Report by Sonia Lin

Introduction
The purpose of this report is to communicate the process and analysis that I undertook to carry out the tasks of this project. I will give an overview of the strategies that I used to get my results. Furthermore, I also touch on the limitations that the strategy has and additional methods used to find further information about the performance of players.

Crawling method and analysis for Task 1
First of all, I went to the main page URL and used inspect element to see where the next link is located. It is located within the 'href' attribute, so I want to take this into account. After entering the main page, I am shown the actual articles with the two links to go to the 'Previous Article' or the 'Next Article.' Therefore, I want to build a code which will crawl through all of these articles then access the 'Next Article' after getting the headlines. I used uReq from urllib.request and beautiful soup to parse the html. I got the 'href' attribute value from the main page and appended this to the end of the URL to access the first website where the articles are held.

My crawling method consists of using a while loop which runs on the condition that the current page is not the same as the last page visited. In the first 'if' statement block within the while loop, I access the first article and I then subsequently use beautiful soup to parse the article.

To obtain the headline for the article, I used inspect element and code to locate the headline location next to the "headline" attribute value. I subsequently append the headline and URL to their respective lists. After the headline is acquired, I go on to find the directory of the next link. The 'else' statement is where the crawling occurs and is similar to the 'if' statement block. The while loop runs over each article, gains the headline and then accesses the next link for the next article. The first URL in the 'if' statement is a stored value whereas in my 'else' statement, the subsequent URLs are matched against the first URL to make sure they do not match. When the URL for my next webpage is equal to the first URL of the article that I accessed, the while loop will stop. Lastly, I converted all my lists into a data frame and consequently, into a csv file format.

Output summary for task 1
The 'task1.csv' file has two column headings title 'Url' and 'Headline' respectively. In each row, there is a website url starting with http:// and subsequent headline title. Overall, there are 100 results. Here are the first 4 results to show the csv file as an example:

	Url	Headline
1	http://comp20008-jh.eng.unimelb.edu.au:9889/main/harapova100.html	Sharapova overcomes tough Molik
2	http://comp20008-jh.eng.unimelb.edu.au:9889/main/enmanove001.html	Henman overcomes rival Rusedski
3	http://comp20008-jh.eng.unimelb.edu.au:9889/main/afinslum002.html	Safin slumps to shock Dubai loss
4	http://comp20008-jh.eng.unimelb.edu.au:9889/main/erreroey003.html	Ferrero eyes return to top form

Summary of Task 2
For task 2a, I parsed the json file and extracted all the names into a list. I created two separate lists of first names and last names only, without middle names, and another list called 'shorternames' where both first names and last names are together. I then similarly implemented a while loop and accessed the articles with code using the method in task 1.

Entering the first article, I converted the article to all lower case letters. I then used regex, .findall(), to get rid of all the punctuation and to only include the word. To find whether the names in the article matched the ones in the json file, I used a for loop to iterate over each article and cross check

whether a name had both its first and last names within the article and whether the first and last names were in the 'shorternames' list. If these conditions were satisfied, the name was added to a list and the first element was indexed to get the first player. After the while loop ends, I created code where the middle names are added back in to create full names.

For part b of the task, I carried out everything exactly as before however, this time I do not remove the punctuation. This is so that I can find the valid match scores and I used regex to achieve this. The first bracketed part of the code serves to find match scores with two digits and a hyphen or slash (/). Then the second bracketed part of the code looks for scores that are tiebreakers. Both parts look for a space, full stop or comma after the score, are grouped and have a + sign after to find recurring matches with these patterns.

I then use a for loop to calculate whether the score found is a valid score. If the score is a tie breaker I ignore it and only take interest in the match scores. I used regex to match the first number on the left hand side of the score, '6-2' and then minus the score to the right hand side of the hyphen, '6-2'. To make sure that only valid scores were counted, I made sure that the minimum absolute difference between the scores is more than or equal to 2 and the maximum is difference is less than or equal to 6. This eliminates scores like, "7-5 2-2" and "7-5 6-2 0-30" to make sure that only valid match scores are counted. I also eliminated single scores such as "6-2" because it states in the instructions that a valid match score has to have a minimum of '2/3 or 3/5' sets won. Lastly, I removed data that did not have player names or a valid score. I converted the resulting data into a .csv file called "task2.csv."

