Jeanelle Abanto
1133815
JABANTO
COMP20008 Assignment 1

The main tasks for the project involve web crawling, scraping, data processing, data visualization and analysis given the website 'http://comp20008-jh.eng.unimelb.edu.au:9889/main/'.

To begin crawling the given website, first thing to do was to set the initial page to crawl, that is to set the initial page to 'http://comp20008-jh.eng.unimelb.edu.au:9889/main/index.html'. Then, parse this webpage as an HTML using the python library **BeautifulSoup**. Search for items tagged as 'a' to find links in the page. Create a list of links to visit and a separate list of visited links to avoid visiting the same link repeatedly. A page limit can be set to limit the number of pages to visit, but since this task involves only a hundred pages, the page limit was omitted.

Since the first task is to extract the headlines from each visited page, scraping for the headline can be inserted within the crawling implementation. In order to find all the links embedded in the current webpage, the page must be parsed as an HTML. To extract the headline, the page must also be parsed as an HTML. Thus, scraping for the class 'headline' can be done before visiting the next webpage. The URL of the current page can be used as the key for the extracted headline resulting to an object containing pairs of {url:headline}. Upon extracting the headline and appending links found to the list of links to visit, the current page can be marked as visited, i.e. it can be added to the list of visited links and removed from the list of links to visit.

Once all the links are visited, the list of links to visit will be empty and all links should be in the visited list. In addition, all headlines should be extracted by the end of the crawling implementation. A series can then be created containing a list of all the URLs visited and their corresponding headline. In this case, there are a hundred URLs found excluding the 'index.html'. This result can then be written to the csv file **task1.csv**. Since the initial page 'index.html' is not included in the analysis, it is not included in the output **task1.csv** file. As a result, a hundred URLs together with their corresponding headlines is written to **task1.csv**. And, in order to make use of the result of this task for succeeding tasks, the series created from this task is therefore used as a return value of the web crawling function.

The next task is to extract the first tennis player name and first complete tennis match score to appear on each article on each page. On the first task, a series of URLs and their corresponding headline was already created. So, for this task it is unnecessary to crawl from the initial page. Making use of the python library **BeautifulSoup**, each of the URLs on the list can be parsed as an HTML to extract the article body. For this case, all items tagged as 'p' were extracted as a whole paragraph.

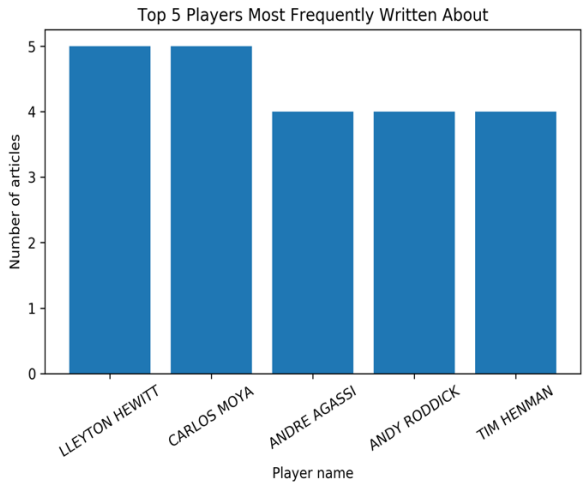
To minimize the search parameters, the extracted paragraph was broken down into sentences. Using the python library **nltk** and the method *sent_tokenize()*, the paragraph was converted into a list of sentences. And so, for each sentence, a search can be performed to find the first tennis player mentioned in the article. Since only those player names listed on the **tennis.json** file will be considered for analysis, the search can be done by finding the names from **tennis.json** file on each

