Crawling

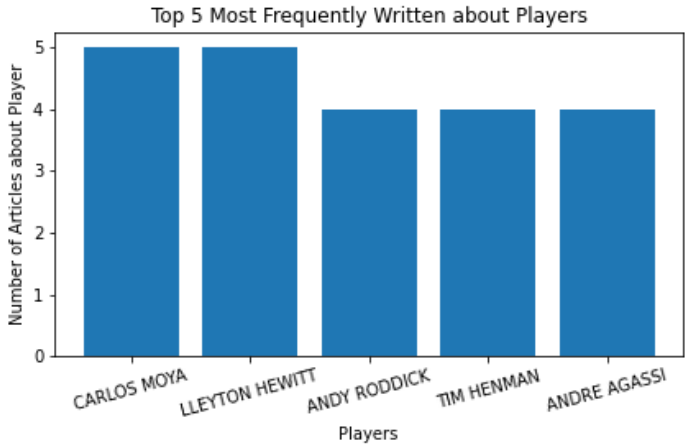
First, the base URL is requested using the “requests” library, e.g. page = requests.get(base_url). This page’s HTML text is then parsed with the library “BeautifulSoup”, e.g soup = BeautifulSoup(page.text,"html.parser"). Since the starting urls page text is now obtained, web crawling through the articles may begin through exploring the pages HTML tags. A continuous while loop is entered to find the first “<a>” tag on the page. Then the “href” attribute of this tag is combined with the base URL using "url join" function from the “urllib” library to get the URL of the next article, appending it to the task 1 dataset. Following the same process as the base URL, the libraries “requests” and “BeautifulSoup” are used for obtaining the html for this article and therefore find the first “<h1>" tag with the class "headline". The text of the tag gives us the headline and therefore is added to the list using the built in “append” function. This process is repeated for every article until we reach a repeated article. In order to prevent a “Crawler Trap”, meaning that the loop doesn’t continue forever and keep crawling the same pages over and over, a conditional statement is placed at the beginning of the loop which breaks out of the loop if the url of the next article is already in the task 1 dataset. When this occurs (all articles have been crawled), a data frame is created from the dataset using the python library “pandas” which is represented as a dictionary and turned into a csv, in this case a total of 100 articles were crawled and stored in the dataset. The csv file’s numbered indexes are removed, and two columns are in place, one containing the URL of every article obtained through web crawling called “url” and another one which contains the headlines of these articles named “headline”.

Scraping

The tennis data in the json file was obtained using the “json” library’s function “load”. Each players data is iterated through and the player names are appended to a list for later use. When appended to the list, the player names were split into their own respective lists in order to account for middle names. E.g "ROGER FEDERER" would be appended as [ROGER, FEDERER] and therefore this “nested list” would have a length of 2, indicating no middle name. Then the dataset of URL’s from the previous task are looped through and the text for each article is obtained using “BeautifulSoup” .The headline for that article is assigned to a string called "article_text" and then a loop is entered using the "findAll function which iterates through all <p> tags on the page, adding them to the article text string. The result of this is a string which contains all the text from that article which could possibly contain the first player name mentioned (in accordance with the names retrieved from the tennis.json file) as well finding possible complete match scores. We can use the “re ” library to apply this regular expression pattern “(((\d)+(-|/)(\d)+ ?)(\((\d)+(-|/)(\d)+\) ?)?){2,5}” to the text, searching for possible scores. The pattern is mainly split up into two groups, one that detects a normal set and another one to detect tiebreakers. The set is combined into one big group in which the pattern is matched if only the outer most group is repeated 2-5 times thus representing a match. The first group matches any two numbers (a least one digit) with a "-" or "/" between them followed by a white space which may occur (0 or 1 times) but because it’s possible that this may be the final set in the match and a "." or ", " may be present instead. The second group matches a similar pattern but looks for brackets encapsulating the set, therefore matching any possible tiebreakers. This second group may occur 0 or 1 times for the pattern to match because sets do not have to result in a tiebreaker. The result of the pattern will be assigned to a variable with "None" being assigned otherwise. It’s important to note that this regular expression has an approximate definition of a complete match and it is possible that incomplete match scores may be matched, but this is a good basis in which we can do further analysis later in the program. The text from the article is then split into a list of all uppercase letters in order to match for player names retrieved from the json file. A loop is entered which iterates over each word in the article and then a nested loop is entered, iterating over the list of player names. There are two conditionals in this loop, one that accounts for a middle name, and one that does not. The first if statement checks if the player has no middle name and attempts to match the current and next index of the article text with the player name. If the

