Multimedia information retrieval: Homework assignment 2

Gideon Hanse, S1630784

February 22, 2019

1. Color based Similar Image Retrieval

(a) **Open the retrieval result ranklist.html in the Debug folder using a web browser and take a screenshot.**

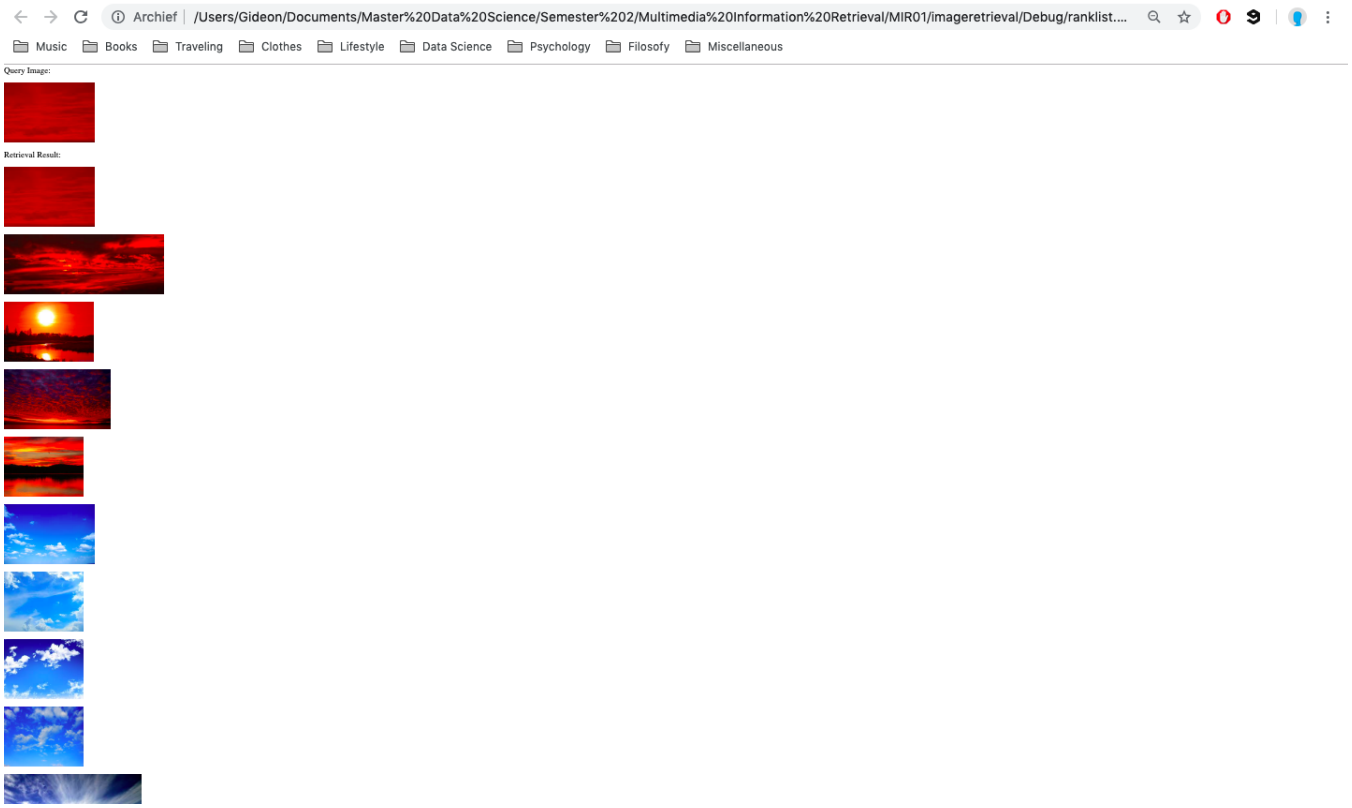
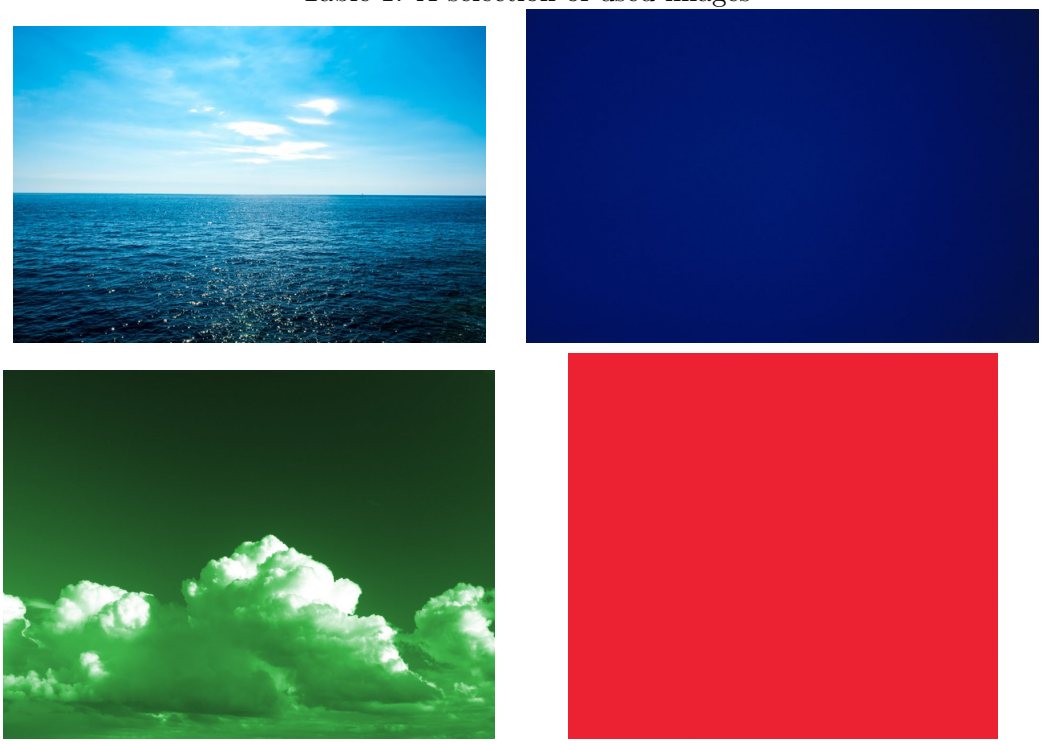


