In order to Webcrawl through the articles, the code from workshop 4 , exercise three was utilized and modified to fit the goal of the assignment. The html parser 'BeautifulSoup' was imported, which allowed the parsing through the html data in the articles' website and extract the information it retained. The seed URL was copied and pasted into the code, as the starting point of the Web Crawling (http://comp20008-jh.eng.unimelb.edu.au:9889/main/). Then a dictionary of 'visited' URLs was made and the seed URL was added to it. This dictionary was to save the URLs that you have visited and parsed through. Using the soup.findall('a') function, all the URLs on the page was found and added to the 'to_visit' list. After these have been established, a while function is used to iterate and parse through the URLs in the 'to_visit' list. At every URL visited, all the URLs found in that site that were neither in the 'to_visit' list or 'visited' dictionary were appended to the 'to_visit' list. Finally, to inhibit the loop from becoming an infinite while loop, it broke the loop when the URL that is for the 'to_visit' list in the 'visited' dictionary, which finishes the web crawling.

The outputs obtained from task one in the form of a csv file contains the URLs of the articles that was crawled and the corresponding headlines to each article. The total amount of articles crawled were found to be a 100, which could be observed from the numbering of the csv file. A certain pattern was found in the URL structures, which was that although each href contained a substring from the headlines, the end of each href was numbered from 1 to 100. Starting from harapova100.html, the next article goes to enmanove001.html and so on. Displaying the information in the csv format also allowed for the clear observation that almost every headline contained the last name of a player.

In order to scrape data from each page, every time a new link from the 'to_visit' list was visited, a series of functions were utilized along with BeatifulSoup and regular expression to parse through and search for the data required. For task 2a (extracting the full name of the first player mentioned in the article), each time a new link in the 'to_visit' list is popped, a custom function takes the soup of the site the link goes to and finds the first player mentioned in the article who is also in the tennis.json. This is done by going through each player in the json.file, check if the name is in the paragraphs or headline of the article using for loops and returning the index value of the paragraph at which the player's name is found. The player with the lowest index value will be the first player found in the article with their full name which is returned as the 'first player found'. The paragraphs and headline were found using soup.findall('p') function and soup.h1.string, then used a for loop to concatenate the headlines with the paragraphs found using findall.

