**Elements of Data Processing**

**Assignment 1 Report**

**Quang Tri Le – 1118563**

**Introduction:**

Crawling and scraping website is an essential automatic activity that derive unstructured data from webpages into a more structured data type. Well written “crawler” (based on personal or organizational purposes) will assist data scientist in visualizing and analyzing the tendency as well as understanding the relationship between given variables. With such information in hand, executive officers will be well advised in order to make unstructured decisions that can create competitive advantage for their own business.

In terms of legal issues, not every website allows crawling and scraping. In fact, each website limits the activities of crawling and scraping in a file usually called robot.txt. Therefore, one must be carefully considered all the legal restriction before implementing a crawler to a certain webpage. Since the crawled websites for this report were given for educational purposes, it was assumed that no legal issue would be encountered by any mean.

The report will attempt to describe the process of making the crawler for websites about rugby articles in order to prove the appropriateness of the program based on given requirement and assumption. Description, analysis and limitation would be discussed in divided sections (tasks) as stated in the specification.

**Methods:**

The program used to crawl and scrape the websites was written in single Python file called “final.py”. The program functions based on the assistance of libraries: request, bs4, pandas, numpy, re, matplotlib and random which are allowed within the scope of the subject. Basic data structures such as dataframe from pandas library and array were extensively used to synthesize related information as well as generating output in “.csv” format and suitable graphs in “.png” format.

**Task 1 & 2:**

Task 2 was done within task 1 in order to utilize the main while loop that crawl and scrape the whole set of given websites.

1. Crawling method and Output of task 1:

To crawl the whole set of webpages, requests library was used to access to the URL of a webpage. Each URL is made of two parts (except for the Welcome page) which are “base\_url” and “tail\_url”. Every website has the same base\_url but unique tail\_url. The tail\_url was found with assistance of bs4 by assigning it to the command:

*tail\_url = soup.find("div", id = "links").find\_all('a')[1]["href"]*

where *soup* is the result of html parsing the webpage in task 2. Once a website is fully scraped, the URL is added to the visited URL list. The loop will break when the URL found in the list.

After crawling was done, there were 147 websites corresponding to 147 unique rugby articles (welcome page not included). The output was saved to “task1.csv” file with two columns: “url” and “headline” for further use in later task.

1. Scraping method and Output of task 2:

Task 2 was done within the same while loop as task 1 since it would scrape data from a crawled website. There were four objects to be scraped from the website: URL of next website to be crawled, headline of current article, legit first appeared team name and legit maximum score.

URL of next website was achieved as discussed from above section. All URL will be stored in an array.

Headline of current article was achieved by the command *soup.h1.text.* All headlines will be stored in an array.

To find for the legit first mentioned name in an article, a loop running through all team names extracted from “rugby.json” was executed in order to find for the team with smallest string index in the text: *text.find(team\_name)*. It is possible that all teams from the “rugby.json” would be mentioned in the article. Therefore, it is safe to find for the appearance of every team in the article and keep the one with smallest index. All legit first appeared team names will be stored in an array.

To extract the largest score of an article, all scores from inside the text were extracted using command:

*re.findall(r' \d{1,3}-\d{1,3}', essay)*

where *essay* is the content of the website (including the headline). The used Regex Expression was still naïve with the assumption that a score would not be used as the start of a paragraph (if so, there would not be any space before the score). With an array of all possible scores in an article, Regex Expression was further used for each score in order to find for the one with largest score (sum of two integers between “-” to be largest):

*score1 = int(re.findall(r'\d{1,3}-', score)[0].strip("-"))*

*score2 = int(re.findall(r'-\d{1,3}', score)[0].strip("-"))*

The score that has the largest sum of *score1* and *score2* in an article will be saved to an array.

After scraping data and produce four core arrays, these arrays were selectively combined into dataframes that would be used to create .csv file for task 1 and task 2.

Task 2 results are reflected in “task2.csv”. There are four columns in the file which are “url”, “headline”, “team” and “score”. There are 64 rows (heading not included) in the file after discarding articles that do not include team name and/or a match score.

**Task 4 & 5:**

1. Task 4:

Chart, bar chart

Description automatically generated

The left sided graph shows the top 5 team with the greatest number of mentioned articles. Data used to make this graph was taken from “task2.csv”. By parsing through the csv file and using dictionary data structure, a dataframe was made including the team name (key in dictionary) and number of mentioned articles (value in dictionary, incremented while looping through the file). This dataframe was then used to plot the bar chart.

