School of Information Management
C Laboratory
Winter 2018

Final Project

created by
Dominik Wurzer November 22, 2018

1 Introduction

You are hired as an Engineer by HackIT - the infamous Wuhan IT start-up which recently secured their series B funding. HackIT is developing a new search engine which they want to incorporate into their information management and data mining products. In particular, HackIT's project focuses on searching financial news articles for relevant information. The product will be of interest to financial analysts, investment bankers and journalists. You are assigned to the engineering team that is supposed to build the first prototype of the core search technology. The design of the search engine development is in its early stage but HackIT's technology division already outlined its basic functionality (see pseudo code in Algorithm 1). Your task is to implement the basic core algorithm and extend it with advanced features.

Your tasks includes:

- **Implementation of the core search engine**
  The search Engine is a mission critical core technology of HackIT. Serving millions of concurrent user searches requires high performance - fast program execution. HackIT's CTO Dr. Dom requires you to implement the entire project in plain C without relying on 3rd party libraries.
- **Enhancing the core search engine with advanced search features**
  In order to market HackIT's search technology as high-tech, you are required to extend the core algorithm with advanced features.
- **Testing**
  To ensure a high quality of service, you are provided with a benchmark set that allows you to test whether your algorithm in working as expected.

2 Data

The HackIT data team prepared a test data set for your search engine, which you can download in an archive form their development server: `http://chinabigdatatraining.com/dataset/dataset.zip`. The data is HackIT property, which owns all rights to it but grants you the privilege to use it for the implementation of their search engine. HackIT does not allow you to miss-use or distribute their data.

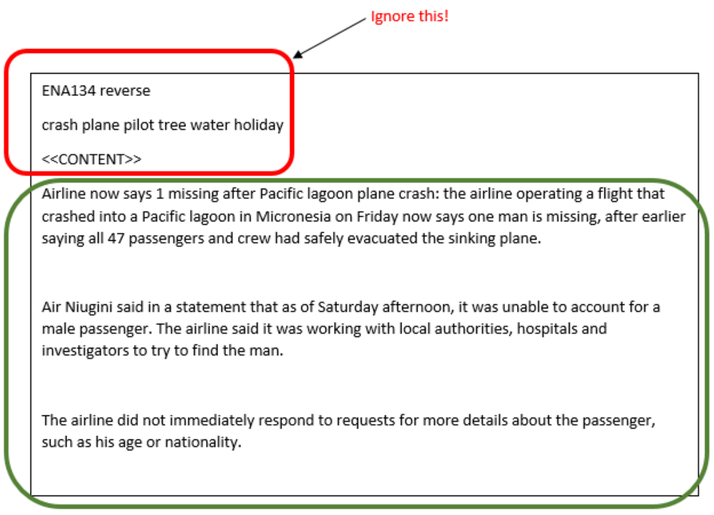
A data acquisition team member notes that the *document-ID* is defined by the document name (excluding the file extension).

For example:

documentX.txt → document-ID: documentX

Further he states that the beginning of each document contains EDNA-codes that are ***NOT*** relevant for your task. The content of each document starts **AFTER** the **«CONTENT»** key word.

For example:


Ignore this!

ENA134 reverse

crash plane pilot tree water holiday

<<CONTENT>>

Airline now says 1 missing after Pacific lagoon plane crash: the airline operating a flight that crashed into a Pacific lagoon in Micronesia on Friday now says one man is missing, after earlier saying all 47 passengers and crew had safely evacuated the sinking plane.

Air Niugini said in a statement that as of Saturday afternoon, it was unable to account for a male passenger. The airline said it was working with local authorities, hospitals and investigators to try to find the man.

The airline did not immediately respond to requests for more details about the passenger, such as his age or nationality.

Figure 1: Example document. The search engine should only consider the terms after the CONTENT tag (framed by the green line) and ignore anything before and including the CONTENT tag (framed by the red line)

3 Term Definition

The task at hand requires you to implement an information retrieval system. Consequently, you are exposed to terminology of research fields covering Information Retrieval, Data Mining and Bid Data applications in general. The following section introduces common terms needed to understand your assignment.

