## Rationale

In society the media and news outlets has a charted role to provide actuate, timely and unbiased information for the general public to stay informed. For this reason, the media needs to have freedom to be politically independent and not sensationalise or doctor facts for a political or business agendas. A common technique sometimes used by various media outlets is to have eye catching headlines that mislead the reader of the truth of the article, colloquialised as “click bait”. This form of news article is reducing the quality of online journalism for larger new outlets such as CNN and VICE due to the skewed or bolded titles usually containing dumbed down version of the article or with no verified information (UWIRE Text, 2018).

A hotly contested statement made about the Australian Broadcasting Corporation (ABC) from the Murdoch media is that it is unfairly biased towards the left political spectrum (Newman ,2019). An editorial in the Australian (owned by News Corp) from a former ABC chairman, claimed that the station had overwhelming support from the leftist political parties such as, GetUp, the Greens and the Labour party (Newman, 2019). Independent news outlets such as Aljazeera News heavily contest ideas stated from New Corp publications stating that the Australian and other Rupert Murdoch news outlets sometimes dismiss scientific fact with false, debunked narratives (Cooke et al., 2020). Therefore, for this assessment current news articles pertaining to the Coronavirus were scraped on the ABC website for future analysis if these articles agreed these allegation stated about the ABC (see **figure 1**). The topic of the coronavirus was selected as it is a current issue within media and it is hypothesised that the allegations made about the ABC are also present in the coverage of the Government’s handling of the pandemic.

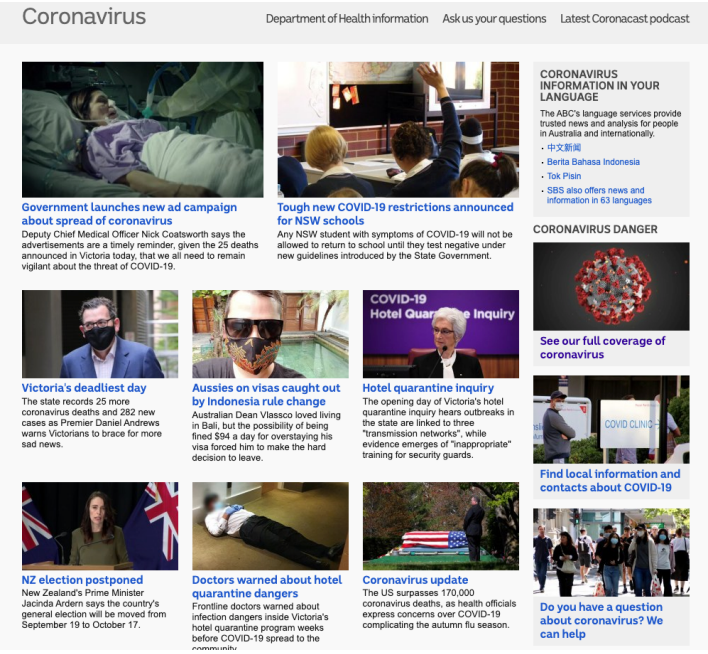


**Figure 1**: Shows a screenshot of the corona virus website.

## Copyright Considerations

For this assessment, the terms of use for the ABC were read and taken into account before text data from the website was scraped. All intellectual property rights in the content, software and systems owned by or licensed to the ABC on any ABC Online Services, including logos, images, names, designs, trademarks and copyright (ABC Content) are reserved to the ABC and its licensors (ABC Terms of Use, 2020). The terms of use also state that content taken from ABC can only be used for personal, non-commercial use (ABC Terms of Use, 2020). This assignment adheres to this as the data will not be used in any commercial setting which would violate the terms and condition. The terms and conditions for the ABC also do not state that web scraping not allowed thus it was interpreted that web scrapping techniques would be allowed as long as data taken was used for personal use only.

## Method

A simple web scraping script was created utilising the Python (3.7.4) packages bs4, urllib, and time (McKinney 2010; Richardson, 2007; Van Rossum & Drake, 2009). These packages were used to access, scrape, aggregate and finally save the data into a csv file for further text analysis.

### Accessing the URL

Initially the page was accessed using the urlopen function from the urllib.request packages which saved the file as an xml format. From there the entire HTML code was parsed into a usable format to access the text using the BeautifulSoup function from the bs4 which saved the entire page in the soup variable as per convention. Each respective article was initially inspected on the webpage html code (see figure 2) which displayed that all article had the same class “doctype-article” and were contained within the <li> html tags. Thus the soup was filtered isolating the articles present on the webpage using the findAll function, articles = soup.findAll("li", class_="doctype-article").

### Aggregating the Text Data

Each article contained within the raw article soup was extracted using a for loop that collected the article title, date, time, link, description and image url. Access of each aspect of article was achieved using , the find() and get() functions. These functions were used to isolate each various aspect of the article soup and saved to their respective variables. As the for loop scraped each respective variable, cleaned variables were printed to the console to inspect the output without having to open the generated csv file.

