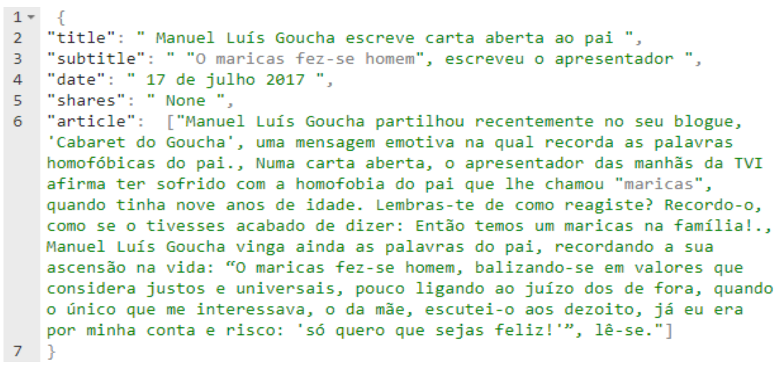


```
1   from selenium.webdriver.common.by import By
2   from bs4 import BeautifulSoup
3   import time
4
5   driver = webdriver.Firefox(executable_path="C:\\chromeDriver\\geckodriver.exe")
6   driver.get=("https://sol.sapo.pt/artigo/572524/manuel-luis-goucha-escreve-carta-aberta-ao-pai")
7   time.sleep(15)
8
9   btnAccept = driver.find_element_by_css_selector("button[class='qc-cmp-button']")
10  btnAccept.click()
11  time.sleep(10)
12
13  dados = driver.find_element_by_id("article-full")
14  html = dados.get_attribute("innerHTML")
15  soup = BeautifulSoup(html, "html.parser")
16
17  header = soup.find("div", {"class":"large-9 medium-8 column artic_content"})
18  title = header.find("header")
19  for titleN in title.find_all("h1"):
20      print('{\n "title": "', titleN.get_text().strip(),'",')
21
22  subTitle = header.find("header")
23  for subTit in subTitle.find("p", {"class":"lead"}).find_next_sibling('p'):
24      print(' "subtitle": "', subTit.strip(),'",')
25
26  body = header.find("div", {"class":"large-8 column corpo"})
27  article = [x.get_text().strip()
28      for x in body.find_all('p')]
29  print(' "postText": ', article, '\n')
```

Figure 6. Extractor of Newspaper Article (fragment)

Then we use Beautiful Soup to parse this HTML and save the HTML code returned by the Selenium, already converted to Beautiful Soup, within the variable *soup* (line 15). The *soup* variable is the tree root parsed from the HTML page that will allow us to browse and search elements in the tree. Within this tree we will look only for the specific elements that we want to extract: *title* - title of the article; *subtitle* - subtitle of the article; *date* - date on which the article was published; *shares* - number of article shares in social networks (when applicable); *article* - text of the article.

For example, to extract the title of the article we will look in the tree for <*div*> whose *class* is "*large-9 medium-8 column artic_content*" (which contains the header and body of the article) and store it in the variable *header*. Within the variable header we will look for the <*header*> tag, and within that tag search for the <*h1*> tag and save the text (title of the article) found in the <*h1*> tag in the variable *titleN*. This extractor stores information extracted in JSON format (see Figure 7).



