Task 2: Web Crawler to scrape TripAdvisor using Selenium (1500 words) - 30% of overall 2062 w

2.1 Domains: (~325 words) 15% 640w

2.1.1 Website URLs to be crawled

TripAdvisor is one of the most influential websites for customers when booking their holiday accommodation. The website reaches 390 million unique visitors each month and lists 465 million reviews and opinions about more than 7 million accommodations, restaurants, and attractions in 49 markets worldwide (Valdivia, Luzon, & Herrera, 2017). Because TripAdvisor has large amounts of data, it has become extremely popular with tourists and hotel managers. Tourists can read the accumulated opinions of everyday tourists. They can also check the popularity index, computed using an algorithm for customer reviews and other published sources such as guidebooks and paper articles.

TripAdvisor has its official hotel description and bubble score rating. However, the website also offers thousands of hotel reviews from previous customers, coupled with the customers own bubble score rating. The bubble score rating is on a scale from 1 to 5, where one bubble represents a terrible experience, and five bubbles represent an excellent experience (Srivishnumohan, 2020). Online customer opinion has become extremely valuable to travellers, but understanding the true sentiment of online user reviews has posed a daunting task given the complexities of the human language.

TripAdvisor has enough standing to be used as a text source, storing numerous reviews of tourist businesses worldwide.

2.1.2 Coverage of the chosen domains

To address the issue of 'how can hotels improve their customer experience', the hotel needs to understand the critical topics and aspects of customer reviews. For example, the important topics when considering accommodation may include features like location and what facilities the accommodation has. Essential aspects of customer reviews may include the friendliness of hotel staff and the cleanliness of the rooms.

This investigation decided to extract the following information from the TripAdvisor website:

- URLs for all the hotels in Townsville.
- Name of the hotels.
- TripAdvisor description of the hotels.
- Customer reviews for the hotels.
- Customer rating for the hotels.
- Date of stay for each user who provided a review.

2.1.3 The natural language data and how these data align to the issue

The natural language data extracted from the TripAdvisor website was:

- Hotel descriptions - The hotel description was the official hotel description provided by TripAdvisor. Not all hotels had a description provided on TripAdvisor.
- Customer review - The customer who had stayed at the accommodation provided the review. The reviews were full of sentiment, both positive and negative. Not all customers who stayed at the hotel provided a review, and some hotels have more reviews than others.

The hotel descriptions and customer reviews give a mix of commercial and personal language. The scraped natural language data will be explored to understand what attributes relate to excellent accommodation in Townsville by providing two different viewpoints of the accommodation in TripAdvisor and the many customer reviews.

2.1.4 Copyright of chosen domain

In copyright law, 'fair use' permits third parties to use copyright-protected works only in a limited way. The more analytics performed on the scraped data, the more the material tends towards fair use. The more 'excerpting' performed on scraped data, the more likely one is liable for copyright infringement. An example of this is where one would not change anything significant about the scraped material and repost snippets of the data as-is. Creating classifications and fields of information will generally mean playing around with scraped data. Organising it differently from what is received from a robot will push the use of information towards the "transformative ", allowing you to create a stronger claim of "fair use" of the information under copyright law.

TripAdvisor content is protected by copyright, but as this is an academic exercise and this report intends to analyse the material, scraping review data from TripAdvisor does not violate copyright laws.

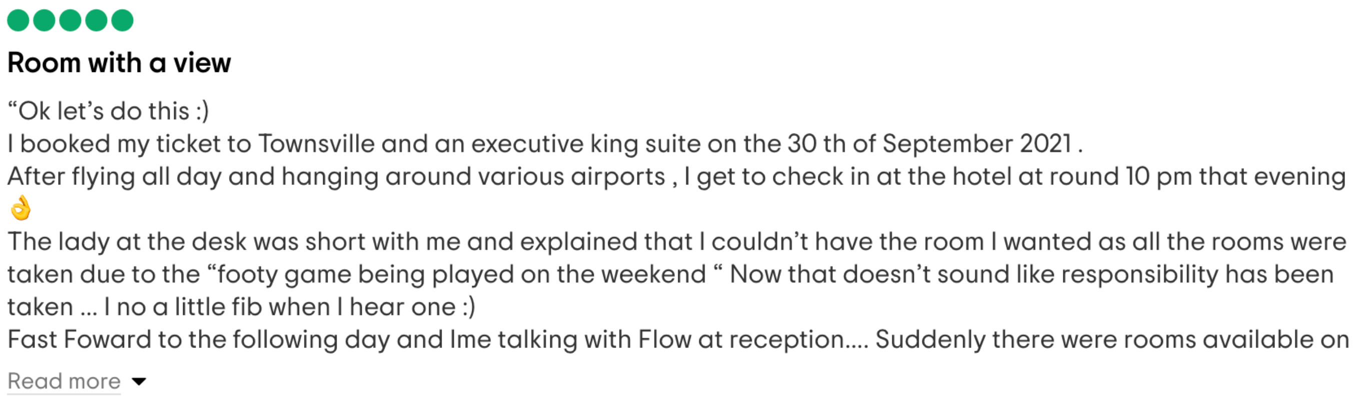
2.2 WebCrawler workflow: (~750 words) 50% 829w

2.2.1 Technology components used for the web crawler

The TripAdvisor website uses lazy loading scripts that edit the HTML document object model (DOM) and break the element refs. Writing a consistent web-crawler program can make it challenging to move through all the URLs and then scrape the required data from each site. When writing this report, a web-crawler application was successfully developed using the **Selenium** package to scrape the required data for this investigation using WAIT_FOR methods. Selenium works by automating web browser activity and replicating functions such as opening a browser and clicking on links. This automated activity mimicked the movements of an actual human website user and was not identified by bot detection. The package requires installing a driver to interface with a browser (Chrome was used in this case). Our program was able to scrape over 18,000 customer reviews, which suggests TripAdvisor may not have bot detection software.

2.2.2 Complexity of the domains where the targeted data resides

Scraping challenges – As highlighted in the screenshot below, TripAdvisor only loads part of the review initially and waits until the user clicks 'Read more' to load the rest.