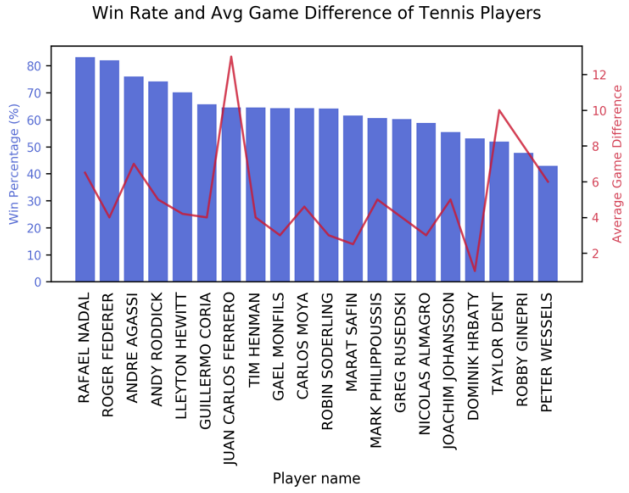
sentence. The python built-in method *find()* returns the position of the first character when a match is found, otherwise it returns a (-1) when a match is not found. Using this method, the name with the lowest position found, that is not (-1), was considered as the first tennis player name written in the article. To make sure that the search is not case-sensitive, all search parameters were converted to upper case during the search. However, it is assumed that full names are mentioned in the article, otherwise a match may not be found until a full name is mentioned. Thus, player names listed in the output will be the first full names found in **tennis.json** file mentioned in the articles. However, full names mentioned in the article but do not exist in the **tennis.json** file, like female player names, are ignored even if they are the first mentioned names, thus resulting to almost half of the articles being discarded. It will be assumed that players of interest are only the male players listed in the **tennis.json** file. Once a player name is found, there will be no need to check succeeding sentences in the paragraph. The URL of the current article is then used as the key for the extracted player name resulting to an object containing pairs of {url:player}.

In order to extract the first complete tennis match score from the articles, the following *regex* was used '(((\d+\-\d+)\s*)+\(\d+[\-/]\d+\)\s*)+((\d+\-\d+))*|(\d+\-\d+\s*){2,}'. This regex can be broken down into two, [(((\d+\-\d+)\s*)+\(\d+[\-/]\d+\)\s*)+((\d+\-\d+))*] and [(\d+\-\d+\s*){2,}]. The first part is used to extract score patterns with a tie-break game, i.e. scores in parentheses. There should be one or more scores without parentheses before a score in parentheses is allowed. After one or more of these score patterns, there could be zero or more scores without parentheses after. The second one is used to extract score patterns without the tie-break game, i.e. scores in parentheses. The {2,} is used to indicate that there should be at least 2 scores in order to be considered a match score. Using the python library **re** and the method *search()* score patterns were extracted. Then, similar to player name extraction, searching for scores was done on each sentence of the paragraph using the list of sentences generated by *sent_tokenize()*. A block of code for checking score validity was implemented wherein equal scores and score differences greater than 6 were assumed invalid. In addition, score results wherein a winner cannot be determined, was assumed incomplete, thus also considered invalid. Otherwise, scores were assumed to be valid and complete. As a result, scores such as [**7-5 2-2**] and [**6-1 3-1 0-30**] were treated invalid and were excluded from the list of extracted scores. Valid scores found are then extracted and stored as an object containing pairs of {url:score}.

The result of the two objects {url:player} and {url:score} can then be merged together with the series created from the first task giving a series containing the url, headline, player, and score. As a result of the merge, URLs without a recorded player name and/or without a recorded score were omitted. From the total of 100 URLs listed from the first task, after merging the three objects there were only 41 URLs listed with player names and valid tennis match scores written to **task2.csv** file. Of the 41 URLs, 20 players with valid tennis match scores were extracted as can be seen on the plot below. The result of the 41 URLs is based on the assumption that the articles do not have duplicates. For as long as the URLs are unique, it will be treated that contents of the articles are also unique. This is also the reason for setting the URLs as the key upon extraction of the player names and the match scores. So, for example the last two URLs listed on the **task2.csv**, although the contents of the article are the same, but the URL and headline are different, it is assumed to be different and thus both are included in the analysis of Carlos Moya's game performance. This is a limitation on the data processing done on this task since checking for duplicate articles was not performed. The assumption is that the articles written on each URL is unique.