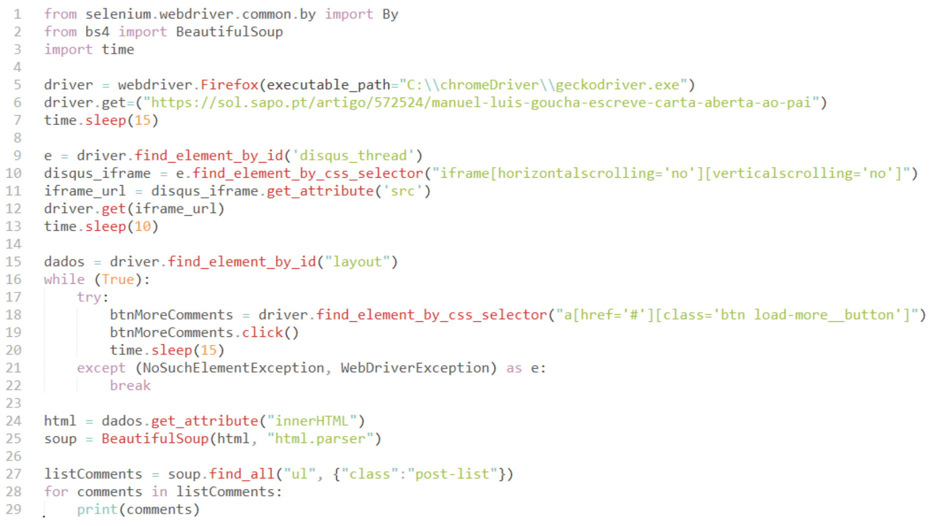
```
 {
"title": " Manuel Luís Goucha escreve carta aberta ao pai ",
"subtitle": " "O maricas fez-se homem", escreveu o apresentador ",
"date": " 17 de julho 2017 ",
"shares": " None ",
"article":  ["Manuel Luís Goucha partilhou recentemente no seu blogue,
'Cabaret do Goucha', uma mensagem emotiva na qual recorda as palavras
homofóbicas do pai., Numa carta aberta, o apresentador das manhãs da TVI
afirma ter sofrido com a homofobia do pai que lhe chamou "maricas",
quando tinha nove anos de idade. Lembras-te de como reagiste? Recordo-o,
como se o tivesses acabado de dizer: Então temos um maricas na família!.,
Manuel Luís Goucha vinga ainda as palavras do pai, recordando a sua
ascensão na vida: "O maricas fez-se homem, balizando-se em valores que
considera justos e universais, pouco ligando ao juízo dos de fora, quando
o único que me interessava, o da mãe, escutei-o aos dezoito, já eu era
por minha conta e risco: 'só quero que sejas feliz!'", lê-se."]
}
```

Figure 7. Extraction of Newspaper Article (example)

The comment thread extractor is shown in Figure 8 and, similarly to the previous extractor, it also uses dynamic and static scraping to perform the extraction. In this case, dynamic scraping plays a very important role because the comment thread is imported from an external source, that is, it is not in the source code of the main page. On the other hand, it is also important when there is an extensive list of comments, as it is necessary to simulate a user clicking on the button in order to see the remaining comments, otherwise the extractor only extracts the comments that are visible (open) in the HTML page at that time.

The search and extraction logic of this extractor is very similar to the previous one, the main difference is that it opens the main page of the news article (line 6), then it will look for the *url* of the sub-page (in the *<iframe>* tag in the attribute *src*) that contains the comment thread (lines 9-11), and finally, opens it to access its information (line 12). When the list of comments is extensive, a button is displayed to click and see the remaining comments. The simulation of that click as if it were a user is performed in lines 16-22. Finally, in lines 27-29 the list of the comment thread is searched and stored in the variable *listComments*. The result of this extraction is an extensive HTML code that contains all the comments.

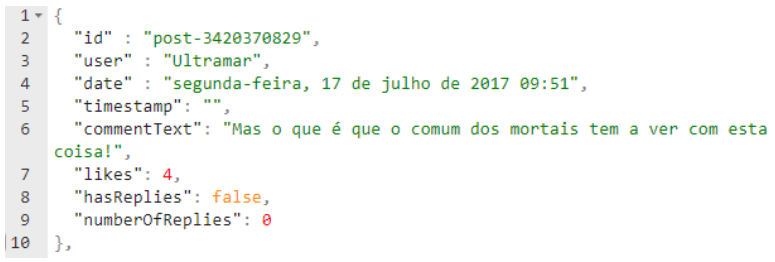
The next step is to filter the extracted information and convert it to JSON format. To achieve this, we constructed a Text Filter and JSON Converter, in Flex, which looks for the information that we want to extract and saves it in JSON format. This phase poses significant challenges: one being that HTML is very verbose and so it takes several steps to extract a simple field; another is the occurrence of responses to nested comments within each other, which makes it difficult to identify where responses end up, given various levels of nesting.



```
1   from selenium.webdriver.common.by import By
2   from bs4 import BeautifulSoup
3   import time
4
5   driver = webdriver.Firefox(executable_path="C:\\chromeDriver\\geckodriver.exe")
6   driver.get=("https://sol.sapo.pt/artigo/572524/manuel-luis-goucha-escreve-carta-aberta-ao-pai")
7   time.sleep(15)
8
9   e = driver.find_element_by_id('disqus_thread')
10  disqus_iframe = e.find_element_by_css_selector("iframe[horizontalscrolling='no'][verticalscrolling='no']")
11  iframe_url = disqus_iframe.get_attribute('src')
12  driver.get(iframe_url)
13  time.sleep(10)
14
15  dados = driver.find_element_by_id("layout")
16  while (True):
17      try:
18          btnMoreComments = driver.find_element_by_css_selector("a[href='#'][class='btn load-more__button']")
19          btnMoreComments.click()
20          time.sleep(15)
21      except (NoSuchElementException, WebDriverException) as e:
22          break
23
24  html = dados.get_attribute("innerHTML")
25  soup = BeautifulSoup(html, "html.parser")
26
27  listComments = soup.find_all("ul", {"class":"post-list"})
28  for comments in listComments:
29      print(comments)
```

