***Task 1.*** First, a seed URL was initialised, and an HTTP GET request was sent to the seed URL by using the Request library, and the urls fetched was stored in a Dictionary. Second, the data on the page was fetched and parsed using the Beautifulsoup library. A parse Tree object (Soup) was created and parsed using Python built-in "lxml" parser. The next URL to visit was extracted from the page and was added to the queue of seed URLs for subsequent crawling.

Third, the fetched URL was added to another queue of visited URLs. A limit was imposed to avoid breaking the site. Each URL was fetched, and the find\_all() method was used to extract all the ‘div’ containers that have an id of <id = ‘headline’>. The headlines of articles were stored as values in a dictionary, which was outputted into a csv file. The output csv file contains two column headings **url** and **headline.** A total of 147 articles were found and their headlines were extracted.

***Task 2.*** We scrape data from each web page by fetching each url using a Beautiful Soup object, and then extracting the content from the HTML script using the ‘p’ tag as a selector and the get\_text() function. The article content was tokenized using *text.split()* function and the regular expression pattern r’\W+’. Once a team name was found, the loop was exited and the name of the first mentioned team was added to an output dictionary. Articles that do not contain team names were discarded and assumed to be irrelevant.

We then find match scores within an article by using regular expressions. First, we tokenize the text using the pattern <r"\w+(?:[-']\w+)\*|'|[-.(]+|\S\w\*"> in which the pattern «\w+([-']\w+)\*» permits us to include word-internal hyphens within tokens, e.g. “16-7” and “27-year-old”.

We then use the pattern r’^\d{1,3}-\d{1,3}$'’ and and the method *re.findall(pattern, text) t*o search for rugby score. This pattern specifies the start and end of the string to be solely digits, and only 3-digit scores were valid. The valid scores were stored in a list., and the total score of each was calculated by first, converting the string into two integer scores using re.split(“-”, string), and then calculating the sum. The list was then sorted and the maximum match score of each team was added to the output dictionary. When two scores with the same maximum were encountered, the one with greater game difference is extracted. This will be useful for analyzing the relationship between average game difference and the number of times each article is mentioned in future analysis.

The output csv file contains headings **url, headline, team** and **score.** Results show that out of 147 articles found in task 1, only 63 articles contain both team names and match scores. Here, we are assuming that only quantitative scores can indicate the performance of a team.