Output summary for task 2

There are four column headings: URL, Headline, Player, Score. With the resulting website URL, headline, full player name and valid match score. Overall, I had 35 results from the results produced

	Url	Headline	Player	Score
1	http://comp20008-jh.eng.unimelb.edu.au:9889/main/enmanove001.html	Henman overcomes rival Rusedski	tim henman	4-6 7-6 (8-6) 6-4
2	http://comp20008-jh.eng.unimelb.edu.au:9889/main/afinslum002.html	Safin slumps to shock Dubai loss	marat safin	7-6 (7-2) 6-4
3	http://comp20008-jh.eng.unimelb.edu.au:9889/main/oddickin004.html	Roddick into San Jose final	andy roddick	7-6 (7-3) 6-3
4	http://comp20008-jh.eng.unimelb.edu.au:9889/main/oungdebu006.html	Young debut cut short by Ginepri	robby ginepri	6-2 6-2

Summary of Task 3

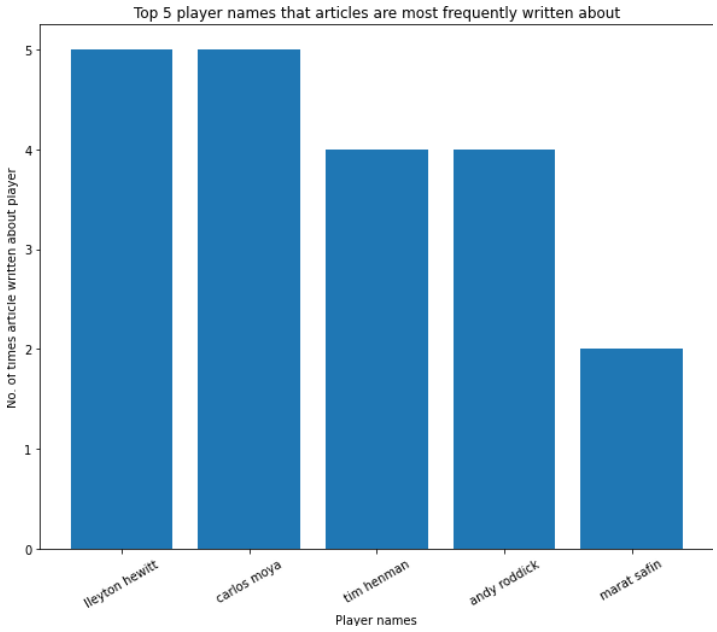
In task 3, my objective is to figure out the average game difference. To avoid redundancy, I have mentioned the method of obtaining the average game difference in my Task 4 and 5 analysis below.

Output summary of Task 3

The final output has the 'player' column and the 'avg_game_difference' column. For each row, it will list the tennis player with their average game difference calculated. In total, I had 18 rows of players. Here is the first 4 rows of the final result.

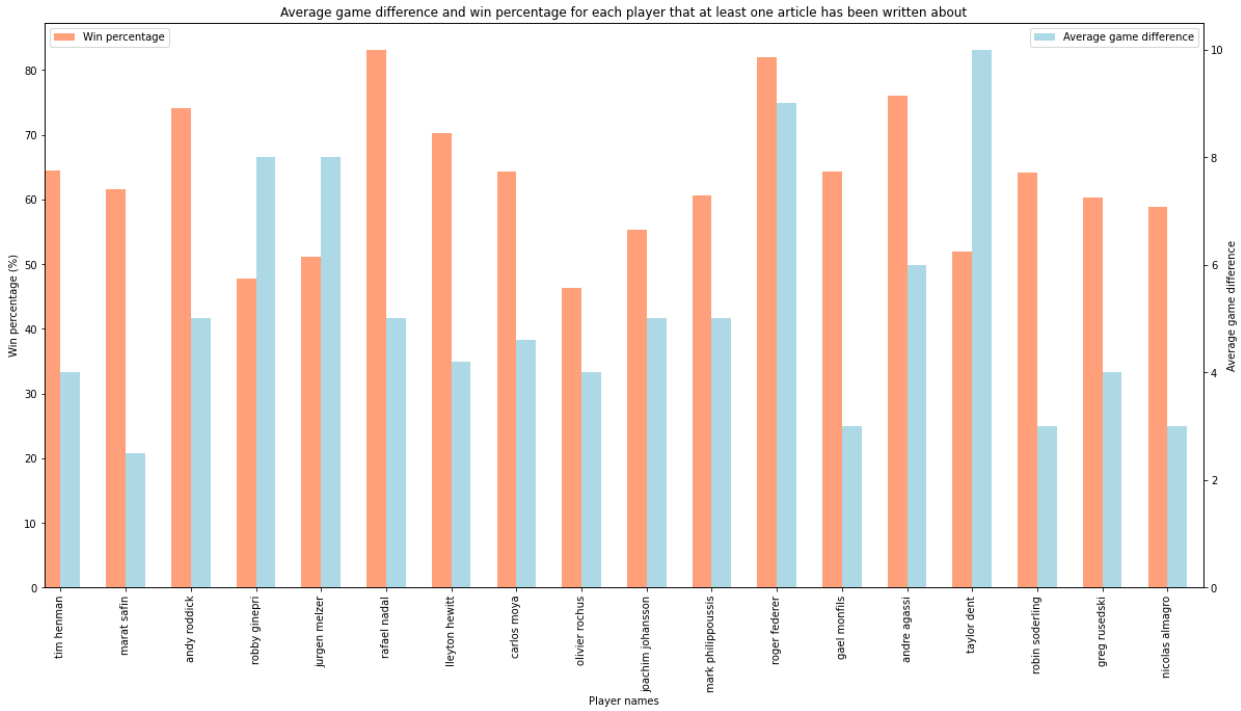
	player	avg_game_difference
1	tim henman	4.0
2	marat safin	2.5
3	andy roddick	5.0
4	robby ginepri	8.0

