HW#9 - Revisiting Web Archiving, Part 3

Ethan Landers

Due: Sunday, December 8, 2024 by 11:59 PM

Q1

To analyze the changes in TimeMaps since HW3, I began by duplicating the relevant files from my HW3 repository: analyze_mementos.py, download_timemaps.py, and utils.py. These scripts provided a foundation for processing and analyzing TimeMaps.

Next, I modified the download_timemaps.py script to save the newly downloaded TimeMaps to a separate directory to ensure the original HW3 TimeMaps were preserved and not overwritten. Using the modified script, I re-downloaded all the TimeMaps over the course of a day. The process was time-intensive due to the volume of queries sent to various memento repositories via MemGator running locally.

After obtaining the new TimeMaps, I created a script called calculate_difference.py. This script utilized functions from the HW3 code to compute the difference in the number of mementos between the old and new TimeMaps. For each TimeMap:

- A negative value indicates that the TimeMap has shrunk, with fewer mementos compared to HW3.
- A value of zero indicates no change in the number of mementos.
- A positive value indicates growth, with more mementos added since HW3.

Also within calculate_difference.py, I calculated the minimum difference as 0 and the maximum difference calculated as 298,918. This indicates that no TimeMap shrank (there were no negative differences), and all TimeMaps grew. Notably, one TimeMap grew by 298,918 mementos, marking it as an outlier in the dataset.

To visualize the results, I generated a boxplot using Seaborn's boxplot() function. This plot illustrates the distribution of differences in memento count across all the TimeMaps. The boxplot is presented in Figure 1.

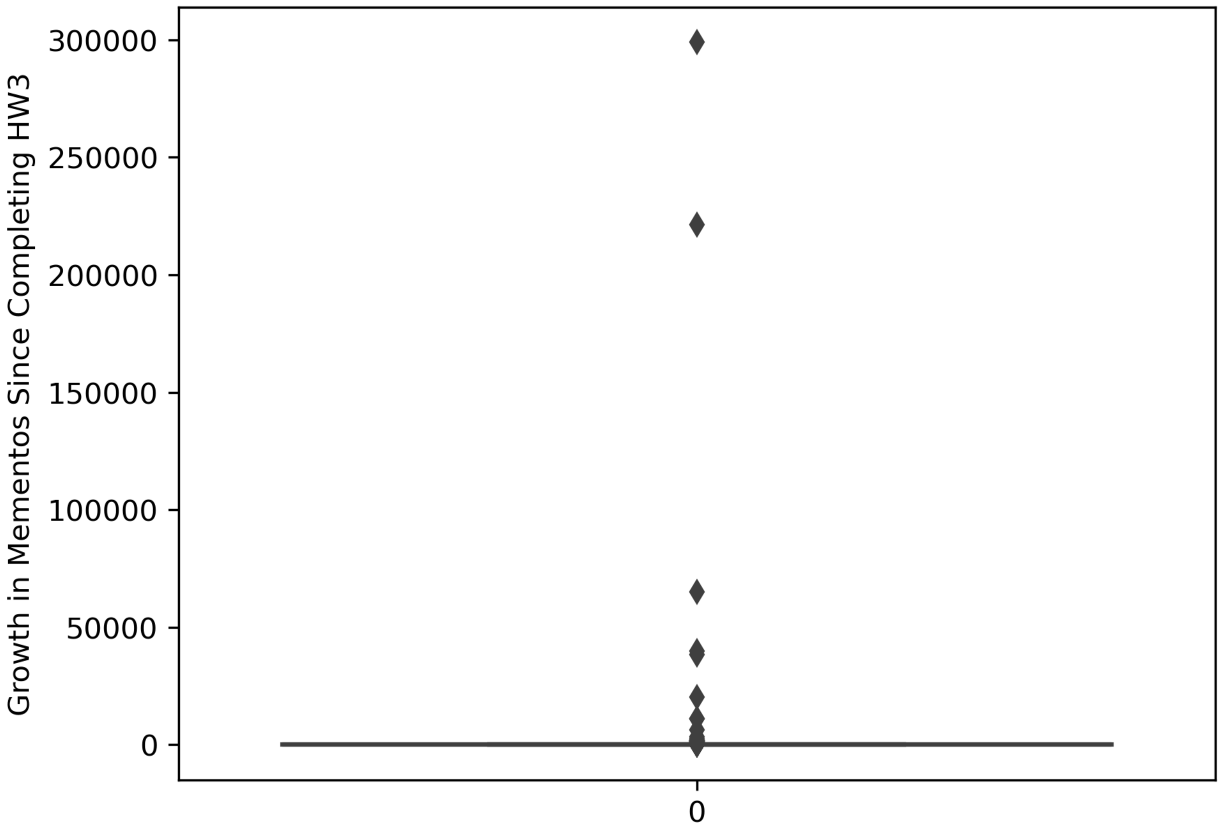

Figure 1: Boxplot of the TimeMap Memento Count Differences

Q2

To re-download the HTML for each webpage and remove the boilerplate, I started by duplicating the download-html.py and process-html.py from HW2. I then updated the paths in these scripts to ensure proper input and output directories were used. I ran each script to download the raw HTML and process it to remove the boilerplate. For consistency, I kept both sets of files (raw HTML and processed content) in separate folders as required and uploaded them to my class GitHub repository.

Q: Do all 500 URIs still return a "200 OK" as their final response (i.e., at the end of possible redirects)?

