**COMP20008 Elements of Data Processing- Project 1 Report**

**Hunter James Thorpe (ID:1079893)**

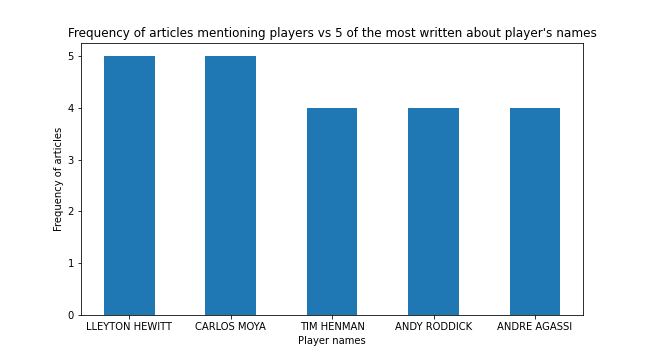
1. **A description of the crawling method and a brief summary the output for Task 1. (2 marks)**

The crawling method begins with the seed url ('<http://comp20008-jh.eng.unimelb.edu.au:9889/main/>'). From here I used the requests module to access the html of the page, followed by parsing it through BeautifulSoup to create a soup object. Following this I combined the seed url with a .a.get(‘href’) object applied to the soup object of the seed page to get the url of the first article on the website. Before applying and iterative approach, I manually crawled to the next article applying the methods .find('p', {"class" : “nextLink”}).a.get(‘href’). I did this so I could maintain a copy of the first article’s url under the variable first\_url. This enabled me to use a while loop to crawl the reminder of the site, ceasing to iterate when the current page’s url is the same as the first site’s url. The output for Task 1 is in the form of a csv file named task1.csv. I wrote header of this file ‘url, headline’ at the beginning of the script body where the file is created. The remaining output is written within the parse\_page function, where the url and headline of every article is written using .writerow() as each article is parsed.

1. **A description of how you scraped data from each page, including any regular expressions used for Task 2 and a brief summary of the output. (3 marks)**

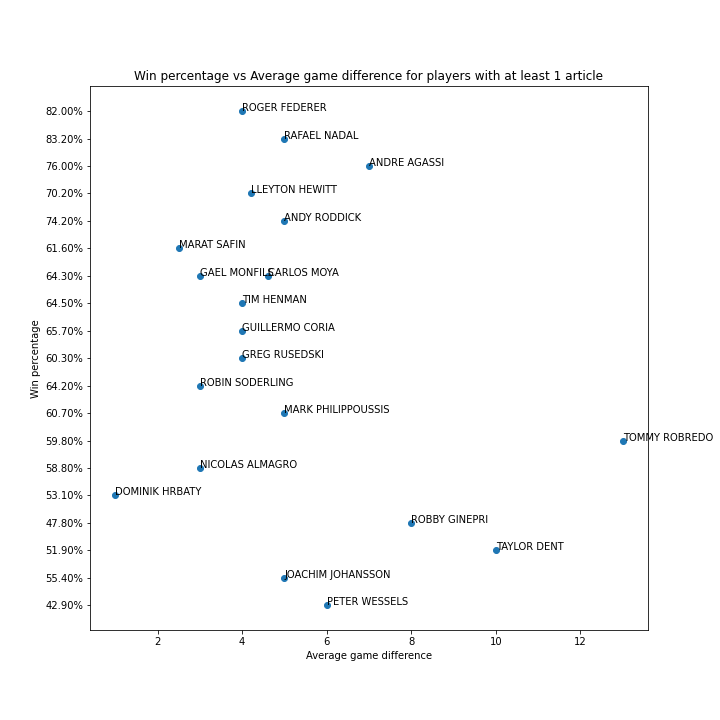
As required under Task 2 my script aimed to scrape each page’s url, headline, first full name of a player, and a completed match score. I scraped each pages url by using the hyperlink of the previous page, using the methods .find('p', {"class" : “nextLink"}).a.get('href'))) on the previous page’s soup object. To obtain the headline, I used the methods .h1.text on the pages soup object. To find the first complete match score, I decomposed the string making up the body of the article into a list, and iterated through each pair of words one at a time, e.g, the string ‘Wimbledon champion Maria Sharapova’ would produce the word pair ‘Wimbledon’ and ‘champion’ on first iteration, followed by ‘champion’ and ‘Maria’ etc. in each iteration of the loop I used the re.sub method with the regular expression ‘[^\w\d\s\)\(-]’ and .strip() method to remove all punctuation from the strings (excluding brackets and dashes) . I then used the re.search method and the regular expressions “^[0-9]-[0-9]$" (for regular sets) and “^\([0-9]{1}-[0-9]{1}\)$" (for tie-breakers) . Once the script found a word pair where each pair was a legitimate set or tiebreaker, I used a while loop to check subsequent word pairs to see if they were part of the match score until I reached a string that did not satisfy the above regular expressions. As each set is read I appended them to a list which is converted to a string and returned at the end of the parsing function.