Figure 8. Extractor of the Newspaper Comment Thread (fragment)

JSON generated by the application has the same fields as JSON extracted on YouTube. Figure 9 shows the JSON result of the HTML extraction.



```
{
  "id" : "post-3420370829",
  "user" : "Ultramar",
  "date" : "segunda-feira, 17 de julho de 2017 09:51",
  "timestamp": "",
  "commentText": "Mas o que é que o comum dos mortais tem a ver com esta
coisa!",
  "likes": 4,
  "hasReplies": false,
  "numberOfReplies": 0
},
```

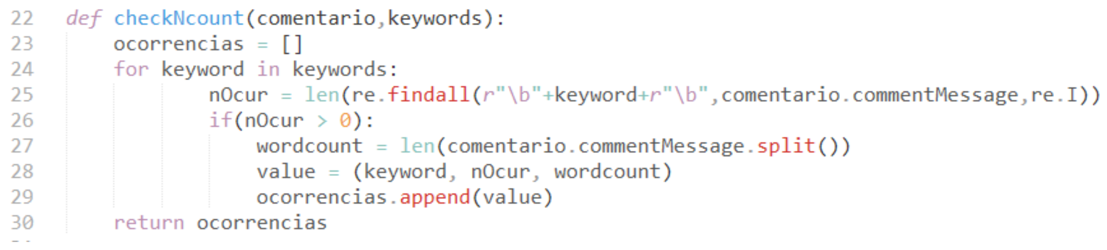
Figure 9. Newspaper Comment Thread in JSON format (fragment of a comment)

It is important to point out that this whole extraction process (newspaper article extractor, comment thread extractor and text filter and JSON converter) is specific to the newspaper “Sol”. This means that for each newspaper it is necessary to create new extractors and a new text filter and JSON Converter, since each newspaper has a different web page.

In Section 6 the analysis of comments to identify discriminatory and prejudice discourse will be discussed.

6. PREJUDICE DISCOURSE DETECTION AND ANALYSIS

It is important to remember that the final objective of the extraction process previously presented is to collect a stream of comment threads to build a comparable linguistic corpus that will be studied mainly in terms of verbal interaction in CMC. After extracting the comments, it is necessary to analyze if the comment thread exhibits a prejudiced and discriminatory stance. To carry out this analysis, we have constructed a Python script that does the statistical analysis for the occurrence of keywords in the comments. This analysis performed by the Python script compares the comment text with the keywords table, which is in JSON format. The code fragment shown in Figure 10 aims to search for the keywords (presented in Section 2) in the comments. When it finds them, it creates a triple analysis with the keyword, the number of occurrences of the keyword and the total number of words that the comment has.

