SEARCH ENGINE

Group 4

Data Structure (CS163)

In modern life, people are working with massive amounts of data. To take advantage of this wealth of information, an efficient search engine plays a crucial role. In this paper, we described the design of a search engine, and our work demonstrated how the basic elements of a search engine can be used in the search engine's basic task. Then, based on the TF-IDF algorithm, an enhanced approach was suggested to guarantee a more efficient retrieval of information resources.

SEARCH ENGINE

DATA STRUCTURE (CS163)

DESIGN ISSUES

Issues

- We got trouble with finding the way sorting the documents which include the keyword. At first, we come up with sorting it by Lexicographic order, meaning that which documents have the number first keyword will be in the top of result, if it is equal then those documents will be sorted by the next one. However, the keyword maybe includes some dummy word or stop words which is not helpful in finding the suitable result. The second idea is sorting documents with the number of visiting by users, but then we got trouble with the data needed to prepare in order to valuate correctly, which requires a great of times to collect. After researching the internet, we have found the appropriate technique for sorting documents based on the relation between keyword and each document, which is TF-IDF.
- How this technique is suitable for us in this engine? Firstly, this approach does not require any data of user's behavior on the prepared database. Secondly, this technique sorting the documents based on the important of each word compared to each other in that keyword, which is far better than the Lexicographic order one.

Data structure

- We chose Trie as our project's main data structure since it is a good information retrieval data structure because search complexities may be lowered to the optimum limit, which is O(keyLength).
- Maps and vectors are also used to save memory and make data access easier for searching engine.
- These items will be contained in a node: an array of children:
  - 39 children's nodes which store 'a' to 'z', '0' to '9', '#', '$', '*' respectively.
  - An unordered hash map to store the filename of word.

ALGORITHM

General method

1. All data is loaded into Trie structures.
2. After loading data successfully, we display a menu of operator options.
3. The system will split and filter the data you want to search into an array of words and detect the operator you want to apply after you enter the data you want to search. Extract stops words from the query.
4. Ranking top 5 results using TF-IDF.
5. Ranking and Display to customer.

Help function

letterToInt(char a) function

When we pass a character into this function, it returns a number that corresponds to the logic stored in the trie structure.

insert(Trie* root, string word)

In this function, we traverse the word. In i-step, we jump to a children[letterToInt (word [i-1])]. We continue to traverse until getting the last element of word. At that node, we save the filename that contains this word.

loadData() function

This function helps us to load all data to Tries structure: dataRoot, stopwordsRoot and thesaurusRoot. Data is loaded into the system depending on the function insert.

getLeaf(string word) function

The goal of this function is to get to the leaf node and acquire a list of files that contain that word.

loadData() function

If there are any undesired characters in string s, we eliminate them and return a new string. For example: wordIgnore("///abctr") will return "abctr".

Search operators

1. **AND** operator

We divide the keyword string into an array of terms and iterate through it. Starting with the first member of the word array, we create a hash map for the list file containing this word. When we acquire the next element, we continue to update the occurrence of the called file. All files with fewer than "current index" occurrences are removed because it violates the AND operator's properties. We continue this process until reaching the final element. After all, we have a list of text files that satisfy AND have properties, we score them using IF-IDF, and we display the top 5 text files with the highest score.

2. **OR** operator

In loading data section, we have already stored the location of each word in each documents using the combination of string (file address). Firstly, we divide the string of keywords into an array of words. After that, we begin iterating through this array. In each element of the word array, we store the list file which contains this word in a hash map. We iterate this hash map to acquire all possible text file that satisfies OR operator properties. After all, we use IF-IDF to score and display the top five text files with the highest score. ad

3. `**-**` operator

Firstly, the engine will delete the word following the "-" sign from the keywords. After that, we use all remain words as keyword, do default searching and load all satisfied text files to a hash map. Finally, we use function **getLeaf()** to get the list file of the removed word and remove the files which appear in both list file and hash map. After all, we use IF-IDF method scoring them and display top 5 text files with highest score.