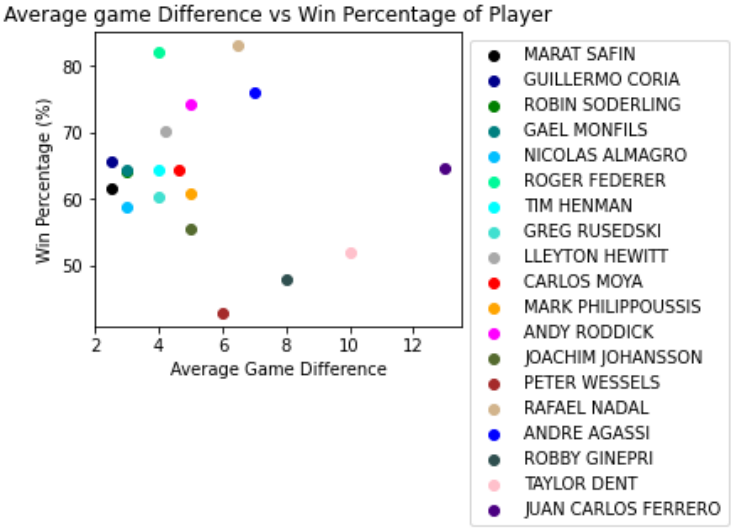
match is successful, then the first player name has been found and the loops are broken out of. The second conditional statement follows the same process but checks if the player has a middle name. When the player name has been found, we can do a proper analysis on the match score to test if it is actually complete. Since the actual game scores inside the tie breakers are trivial, regular expressions can be used to discard any scores inside brackets. Regular expressions "findall" method can be used to output all the numbers in the match score into a list. The “zip” function is applied to the list of numbers so we can iterate over tuples, with each one containing the first and second players game score for a set. The first conditional in the loop, detects that if the either one of the scores is "7" then the difference between them must be 1 or 2. The next one accounts for game scores that exceed 7, which must have an absolute game difference of 2. And lastly, one player must have won the set at 6 games and therefore the other player must have a score below 5. If none of conditionals are met, then it can be concluded that the match score is complete and therefore discarded. If the match is complete, then the articles URL, headline, first player mentioned, and complete match score is appended to the dataset. The dataset represented as a dictionary is converted to a pandas data frame and then into a csv file with headings; url, headline, player, score containing information about each articles URL, headline, first player mentioned and first complete match score respectively

Plot Analysis



The plot from task 4 exhibits the top 5 most frequently mentioned players, in accordance with the tennis.json file. A Dictionary is created, with player names as the keys and their frequency count as the values. A for loop is used to iterate through all the 41 player names obtained from each valid article in task 2. If a player does not yet have a key assigned to them in the dictionary a frequency value of 1 would be assigned, if they do then add 1 to the frequency value instead. The player names in the dictionary are then sorted in descending frequency values and the first 5 player names and their frequency count are plotted using the library “Matplotlib” to form a bar chart, with the player names on the x-axis and their number of times mentioned on the y-axis. A bar chart was chosen in order to convey the number of articles each player has written about them, there is no aim to find a correlation/trend between the two variables. The range of the frequency data is only 1, suggesting that the top 5 mentioned players have very similar values. There may be the possibility of articles which contain the exact same information but different URLs, possibly corrupting the data and misrepresenting how many articles may actually be written about a certain player. The most frequently mentioned players may be correlated with higher win percentages as the best performing

players tend to generally be the most popular among crowds hence more articles being written about their wins during tennis matches occurring between 2004-2005. It could be assumed that the first player mentioned is the winner of the first complete match score and therefore this player may have more wins but this is a generalization which cannot be taken for certain because the article may be about the losing player hence misrepresenting the data.