To verify the status of the URIs, I created a script named check_status_codes.py to check the HTTP response codes of each URI. The script takes as input the URI mapping file generated by download-html.py, which contains all the relevant URIs to HW2 and this assignment. The script uses a dictionary to store the status codes for each URI and a Counter collection to tally different response categories (200OK or non-200 OK). Depending on the status code received, the appropriate counter is incremented.

After processing, the script saves the status codes to a file named status_codes.json for use in later questions in HW9. Additionally, the script outputs a summary of the status codes to provide a quick overview of the results. Upon running the script, I found that 505 URIs returned 200 OK responses while 24 URIs returned non-200 OK responses.

Q3

I created a script named compare_webpage_sizes.py to compare the sizes of resulting text from HW2 and the current text. To achieve this, I first implemented a function called get_file_sizes.py to compute the sizes of all files in a given directory in bytes. I then supplied four directories to this function to retrieve the file sizes for both the raw and processed HTML files from HW2 and the current work. After calculating the differences in sizes for both raw and processed HTML sizes, I plotted the results as boxplots for visualization (refer to Figure 2).

Overall, there was a noticeable increase in text size between the completion of HW2 and the current work for both raw and processed HTMLs. While some shrinkage occurred in text size for both types, the growth was more significant. Notable outliers include one raw HTML that grew by 100,000 bytes and another that decreased by 75,000 bytes.

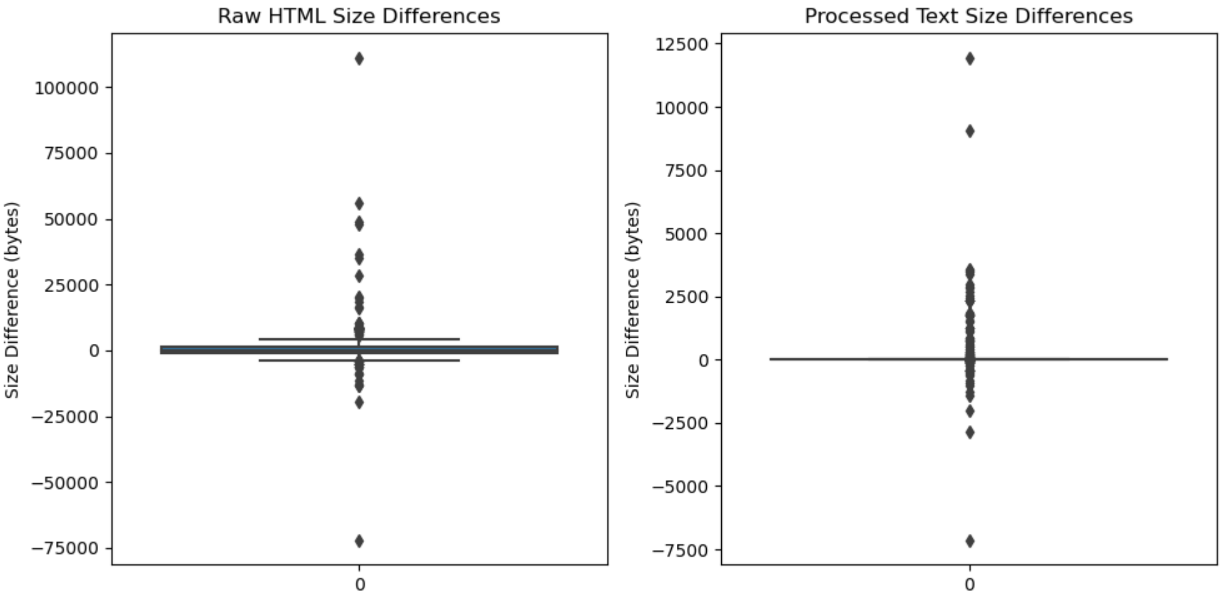

Figure 2: Raw and Processed HTML Text Size Differences

Q4

I created a script named explore_webpage_differences.py to analyze the top three URIs that exhibited the most significant changes. This script utilized the JSON file generated by check_status_codes.py as the source of the status codes for Q4. I began by filtering the URIs to include only those that returned a 200 OK response.

Next, the script iterated through these filtered URIs to compare the sizes of the processed files from HW2 and the current assignment. The size differences were calculated, sorted in descending order, and the top three URIs with the largest changes were selected. Below are the output statistics generated by the script:

```
Top 3 URIs with the most significant changes:
27e75055b4c3c78a42f7b280bfa64e92.html: 11964 bytes
b10c01549227c83825e3c82761084462.html: 9046 bytes
1787bbdde8554a5efd2c2d27d75fec61.html: 7134 bytes
```

In the same script, I used the difflib Python library to inspect line-by-line the differences between the top three URIs. The first URI corresponds to Old Dominion University, where the output shows outdated information was removed and replaced with new, relevant details about university operations. The second URI appears to belong to a music hall, where similar updates were made to reflect timely and relevant content. The final URI hosts abstracts of certain publications, and the differences suggest a new abstract was added to the page.

References

- difflib.unified_diff, `https://docs.python.org/3/library/difflib.html`
- Matplotlib 3.9.3 documentation, `https://matplotlib.org/stable/index.html`
- os.path, `https://docs.python.org/3/library/os.path.html`
- Python Counter Collection Documentation, `https://docs.python.org/3/library/collections.html#collections.Counter`
- Python Requests Library Documentation, `https://docs.python-requests.org/en/latest/user/quickstart/#response-status-codes`