4. `**intitle:**` operator

This operator is likely to OR operator. However, for each searched term, we load to a hash map the files that contain the word in the initial lines.

5. `**+**` operator

We treat the substring before `+` as a normal query and the other behind `+` has to be appear in document.

6. `**filetype:**` operator

This operator is easy to handle, we search in the '___index.txt' file to look for the matching file type and print out the first five files.

7. Search for a price

The mechanism of this operator is the same as that of the AND operator.

8. Search **hashtags**

We treat this operator the same as exact match operator.

9. Search for an exact match.

In loading data section, we have already stored the location of each word in each documents using the combination of string (file address) and vector (the location of each one in that document) so that we can reduce the time enumerate every word in each file. Firstly, we treat the query as normal keyword and get the list of ranked documents. Secondly, we investigate every single file in the list and check if any file has the continuous keyword by locate the address of first word in query, then checking if the next one is match with the corresponding word or not, etc.... If it is true, the file will be printed out in that order.

10. Search for wildcards or unknown words.

This operator is likely to the eighth, there will be a small difference in checking the continuous step. The condition `if the corresponding word must be the same` will be ignore if the character '*' appear.

11. Search withing range operator

This operator is likely to the seventh one, the keyword will be first handled to split the price off and use the seventh operator for convenient but will be plus another condition that is have to be in between the price range.

12. `**~**` operator

We store the synonyms database into a separate Trie, when this operator gets called, we find the synonyms of that keyword, connect them to a string with space in between then we treat them as a normal keyword. *Example*: "set" is synonyms with "configure", "collection" and "change", the keyword then become "configure collection change" and will be treated as normal keyword.
We use thesaurus database public by WordNet.

OPTIMIZATION

Complexity

Runtime complexity:

- O(N ^ M) where N is the possible character count and M is the average word length.
- In case data consists of 100 files, loading database costs 1.536952 seconds (average).
- In case data consists of 11 000 files, loading database costs 172.1386 seconds.
- Total time of 1000 random queries: 30.916124 seconds.

Space complexity:

- Loading database: O (N * M), where N is the number of words in database and M is the length of longest string.
- Searching: O(keyLength)

Optimization method

Use Trie as main data structure:

We chose Trie as the primary data structure since it optimizes the system's space. We regularly come across the same words during loading data. Rather than saving them separately, Trie allows you to save information of a single word in the same address. As a result, Trie is able to optimize the system space.

Ignore stop words:

We load stop words data to a Trie called stopwordsRoot. When customer enters an keywords to search, the engine will ignore all stop words which exists in keywords. This process helps us saving running time a lot.

Standardize letter to lowercase:

All words in text files are convert to lowercase before loading to Trie structures. This helps us not only save memory but also optimize searching speed.

When customers enter something like: “Apple AND oRange”, the system will convert it to “apple and orange”. This method makes the term we're looking for easy match the Trie data.

Rank using IF-IDF method:

TF-IDF stands for Term Frequency - Inverse Document Frequency. The TF-IDF weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus (data-set).

When we have a list of text files and a list of words,we use idf-if formula to score them and display 5 files with highest ranking.

- Term frequency: $$TF(t,d) = \frac{f(t,d)}{\max\{f(w,d) : w \in d\}}$$
  - $f(t,d)$: number of occurrences of word t in the text d.
  - $\max\{f(w,d) : w \in d\}$: the maximum number of occurrences of any word in the text.

- Inverse Document Frequency: $$IDF(t,d) = \ln \frac{|D|}{|\{d \in D : t \in d\}|}$$
  - $|D|$: total number of text files in set A.
  - $|\{d \in D : t \in d\}|$: total number of text files which contain word t.

IF-IDF Formula:

$$TFIDF = tf(t,d) \times IDF(t,d)$$

When comparing files, the TF-IDF value is also used as the keyword score.

TEST SAMPLES

1. **AND** operator

Input keyword (0 to exit): summer AND mushrooms

LINK: Data41.txt

...this **summer** These great new reads take you on journeys from .....foraged **mushrooms** in Sydney. This smorgasbord of a tale will have ...

