Remarks: Throughout this report, first named team will also be referred to as featured team or first mentioned team.

**Method of retrieving headlines from each article (Task 1)**

Crawling method: This program takes a seed URL as a starting point and crawls for a maximum of 300 distinct pages. The implementation relies on standard Python libraries *requests* and *BeautifulSoup* to make HTTP requests, retrieve and parse data from each website. The contents of the pages were determined to be HTML-structured in advance; hence, only the HTML parser is used. The crawler retains every URL it visits in a list of visited links in order to not repeat any data or fall into an infinite cycle. It searches for additional URLs inside the current page content by searching for *<a>* tags using *BeautifulSoup*’s *findAll()* method; then takes the *href* attributes of the results and joins them with the seed URL to compile new URLs. The program appends any unvisited URLs into a to-visit list for later use, before moving on to the next link in that list, continuing until there are no more URLs to visit or the page limit is reached. The crawler starts at this URL: <http://comp20008-jh.eng.unimelb.edu.au:9889/main/index.html>

Searching for headlines: The program uses *BeautifulSoup*’s *find()* method to find HTML tags where the *id* attribute is *‘headline’*. From testing, the crawler retrieved 147 URLs that were directly linked to one another, excluding the initial seed URL. Of the 147 URLs retrieved, exactly 147 article headlines were found, signifying that all articles contained a headline at the time of testing. This data is stored inside an article-headlines dictionary with the URLs as indexes.

Output: The program converts the dictionary of headlines into a *pandas Series* in order to sort the data by URL’s. It then produces a CSV file that presents a table of URLs and their corresponding headlines using the *csv* library. In this case, 147 pairs of URL-headline were recorded from the test.

**Method of retrieving teams and highest scores from each article (Task 2)**

Searching for teams: The program opens the *rugby.json* file and uses the standard *json* library to retrieve information about the teams it needs to search for. It does so by iterating through all the *team* attributes of the json file and adding the *name* attribute of each *team* into a regular expression string in the format of:

“(<*team name 1*>|<*team name 2*>| … |<*team name n*>)”

where each <team name> represents a team, i.e. England. This regular expression is used with the *re* library’s *search()* method to search for the first occurrence of a team in each article’s body segment, ignoring the head segment of the page HTML which provides irrelevant information. This first mentioned team is assumed to be the featured team of the article. If such a team is found, it is put into a featured-teams dictionary with the URL of the page as the index.

Searching for match scores: The program searches for match scores using the following regular expression:

"[0-3]?[0-9]?[0-9]-[0-3]?[0-9]?[0-9]"

which matches from 1 to 3 digits on each side of a ‘-‘. The scores matched are limited to three digits and a maximum value of 399, given that no rugby match has ever recorded a higher score than this. Therefore, any patterns between 0-0 and 399-399 are considered a score. The above regular expression is used with the *re* library’s *findAll()* method to find all the occurrences of a match score in each article. All the scores found are iterated through and the highest overall sum is assumed to be associated with that article. If such a score is found, it is put into a highest-match-scores dictionary with the URL of the page as the index.

Output: Using the *pandas* library, the featured-team and highest-match-score dictionaries are converted into *Series*. The program performs an inner join on the three data *Series* produced so far: article-headlines, featured-teams and highest-match-scores; by merging them into a single *DataFrame* and eliminating records without team or score attributes. The resulting merged table contains records with URLs as indexes and three columns corresponding to headline, team and score. This *DataFrame* is translated into a CSV file using its own *to\_csv()* method. From the test, the CSV file produced contained 65 articles out of the original 147 articles, indicating that 82 articles neither featured a team nor contained a valid match score. The *DataFrame* produced here is used for all further data analyses in tasks 3, 4 and 5; and will be referred to as the ‘filtered dataset’.

**Analysis of the top 5 most frequently featured teams (Task 4)**

The following graph for task 4 was generated by the program at the time of testing.

A screenshot of a cell phone

Description automatically generated

The data used: {Italy: 6, France: 8, Wales: 12, Ireland: 13, England: 20}

Analysis: From the graph, England is the most frequently featured team in the filtered dataset. The mean value for these five teams is 11.8 articles and the median value is 12 articles held by Wales; there is no mode value for this graph. The data looks to be growing linearly; however, no conclusion can be made about the linearity or growth rate of the data given that the sample size is too small.