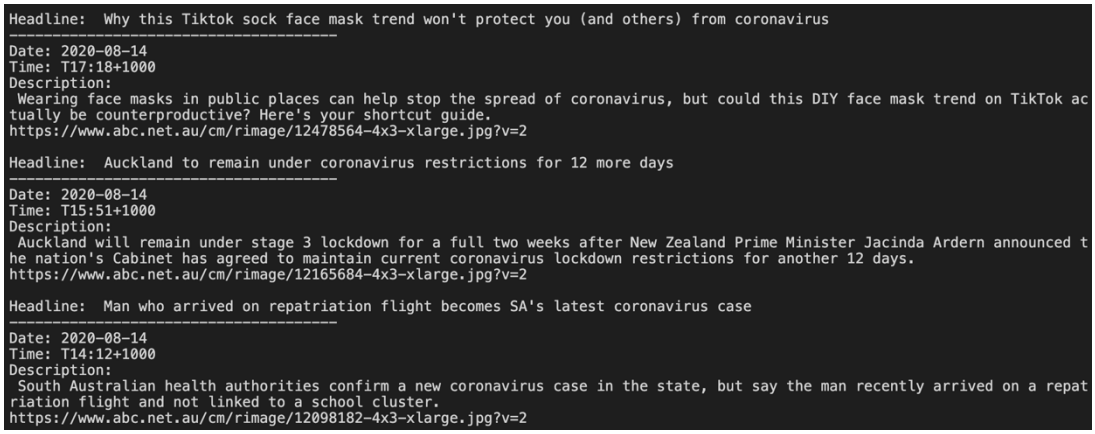


**Figure 2**: Shows a screenshot of the web scraper in action.

Each variable was then added to a python dictionary called “new_row” which was then appended to a data frame (news_items) for further processing. If a certain article on the webpage did not contain one of the respective variable that was scraped, an empty sting was input into the list using multiple if statements.

```
if not description:
    description = ""
if not link:
     link = ""
…
```

To enrich the dataset further, text from each individual article was accessed using concatenated link from the news_items data frame. By accessing the each article displayed on the ABC coronavirus webpage, this increased the amount data in the corpus and thus would improve the accuracy for future analysis.

```
article_text = []
for i in range(len(news_items['link'])):
    individual_page = urlopen(news_items['link'][i])
    individual_page = individual_page.read()
    individual_page_soup = BeautifulSoup(individual_page, 'lxml')
    #print(soup.prettify())
    raw_text = individual_page_soup.findAll("p", class_="_1SzQc")
    cleaned_text = []
    for paragraph in raw_text:
        text = paragraph.get_text().strip()
        cleaned_text.append("".join(text))
    article_text.append(cleaned_text)

news_items['articles'] = article_text
```

The final action of the web scraper saved the aggregated data frame to a csv called news_items.csv. To ensure that previous articles were not deleted every time the coronavirus webpage was scraped, an if-else statement was implemented to see if

the news_item.csv existed in the directory using the os Python package (Van Rossum & Drake, 2009). If the csv file already existed in the working directory, the csv file was read into the script and the newly scraped articles were appended to the end of the file. Measures were taken to ensure this process only added new articles to the csv.

```
if os.path.exists('news_items.csv'):
    news_items_file =  pd.read_csv('news_items.csv')
    news_items_file = news_items_file.drop("Unnamed: 0", axis = 1)
    news_items_file = news_items_file.append(news_items, ignore_index=True)
    new_items_file = news_items_file.reset_index()
    title = news_items_file['title'].drop_duplicates(keep = 'first')
    values  = list(title.index)
    filtered_news_items = news_items_file.loc[values]
    filtered_news_items.to_csv('news_items.csv')
else:
    news_items.to_csv("news_items.csv")
```

## Problems That Occurred and Solution

The most challenging aspect of web scraper was scraping the image url from the article soup. This was because each image had a respective image ID and differing image sizes that were contained within its respective url. To overcome the first hurdle (image ID), the html text for each image was inspected which revealed that each image ID was contained in the element ‘data-image-cmid’. This was accessed using the get() function which pulled the ID and added it to the image_url.

```
image_id1 = article.get('data-image-cmid')
image_url = "https://www.abc.net.au/cm/rimage/{image_id}-{image_size}.jpg?v=2"
```

To account for the varying image sizes, a try and expect statement was used inside a for loop to see if the url would open with various image sizes. If the url did not open with one particular size, the except statement would move on to the next size until the image opened.

```
image_size = ["4x3-xlarge", "16x9-xlarge"]
for size in image_size:
    url = image_url.format(image_id=image_id1, image_size=size)

    try:
        pages = urlopen(url)
        print(url)
        url_ = url
        break
     except:
        continue
```

## Preliminary Findings

No preliminary findings were able to be assessed to confirm or deny the allegations made by the Murdoch media about the ABC. The data scraped for the purposes of this assessment is not enough to complete this investigation accurately, as vast amounts of historical text data is required in order to perform an accurate sentiment analysis on the generated articles. To overcome this, this article was written so to add to the current csv for future automatic web scraping of the ABC.

## References


ABC Terms of Use. (2020). Retrieved from https://about.abc.net.au/terms-of-use/

Cooke, R., Remeikis, A., Tiffen, R., & Painter, J. (2020, January 25). The Murdoch media: Polluting Australia's airwaves? Retrieved from https://www.aljazeera.com/programmes/listeningpost/2020/01/murdoch-media-polluting-australia-airwaves-200125010210055.html

Click-bait articles negatively impact journalism. (2018). UWIRE Text Retrieved from https://go-gale-com.elibrary.jcu.edu.au/ps/i.do?p=STND&u=james_cook&id=GALE%7CA558015293&v=2.1&it=r&sid=summon

McKinney, W., & others. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (Vol. 445, pp. 51–56).

Newman, (2019). The Australian. Forces from within are destroying the ABC. Retrieved from https://www.theaustralian.com.au/remote/check_cookie.html?url=https%3a%2f%2fwww.theaustralian.com.au%2fcommentary%2fforces-from-within-are-destroying-the-abc%2fnews-story%2f5366e9180672ab1bb16e5f65c9e10eb9

Richardson, L. (2007). Beautiful soup documentation. April.

Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.