Figure 1: Ranklist screenshot

(b) **Read the file, maintest.cpp. Download some jpeg images (preferably small to medium size - less than 800x800 pixels) and try out some more color queries. Evaluate scientifically - describe when it works and does not?**

I downloaded several images consisting of different colors and properties. Among the images where red sunsets, blue skies, green landscapes and plain colors, such as the following:

Table 1: A selection of used images



I addition to using downloaded images, I modified some of the provided query images. For example, I changed the hue, saturation or brightness for some of the images to see to which extinct the image was still seen as similar, or seen as completely different.
By trying color queries in this way, I managed to get some interesting results:

i. In general, the main color of the image seems of the most importance in deciding if an image is similar to the query image. For example, a plain red image is determined to be more similar to a red sunset than to a blue sky. Also, a picture of the blue ocean is more similar to a blue sky than it is to a green sky or a green ocean.

ii. This is also shown by changing the hue of an image. In the cases where I only changed the hue of a picture, it is not recognized as the same image, even though the content of the images are exactly the same.

iii. Brightness does not seem to play an important role in determining similarity, which is explained by the fact that only bins for hue and saturation are created in the source code. In most cases of changing the brightness of a particular query image, it was still considered to be very similar.

iv. Changing the saturation of the picture has two effects. Namely, increasing saturation does not change the similarity that much. However, decreasing the saturation of an image drastically decreases the similarity of the query image and the changed image.

v. Images that have both a different saturation and hue than the query image are considered to be completely different, even though the content may be the same.

vi. Images like rainbows or color spectra are generally ranked somewhere in the middle with regards to the similarity with a query image, because they almost always contain some of the colors in the query image, as well as many other colors.