LINK: Data368.txt

...this **summer** These great new reads take you on journeys from .....foraged **mushrooms** in Sydney. This smorgasbord of a tale will have ...

LINK: 12 travel books you won't be able to put down this summer.txt

...this **summer** These great new reads take you on journeys from .....foraged **mushrooms** in Sydney. This smorgasbord of a tale will have ...

2. **OR** operator

Input keyword (0 to exit): snake OR horse

LINK: Data466.txt

...Trojan **horse** affair, in which Islamic extremists supposedly attempted to take ...

LINK: Data139.txt

...Trojan **horse** affair, in which Islamic extremists supposedly attempted to take ...

LINK: 44 2.txt

...Trojan **horse** affair, in which Islamic extremists supposedly attempted to take ...

LINK: burmese python.txt

...large **snake** of choice among reptile owners. Unfortunately these potentially huge ...

LINK: Data648.txt

...large **snake** of choice among reptile owners. Unfortunately these potentially huge ...

3. **`-`** operator

Input keyword (0 to exit): snake -large

LINK: Data649.txt

...coral **snake**. The skin of caecilians is smooth and slimy, and ...

LINK: Data322.txt

...coral **snake**. The skin of caecilians is smooth and slimy, and ...

LINK: Caecilians.txt

...coral **snake**. The skin of caecilians is smooth and slimy, and ...

LINK: Data659.txt

...wire **snake** around its deck. Two life-sized human dummies in orange ...

LINK: Data656.txt

...wire **snake** around its deck. Two life-sized human dummies in orange ...

4. **`intitle:`** operator

Input keyword (0 to exit): intitle: Journeys Paris

LINK: Data41.txt

...on **journeys** from Papua New Guinea to **Paris**. BOOKS ARE ESSENTIAL items ...

LINK: Data368.txt

...on **journeys** from Papua New Guinea to **Paris**. BOOKS ARE ESSENTIAL items ...

LINK: 12 travel books you won't be able to put down this summer.txt

...on **journeys** from Papua New Guinea to **Paris**. BOOKS ARE ESSENTIAL items ...

5. **`+`** operator

Input keyword (0 to exit): campaign +appearance

LINK: 020.txt

...recent **campaign appearance**, Mr Erdogan warned that Australians hostile to Islam would ...

LINK: 033.txt

...joint **appearance** in a cover story about how strongmen subvert democracy. ...

LINK: 058.txt

...the **appearance** at last yearâ€™s Paris Air Show of a Chinese ...

LINK: 067.txt

...outward **appearance**. The details vary from country to country, but it ...A campaign poster showed Mr Soros grinning evilly and embracing opposition ...

LINK: 097.txt

...the **appearance** of the countryside, is spot-on. Advocates may point to ...

6. **`filetype:`** operator

Input keyword (0 to exit): filetype:1.txt

1.txt

0**1.txt**

00**1.txt**

1**1.txt**

01**1.txt**

7. Search for a price

Input keyword (0 to exit): salary $200

LINK: Data90.txt

...her **salary** of just **$200** a month, she pays for a private ...

LINK: Data417.txt

...her **salary** of just **$200** a month, she pays for a private ...

LINK: 027.txt

...her **salary** of just **$200** a month, she pays for a private ...

8. Search **hashtags**

Input keyword (0 to exit): #OldEnoughToVoteOurselves

LINK: 012.txt

...hashtag **#OldEnoughToVoteOurselves** to trend on Twitter in Thai); and after the ...

LINK: Data371.txt

...hashtag **#OldEnoughToVoteOurselves** to trend on Twitter in Thai); and after the ...

LINK: Data44.txt

...hashtag **#OldEnoughToVoteOurselves** to trend on Twitter in Thai); and after the ...

9. Search for an exact match.

Input keyword (0 to exit): "mitochondrial DNA"

LINK: Ancient DNA reveals new twists in Neanderthal migration.txt

...Ancient DNA reveals new twists in Neanderthal migration Genetic surprises pulled .....both **mitochondrial DNA**, which is a genetic fraction passed from mother to ...