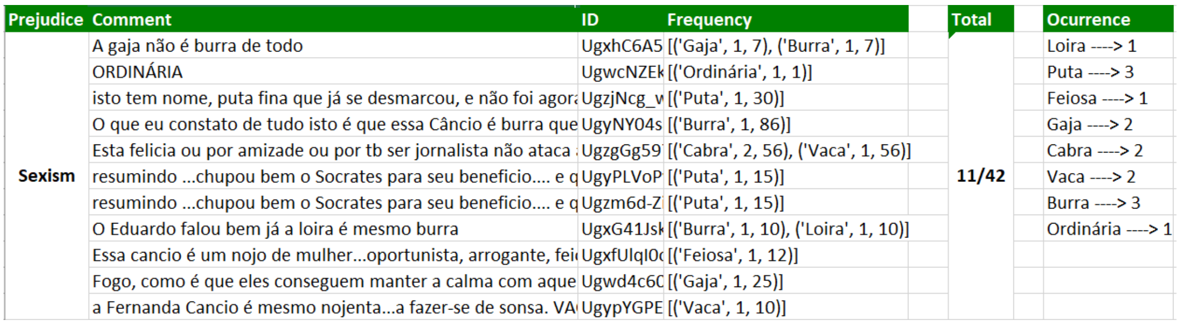


```python
def checkNcount(comentario,keywords):
    ocorrencias = []
    for keyword in keywords:
            nOcur = len(re.findall(r"\b"+keyword+r"\b",comentario.commentMessage,re.I))
            if(nOcur > 0):
                wordcount = len(comentario.commentMessage.split())
                value = (keyword, nOcur, wordcount)
                ocorrencias.append(value)
    return ocorrencias
```

Figure 10. Statistical analysis for the occurrence of keywords in the comments (fragment)

This script in Python does the statistical analysis by: type of prejudice in a comment thread; frequency with which this concept appears in a particular comment; total number of comments that have keywords out of the total number of comments of the thread; which keywords are present in the comments and how often they appear in the comment thread; which type of prejudice is present in the comment thread; which is the most prevalent prejudice in a comment thread.

The analysis that is performed for a file with a comment thread can also be performed for a directory that contains several files with comment threads. The output of the statistical analysis for the occurrence of keywords in the comments is exported to a CSV file. An example of this result is shown in Figure 11. This example corresponds to the JSON sample of YouTube extraction in Figure 3.



Prejudice	Comment	ID	Frequency	Total	Ocurrence
	A gaja não é burra de todo	UgxhC6A5	[('Gaja', 1, 7), ('Burra', 1, 7)]		Loira ----> 1
	ORDINÁRIA	UgwcNZEk	[('Ordinária', 1, 1)]		Puta ----> 3
	isto tem nome, puta fina que já se desmarcou, e não foi agora	UgzjNcg_v	[('Puta', 1, 30)]		Feiosa ----> 1
	O que eu constato de tudo isto é que essa Câncio é burra que	UgyNY04s	[('Burra', 1, 86)]		Gaja ----> 2
	Esta felicia ou por amizade ou por tb ser jornalista não ataca	UgzgGg59	[('Cabra', 2, 56), ('Vaca', 1, 56)]		Cabra ----> 2
Sexism	resumindo ...chupou bem o Socrates para seu beneficio.... e q	UgyPLVoP	[('Puta', 1, 15)]	**11/42**	Vaca ----> 2
	resumindo ...chupou bem o Socrates para seu beneficio.... e q	Ugzm6d-Z	[('Puta', 1, 15)]		Burra ----> 3
	O Eduardo falou bem já a loira é mesmo burra	UgxG41Jsk	[('Burra', 1, 10), ('Loira', 1, 10)]		Ordinária ----> 1
	Essa cancio é um nojo de mulher...oportunista, arrogante, fei	UgxfUlqI0	[('Feiosa', 1, 12)]		
	Fogo, como é que eles conseguem manter a calma com aque	Ugwd4c6C	[('Gaja', 1, 25)]		
	a Fernanda Cancio é mesmo nojenta...a fazer-se de sonsa. VA	UgypYGPE	[('Vaca', 1, 10)]		

Figure 11. Result of statistical analysis for the occurrence of keywords in the comments

Finally, we have built a Web interface that integrates: the statistical analysis module; another module to fill in missing metadata fields; another component that allows for the inclusion of new categories of prejudice types and sociolinguistic variables, and for the addition of new keywords in English and Portuguese to each sociolinguistic variable.

Additionally, after performing the statistical analysis and populating the missing metadata, it still stores the resulting Json file in the database.

Section 7 presents our conclusions and future work.

7. CONCLUSION

With the globalized rise in online communication, people worldwide are spending increasing amounts of time, attention, and energy on social networks (Facebook, YouTube, Twitter, Reddit, etc.). They navigate these platforms on a daily basis, and use them to socialize, connect, express opinions and, crucially, absorb opinions. Dialogue exchange with virtual strangers allows Internet surfers to discuss and argue about a variety of topics, some of which are often openly polemical, without ever coming face to face with one another. The anonymity and distance that characterize this interaction are partly to blame for a generalized willingness to offend and a rising propensity to discriminate, persecute and express prejudice.