2. **Downloading and Parsing Weblinks**

(a) **Write two functions based on the tutorial C code which takes as input a web URL, downloads the webpage, and outputs the list of weblinks.**
I managed to implement two separate functions, one that downloads a webpage and returns a string containing the html source code of the webpage.
The other function retrieves links from this webpage and outputs them into a string. It

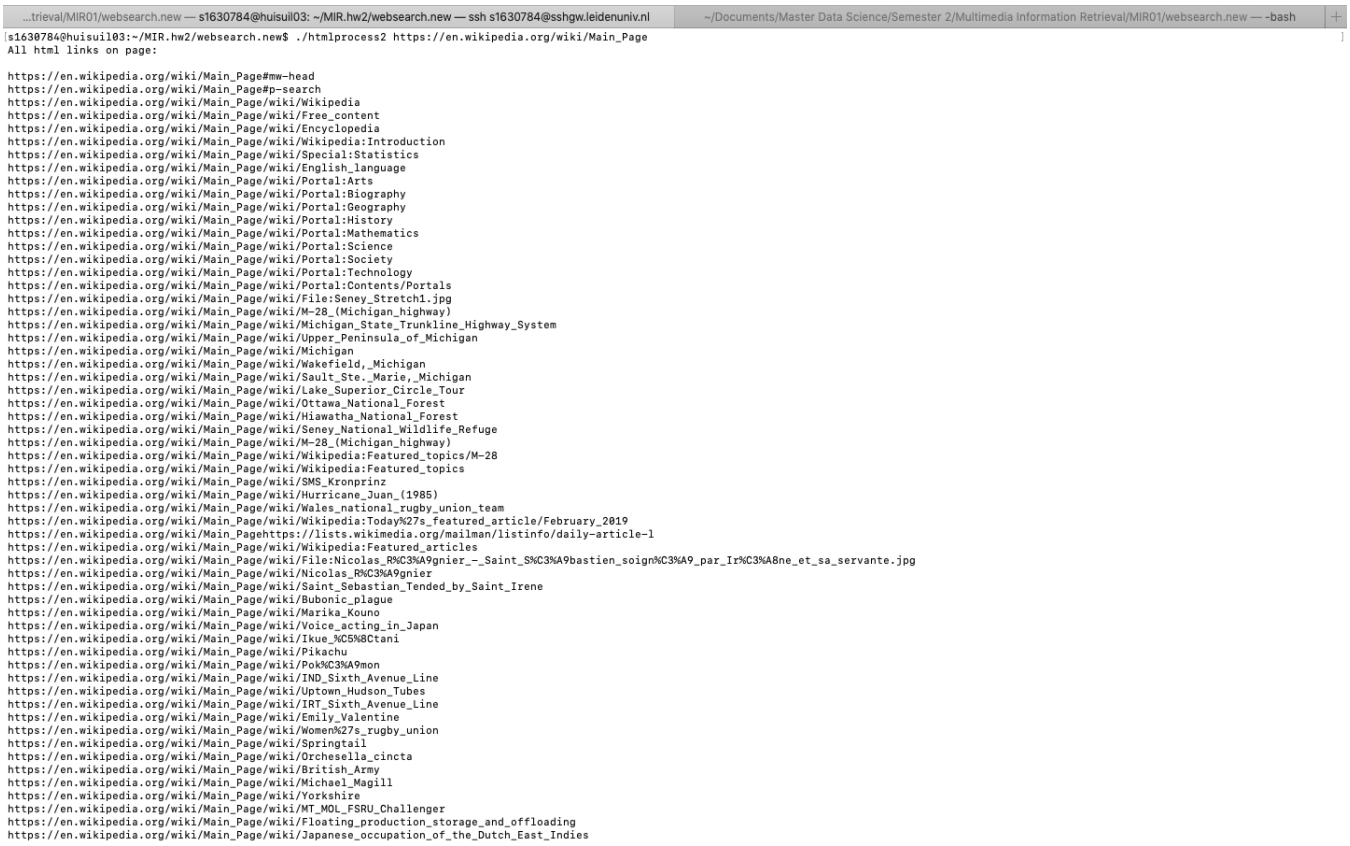


Figure 2: Returned list of links from Wikipedia

works very well in most of the cases. However, I must have made a small mistake in memory allocation, since I get a segmentation fault in some cases, particularly with big webpages.