In searching for the first full name mentioned, within the same iteration I used a similar regular expression except without the exceptions for brackets and dashes (‘[^\w\d\s]’), in combination with the .upper() method to remove all punctuation and make the strings uppercase to match the data from the JSON file. Outside of this function I parsed the JSON file data by creating a dictionary called player\_dict where the keys are the full names of the players and the values are lists of the form [win percentage, number of complete articles about player, total game difference]. I used the .keys() method on this data within the loop to compare the full player names to a string composed of the two pair words of the current iteration. If a match was found I updated the variable player\_value to represent the players name in the form of a string. At the end of each iteration, an if statement checks if a player value and score value has been obtained, and if it has it updates the relevant data structures within the program and breaks the loop. The output of task 2 is similar to task2 however there is two extra columns headed player and score where the full name of the first player mentioned and the first full match score mentioned is listed.

1. **An analysis of the information shown in the two plots produced for Tasks 4 & 5, including a brief summary of the data used. The plots are to be shown (included) along with your analysis. (4 marks)**

I chose to generate a bar chart (figure 1) as my response to task 4. The chart plots the names of five players that articles are most frequently written about against the frequency of articles written about each player. The data used for this plot comes from the variable player\_dict, which contains data taken from ‘tennis (1).json’ (player names), and data scraped from the articles (frequency of articles written on each player). The chart shows that the two most frequently written about players, Lleyton Hewitt and Carlos Moya, have both had five articles written about them from the data set. The remaining three players described in the plot, Tim Henman, Andy Roddick, and Andre Agassi, have all had 4 articles written about them.

**Figure 1:** Bar chart showing Frequency of articles mentioning players vs 5 of the most written about player’s names (constructed via Matplotlib)

In response to task 5, I constructed a scatter plot (figure 2), that plotted players win percentages vs their average game difference. The data used for this representation came from the variable player\_dict. The win percentages and player names where transferred into this dictionary from ‘tennis (1).json’, and the average game difference was calculated using two ongoing tallies: a players total game difference divided by the number of times a player is mentioned in an article. The plot shows that 15/ 20 (75%) of the players with at least one article written about them have an average game difference between 2 and 6. Furthermore, the plot shows no correlation between average game difference and win percentage.

**Figure 2:** Scatter plot showing Win percentage vs Average game difference for players with at least one article written about them (constructed via Matplotlib)

1. **A discussion of the appropriateness of associating the first named player in the article with the first match score. (2 marks)**

Whilst in most cases the first player whose full name is mentioned in the article has generally participated in the first complete match score, this is not always the case. The author of an article may choose to discuss an opponents previous match score to give context to the article, resulting in an erroneous association between the first player named and the first match score. Additionally the author may choose to discuss a previous game of the tournament or venue involving two seperate players, resulting in a similar error. Furthermore, the author could address a previous match of the first named player, resulting in the player’s previous match potentially being computed twice while neglecting the match score associated with the current article. Another possible source of error, although less likely, is that the first fully named player did not participate in the match that resulted in the first match score. All of these possible sources of error have similar impacts on the data, in that they make an erroneous association and fail to make the correct one, leading to a potentially severe impact on the information that is gleamed from the dataset e.g. average game difference. A potentially more accurate way of associating players and match scores would be to designate the first complete match score to the two most frequently named players (either full name or last name) in the article. Whilst this method still has possibility for error, particularly in cases where more than one match score is mentioned in the article, it is more likely recognise the correct relationship between the match score and the players involved in this match.

1. **At least one suggested method for how you could figure out from the contents of the article whether the first named player won or lost the match being reported on. (1 mark)**

One possible method for identifying if the first name player won or lost the match could resolve around a set of words and phrases frequently used by sports journalists such as ‘defeated’, ‘triumphed’, ‘overcomes’, ‘slumps to’, ‘shocks’, etc. By using each of this phrases and their associated meaning in context, an algorithm could be developed which locates the first instance of one of this such phrases, and then finds the first instance of a players name and its relation in sequence to this key phrase. For example, if the word ‘defeated’ was located in the body of an article, and the first occurrence of a players name occurred before this word, the algorithm would attribute the match win to this player. However if a phrase such as ‘slumps to’ was located the opposite would be true, as it is most likely the phrase is being used to indicate the player has lost to their opponent.

1. **A discussion of what other information could be extracted from the articles to better understand player performance and a brief suggestion for how this could be done. (1 mark)**

One other type of information that could be extracted regarding a player’s performance could be their most common injuries and ailments. For example if an article about Roger Federer includes his name and words such as ‘ankle’, ‘elbow’, ‘sprain’ or ‘foot’ an association between this data and the player could be made. Once sufficient data from numerous articles is collected, inferences could be made regarding the player’s injury history. Suppose a significant number of articles mentioning Federer included the phrase ‘ankle sprain’ this information could be used to support the hypothesis that this injury is an ongoing issue for the player. One problem with this method is that the injury referenced in the article could be wrongly attributed to the wrong player, and a fairly sophisticated approach would have to be adopted to eliminate the chance of this happening completely.