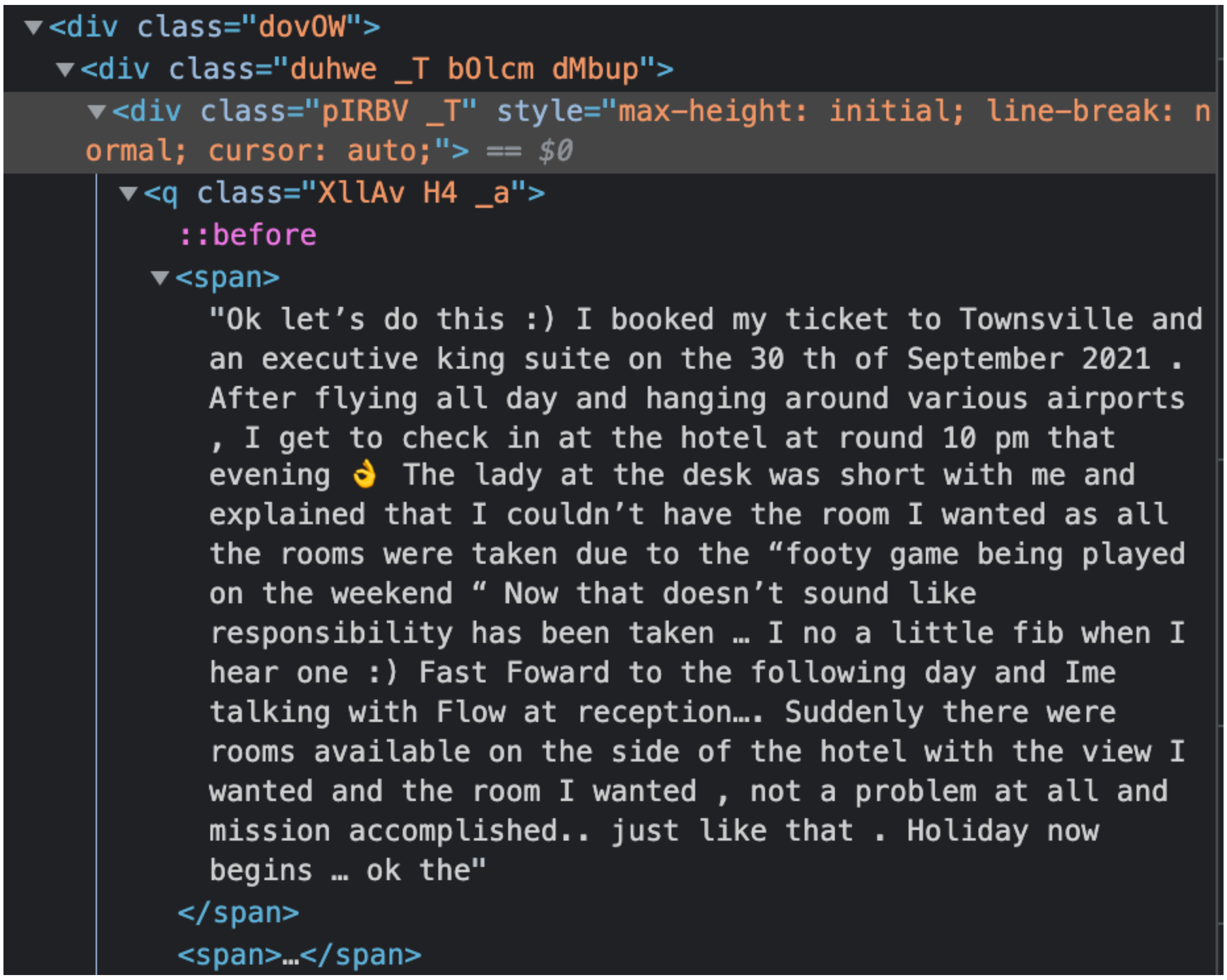


●●●●●

Room with a view

"Ok let's do this :)
I booked my ticket to Townsville and an executive king suite on the 30 th of September 2021 .
After flying all day and hanging around various airports , I get to check in at the hotel at round 10 pm that evening 👌
The lady at the desk was short with me and explained that I couldn't have the room I wanted as all the rooms were taken due to the "footy game being played on the weekend " Now that doesn't sound like responsibility has been taken ... I no a little fib when I hear one :)
Fast Foward to the following day and Ime talking with Flow at reception.... Suddenly there were rooms available on

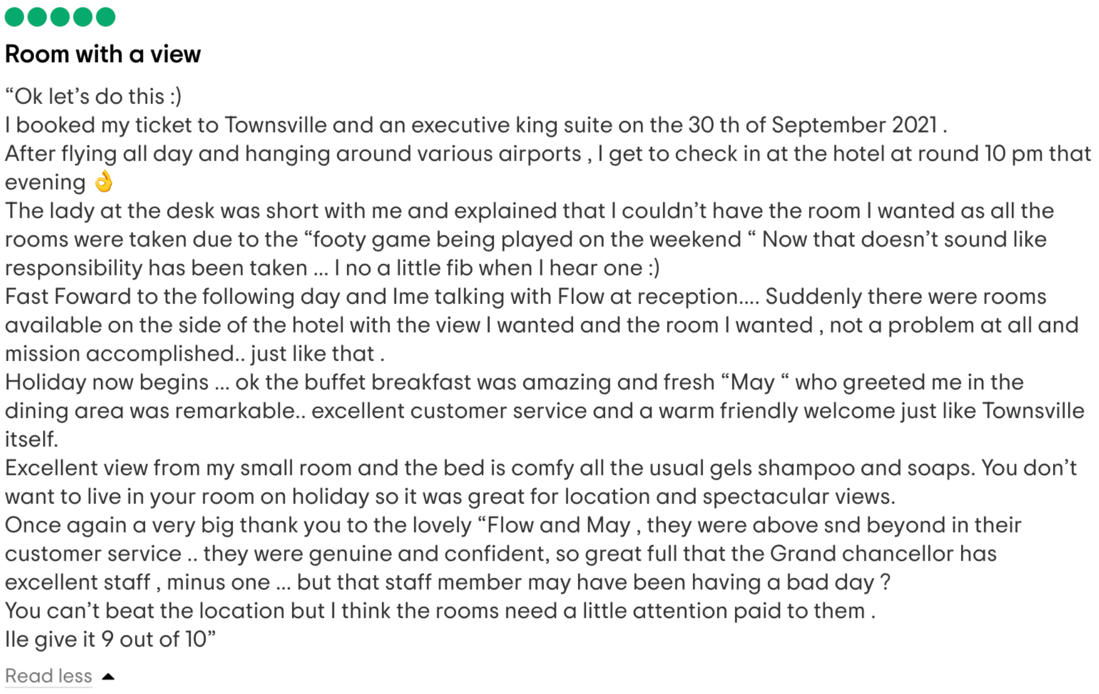
The following is a screenshot of the inspect element tool before clicking the 'Read more' button.



```
▼<div class="dovOW">
  ▼<div class="duhwe _T bOlcm dMbup">
    ▼<div class="pIRBV _T" style="max-height: initial; line-break: n
    ormal; cursor: auto;"> == $0
      ▼<q class="XllAv H4 _a">
          ::before
        ▼<span>
            "Ok let’s do this :) I booked my ticket to Townsville and
            an executive king suite on the 30 th of September 2021 .
            After flying all day and hanging around various airports
            , I get to check in at the hotel at round 10 pm that
            evening 👌 The lady at the desk was short with me and
            explained that I couldn’t have the room I wanted as all
            the rooms were taken due to the “footy game being played
            on the weekend “ Now that doesn’t sound like
            responsibility has been taken … I no a little fib when I
            hear one :) Fast Foward to the following day and Ime
            talking with Flow at reception…. Suddenly there were
            rooms available on the side of the hotel with the view I
            wanted and the room I wanted , not a problem at all and
            mission accomplished.. just like that . Holiday now
            begins … ok the"
          </span>
          <span>…</span>
```

Fig 1. Data table dictionary.

Below is the same review once the ‘Read more’ button is clicked.