(b) **Parsing Image Links**
The next goal was to retrieve only links of images. In order to do this I could slightly modify my Attribute function to make the parser look for img links instead of href links. The code I wrote works well, but in this case I also get segmentation faults for some webpages. I think this is caused by the same problem as in assignment 2a, and I'll be working on fixing this memory problem. Unfortunately, I did not manage to repair the fault before this assignment's deadline.

```
s1630784@huisuil03:~/MIR.hw2/websearch.new$ ./htmlprocess2 https://en.wikipedia.org/wiki/Main_Page
All image links on page:

https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/4/43/Seney_Stretch1.jpg/195px-Seney_Stretch1.jpg
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/3/34/Nicolas_R%C3%A9gnier_-_Saint_S%C3%A9bastien_soign%C3%A9_par_Ir%C3%A8ne_et_sa_servante.jpg/155px-Nicolas_R%C3%A9gnier_-_Saint_S%C3%A9bastien_soign%C3%A9_par_Ir%C3%A8ne_et_sa_servante.jpg
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/a/a0/Karl_Lagerfeld_2014.jpg/120px-Karl_Lagerfeld_2014.jpg
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/8/8b/Robert2_of_Scotland.jpg/100px-Robert2_of_Scotland.jpg
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/7/71/Albert_Reiss_LOC_ggbain-25651.jpg/280px-Albert_Reiss_LOC_ggbain-25651.jpg
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/en/thumb/4/4a/Commons-logo.svg/31px-Commons-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/3/3d/Mediawiki-logo.png/35px-Mediawiki-logo.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/7/75/Wikimedia_Community_Logo.svg/35px-Wikimedia_Community_Logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikibooks-logo.svg/35px-Wikibooks-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/f/ff/Wikidata-logo.svg/47px-Wikidata-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/2/24/Wikinews-logo.svg/51px-Wikinews-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/f/fa/Wikiquote-logo.svg/35px-Wikiquote-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/4/4c/Wikisource-logo.svg/35px-Wikisource-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/d/df/Wikispecies-logo.svg/35px-Wikispecies-logo.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/0/0b/Wikiversity_logo_2017.svg/41px-Wikiversity_logo_2017.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Wikivoyage-Logo-v3-icon.svg/35px-Wikivoyage-Logo-v3-icon.svg.png
https://en.wikipedia.org/wiki/Main_Page//upload.wikimedia.org/wikipedia/en/thumb/0/06/Wiktionary-logo-v2.svg/35px-Wiktionary-logo-v2.svg.png
https://en.wikipedia.org/wiki/Main_Page//en.wikipedia.org/wiki/Special:CentralAutoLogin/start?type=1x1
https://en.wikipedia.org/wiki/Main_Page/static/images/wikimedia-button.png
https://en.wikipedia.org/wiki/Main_Page/static/images/poweredby_mediawiki_88x31.png
Segmentation fault (core dumped)
s1630784@huisuil03:~/MIR.hw2/websearch.new$
```

Figure 3: Returned list of image links from Wikipedia

3. HTML parser shootout

(a) **Download using either your own source code or a browser 10 html webpages**
I downloaded the ten webpages to local html files using a browser.

(b) **Write code to load the webpages into RAM memory and parse each of them X times.**
Since my own code still has a memory allocation problem, I decided to alter the getlinks.c example file. I wrote code to load all of the webpages into main memory subsequently to parse them 1000 times.

(c) **For a different parser do the same process as in (b) with the same X and compare the speeds.**
As for a different parser, I used the Beautiful Soup library in Python. In the code I provided in addition to this report can be seen that I simply looped over the different html files and extracted the weblinks out of them. // In order to be able to compare the C haut parser to the Python Beautiful Soup parser, I conducted a small experiment. I let both parsers extract the weblinks out of the ten provided html pages 1000 times. This gave me the following results:

Parser	C Haut	Python Beautiful Soup
Elapsed time (s)	47.589	140.703

Table 2: Time performances averaged over 5 runs

Based on the time it took to parse every webpage a 1000 times, we could conclude that the C haut parser performs much better, as it is about three times as fast as the Beautiful Soup parser. The performance of link retrieval is exactly the same for both parsers, as they found the exact same amount of links in the same order for all ten sites. As a conclusion, it can be stated that the two parsers have a similar retrieval performance. However, since the C haut parser is so much faster than the beautiful soup parser it would be fair to say that the haut parser triumphs in this shootout.