- **Search Engine**
  A program that searches for and identifies information in a data set that correspond to keywords or characters specified by the user. An example for a commercial website is Google or Baidu which focus on searching particular web sites on the World Wide Web.
- **Query**
  A set of search key words entered by the user of a search engine. The queury expresses the information need of the search engine user. For example, a user might be interested in information about a product release of a particular phone. His or her search query might look like this: "release date of iphone 2019"
- **Query Terms**
  Search terms that make up the query. The example query "release date of iphone 2019" consists of 5 query terms that together make up the search query and express the information need of the search engine user.
- **Script**
  A computer script is a list of commands that are executed by a certain program or scripting engine.
- **Document ID**
  A unique identifier for documents, i.e.: a number or term that describes exactly 1 document.
- **Query ID**
  A unique identifier for queries, i.e.: a number or term that describes exactly 1 query.
- **Bug**
  A software bug is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways.

4 Tasks

You are a software engineer at HackIT and tasked to implement their search engine. When executing the search engine the user should be required to specify the *"search mode"* using a command line parameter. The user should be able to choose from 2 modes:

- **Manual Guided Search**
  Manual guided search (command line key-word *"manual"*) allows the user to type query terms into the search engine by hand (manually). After submitting the search terms (enter key pressed) the search engine should print all document IDs to the screen that match at least one of the query terms.
- **Script Guided Search**
  Script guided search (command line key-word *"script"*) allows the user to create a query file called "*query.txt*" in the same file location as the search engine. The query file contains search queries (1 per line) identified by the query ID.

  Example query.txt:
  q1 house mouse
  q2 star car bar

  In this example q1 and q2 are the query-IDs of query 1 and query 2. Query 1 (q1) consists of 2 query terms (house and mouse), whereas query 2 consists of 3 query terms (star, car, bar)

Carry out the following steps:

.1. **Acquire Data Set**
Create a new working directory called "SearchEngine" and download the data set. Unpack the archive in your working directory and inspect the files manually. Pay attention to which part of the documents are relevant to your search engine, as stated in section Data.

.2. **Creating Source File** ***(1 Point)***
Create a source code file named **searchEngineXXX.c** (XXX should be replaced with your student-ID). The source code file holds the entire code for your search engine.

.3. **Implementing the Core Search Engine Algorithm** ***(40 Points)***
Implement the core search engine that offers two searching modes - manual and script. The search mode should be specified as a command line parameter using the option "manual" or "script". Algorithm 1 illustrates the pseudo code of the core search engine.

Algorithm 1 : Pseudo Code Search Engine HackIT

```
1:  check command line parameter for search mode
2:  if search mode = "manual" then
3:      ask user to enter query term (qt), separated by space and confirm by enter key
4:      query array $QA = \{qt\}$ /* → each term is one element in QA */
5:      for all document $d \in$ training data set do
6:          for all terms $t \in d$ do
7:              if $t \in QA$ then
8:                  print id of document d to screen (one per line)
9:              end-if
10:         end-for
11:     end-for
12: else
13:     if search mode = "script" then
14:         read query file
15:         for all query $q \in$ query file do
16:             query array $QA = \{$query term $qt \in q\}$ /* → each term is one element in QA */
17:             for all document $d \in$ training data set do
18:                 for all terms $t \in d$ do
19:                     if $t \in QA$ then
20:                         print id of q and d to screen (one per line)
21:                     end-if
22:                 end-for
23:             end-for
24:         end-for
25:     end-if
26: end-if
```

.4. **Correcting the Core Search Engine Algorithm *(14 Points)***
When creating the pseudo code for the search engine, the engineers of HackIT made a mistake as their version of the algorithm prints the document ID for each query term that matches a document term. This is of course a bug and not acceptable. You are required to ensure that each document ID is only printed a single time to screen, independently of the number of query terms that match it.