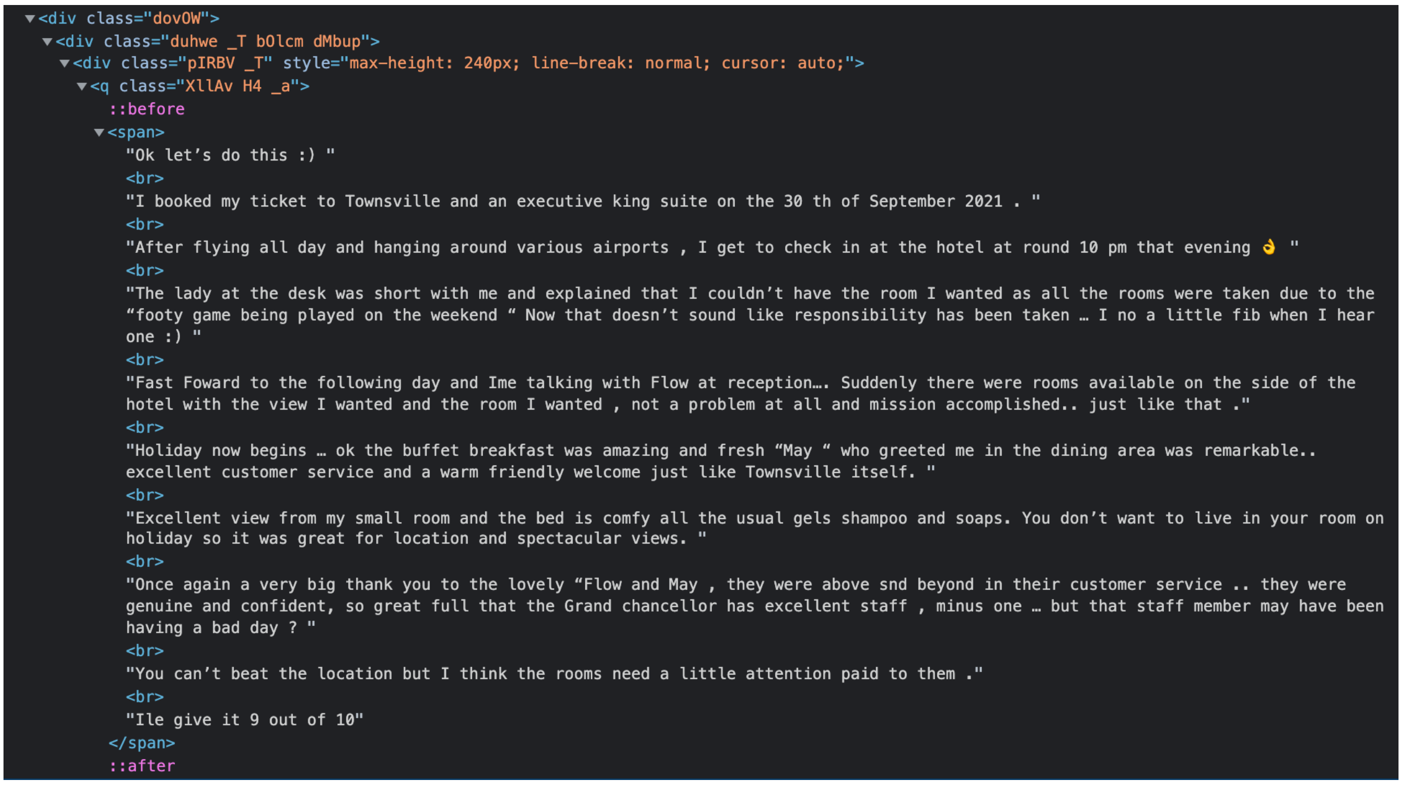
●●●●●

Room with a view

“Ok let’s do this :)
I booked my ticket to Townsville and an executive king suite on the 30 th of September 2021 .
After flying all day and hanging around various airports , I get to check in at the hotel at round 10 pm that evening 👌
The lady at the desk was short with me and explained that I couldn’t have the room I wanted as all the rooms were taken due to the “footy game being played on the weekend “ Now that doesn’t sound like responsibility has been taken … I no a little fib when I hear one :)
Fast Foward to the following day and Ime talking with Flow at reception…. Suddenly there were rooms available on the side of the hotel with the view I wanted and the room I wanted , not a problem at all and mission accomplished.. just like that .
Holiday now begins … ok the buffet breakfast was amazing and fresh “May “ who greeted me in the dining area was remarkable.. excellent customer service and a warm friendly welcome just like Townsville itself.
Excellent view from my small room and the bed is comfy all the usual gels shampoo and soaps. You don’t want to live in your room on holiday so it was great for location and spectacular views.
Once again a very big thank you to the lovely “Flow and May , they were above snd beyond in their customer service .. they were genuine and confident, so great full that the Grand chancellor has excellent staff , minus one … but that staff member may have been having a bad day ?
You can’t beat the location but I think the rooms need a little attention paid to them .
Ile give it 9 out of 10”

Read less ▲

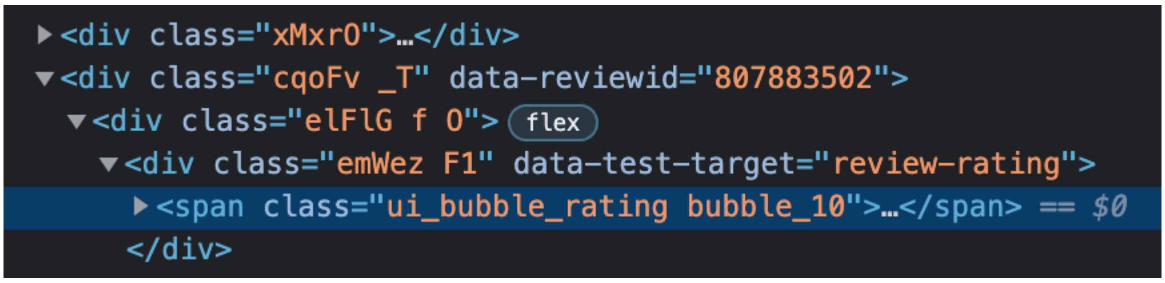
Below is a screenshot of the inspect element after clicking the 'Read more' button.



```
▼<div class="dovOW">
  ▼<div class="duhwe _T bOlcm dMbup">
    ▼<div class="pIRBV _T" style="max-height: 240px; line-break: normal; cursor: auto;">
      ▼<q class="XllAv H4 _a">
          ::before
        ▼<span>
            "Ok let’s do this :) "
            <br>
            "I booked my ticket to Townsville and an executive king suite on the 30 th of September 2021 . "
            <br>
            "After flying all day and hanging around various airports , I get to check in at the hotel at round 10 pm that evening 👌 "
            <br>
            "The lady at the desk was short with me and explained that I couldn’t have the room I wanted as all the rooms were taken due to the
            “footy game being played on the weekend “ Now that doesn’t sound like responsibility has been taken … I no a little fib when I hear
            one :) "
            <br>
            "Fast Foward to the following day and Ime talking with Flow at reception…. Suddenly there were rooms available on the side of the
            hotel with the view I wanted and the room I wanted , not a problem at all and mission accomplished.. just like that ."
            <br>
            "Holiday now begins … ok the buffet breakfast was amazing and fresh “May “ who greeted me in the dining area was remarkable..
            excellent customer service and a warm friendly welcome just like Townsville itself. "
            <br>
            "Excellent view from my small room and the bed is comfy all the usual gels shampoo and soaps. You don’t want to live in your room on
            holiday so it was great for location and spectacular views. "
            <br>
            "Once again a very big thank you to the lovely “Flow and May , they were above snd beyond in their customer service .. they were
            genuine and confident, so great full that the Grand chancellor has excellent staff , minus one … but that staff member may have been
            having a bad day ? "
            <br>
            "You can’t beat the location but I think the rooms need a little attention paid to them ."
            <br>
            "Ile give it 9 out of 10"
          </span>
          ::after
```

Once clicked, further Javascript is executed, making it challenging to scrape the whole review. **Selenium** package launched the browser and simulated pressing the 'Read more' link, which executed the JavaScript to reveal the complete review for scraping. It then needed to be waited for until the data populated successfully.

Another example of the TripAdvisor website's complexity is the user's graphical score as part of their review. The image is made up of five circles, some of which are filled in green. The number of green circles indicates a score out of 5. There is no textual indication of the score, making it difficult to scrape. This study identified that the CSS class of the SPAN element holds an indication of the score.



```
▶<div class="xMxrO">…</div>
▼<div class="cqoFv _T" data-reviewid="807883502">
  ▼<div class="elFlG f O"> flex
    ▼<div class="emWez F1" data-test-target="review-rating">
      ▶<span class="ui_bubble_rating bubble_10">…</span> == $0
      </div>
```

By performing a string manipulation on the class name and retaining a substring of the last two index positions, the code extracted the review rating.

2.2.3 Sequencing of the crawler, using the complexity, data structures and website access restrictions to optimise the crawler

This report will analyse language data from user reviews of hotels in Townsville using **Python version 3.9.0**. Three web-crawler programs were used to extract the required data from TripAdvisor. The sequencing of the web-crawler programs is as follows:

- *'urlCrawler'* – Takes the initial query of 'https://www.tripadvisor.com.au/Hotels-g255073-Townsville_Queensland-Hotels.html'. Crawls through every Townsville hotel on TripAdvisor and collect the URLs. Save the file as hotelUrls.csv
- *'hotelInfoScraper'* - Use the list of URLs from hotelUrls.csv to crawl through all web pages to scrape the TripAdvisor hotel descriptions and hotel names. Save the file as hotelData.csv.
- *'hotelReviewScraper'* - Perform a third web crawl using the list of URLs to extract the customer reviews for each hotel, the bubble score rating for each review and the date of the stay for each review. Save the file as hotelReviews.csv.