***Tasks 4 & 5***

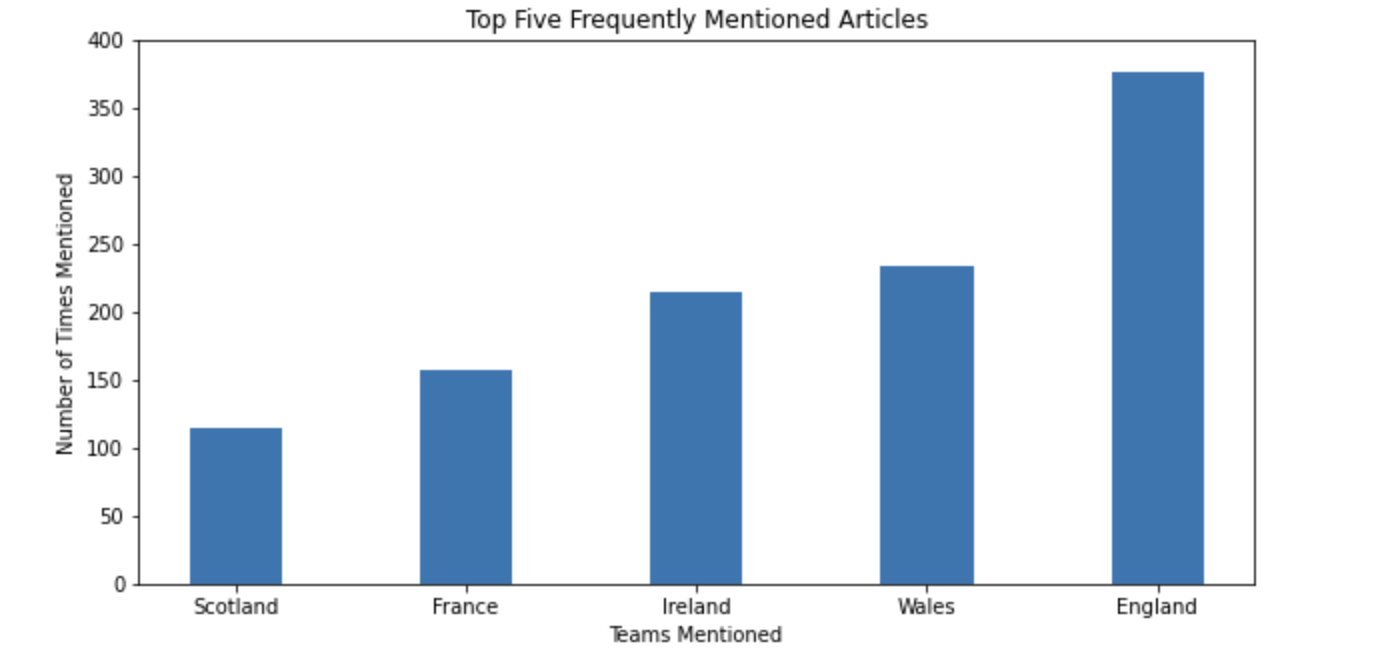


Figure 1.

In task 4, the top 5 mentioned teams in articles and the number of times mentioned was plotted into a bar chart (Fig. 1). The categorical variable, the team names, was plotted on the horizontal axis and the numerical variable, the frequency mentioned, was plotted on the vertical axis. The relative times of teams being mentioned can be visualized. The range in the frequency of being mentioned is 261, with the England team being mentioned most frequently at 376 times, and Scotland being mentioned least frequently at 115 times.

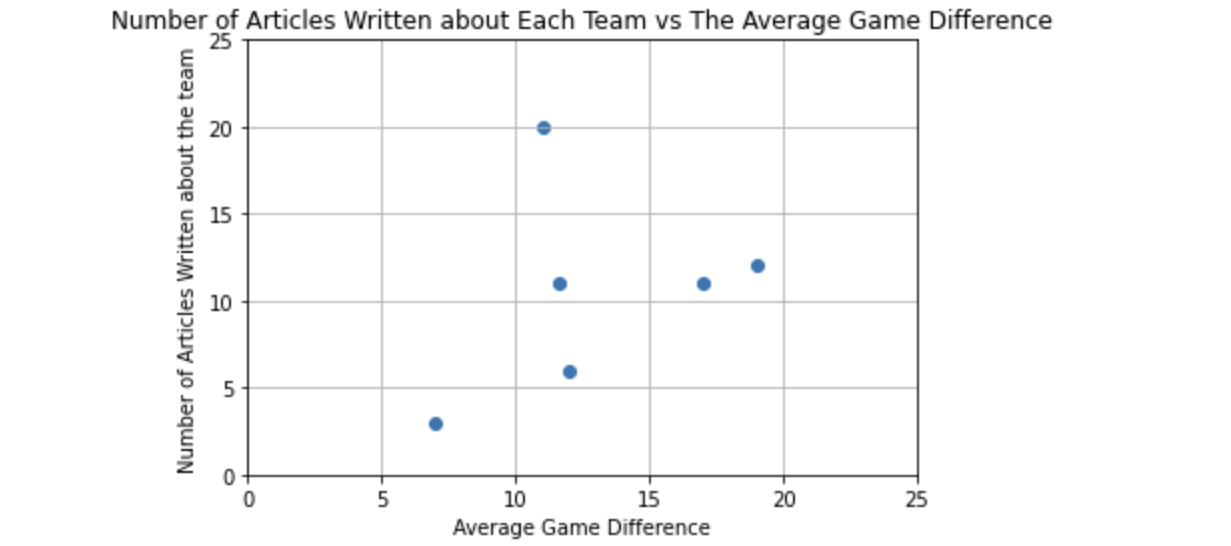


Figure 2.

In task 5, a scatter plot was used to compare the average game score difference and the number of times each team was mentioned in the articles (Fig. 2). Both axes display continuous variables. From the graph, a positive correlation can be shown by excluding the point A as an outlier (the blue line in Fig. 2), which shows that the greater the average game difference of each team, the greater the number of articles written about them.

However, a negative correlation (the red line in Fig. 3) can also be shown by excluding point B and C as outliers. This indicates that the data sample is too small, and more data points (i.e. more teams) are needed to draw a reliable conclusion about the correlation. In addition, although a correlation is identified, the causal relationship between two variables cannot be concluded, i.e. a bigger average game difference does not cause more articles written about it, since both variables may be influenced by a third factor such as the source country of articles. Pearson correlation is used to assess linear correlation between the average game difference and the number of articles written about them. A correlation coefficient r of + 0.91 indicates a strong positive correlation between the two variables.

The scatter plot is a powerful representation of the general trend between two variables and provides a clue for predicting its underlying causal relationship. i.e. Teams with greater average game difference are likely to be written by more articles.