.5. **Advanced Search Engine I - Exact Search *(15 Points)***
Extend the core search engine by implementing an exact search feature as a command line option "exactSearch". This feature should be applicable for both search modes (manual & script) and compatible with all other advanced search features. If the search engine is executed with "exactSearch" command line parameter, print only those documents to screen that match **ALL** query terms.

For example:
$d1\{a, b, c\}$
$d2\{a, g, d\}$
$d3\{a, b\}$
$d4\{a, b, r, f\}$

$q1\{a, b, f\}$

When "exactSerach" is active only document d4 should be printed as it is the only document that matches all query terms of query q1, i.e: all query terms (a,b,f) of q1 also occur in d4.

.6. **Advanced Search Engine II - Top Search *(15 Points)***
Extend your search engine by implementing a top search feature as a command line option "topSearch". This feature should be applicable for both search modes (manual & script) and compatible with all other advanced search features. If the search engine is executed with "topSearch" command line parameter, print only the document containing the most amount of query terms (biggest overlap between query and document) to the screen.

For example:
$d1\{a, a, a, a, c\}$
$d2\{a, g, d, d\}$
$d3\{a, b, x\}$
$d4\{a, b, r, f\}$

$q1\{a, b, x\}$

When "topSearch" is active only document d1 should be printed to the screen when searching for query q1. Note that document d3 matches all query terms in q1 but the overlap of d3 and q1 = 3, whereas the overlap of d1 and q1 = 4 as repeated query terms are taken into account.

.7. **Advanced Search Engine III - Top K Search *(15 Points)***
Extend your search engine by implementing a ranked search feature as a command line option "topKSearch". This feature should be applicable for both search modes (manual & script) and compatible with all other advanced search features. If the search engine is executed with "topKSearch" command line parameter, print the 3 documents with the biggest overlap with the search query.

For example:
$d1\{a, b, c\}$
$d2\{a, g, d\}$
$d3\{a, b, f, d\}$
$d4\{a, b, b, r, f\}$
$d5\{g, r, d\}$

$q1\{a, b, f\}$

When searching documents d1 - d5 with query q1, you should print the following 3 results:
$q1$ $d4$
$q1$ $d3$
$q1$ $d1$

Note that the document order is not important here. The top 3 documents are determined by the overlap between the query and documents:
overlap of d4&q1 = 4; d3&q1 = 3; d1&q1 = 2; d2&q1 = 1; d5&q1 = 0;

5 Benchmark Testing

HackIT provides its engineers with a benchmark set of queries with their corresponding correct search results. You can download this benchmark set from their development server:
`http://chinabigdatatraining.com/dataset/testset.zip`. When implementing your search engine, you can run it on the data set and with the test queries provided by the benchmark set and compare your results.

6 Output Format

This section defines the required output format for the script search mode. It is important that the script mode of your search engine follows the output format correctly to ensure that your algorithm can be assessed. Whenever your core algorithm processes a certain query in script mode and finds a document that contains one or more of the query terms, you are required to print the following line to the screen:

query-ID document-ID

Each query/document ID pair should be in its own line and separated by a single space (blank). The output format of the manual mode is up to you and not further specified by HackIT.

7 Submission

Digital Submission:
In order to evaluate your performance you are required to submit your the source code file (***NOT*** executable). The evaluation result is based on the correct implementation and result of your algorithm. The total number of possible points is 100, which are distributed across the individual functions of your search engine (as specified in Section 3). The technology division of HackIT will evaluate your search engine by automated testing, which requires you to pay close attention to the correct naming of the source code filename as well as all command line parameters and output format. Name all function exactly as stated in this document.

NOTE: If you fail to meet the exact naming convention the automated evaluation script will fail and your submission result will be negative.

HackIT will release the submission system by next week via WeChat group. You will be provided with a link to an online submission system including your username and password. You are required to submit a digital version of your source code using the provided submission link in combination with your username and password before the end of the course.

Physical Submission:
In addition to the digital submission of your source code, you are required to hand in a physical (printed) version of your lab report before the end of the course.

HackIT's head of technology division Dr. Dom wishes you good luck, fun and success! :)