The following code (please refer to the appendix for full code) was used to crawl through all web pages and collect the URLs associated with hotels in Townsville on TripAdvisor:

```python
defaultTimeout = 30
defaultRetryWaitTime = 0.5

url = 'https://www.tripadvisor.com.au/Hotels-g255073-Townsville_Queensland-Hotels.html'

#Creates a DataFrame for urls we can save later
hotels = pd.DataFrame(columns = ['url'])
hotels.index.name = "hotelId"

#stores the nextPage to parse
nextPage = url

currentPageNumber = 1

getPageRetryMax = 5
currentTry = 1

try:
    while nextPage != "":

        if currentTry > getPageRetryMax:
            raise Exception("Failed too many times")

        print(f"Downloading current page: {currentPageNumber}")
        driver.get(nextPage)

        print("Adding hotel URLs")
        hotelsNew = None
        try:
            hotelsNew = getHotels(driver)
        except Exception:
            currentTry = currentTry + 1
            continue

        print("Getting next page link")
        try:
            nextPage = getNextPage(driver, currentPageNumber)
        except Exception:
            currentTry = currentTry + 1
            continue

        for hotelNew in hotelsNew:
            hotels.loc[len(hotels)] = [hotelNew]

        currentPageNumber = currentPageNumber + 1
except Exception:
    print("Failed too many times")
finally:
    print("Saving data to disk")
    hotels.to_csv("hotelUrls.csv", index = True, header = True)
    driver.quit()

print("Program finished")
```

Elements were identified and extracted using the HTML inspect navigation panel corresponding to the page in the chrome browser. The following code (please refer to the appendix for full code) was used to scrape the TripAdvisor hotel description and hotel name from each URL collected in the previous code:

```
defaultTimeout = 30

hotelData = pd.read_csv('hotelUrls.csv')

totalHotelsToScrape = len(hotelData)
print(f"Preparing to scrape {totalHotelsToScrape} hotels")

hotelNames = []
hotelDescriptions = []

getPageRetryMax = 5
currentTry = 1

for index, row in hotelData.iterrows():
    currentTry = 1
    while True:

        if(currentTry > getPageRetryMax):
            print(f"Failed to retrieve information from hotel at index {index}, skipping")
            hotelNames.append("")
            hotelDescriptions.append("")
            break

        try:
            #There would be a more excellent way to track this index, but this works
            currentHotelIndex = len(hotelNames)

            currentPageUrl = row['url']
            driver.get(currentPageUrl)

            #Get hotel name from page, it has an ID
            hotelNameElement = WebDriverWait(driver, defaultTimeout).until(
                EC.presence_of_element_located((By.ID, "HEADING"))
            )

            hotelName = hotelNameElement.text

            aboutElement = WebDriverWait(driver, defaultTimeout).until(
                EC.presence_of_element_located((By.ID, "ABOUT_TAB"))
            )

            descriptionElement = WebDriverWait(aboutElement, defaultTimeout).until(
                EC.presence_of_element_located((By.CSS_SELECTOR, "div[class='pIRBV _T']"))
            )

            descriptionText = descriptionElement.text.replace('\n', " ")

            #If the name is blank/small just put a blank string in the array to keep our
column aligned with hotelData
            if(len(hotelName) < 5):
                hotelNames.append("")
            else:
                hotelNames.append(hotelName)

            #If the description is blank/small just put a blank string in the array to
keep our column aligned with hotelData
            if(len(descriptionText) < 5):
                hotelDescriptions.append("")
            else:
                hotelDescriptions.append(descriptionText)

            print(f"Scraped hotel {currentHotelIndex + 1} of {totalHotelsToScrape},
hotelName: {hotelName}, len(descriptionText): {len(descriptionText)}")
            break
        except Exception:
            print(f"Failed to get data from page at index {index}, retrying...")
            currentTry = currentTry + 1
            continue
```

```
hotelData['hotelNames'] = hotelNames
hotelData['hotelDescriptions'] = hotelDescriptions

hotelData.to_csv("hotelData.csv", index = False, header = True)
driver.quit()
```

The hotel descriptions were saved to a csv file, ‘hotelData.csv’. The following code (please refer to appendix for full code) was used to collect the customer reviews for every hotel, the customer rating, and the date of their stay by crawling through the pages by URL:

```
# hotelId, reviewId, stars, date, review
hotelData = pd.read_csv('hotelUrls.csv')
hotelFlag = [0] * len(hotelData)
hotelData['hotelFlag'] = hotelFlag

hotelReviews = pd.DataFrame(columns = ['hotelId', 'reviewId', 'stars', 'date', 'review'])
pageNumber = 1

while anyHotelsToCheck():
    print(f"Scraping each hotel page for reviews, looking at their page number 
{pageNumber}")
    getReviewsWithPageNumber(pageNumber)
    print(f"Finished scraping each hotel's page {pageNumber} for reviews, currently we 
have {len(hotelReviews)} reviews")
    pageNumber = pageNumber + 1

print("No more hotel reviews to scan")

driver.quit()
```

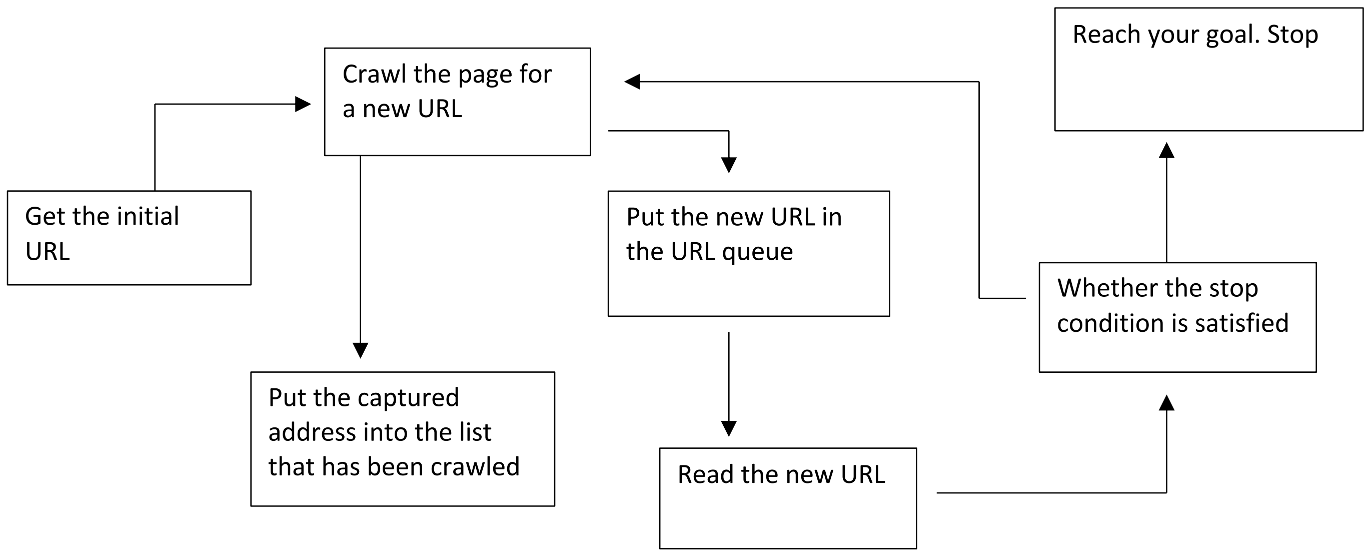
A flag system was used to deal with how the ‘*hotelReviewScraper*’ program would only look for the appropriate number of pages for customer reviews. The program detected if the hotel had no more reviews to scrape if the current page was ‘no more reviews’ or the 'next button was not clickable. Fault-tolerant coding practices were emphasised to ensure the code was able to run successfully on the first attempt and prevent being blocked by bot detection. An example of this is the significant retry logic applied to code based on exception throwing. Waiting loops were used to allow for the lazy loading scripts, as mentioned in part *2.2.1*. Also, print statements were added to the loops to understand the efficiency of the code and potentially minimise bot detection with early termination if a bug was encountered.

2.2.4 Data Storage

All output from the three web crawler programs were saved as a comma-separated values file:

- hotelUrls.csv
- hotelData.csv
- hotelReviews.csv

Demonstrate: Robust, fault tolerance coding practices / Illustration


Get the initial URL
Crawl the page for a new URL
Put the captured address into the list that has been crawled
Put the new URL in the URL queue
Read the new URL
Whether the stop condition is satisfied
Reach your goal. Stop

2.3 Harvested Corpus EDA: (~325 words) 35% 593w

2.3.1 Data Wrangling

Hotels that were missing the description were removed from the 'hotelData' data using the following code:

```
# Remove rows with missing descriptions
dfHotelData = dfHotelData.dropna( how='any', subset=['hotelDescriptions'])
```

In the 'hotel reviews data, the review rating was changed to 1–5 (instead of 10-50) to make the data easier to interpret.

