Information retrieval project - 2024

ID : Ariel Siman Tov – 209499821

Tal Klein - 209234103

Emails: tak@post.bgu.ac.il

arielsi@post.bgu.ac.il

link to Github repo:

https://github.com/TalKleinBgu/IR_Project

link to bucket :

https://console.cloud.google.com/storage/browser/209234103_final

The key experiments we ran on our IR engine were as follows:

1. The first implementation of our engine included: inverted index on title + stemming, Inverted index on body + stemming (stemming on the queries as well) and using ranking method of Cosine similarity. We have defined a final ranking function, which set different weight for the title, the body, and the PageRank. **This implementation led us to relatively good results: AVG_Time: 9.7 sec | AVG_Hermonic_Score (harmonic mean of Precision@5 F1@30): 0.0334333 | AVG_Precision@10: 0.063333**
2. In attempt to improve our IR engine, we tried to change our ranking method to BM25 by the equation (which we saw in relevant articles): $$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

   Which includes this IDF equation: $$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

   This implementation led as to better results: AVG_Time: 9.8 | AVG_Hermonic_Score: 0.53016666 | AVG_Precision@10: 0.71666

3. At this point, we wanted to try to improve out **AVG_Time** so we decided to change several things in our implementation:
   a) **Use treads:** we implement to layers of tread:
      - **External treads:** Using two threads, we performed a separate parallel calculation for the similarity between the title and the query and for the similarity between the body and the query.
      - **Inner treads:** we set a tread to every word in the query. (example - for query with 3 words we used 3 treads).

   b) **Use Heap data structure:** To enhance the efficiency of our candidate ranking process, we have decided to use Heap data structure in our implementation. Additionally, we have imposed a limitation on the size of heap pops, restricting them to 7500 for optimal performance.

   c) **Use "Impact Ordering":** We Set thresholds in two levels:
      - **Term frequency in Body > 3** - We wanted to neglect the effect of terms that correlate with the query with a small frequency on the ranking, in order to optimize the ranking function and shorten the running time (because we need to go through fewer terms).

- **<u>Dictionary size < 50,000 documents</u>** - In order to minimize the running time. This is possible because we have sorted the IDF and TF of the query so that only the documents with the best relevance potential are considered. So we "see" only 50,000 Documents.

This implementation led as to better results: AVG_Time: 1.323 | AVG_Hermonic_Score: 0.401933 | AVG_Precision@10: 0.57

4. We tried to add NLP model such as Word2Vec to our implementation in order to improve our matches by doing query expansion. We add to the query the 4 most similar words. We tried to use Glove-wiki-gigaword-300[1] library, which already train Wikipedia corpus as Word2Vec model. Unfortunately, this attempt did not improve the results of our IR engine in every aspect (Time & Accuracy): **AVG_Time: 2.20624 | AVG_Hermonic_Score: 0.1929666 | AVG_Precision@10: 0.276666**
5. We tried to remove the stemming from our implementation because we saw in the lectures that this method improves Recall rate and reduce the precision rate. In our first attempt, we tried version with no stemming (or prepossessing) - this attempt did not improve the results of our IR engine in total. Subsequently, we try lemmatization as an alternative to stemming and observed that this approach led to more favorable results:

   <u>Lemmatization results:</u> AVG_Time: 1.0945 | AVG_Hermonic_Score: 0.3944999 | AVG_Precision@10: 0.553333

   <u>No stemming and no lemmatization results:</u> AVG_Time: 0.9793 | AVG_Hermonic_Score: 0.3855 | AVG_Precision@10: 0.5166
6. In our final attempt, we tried to enhance the **performance rate** of our IR engine by implementing targeted optimizations such as (**<u>with Lemmatization</u>**):
   a) Adding Pageview to our final ranking function.
   b) Improve our Tokenization: In cases where a number appears in the query, we have added the transcription of that number to the query in order to match more results.
   c) In the " Impact Ordering" we increase the Dictionary size to 150,000 to get better results.