Task 4 and Task 5 analysis and summary of data used



The graph shows the Top 5 player names that the articles are most frequently written about. Lleyton Hewitt and Carlos Moya are both the most mentioned players, with 5 articles each. Tim Henman and Andy Roddick are second equal and are mentioned in 4 articles each. Marat Safin is 5th, with his name mentioned in 2 articles.

To get this data, I took the csv file 'task2.csv' and used code to count the number of times an individual player was mentioned in the column 'Player'. I stored this information in a dictionary format and collected the top 5 highest key, value pairs in the dictionary. I graphed this as a bar graph.



In the graph for task 5, win percentage is overall higher than average game difference for players. Average game difference varies among players and does not seem to be affected by win percentage or vice versa. Average game difference may be lower because we only calculated the scores based off a sample of articles whereas win percentage is based off all the games played in their career.

The data was in two components: win percentage and average game difference. To get the win percentage data for each player, I created a list of all players written at least once and parsed the json file. I then used a for loop to go through and get the win percentage of the player mentioned at least once within the list.

To get average game difference, I took all the match scores and firstly removed the tie breakers because it they don't count towards the average difference. I used regex to identify all the bracketed values and I used re.sub to substitute the tie breakers with blank spaces. I then utilised a for loop to

iterate through all the values down the column of the data frame and identify the difference of the number to the left of the hyphen and the right of the hyphen (using regex). I then summed all the game scores for each player and used the method .abs() to make it positive. Furthermore, I combined the list of players with the absolute value into a dictionary so that each player is only mentioned once but their absolute value is stored. To get the average, I used the mean method from the statistics library to average all the scores stored as values. Lastly I put this into the same graph with the win percentage by using the .twinx() method so that there is two axes.

Appropriateness of the method within this project

In my opinion, it is appropriate to associate the first named player with the first match score. By taking a look at the headlines, they will usually mention the relevant tennis player names first and then the opposing player. Furthermore, we can assume that when the relevant tennis player is associated with the first match score because it makes sense for the article to include a score referring to the player no matter if they won or lost. For example, the first article is "Sharapova overcomes tough Molik," and the subsequent paragraph with the first match score is "The second seed beat fourth-seeded Australian Alicia Molik 4-6 6-1 6-4." The score refers to Sharapova and therefore, we can assume that this pattern is carried forwards in all the other articles. Our project however, is probably not accurate because the json file containing tennis player names do not contain all tennis players but only a select few. There is also a small room for error as the name may not be mentioned in the headline. However, looking at the task1.csv, we see that most of the time it is so we can be confident in our method and assume that it is correct most of the time.

Suggested method to find if the first named player won or lost the match being reported on

There could be a dictionary of positively associated words to winning such as 'won', 'win', 'succeed' and a dictionary of negatively associated words to winning such as 'defeat', 'beaten', 'loss'. Of course some words can have multiple meanings for example. The word 'beat' can have a positive signal as it likely indicates that someone won against another tennis player. However, 'beaten' can have negative connotations because it likely refers to the player having lost. Therefore, it is important to categorise these words into their respective positive or negative categories. Then we can use a crawling method to go through all the articles counting up all the words that match positive and negatively. The regular expression ".findall()" will most likely be used for this. After this, we can then write a code to determine which dictionary had the most points for each category of positive vs. negative association. The articles with more positively associated words will print out 'win' whereas the articles with the more negatively associated words will print out 'loss' for the first player mentioned. I will make the assumption that the json file of names will have a more complete file of names so that it is able to capture more names.

Method to extract additional player information from articles which may indicate player performance

I believe that the use of multiple names and match scores mentioned in the articles could be extracted to better understand player performance. A player could be mentioned multiple times in the article with different winning or losing match scores and we can use coding to associate the scores to the player. In addition, other names of players are mentioned so we can take advantage of this and find out if there is a score that is associated with that name. Using regular expression, I will specifically search for all the paragraphs which contain a valid match score in it. Assuming I have a file containing most of the tennis player names, I will find the first player name mentioned in each paragraph. Secondly, I will calculate the match score within the paragraph and then associate that match score with the first player mentioned in that paragraph. This score can then be put into a dictionary of names and the average calculated in the end to get a better understanding of their performance. We can then finally make a graph comparing all their averages and hence, performance.