Poor queries:

- **master of software engineering** (runtime performance): There are two many words which the time taken to finish this query is too long.
  Technique used to improve runtime performance:
  1. Pre-calculated the tf-idf value and normalized it.
  2. Find intersection of result posting by sorting it first
  3. Use index of index to go to the file quicker.
- Computer engineering (ranking performance): Some website mentions this word many times. But not include in the title.
  Technique used to improve runtime performance:
  1. Treat the important words, titles and headings more important than other words.
- **Dumping the hybrid tree from memory to disk** (ranking performance): There are duplicated webpages in the library and both of them shows up in the result list, which does not make sense. Also the terms is less specific that the term "the" and "from" might have negative influence on the result.
  Technique used to improve ranking performance:
  1. Remove duplicate pages from the library with near similarity.
  2. Use cosine similarity instead of plain tf-idf for ranking, making sure that the result is relevant without greatly influenced by popular terms.
- **aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa** (runtime performance): The term does not exist in the index table, but the searcher needs to spend a long time to find out.
  Technique used to improve runtime performance:
  1. Check the Index of Index for both char_position and the next char_position, that as soon as the iterator step over to the next char_position, it knows that the term does not exist without iterating through the rest of the index table.
- **sciencecomp uter scie ncecomputer science computer sciencecompu ter** (runtime performance): The query includes terms never showed up in any pages, nor exist in the index table, which will result in no result regardless. 0.5s at the beginning, then it has been reduced to 0.25s.
  Technique used to improve runtime performance:
  1. Remove the non-existing terms from the query, preventing the dictionary key error and unnecessary memory spend.
- **zz** (runtime performance): The query includes terms that does not exist in index table, and also does not match any key in IndexOfIndex, which will spend great time on searching.
  Technique used to improve runtime performance:
  1. Additionally check the condition when the key in IndexOfIndex is not matched, if it is the case then it must not exist in the index table. Remove it from the query to ensure the other terms are not affected.
- **University of California** (ranking performance): The query is so popular that most of the result have the similar score, which does not really help our searching rank.
  Technique used to improve ranking performance:

1. Use additional source of scoring for ranking purpose, e.g., title.
2. Eliminate similar pages from the library, that can eliminate many subpages from the same main page.

- **page** (ranking performance): The query is also so popular because of its multiple meanings, and greatly used in some other general cases like for the page number.
  Technique used to improve ranking performance:
  1. We use tf-idf instead of term frequency to ensure that it is not a disturbance text from the page number but the actual focus.
- **walking walked walker walks** (ranking performance): The query contains variant of words, that most likely should be consider as one term.
  Technique used to improve ranking performance:
  1. Use stemming on both the indexer and the query, to make sure the variant of words do not influence the result for anything like English tense or plural.
- **a aa b bb c cc d dd computer science course** (runtime performance): The query contains a lot of popular terms with huge meaningless result list first, and the term with smaller result stays at the end, which greatly damage the runtime performance.
  Technique used to improve runtime performance:
  1. Sort by length before the set intersection among each terms, and stop the process early once it determines no result is in the intersection.

Good queries:

- Computer science
- Python
- Intelligent Artifact
- ACM
- **Machine learning**
- 74743581845224e
- eugenebypass
- unity3d
- unreal
- cristina lopes

Why we choose them: the good queries have small number of terms, that the runtime speed is not greatly impacted by the query length. Some of them appear on the title frequently, which helps the evaluation and give the most relevant result. The results are more relevant since the terms used are specific and do not have multiple meanings, which ensured the ranking performance.