1. Chart, scatter chart

   Description automatically generatedTask 5:

The left sided graph shows the relationship between the number of mentioned articles of each team and average game difference of corresponding team. For average game difference, the data (array of integer) was made in task 3 using the dataframe from task 2 that contains the score and corresponding team name. Score difference in each article was achieved using regex. The average game difference of a team was calculated as *(sum of all game difference)/(number of mentioned articles)*. A dataframe storing average game difference and number of mentioned articles were then used to plot a scatter as shown.

1. Analysis of task 4 & 5:

Task 4 produced a bar chart showing five most frequently mentioned teams in articles. Assuming the crawling method to be ideal, the graph was made merely for the sake of displaying information without any further implication. However, the crawling method is not realistic since for every article, multiple valid teams could be mentioned, thus there are overlaps articles between valid teams.

Task 5’s scatter plot shows relationship between average game difference and number of mentioned articles of a team. From the scatter plot, it is observed that most of average game differences lie within the range 10 to just below 20 with outliers New Zealand and Scotland. Other than that implication, the scatter does not show any further useful information such as team performance since the average game difference was taken as absolute values according to the assignment specification. If there was a rule of how a match score forms (e.g. home team gets mentioned first in a score) and negative difference was considered, the scatter plot might convey more interesting information.

**Discussion of the association between first named team with first match score:**

There are number of ways to interpret first match score that does not necessarily mean the final score of the match. First match score in a rugby article could be referred to as the score of **first half** of the match or a milestone in the game that is worth mentioning. Assuming taking the first named team is an ideal method, associating it with the first match score might make the program scrape false information about result of the match. There is still chance that first match score encountered could also be the final score of the match. However, using such method would not cover the situation where multiple scores are reported in the article. Therefore, taking the largest score of the article, although not a perfect solution (it might refer to another match), is a simple but more well-rounded solution.

In terms of first named team, the problem had been discussed in section c. of **Task 4 & 5** where multiple teams are mentioned in the articles. Since the assignment specification assumes that this is valid, the problem would not be further discussed.

In short, by associating the first named team with first match score, there is a high risk in scraping false result of a match which leads to false plotting and analysis.

**2 suggestions that might predict first named team win or lose:**

1. Home team gets mentioned first in the score:

In sport broadcasting, the score usually shows the point of the home team first (at the left-hand side). Therefore, when being reported on the article, the score tends to be kept in the same order. The program can make used of this to determine whether home team lost or won in that match providing the method checking if first named team is the home team or not. We can further verify this by checking what stadium is mentioned in the article. If the stadium is not first named team’s, chance is that the first named team is a guest team. Winning or losing of first named team will then be decided based on the sign of score difference.

This method works well in the articles where stadium name is mentioned. However, if the article does not include any of them, first named team by default should be the home team. Such default assumption could lead to catastrophe if most of the articles do not include stadium name. On the other hand, we can still make use of the conventional score order and create a better method to determine the home team and guest team.

1. Account negative values for game difference:

This method was first mentioned in section c. of **Task 4 & 5**. Assuming the first named team gets mentioned first in the score, the raw difference of the match score is recorded without taking the absolute values. If the difference is positive, the first named team won and vice versa.

The method does not require any extra information such as stadium name and could work in all valid articles. However, there are still assumptions to be made which lower the accuracy of the prediction. Moreover, taking negative values for game difference could spread the scatter plot in a larger range which make it harder to identify frequent range and outliers.

**Other extractable information and suggestion for better understanding team performance:**

In sport articles, there are usually comments made by the coach or players after or before the match occurs. These comments are usually in double quotes which we can utilize regex to extract the content. With the modified “rugby.json” file including extra information about the players of a team, it is certainly possible to know the person (from which side) is making the comment. With the help of natural language processing, descriptive words such as “good”, “bad”, “confident”, “disappointing”, “happy” … could be extracted to understand better the team condition and performance for every match.

Furthermore, bigger picture could be achieved by extracting date and times of the match in the article for making time-based graph of team performance during a season for better visualization and analysis. This could assist the head of the team to make suitable plan for the future of the club.

**Conclusions:**

Although the written program was still naïve because it is still based on number of assumptions from the specification that lead to low reliability if implemented in practical scenario, there are still space for potential improvements. With that being said, the program can potentially work fine in a more “structured” rugby articles and the framework could still be applied for more complex tasks.