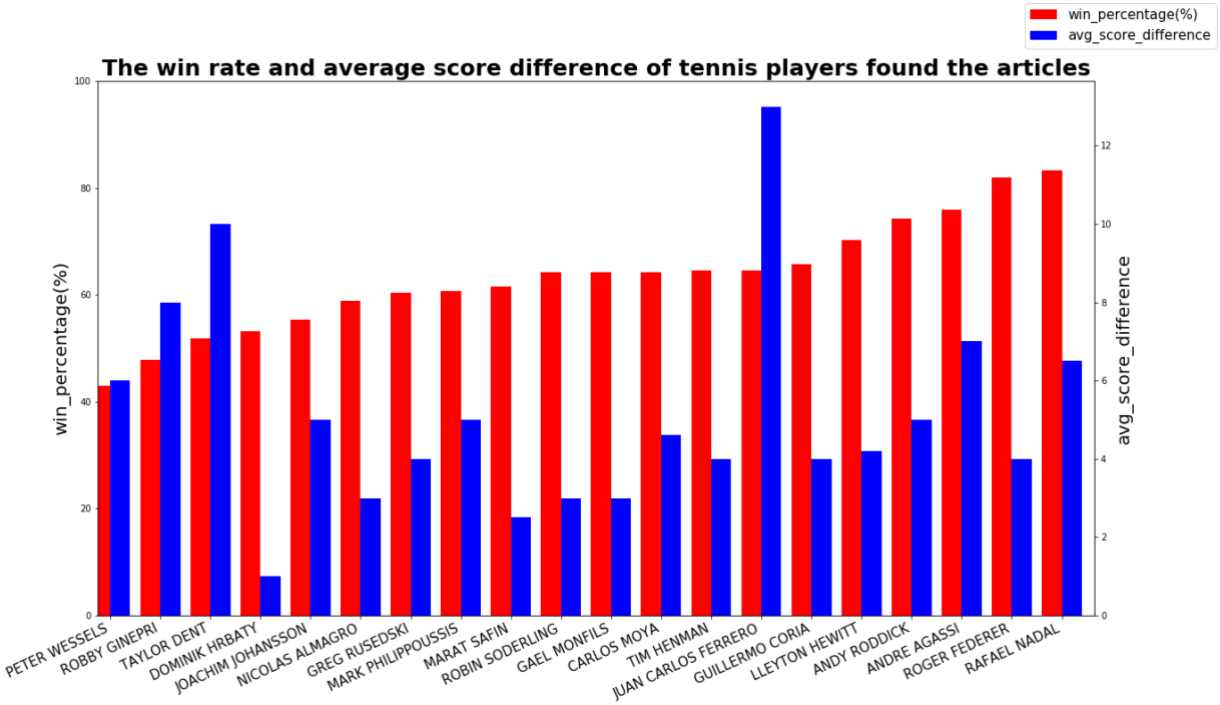
For task 2b, to find the first complete match score identified in the article, three custom functions were used, 'find_match_scores', 'check_for scores' and 'remove_tiebreak_points'. Find_match_scores will return the final match score found and 'check_for_scores' is a recursive function that actually goes through the paragraphs and headline to see whether there is a score or not. Remove_tiebreak_points is a simple function that removes the tiebreaker points/points in the brackets. Firstly, when the find_match_score function is called, it concatenates all the paragraphs and headline just like 2a and then calls check_for_scores. Check_for_scores utilizes re.search and searches for the pattern '\d+[-|\/]\d+' in the paragraphs while using .start() which returns the index of the start of the match. If the .search().start() function found something, the index is used to continue character by character while concatenating it to an empty string until it reaches a character other than numbers, a dash, forward slash or brackets. This completes the potential 'full score' string of the match. However, if the found string contains less than 2 scores, then it is not a valid match. Hence the findall() is used on the string with the same pattern ('\d+[-|\/]\d+') and is checked if the length (len()) of the output is below 2. If that is the case, it is discarded and a for loop is used to shorten the paragraph up to the index of the 'invalid match score' in order to call the function recursively, until a valid match score is found or the paragraph is shortened to nothing. Once the score is returned from check_for_scores, it is checked whether it returned None, which results in find_match_scores to return None as well. However, if it did return a score, it

is then passed through remove_tiebreak_points, which uses a while loop to go through the string and remove the substring from '(' to ')' and returning it. After these functions returns, find_match_scores process the score to check the validity of it. Its first condition is that both points in a score must be higher than 6, the second is that if one point is 6, then the other points of that score must not be higher than 8 and the third condition is that if both points are higher than 6, then the difference between the two points should be smaller than 2. If any of these conditions are violated, the function returns None. Otherwise, the function returns the validated full match score which is then appended to the list of valid match scores.

The output of this task showed 39 results and all the results found were valid match scores. A pattern was observed in the csv output where the first match of the names found in the article mostly are mentioned in the headlines of the article as well. This shows that most articles in this assignment puts the name of the person in the headlines also puts the full name of that person in the article first. This may mean that the first name found in the article is highly likely to have the article about him or her.

Although the first person found in the article matches the headlines, the method of searching for match scores is not sensitive to whether who specifically won or lost or if this was the actual game of the players discussed in the article. It also cannot tell whether the actual game was 5 or 3 games. The scores are merely the very first valid score of the article that is 'possibly' the score of the player discussed in the article. Hence, although the scores obtained possibly may be valid, they do not necessarily represent any match scores of the first player found. Ultimately, it may be appropriate to relate the article to the first player found, but it's not fully appropriate to assume that the first match score found belongs to the first player found.