Most of the data wrangling was applied in the web crawler scripts, which left little wrangling to perform following the corpus harvest.

2.3.2 Summary of the generated corpus

The hotelData.csv file is comprised of four columns: ‘hotelID’, ‘URL, 'hotelNames’ and ‘hotelDescriptions’ (Table 1). The data has 115 observations with 25 rows missing the hotel name and description.

Field ID	**Description**	**Totals**
hotelId	ID for each hotel in Townsville.	115
URL	URL of the hotel listed on TripAdvisor.	115
hotelNames	The name of the hotel.	90
hotelDescriptions	The TripAdvisor description of the hotel.	90

Table 1. Data table dictionary for hotelData.csv.

The hotelReviews.csv file is comprised of five columns: ‘hotelId’, ‘reviewId’, ‘stars’, ‘date’ and ‘hotelReviews’ (Table 2). The data has 18242 observations with no missing data.

Field ID	**Description**	**Totals**
hotelId	ID for each hotel in Townsville.	18242
reviewId	ID for each review assigned to a hotel.	18242
stars	The user rating from each review.	18242
date	Date of stay from each user review.	18242
reviews	The review posted from the user review to TripAdvisor describes their stay at the hotel.	18242

Table 2. Data table dictionary for hotelReviews.csv.

2.3.3 Visualisation of the corpus

To understand the harvest corpus of both the hotelData.csv and hotelReviews.csv initial data exploration and analysis was performed.

2.3.3.1 Visual representation of 'hotelData.csv' data.

A bar plot was created to represent the number of words in each hotel description. Most hotel descriptions were between 50 – 100 words (Fig. 1).

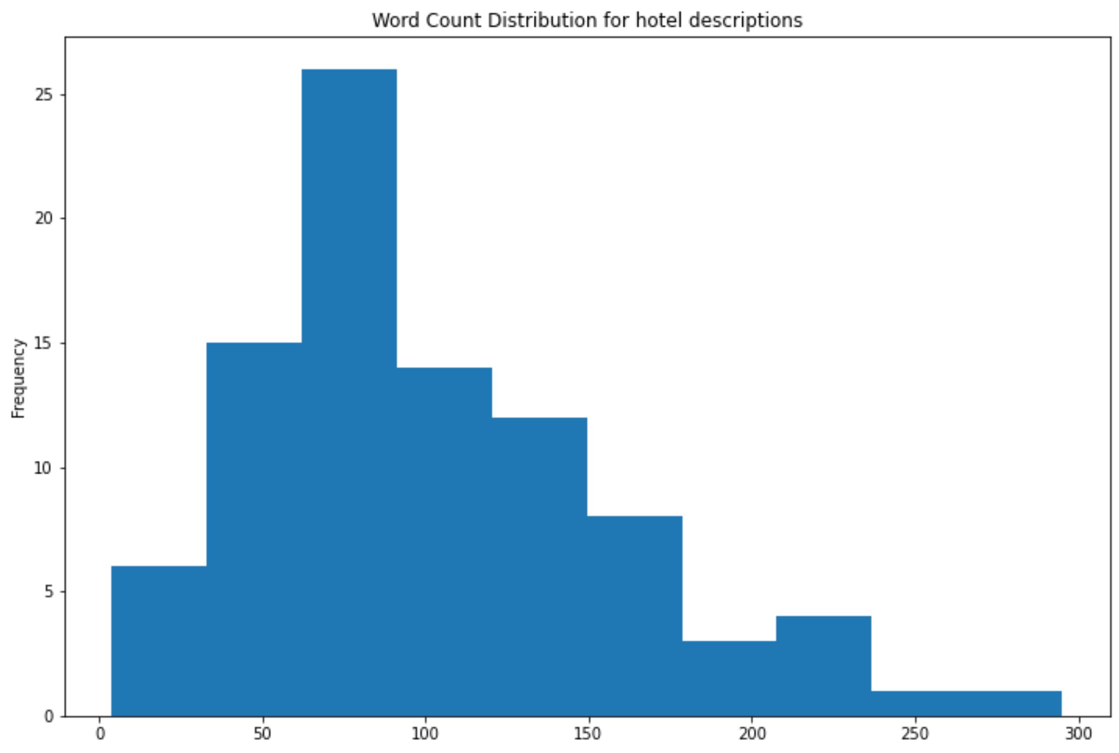


Fig 1. Word count distribution for hotel description from hotelData.csv.

A bigram was created to understand the frequency of two-word sequences in the hotel descriptions. The hotel descriptions' three most frequent two-word sequences were 'self contained', 'air conditioning, and 'The strand' (Fig. 2).

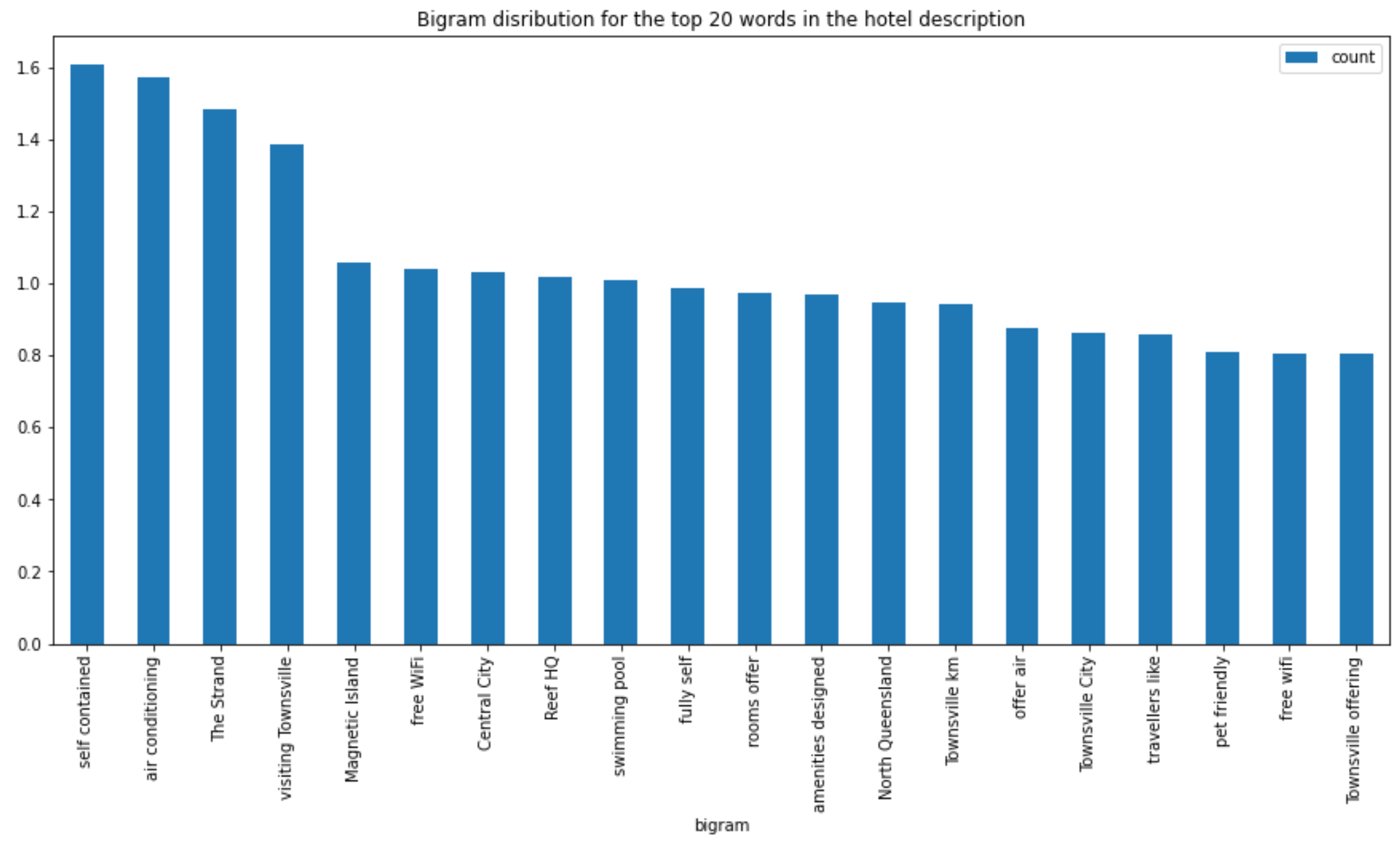


Fig 2. Bigram distribution for hotel description from hotelData.csv.

