Search Engine Analysis
Team #2 - Semester

***Abstract*—This document shows our analysis of the performance of our Search Engine and how we performed those analysis.**

I. The Experiment

A. Running One Experiment

This command runs the whole server in analysis mode and closes it after finishing the experiment.

```
$ env PAM=1 mvn
```

B. PerfAnalyser

`PerfAnalyser.java` is the java class responsible for conducting one experiment and closing the server afterward. It does the following:

1) Launches `$TOTAL_THREADS` of threads, each calls `Query Processor` with a random query.
2) After timeout of `$TIMEOUT_MS`, `PerfAnalyser.java` interrupts threads that didn't finish, then collects the time of the rest of the threads.
3) Calculates the average time of all threads, and calculates the number of timeouted threads.
4) Repeats this experiment one time again with the ranker disabled.
5) Queries the size of all crawled documents and the number of indexed keywords.
6) Serializes all the collected data into json file whose name follows the pattern `{performance-analysis-${TIME}.json}` and saves it into current working directory.

C. Repeating

You need to conduct this experiment multiple times during different stages of search engine running. Then plot the results to be able to answer the performance questions.

To plot the results with the python script:

```
$ python3 plot.py $PWD perf*.json
```

II. Results

Setting `$TOTAL_THREADS` = 200, `$TIMEOUT_MS` = 2 Minutes and running `PerfAnalyser.java` 10 times at different stages of database building. We got the figures 1 and 2.

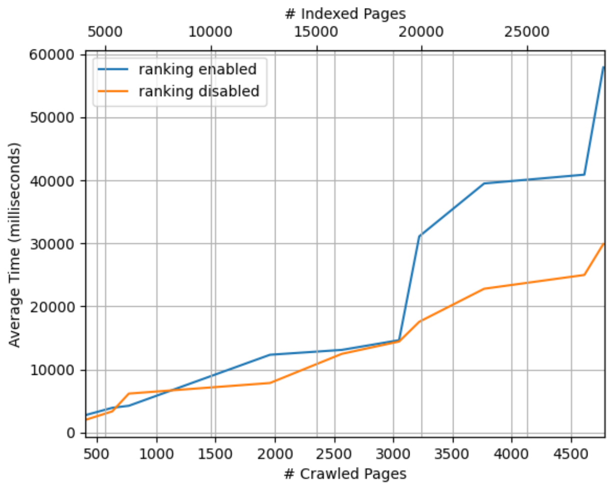

Fig. 1: Average Time vs. Num. Crawled Pages and Indexed keywords

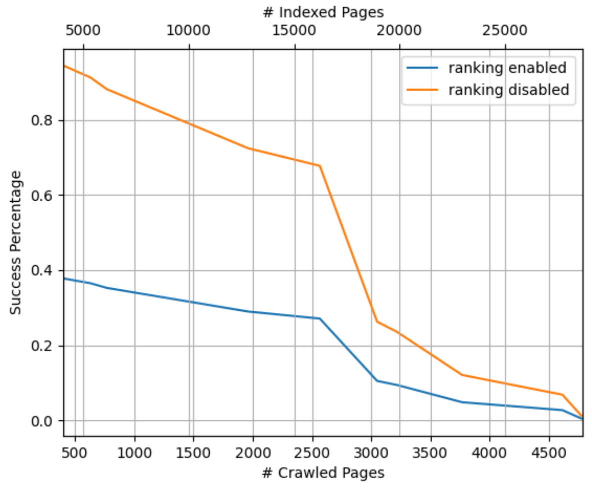

Fig. 2: Success Precentage vs. Num. Crawled Pages and Indexed keywords

III. Conclusion

We notice that with the increase of indexed words and fetched documents, the performance slows down significantly, specially with ranking enabled.