LINK: Data312.txt

...Ancient DNA reveals new twists in Neanderthal migration Genetic surprises pulled .....both **mitochondrial DNA**, which is a genetic fraction passed from mother to ...

LINK: Data639.txt

...Ancient DNA reveals new twists in Neanderthal migration Genetic surprises pulled .....both **mitochondrial DNA**, which is a genetic fraction passed from mother to ...

10. Search for wildcards or unknown words.

Input keyword (0 to exit): "This Teenager * * More"

LINK: 5 Times People Used Trees to Change the World.txt

...them. **This Teenager** Has Planted **More** Than 14 Billion Trees Felix Finkbeiner has .....and more, and Payeng must protect it from a new threatâ€”the ...

LINK: Data15.txt

...them. **This Teenager** Has Planted **More** Than 14 Billion Trees Felix Finkbeiner has .....and more, and Payeng must protect it from a new threatâ€”the ...

LINK: Data342.txt

...them. **This Teenager** Has Planted **More** Than 14 Billion Trees Felix Finkbeiner has .....and more, and Payeng must protect it from a new threatâ€”the ...

LINK: Data669.txt

...them**. This Teenager** Has Planted **More** Than 14 Billion Trees Felix Finkbeiner has .....and more, and Payeng must protect it from a new threatâ€”the ...

LINK: 091.txt

...a **more** extreme case. In 2015, 47% of unmarried 20- to .....a **teenager**. She contacted her own mother twice in eight weeks.Worries ...

11. Search withing range operator

Input keyword (0 to exit): company $1..$1000

LINK: Data539.txt

...on **$200** million worth of Chinese goods to 25 percent from .....10. **Company** shares have hovered near **$42** per share since their debut .....roughly **$1** billion in the first quarter alone. While revenue jumped .....to **$3.1** billion for the quarter, revenue growth slowed compared to ...

LINK: Data212.txt

...on **$200** million worth of Chinese goods to 25 percent from .....10. **Company** shares have hovered near **$42** per share since their debut .....roughly **$1** billion in the first quarter alone. While revenue jumped .....to **$3.1** billion for the quarter, revenue growth slowed compared to ...

LINK: 68 2.txt

...on **$200** million worth of Chinese goods to 25 percent from .....10. **Company** shares have hovered near **$42** per share since their debut .....roughly **$1** billion in the first quarter alone. While revenue jumped .....to **$3.1** billion for the quarter, revenue growth slowed compared to ...

LINK: Data536.txt

...for **$9.99** per order, a move that indicates the retailer is .....a **$99** annual membership fee or **$14** for a monthly membership (an .....of **$35** on its most popular items. The retail giant currently .....pay **$119** a year to one-day delivery. The online shopping giant .....The **company** previously offered one-day shipping on select items and free ...

LINK: Data209.txt

...for **$9.99** per order, a move that indicates the retailer is .....a **$99** annual membership fee or **$14** for a monthly membership (an .....of **$35** on its most popular items. The retail giant currently .....pay **$119** a year to one-day delivery. The online shopping giant .....The **company** previously offered one-day shipping on select items and free ...

12. `~` operator

Input keyword (0 to exit): ~house

LINK: 008.txt

...Bataclan **theatre** in Paris in 2015, Mr Belattar started his shows ...â€œTrumpâ€ **sign** on top. The gag somehow seems less funny today. ...his **family**. And Mr Trump? He could well improv his way ...

LINK: 89.txt

...the **home theatre** software launched with XP and not many people used ...

LINK: Data276.txt

...the **home theatre** software launched with XP and not many people used ...

LINK: Data28.txt

...Bataclan **theatre** in Paris in 2015, Mr Belattar started his shows ...â€œTrumpâ€ **sign** on top. The gag somehow seems less funny today. ...his **family**. And Mr Trump? He could well improv his way ...

LINK: Data355.txt

...Bataclan **theatre** in Paris in 2015, Mr Belattar started his shows ...â€œTrumpâ€ **sign** on top. The gag somehow seems less funny today. ...his **family**. And Mr Trump? He could well improv his way ...