A trigram was created to understand the frequency of three-word sequences in the hotel descriptions. The hotel descriptions' most frequent three-word sequences were 'motel hall hire', 'fully self-contained, and ‘offer air conditioning’ (Fig. 3).

After exploring the bigram and trigram of hotel descriptions, the study identified that the hotel most talked about the accommodation being self-contained and having air conditioning. The descriptions also frequently mentioned 'The strand', a seaside foreshore in Townsville that has views of the Port of Townsville and Magnetic Island. This bigram speaks to the location of the accommodation.

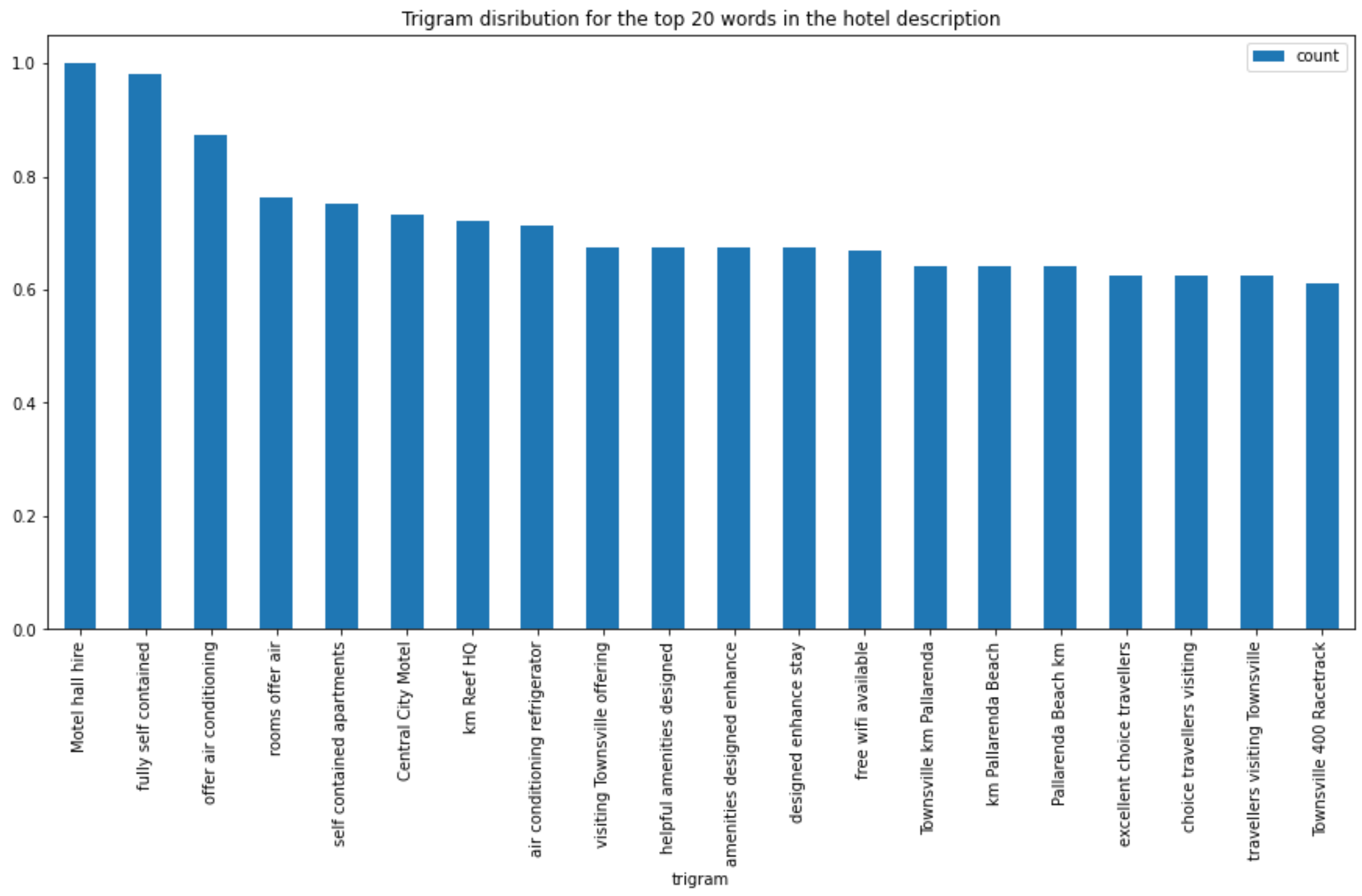


Fig 3. Trigram distribution for hotel description from hotelData.csv.

A word cloud was created to visualise the frequent individual words in the hotel description (Fig. 4). The prominent words in the word cloud are 'located', 'Townsville', 'apartment' and 'beachfront'.

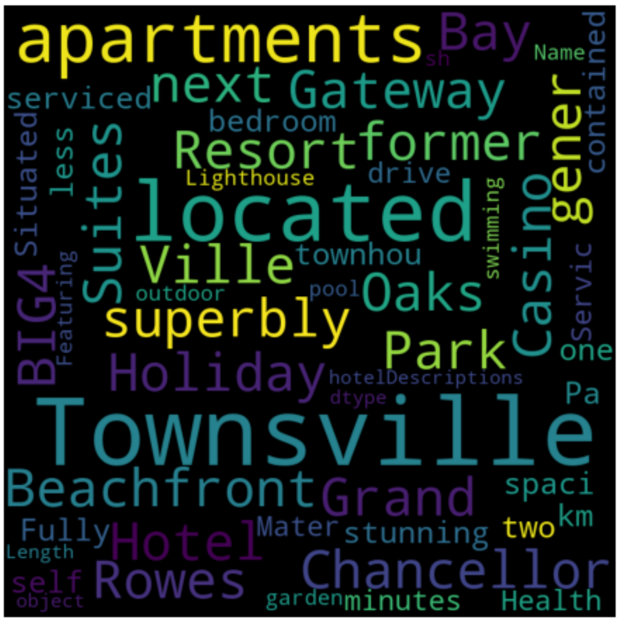


Fig 4. Word cloud for hotel description from hotelData.csv.

2.3.3.2 Visual representation of ‘hotelReviews.csv’ data.

A bar plot was created to represent the total word count for hotel reviews. Most of the reviews contained 50-100 words (Fig. 5).