*A discussion of the appropriateness of associating the first named team in the article with the first match score.*

The association of the first named team in the article to the first match score is not very appropriate. Given that both are first mentioned, it could be assumed that they are related, and this assumption is more valid than extracting the maximum match score as seen in task 2. Yet, this assumption still does not generalize not well to minor variations in the input data, and the largely inevitable noises and language variations pose hurdles to the application of this rule. Although both are first mentioned, we cannot assume that no other teams exist between the two, i.e. in the order of [Team 1] [Team 2][Score 1]. For example, “England beats France, whose 16-17 defeat to New Zealand was ..” In this case, the score 16-17 is actually referring to the second mentioned team France not England, and our assumption is not appropriate. If we want to apply the rule, we need to make sure no other teams are mentioned in between the two.

This assumption can also be very risky since the distance between two entities (the first match score and the first team mentioned) is not known. I.e. They may not appear in the same sentence. Due to the inability of making semantic inferences and detecting language contexts, the relationship between the first named team and the first match score cannot be safely assumed only based on the order they are mentioned.

*Two suggested methods for figuring out whether the first named team won or lost the match.*

The first method involves using rule-based systems to look for patterns. We will first create a pattern dictionary which include a list of synonyms of ‘win’ and ‘lose’ (e.g. ‘win’, ‘defeat’ and ‘beat’, and ‘lose’, ‘failing’). Text is then preprocessed using tokenization, case folding, stop word removal stemming and lemmatization. We will then use regular expressions to search for all sentences containing both the first named team and dictionary words in tokenized texts. By counting and comparing the occurrence of positive and negative dictionary words, we can determine whether the first named team won the match.

This method has high precision in extracting relevant relations between the first named team and ‘win’ or ‘lose’, since the keywords used exactly match the information of interest (e.g. win or lose). However, this method is only extracting relations on the sentence level, while relations can span over sentences and even cross documents. Thus, this method cannot capture long range relations. It also cannot make semantic inferences that lead to false negatives. For example, in *‘We have lost a game we should have won*’ means the opposite thing to ‘we win’, which would be captured by this system when searching for the word ‘win’.

In addition, the manual implementation of rules means that minor deviations in the target sentences would lead to linear growth of the number of rules required, while the size of the dictionary is limited. Also, this method has little learning capacity, so it does not improve over time, unless the rules are changed manually.

The second method involves using machine learning algorithms to determine whether the first named team won or lost. In this method, it will be assumed that the greater positive comments or descriptive words within the article means the first-named team wins. The first step involves converting unstructured texts into numerical representation by first, pre-processing the text through tokenization, case-folding, stemming stop-word removal and lemmatization, and second, using Bag-of-words representation to convert texts into numerical vectors. Then, the machine learning algorithm will be fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. win, lose) to produce a classification model. After training, the model will be used to make accurate predictions about team performance in unseen articles.

This method is much faster and more accurate. When input data differs only slightly, machine learning models can generalize better and result in a higher chance of extracting relevant information, compared to the need to manually re-define rules. Also, the performance improves as the datasets grow bigger.

However, we require large amounts of hand-crafted, structured training data to train the classifier, and it is time-consuming in this case, where pre-labelled articles are non-existent. Also, this method requires pre-processing input unstructured text into numerical matrix, which is computationally burdensome.

To conclude, these problems can be resolved by combining rule-based systems with machine learning. For example, we can employ rule-based pattern extractor using link grammar parser and Stanford PoS (Part-of-Speech) tagger to label texts and construct training sets, and then use a semi-supervised machine learning algorithm to predict unlabelled articles.

***Future Information extraction directions***

We can extract information about the total games won by each team based on their match scores, and compare them by constructing a bar graph. This would be an additional indicator of the performance of winning and losing teams. To approach this task, we would first identify two named teams, then extract relations between these two teams.

Specifically, we can initially look for all triples of the form (X, α, Y), where X and Y are the two named teams, and α is the string of words that intervenes between X and Y. Next, we extract all substrings that intervene between X and Y into a list. We can then use regular expressions to pull out just those instances of α (relations) that are match scores. For example, we can use the pattern r'^\d{1,3}-\d{1,3}$'’ and the method *re.search()* to extract substrings of interest. By comparing the two scores of two teams, we can then identify the winning and the losing team. We increment the number of times that each team wins and find the number until we process all articles. Finally, we graph the teams and the number of times they have won using a bar chart. The bar chart would provide a powerful representation of the performance of each team in terms of the number of times they win.