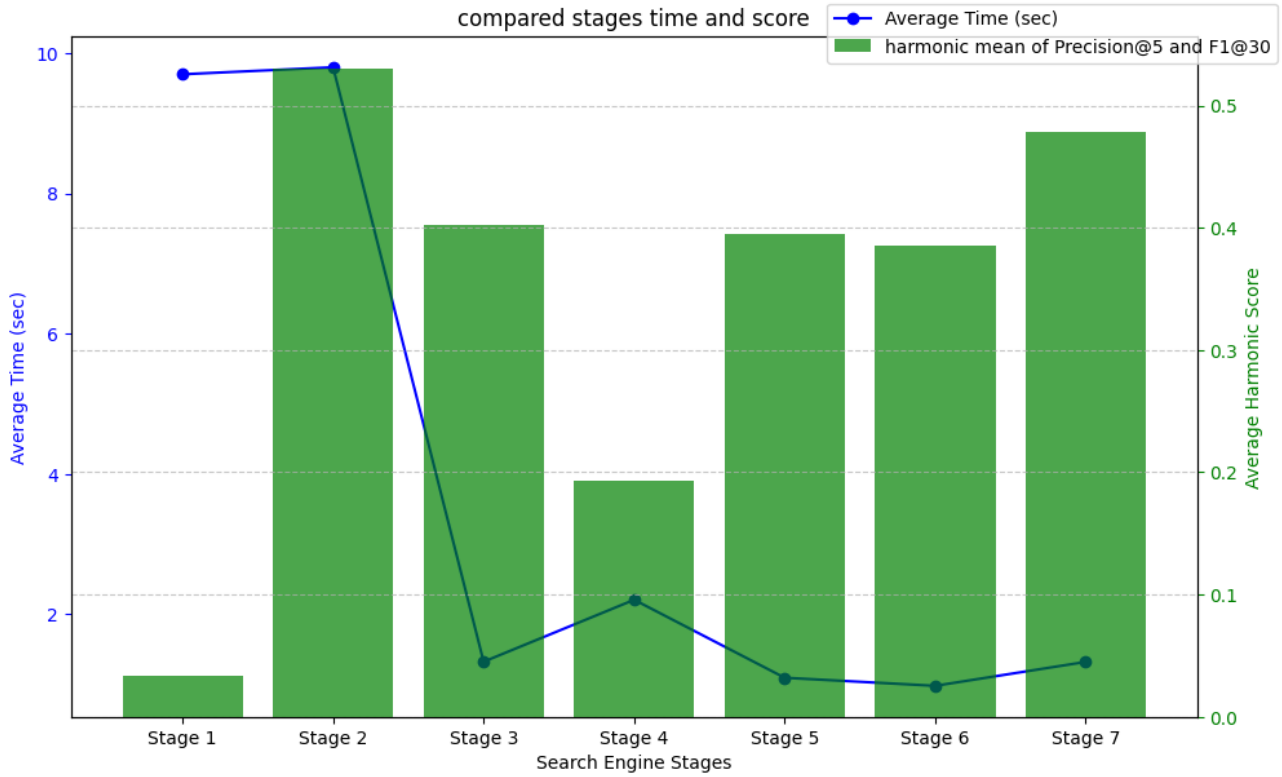
      This led us to the best results of: AVG_Time: 1.32065 | AVG_Hermonic_Score: 0.4781333 | Average Precision@10: 0.646666

<u>Weights of our final implementation after testing:</u>

$$12 * BM25_{body} + 4 * BM25_{Title} + 2 * log_2 PageRank + 4 * log_2 PageViews$$

[1] https://huggingface.co/fse/glove-wiki-gigaword-300

Graph of engine performance and retrieval time of each major step in our implementation



Takeaways and key findings:

- The choice of a ranking method significantly impacts our engine's performance. Adopting the BM25 ranking method consistently improves our results.
- Optimization strategies involving multi-threading (In several stages), suitable data structure (heap in our case) and ordering the candidate, contributed to enhanced efficiency.
- NLP models, such as Word2Vec, has a big potential to improve performance. Training a model on static corpus (such as Wikipedia), and using method like Doc2Vec, may improve our IR engine performance. Unfortunately, our Word2Vec implementation did not yield substantial improvements to the results, and we didn't succeed to implement Doc2Vec properly.
- The decision to remove stemming and lemmatization in certain cases resulted in improved performance in our understanding for two reasons:
  a. Time complexity - Retrieving our queries will be faster without stemming or lemmatization.
  b. Evaluate method which lay mostly on precision - we learn in the lecture that stemming improves Recall rate and reduce the precision rate.
- We believe that our IR engine's performance could be enhanced by implementing **Biword-Indexes.** This approach would particularly benefit queries that involve sequences of words with contextual significance, such as "World War II". Unfortunately, we did not have time to try implement this method.