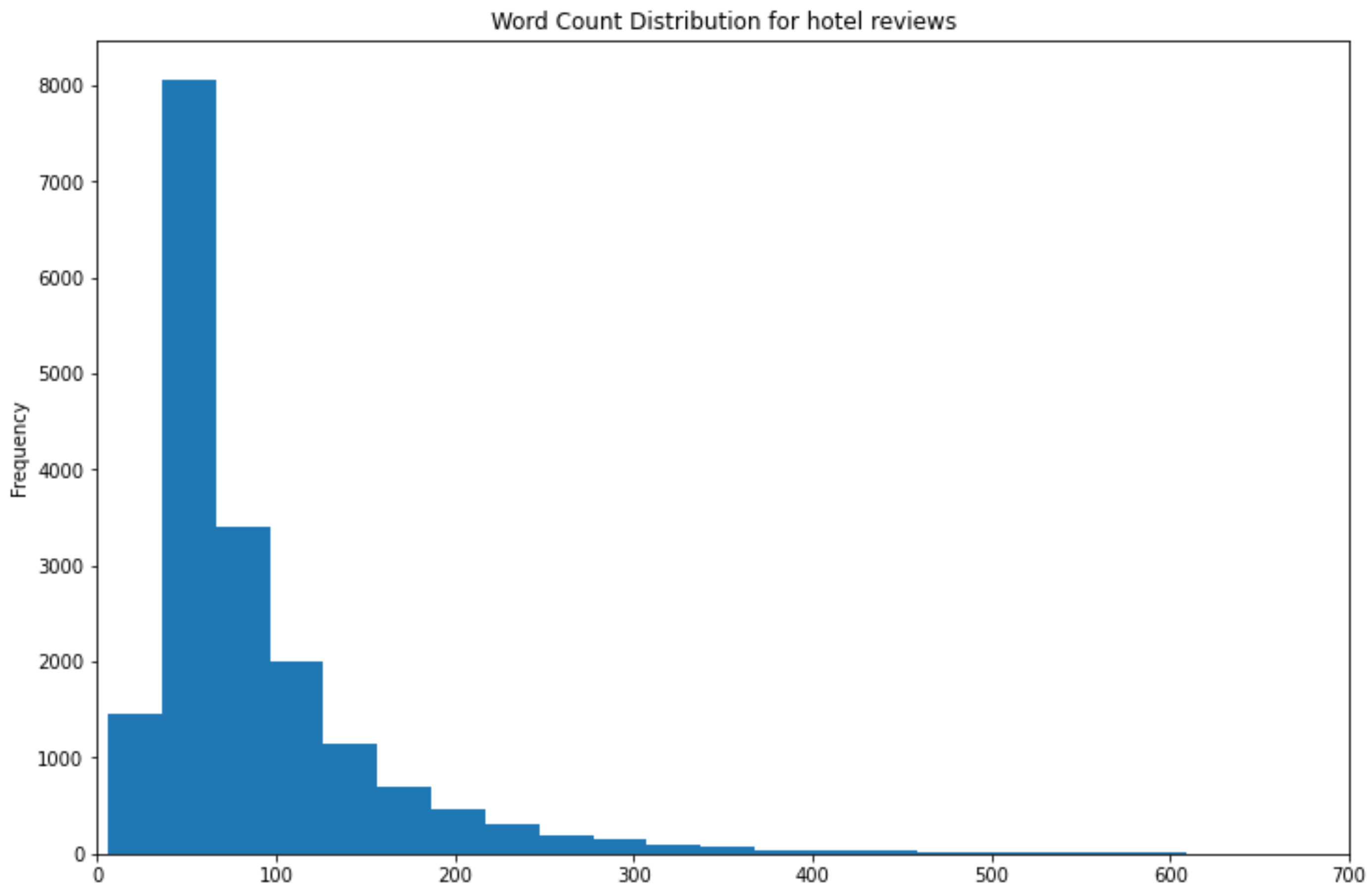


Fig 5. Word count distribution for hotel reviews from hotelReviews.csv.

A bigram was created to understand the frequency of two-word sequences in the hotel reviews. The hotel reviews' three most frequent two-word sequences were 'the room, 'the staff, and 'friendly helpful' (Fig. 6).

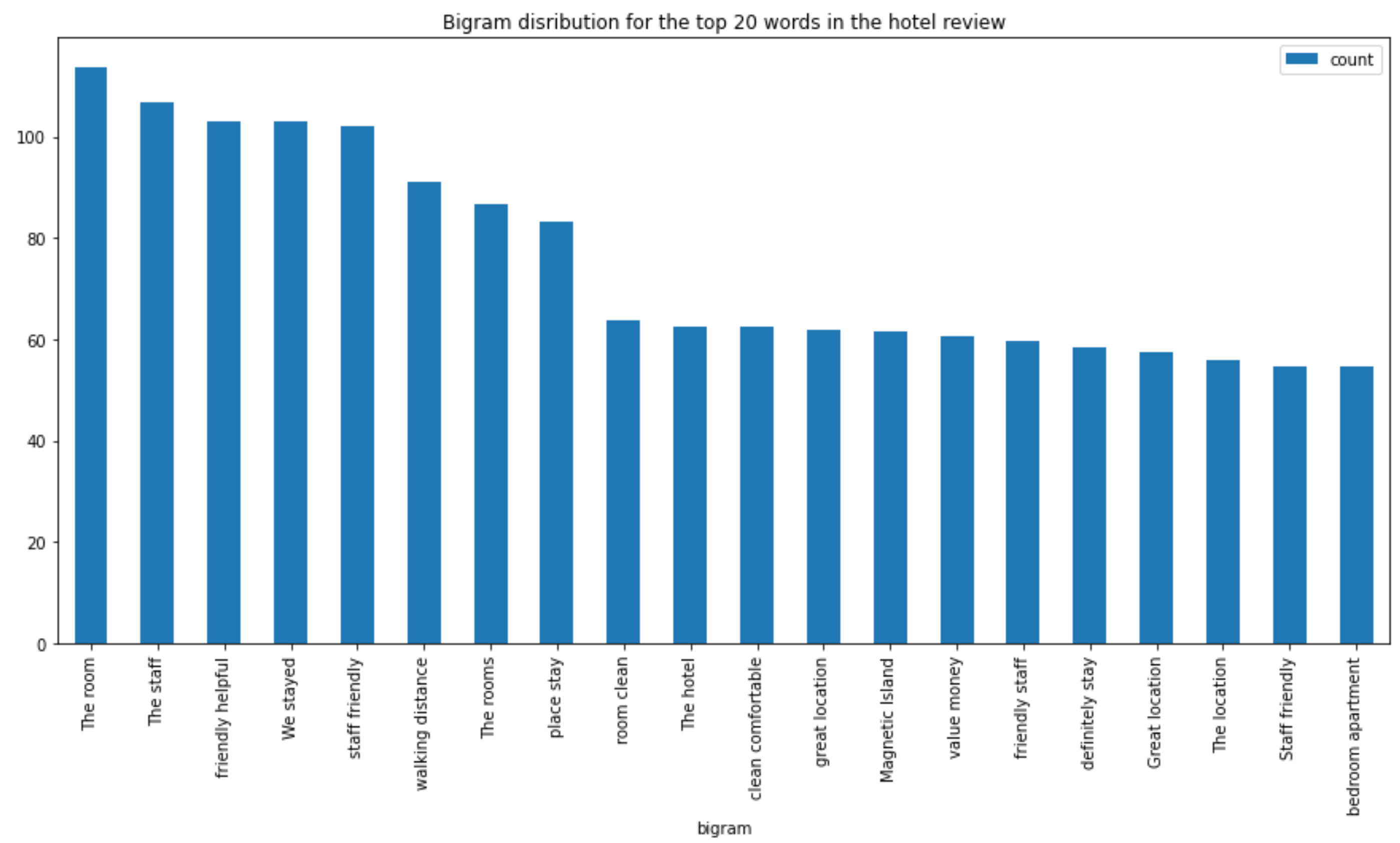


Fig 6. Bigram distribution for hotel reviews from hotelReviews.csv.

A trigram was created to understand the frequency of three-word sequences in the hotel reviews. The three most frequent three-word sequences in the hotel reviews were ‘staff friendly helpful, ‘the staff friendly’, and ‘the room clean’ (Fig. 7).

After exploring the bigram and trigram of hotel descriptions, the study identified that hotel reviews mostly talked about how friendly and helpful the staff was and how clean the room was. This is quite a contrast in comparison to the features of the room and location discussed in the hotel descriptions.