**Analysis of Average game difference vs Frequency of featured articles (Task 5)**

The following graph for task 5 was generated by the program at the time of testing:

A screenshot of a cell phone

Description automatically generated

The data used:

Number of articles:

{Scotland: 2, New Zealand: 4, Italy: 6, France: 8, Wales: 12, Ireland: 13, England: 20}

Average game difference (rounded to two decimal places):

{Scotland: 5, New Zealand: 30.5, Italy: 13.66, France: 12.13, Wales: 10.83, Ireland: 17.23, England: 10.5}

Analysis: This data was sorted by increasing number of articles per team. From the data, we can see once again that the jump in articles between teams do not exceed 4 articles, with the exception of Ireland to England. This shows a better indication of linear growth in the field of number of articles, however, there is still no firm conclusion of such linearity. In terms of average game difference, it is apparent that given this particular ordering of teams, there is no linear growth or decay pattern in the average game difference data– the red bars are not strictly increasing or strictly decreasing. The Pearson correlation coefficient was calculated for these two data vectors with a result of -0.1748, indicating a weak negative correlation. This result implies that there is a small degree of linear correlation between the data; however, the data for number of articles still cannot be used to predict values for average game difference with high accuracy.

**Discussion of the associating the first named team with the first match score**

Associating first named team with first match score: Naturally, news articles often mention the main topic of discussion at the very start of the article as part of the introduction into the topic. Therefore, it is common that if a rugby article is written about a team and a match, these would be mentioned at the beginning of the article. Oftentimes, these pieces of information may also be included in the headline. As a result, it is reasonable to assume that the first named team and the first match score in an article are directly related.

Associating first named team with highest match score: In comparison, the method that was implemented into the crawler is tailored to find the highest match score in the article. Articles can contain information about multiple different matches, which can be located anywhere in the text; thus, there are no logical correlations between the first named team and the highest match score. Several reasons to discuss different match scores within an article include:

* There was another really interesting match that day.
* The article is comparing this match to a similar match.
* The article is giving a suggestion of what the score should have been for this match.
* etc…

Therefore, it is more suitable to associate the first named team with the first match score in preference to the highest match score in the article.

**Two suggested methods for determining whether the first named team won the match**

Method 1: Use the bag of words from the article**:** The program can be configured to scrape all individual words from an article and count how many times each word occurred. This will result in a bag of words being generated for each article. From this data, words can be weighted depending on their positive or negative connotations. By comparing the number of “positive” versus “negative” words, a prediction can be made about whether the team won or lost the match. An advantage of this method is that given the featured team is correctly identified, the outcome should yield substantially valid results as articles often focus more on the featured team. A disadvantage of it is the processing, lemmatisation and weighting that goes into each individual word, making the implementation harder. An example where this method does not work well is when an article mentions another match where the result is the opposite of the mentioned match, giving false indications of performance.

Method 2: Count the number of times the team was mentioned compared to other teams: The program can be configured to scrape all occurrences of a team name in the text and count the number of times each team is mentioned throughout the article. This will result in a table of teams corresponding to their number of appearances in each article. Using this data, the number of times the featured team is mentioned can be compared to the number of times other teams are mentioned. If the ratio leans towards the featured team (i.e. this team is mentioned a substantial amount of times compared to other teams), it can be assumed that this team won the match. An advantage of this method, contrary to method 1, is that it is simple to implement and does not require much natural language processing. A disadvantage of it is the invalidity that the results may yield if the articles talk about the featured team for their loss instead of victory. An example where this method does not work well is when an article gives a summary of all the matches in a day, hence mentioning a variety of different teams; this means that the first named team will be mentioned proportionately less, and the match will be considered a loss regardless of the actual outcome.

**Other information to extract from the articles**

The date and time of the match as well as the location where it was played can be sought from scraping the websites. This will give an indication of whether team performance is correlated to time or location of play.

Data about the opposing team can also be collected, where a simple strategy would be to find the second mentioned team in the article (after the first). From this data, it can be determined whether team performance is correlated to which team is playing against the featured team.

Individual player names can also be collected by scraping through and finding words starting with capital letters that are not locations. This data may give an indication of which players contributed to a win or loss of a team.