Qualitative evaluation of the top 10 results:

Bad Query: "When did World War II end?" | Time – 3.595899 | Hermonic_Score - 0.154

Our engine faced challenges with a specific query due to our tokenization process, which excluded words with fewer than 2 letters. Consequently, the term "II" was omitted from the query, leading to inaccuracies as the engine referred to World War in a general context, not specifically to World War II. Moreover, the extensive relevance of documents on this topic resulted in prolonged query execution times. In an attempt to address this issue, we adjusted our tokenization to include 2-letter words but observed no significant improvement. This adjustment, in fact, led to a reduction in the overall performance of our engine, likely due to an increased volume of 2-letter words in version II titles and bodies. To solve the problem, we can consider the phrase "World War II" as one word in the inverted index phrase or try to use more powerful NLP models.

Our result:

```
1) Document id : 28514827, Title:The war to end war
2) Document id : 4764461, Title:World War I
3) Document id : 32927, Title:World War II
4) Document id : 360422, Title:End of World War II in Europe
5) Document id : 2198844, Title:Allies of World War II
6) Document id : 32611, Title:Vietnam War
7) Document id : 7062377, Title:Aftermath of World War II
8) Document id : 65307, Title:World War III
9) Document id : 16772, Title:Korean War
10) Document id : 34059, Title:War of 1812
```

Good Query: "Computer" | Time – 0.3243 | Hermonic_Score - 0.754

This query performs well for several reasons:

- Short and Common Word: "Computer" is a concise and frequently used term, making it easier to match within documents and resulting in faster query execution times.
- High Frequency of Occurrence: Given the widespread use of the word "Computer" across various contexts, there is a higher likelihood of relevant documents containing this term. This contributes to a higher Harmonic Score, indicating the query's effectiveness.
- General Relevance and Non-Hidden-Semantic meaning: This versatility stems from the non-hidden semantic meaning of "Computer," which is inherently clear and widely understood, facilitating its effective utilization as a query term without ambiguity or domain-specific connotations.

Our result:

```
1) Document id : 7878457, Title:Computer
2) Document id : 5323, Title:Computer science
3) Document id : 21808348, Title:Computer hardware
4) Document id : 18457137, Title:Personal computer
5) Document id : 5783, Title:Computer program
6) Document id : 50408, Title:Computer engineering
7) Document id : 18567210, Title:Computer graphics
8) Document id : 25652303, Title:Computer architecture
9) Document id : 10175073, Title:3D computer graphics
10) Document id : 328784, Title:Computer scientist
```

List all index files:

```
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body
5.54 GiB     gs://209234103_final/Body
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body_BM25
5.7 GiB      gs://209234103_final/Body_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body_lemm
5.79 GiB     gs://209234103_final/Body_lemm
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body_lemm_BM25
5.95 GiB     gs://209234103_final/Body_lemm_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body_no_stem
6 GiB        gs://209234103_final/Body_no_stem
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Body_no_stem_BM25
6.16 GiB     gs://209234103_final/Body_no_stem_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title
250.33 MiB   gs://209234103_final/Title
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title_BM25
488.14 MiB   gs://209234103_final/Title_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title_lemm
333.59 MiB   gs://209234103_final/Title_lemm
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title_lemm_BM25
496.37 MiB   gs://209234103_final/Title_lemm_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title_no_stem
410.78 MiB   gs://209234103_final/Title_no_stem
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Title_no_stem_BM25
497.49 MiB   gs://209234103_final/Title_no_stem_BM25
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/PageRank
84.69 MiB    gs://209234103_final/PageRank
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/PageViews
73.5 MiB     gs://209234103_final/PageViews
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/Word2Vec_Model
0 B          gs://209234103_final/Word2Vec_Model
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final/doc_to_title_dict
168.88 MiB   gs://209234103_final/doc_to_title_dict
tak@cloudshell:~ (boreal-mode-413417)$ gsutil du -sh gs://209234103_final
52.63 GiB    gs://209234103_final
tak@cloudshell:~ (boreal-mode-413417)$
```