PLOT 1 (Task 5)

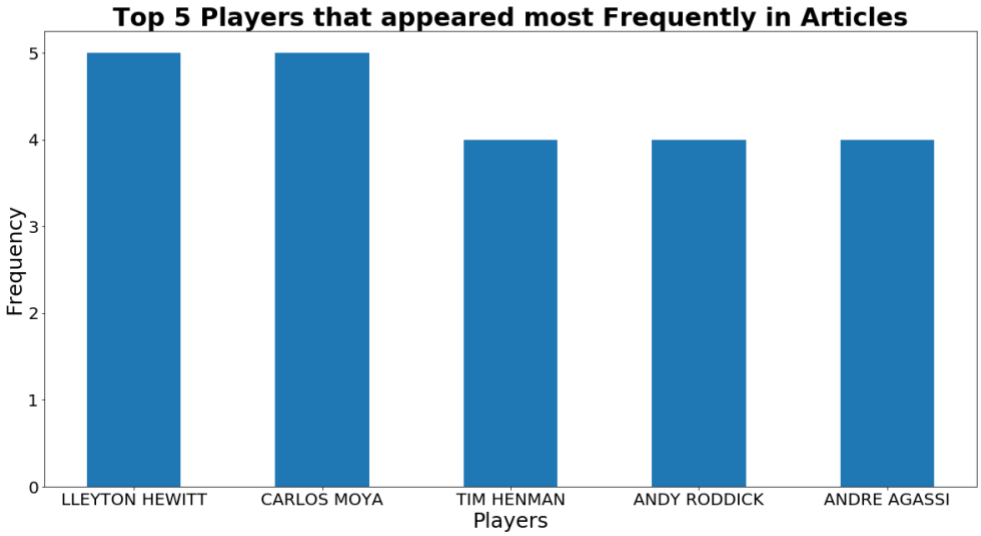


The plot found in task 5 displays the win rate of each player and the average score difference that is acquired from the calculations. The red bars represent the win rate and the blue bars represent the average score difference. The highest win rate is around 84 percent and the lowest is around 43 percent. For the average score difference, the highest is 13 and the lowest is 1 is set up so that the win rate is ascending from left to right, which allows the graph to compare it with the trend of the average score difference. The average score difference data can be observed to have no particular trend at all, much the less parallel with the ascending

win rate trend. Hence, only observing the data from the graph, it can be concluded that the win rate does not affect the average score difference of the players.

However, although the win rate is data extracted from the json file, the average score difference is data obtained from the same method as the match scores data in task 2b, hence the same credibility applies to this data as well. Also, if the frequency of the appearance of players in articles is low, the average score difference has a higher chance of being high. Since the frequency of all players are not the same, this data is skewed. As it is a discredited set of data from only a few scores, the average score difference is not an exact representation of the player's real average score difference.

PLOT 2 (Task 4)



The plot found in task 4 shows the top 5 players that appeared most frequently in articles. First was Lleyton Hewitt and Carlos Moya who both appeared 5 times and Tim Henman, Andy Roddick and Andre Agassi appeared 4 times each in the articles. This data on its own does not convey much information and hence should be compared to the plot from task 5. Although Lleyton Hewitt and Carlos Moya are in the higher section of win rates and similar to Tim Henman's win rate, they are still much lower than Andy Roddick and Andre Agassi. In contrast, Hewitt and Moya are one above the other two in the frequency of appearance in articles, which may convey that win rate is not necessarily related to appearance in articles. However, the difference in frequency is slight and all five of them are definitely in the higher section of win rates. Hence this can highlight that the higher the win rate, the higher the appearance in articles.

Task 4's set of data may also be skewed as the number of articles that were crawled is extremely low and if the sample size is low, the lower the chance that the frequency data represents the population mean.

In order to attempt to find out the outcome of the player's match in the article, a method such of making a dictionary of positive, victory induced words such as: 'win, won, recovered, successful…' and a dictionary of negative, defeat induced words such as: 'lose, lost, twisted, injury…' then going through the article to see which words were used more. If the positive words were used more, the article may be more likely to be about the win of the player's tennis match and vice versa.

To extract more information about the player's performance, every time the player's name is mentioned, that paragraph could be extracted and like the method of finding out who was

victorious, use the same (or a modified) dictionaries and check in that particular paragraph which words were used more. This may give a slight indication whether the player's performance was good or bad.