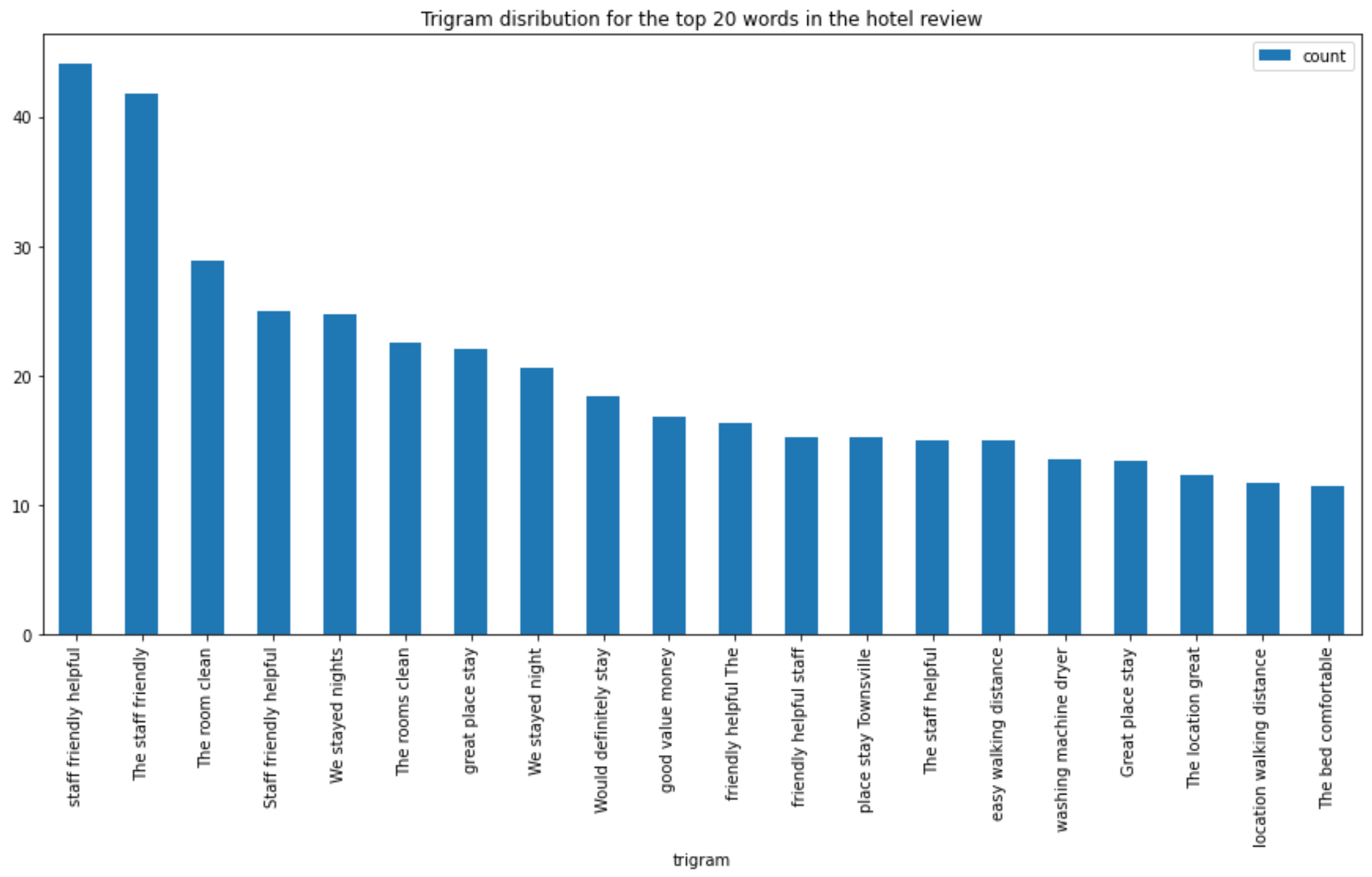


Fig 7. Trigram distribution for hotel reviews from hotelReviews.csv.

A bar plot was created to visualise the distribution of the various rating in the hotel reviews. Most people who left reviews rated their stay as 4 and 5 stars (Fig. 8).

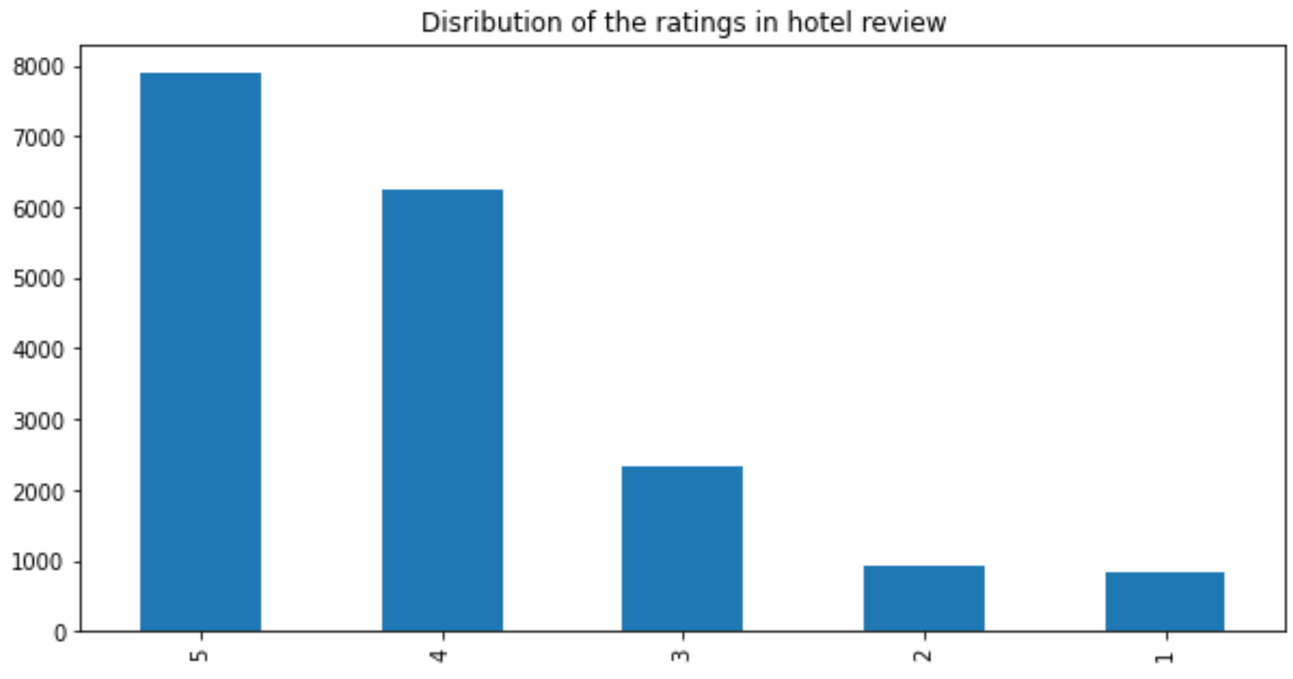


Fig 8. Distribution of ratings for hotel reviews from hotelReviews.csv.

A word cloud was created to visualise the frequent individual words in the hotel reviews (Fig. 9). The prominent words in the word cloud are 'room', 'Townsville', 'breakfast', 'great' and 'close'.

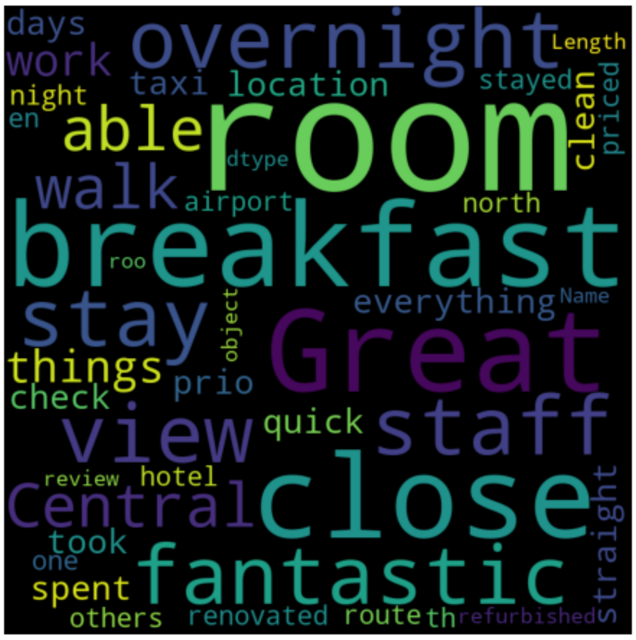


Fig 9. Word cloud for hotel reviews from hotelReviews.csv.

2.3.4 Descriptive statistics of corpus

The number of reviews per hotel ranges from 0 – 1179 in the *hotelData.csv*.