This scatter plot shows the average game difference (x-axis) and win percentage (y-axis) of each player that contained at least one valid complete match score. A scatter plot was chosen due to the two numerical pieces of data and one categorical. The win percentages for each player were retrieved from the data obtained in the tennis.json file through iteration. The win percentages in the file contained a “%” at the so the “strip” function was used, and the percentage was appended to the task 5 dataset. The average game difference was calculated using the task 3 data by grouping the articles with the same player names using the pandas functions “groupby” and “mean” to calculate the average game difference. Each point on the plot represents a player, color coded with a legend to the side displaying which player corresponds to what color. The plot displays a cluster in the lower range of average game difference (≤5) with majority of that data in this range. Players with a low average game difference suggest that they are consistently being matched against other players with similar skill level. Since the absolute value was taken, and the game differences weren’t signed at all (signaling a win or a loss) little correlation should be expected and it doesn’t make much sense to plot this against the win percentage. If instead the average game differences considered the win/loss aspect by applying negatives, it would be expected that some form of positive correlation would be seen in the graph. Another thing to note is that for many players, only one article was included and therefore making it hard to make any real assumptions about these players performance from the given data.

Appropriateness of the method for deriving the first named player

In general, associating the first complete match score with the first player mentioned in the article seems to be a reasonable and effective method. If the primary aim here is to simply determine if the first player mentioned is in the match of the complete match score, regardless of win or loss, than it can generally be assumed that the player is associated with the score. It would be unlikely that this would not be the case, considering that the general case is that the first player mentioned is early in the article and a match score would shortly follow. However there are certain instances where this may not be the case, in which maybe the article starts off talking about previous matches of a player and so happens to mention the name of another player (also in the tennis.json file) in a previous game, hence this method would prove ineffective. E.g "After … beat .. last week, … successfully rose

up and took down … scoring…" and so on. An article may be also written about the second player mentioned which would affect the accuracy of the number of times the player is mentioned but would not affect average game difference since the winner and loser both have the same absolute value. But from some basic eyeballing of the articles, the current and relatively simplistic method would appear effective. Overall, this method of pairing the first player mentioned with the first match score is a simple and competent method for obtaining player information. However rare cases may contradict this, possibly resulting in invalid data. More complex methods can be formed in order to counter this such as to possibly associate the complete match score with the player name that is closest to the score in the article instead

Suggestion of the method for determining win/loss of the first player

One possibility is to detect keywords that are indicative of a win/loss that are between the text of the first player mentioned and the second player mentioned. Approximate string searching can be implemented to detect for words such as "beat, dominated, defeated" which would suggest the first player mentioned won but it is also important to note how the words preceding and following these keywords may interact to suggest a win or loss. For example, "… defeated the swiss …"" or "… was defeated by the swiss…". They both tell us different things about the winning status of the first player, yet the word "defeated " is contained in the text between both the players. This can be accounted for by having separate collections of keywords and other words which may influence the meaning of the keywords in the context. A collection of words such as "by" or "was" and what position they should be in relative to certain keyword can be represented as dictionary with index's as the values. For example to indicate the first player won, there may be no other words between the keyword and the other player in the text, but if the first player lost, something such as a "by" might come between the keyword and the player name. This combination of keywords and surrounding words in the article text would prove to be an effective method to determine a win or loss for the first player

Suggestion of information extraction and processing method for understanding player performance

The average game difference may be recorded again but instead of taking the absolute value, it is possible to take the signed value instead. This would be useful only if it was known if the first player won the match or not, which could be figured out using the method described in the previous section detailing how keywords which are suggestive of a win/loss could be analyzed in sentences. Using this method to find the winning status of the first player mentioned, a positive or negative value can be assigned to the game difference and be recorded in the players dataset. A similar process can follow task 3 using pandas features, in which the mean is taken of all players recorded game differences to give a signed average game difference. If we search for the second player mentioned in the article and it is assumed that this person also relates to the first complete match score, the player that won will be assigned the positive value and the losing player, the negative value and thus since there is more data added per article, more accurate and better representation of player performance can be recorded. The signed average game difference is important because it not only presents a rough idea and how much they win or lose but also how much they win or lose by, and the magnitude of the average game difference suggests a great overview on the players general performance and can be correlated with other pieces of data such as win percentage.