Our goal is to extract these interactions (comment threads) from various sources and analyze them. To examine the different types of discriminatory discourse we have created a table with keywords, categorized by type of prejudice and the respective sociolinguistic variables. To collect the dialogue texts and extract the comment threads from Facebook and YouTube, as well as to extract the comment threads from the newspaper page and the newspaper article, we are using two existing tools and creating two scripts. As the output format is not always the same, it is necessary to create scripts that convert the extraction outputs to JSON. Finally, as we intend to analyze the type of discourse present in the comments, we have created a script to perform several analyses, one of which is to identify the prejudiced keywords in the comments and to count how often they appear in each comment and in the thread in general. Our project is aligned with many others in the CMC Community (Cougnon et al., 2019; Pahor de Mati et al., 2019; Franza and Fišer, 2019; Beißwenger et al., 2019a; Beißwenger et al., 2019b; Lombart, 2019), projects conducting various studies related to social media discourse, in particular socially unacceptable discourse. In our case, we aim at integrating multiple sources and analyzing a broader group of social variables.

As future work we intend to implement extractors for other Portuguese and English newspapers. Another goal is to extract Twitter and Instagram comments. We also aim to improve the extraction of YouTube comment threads, that is, to create a script to extract information about publication, similar to the extractor of the newspaper article.

ACKNOWLEDGEMENT

This work has been supported by FCT - Fundação para a Ciência e Tecnologia within the Project Scope: PTDC/LLT-LIN/29304/2017. We are also indebted to our students, André Salgueiro, Bruno Carvalho and Fábio Araújo, for their valuable help in developing the statistical analysis and visualization components. We also wish to thank Daria Bębeniec for her careful reading of the previous versions of the paper and her helpful comments and suggestions.

REFERENCES

Beißwenger, M. and Fladrich, M. and Imo, W. and Ziegler, E., 2019a. Collecting and Analyzing a Corpus of WhatsApp Interactions Using the MoCoDa2 Web Interfaces. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 72.

Beißwenger, M. and Herzberg, L. and Lüngen, H. and Wigham, C., 2019b. cmc-core: A basic schema for encoding CMC corpora in TEI. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 73.

Cougnon, L.A. and Coppin, J. and Gutierrez Figueroa, V., 2019. A Mixed Quantitative-Qualitative Approach to Disagreement in Online News Comments on Social Networking Sites. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 31-35.

Coupland, N., 2007. *Style: Language Variation and Identity*. Cambridge University Press.

Franza, J. and Fišer, D., 2019. The lexical inventory of Slovene socially unacceptable discourse on Facebook. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 42-46.

Jarmul, K. and Lawson, R., 2017. *Python Web Scraping*. Packt Publishing.

Klostermann, P., 2015. Youtube comment scraper. http://ytcomments.klostermann.ca/. (Accessed in: 2019-01-20).

Labov, W., 2001. *Principles of Linguistic Change: Social Factors*, Vol. 2. Blackwell.

Lambert Graham, S., 2007. Disagreeing to agree: Conflict, (im)politeness and identity in a computer-mediated community, *Journal of Pragmatics*, Vol. 39, No. 4, pp. 742 – 759. Special Issue: Identity Perspectives on Face and (Im)Politeness.

Lombart, E., 2019. Text-based messages environment types. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 75.

Mitchell, R., 2015. *Web Scraping with Python: Collecting Data from the Modern Web*. O'Reilly Media.

Pahor de Maiti, K. and Fišer, D. and Ljubešić, N., 2019. How haters write: analysis of nonstandard language in online hate speech. *Proceedings of the 7th Conference on CMC and Social Media Corpora for the Humanities (CMC-Corpora2019).* Cergy-Pontoise, France, Eds. Longhi, J. and Marinica, C., pp. 36-41.

Prutsachainimmit, K. and Nadee, W., 2018. Towards data extraction of dynamic content from javascript web applications, *International Conference on Information Networking (ICOIN)*, pp. 750–754.

Rieder, B., 2013. Studying facebook via data extraction: The netvizz application. *Proceedings of the 5th Annual ACM Web Science Conference*, WebSci '13. New York, NY, USA, pp. 346–355, ACM.