Using the record containing the URLs, headlines, player names and scores, the average game difference of each player was computed by grouping the scores by player names, getting the absolute value of the game difference of each recorded match scores, then getting the mean value. The number of complete match scores recorded on each player can be assumed to be the number of times an article is written about the player. As shown in images below, the top 5 players most frequently written about all belong to the top 100 players as cross referenced with **tennis.json** file. But as seen on the plot to the right, among the twenty extracted players, these five players belong to the top 10 players with the highest win percentage. It is quite expected that seeded players will have more articles written about them. On average, most of the players have game differences ranging between 2~6 regardless of the win percentage. Although there were some with game differences outside the average including Juan Carlos Ferrero, Dominik Hrbaty, Taylor Dent, and Robby Ginepri. These players who seem to have average game differences 'beyond' 2~6 actually have only 1 complete match scores recorded when cross referenced with the output on **task2.csv**. But these are still included in the analysis since the average game difference was computed on the basis that 'at least one' article is written about them. Taylor Dent and Robby Ginepri had an overwhelming win, whereas Juan Carlos Ferrero had a crushing defeat resulting to higher average game difference for these players. Dominik Hrbaty on the other hand had a score that actually belongs to female tennis player Daniela Hantuchova who recovered from a terrible start. As previously mentioned, female player names were ignored since these names are not recorded in the **tennis.json** file which is a limitation on this analysis. Thus, even if the first mentioned score belongs to a female player, if a male player name is found, the score will be assumed to belong to the male player.





The data used as input to generate these graphs was the result of task2 and task3, i.e. the list of url with player names and complete match scores recorded, and the average game difference computed by grouping the recorded match scores by player names. It is assumed that the first complete match score is the only score corresponding to the first mentioned player in each article because there may also be other players mentioned with their corresponding scores in the article. For this task, only the first mentioned player and first complete match scores are of interest. And, the win percentage of each of the 20 players was extracted from **tennis.json** file by searching for items tagged as 'wonPct'.

The above analysis is based on the assumption that the first complete match scores belong to the first mentioned player in each of the articles. Although this may be true for most articles, but some of the first mentioned scores actually belong to another player. As in the case of Dominik Hrbaty, the score recorded belongs to a female player whose name is not found in the **tennis.json** file. Another example would be an article where Andy Roddick was the first mentioned player, but the first mentioned score actually belongs to Andre Agassi. It may seem as if the score may still belong to Andy Roddick assuming Andre Agassi was his opponent. But, Andy Rodick's match performance and score was mentioned a sentence after Andre Agassi's match result. Also, a match is played by at least two players. Assuming the articles are about singles match, the mentioned score should belong to two players. For example, in the article 'http://comp20008-jh.eng.unimelb.edu.au:9889/main/adalputs071.html' the match score [**6-7 (6/8) 6-2 7-6 (8/6) 6-2**] which was recorded as Rafael Nadal's score as he is the first mentioned player, should also be recorded as Andy Roddick's score being Rafael Nadal's opponent. Doing so will give more data for the computation of each player's average game difference. Associating the first named player with the first match score can be one way to start gathering and analyzing data but as can be seen in this report, it is not sufficient and as in the given examples above, it may not be accurate especially if there are player names (e.g. female names) being ignored during the search.

Going back on the assumption that the first mentioned score belongs to the first mentioned player, one way to find out if the player won or lost the match is by analyzing verbs or keywords used in the sentence where the name appeared and/or where the score appeared. Similar to the machine learning implemented on movie reviews, whether the reviews are positive or not, it can be used to check if the contents being reported is a win (positive) or a loss (negative). Articles containing words or phrases such as 'victory', 'win', 'lead', 'beat', 'defeated', 'to defeat', etc. can be used to learn that the player won the match. Otherwise, words such as 'loss', 'fall', 'losing', 'defeat', 'defeated by', etc. can be used to learn that the player lost the match. It can be given a training set of losing articles and a set of winning articles to learn to identify a win or a loss more accurately.

Other information that can be extracted is the number of sets won in a match, and the opponent of a player. If it is determined that the mentioned player won the match, it can be assumed that the first score in a pair, say [**6-1**] belongs to this player and the second score belongs to the opponent. If the player lost the match, the second score in a pair belongs to the player. Determining if a set is won or lost can be checked using the difference in scores. If the player won the match, a set is won if the difference is positive. However, if the player lost the match, a set is won if the difference is negative. Getting the total number of sets won for each player can be another way to analyze the win percentage of a player, i.e. the number of sets won over the total number of sets played. Doing this for both the player and the opponent gives the performance of both players assuming both player names exist on the **tennis.json** file. This also gives more information on the game difference of the extracted scores. If for example Roger Federer played against a non-seeded player, the game difference may be higher due to an overwhelming win, in contrast to playing against Rafael Nadal which is expected to be a close match, i.e. it may have a lower game difference.