Searchflow

Building A Search System based on Stackoverflow.com

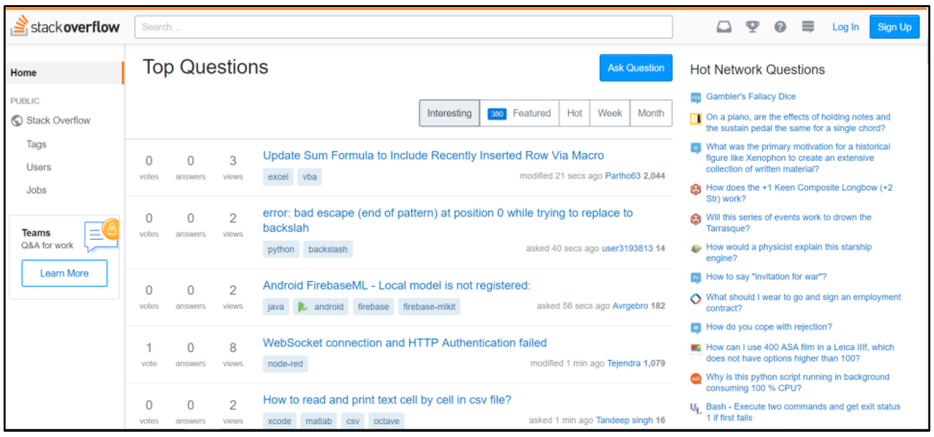
Kevin Chen Trieu[1], Oliver Budke[1], Petros Debesay[1], and Pritom Kumar Mondal[1]
[1] School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University.

Abstract. Stackoverflow is an online library of answers to questions of professional and enthusiast programmers. One document on Stackoverflow consists of usually one question by a programmer and several replies from other users. Each reply may have some comments. Questions are labelled with different tags and similar questions are also linked. Our search system named Searchflow is aimed at indexing these documents. Therefore, about one million questions and answers have been retrieved from the website. Indexing and searching methods included in Searchflow provide relevant documents regarding a user's information need. A state-of-the-art graphical user interface helps the user interact with the database. Ultimately, the contribution of this work is to assist users in finding solutions for computer-science related problems that are well documented in the Stackoverflow library.

Keywords: Information Retrieval, Web Search, Search Engine.

1 Introduction

Professional and enthusiast programmers come across problems in their everyday work. However, these problems often have been encountered previously by other programmers who are willing to share their acquired knowledge about the solution. Jeff Atwood and Joel Spolsky, both having software development blogs with a huge following in 2007, built a question & answer platform called Stackoverflow to alleviate this situation in the daily life of a programmer. Their goal has been to build a library of detailed answers to every question about programming with the help of a community. [1]



Top Questions

0 votes 0 answers 3 views
Update Sum Formula to Include Recently Inserted Row Via Macro
excel vba
modified 21 secs ago Partho63 2,044

0 votes 0 answers 2 views
error: bad escape (end of pattern) at position 0 while trying to replace to backslah
python backslash
asked 40 secs ago user3193813 14

0 votes 0 answers 2 views
Android FirebaseML - Local model is not registered:
java android firebase firebase-mlkit
asked 56 secs ago Avrgebro 182

1 vote 0 answers 8 views
WebSocket connection and HTTP Authentication failed
node-red
modified 1 min ago Tejendra 1,079

0 votes 0 answers 2 views
How to read and print text cell by cell in csv file?
xcode matlab csv octave
asked 1 min ago Tandeep singh 16

Hot Network Questions

- Gambler's Fallacy Dice
- On a piano, are the effects of holding notes and the sustain pedal the same for a single chord?
- What was the primary motivation for a historical figure like Xenophon to create an extensive collection of written material?
- How does the +1 Keen Composite Longbow (+2 Str) work?
- Will this series of events work to drown the Tarrasque?
- How would a physicist explain this starship engine?
- How to say "invitation for war"?
- What should I wear to go and sign an employment contract?
- How do you cope with rejection?
- How can I use 400 ASA film in a Leica IIIf, which does not have options higher than 100?
- Why is this python script running in background consuming 100 % CPU?
- Bash - Execute two commands and get exit status 1 if first fails

Fig. 1. Screenshot of Stack Overflow as of May 2019

All questions are tagged with their subject areas. Each can have up to 5 tags, since a question might be related to several subjects. Good answers to questions are voted up and rise to the top. The best answers show up first so that they are always easy to find. The person who asked can mark one answer as accepted. Accepting, however, does not mean it is the best answer, it just means that it worked for the person who asked. Comments can be used to ask for more information or to clarify a question or answer. As of today, with 17.7 million posted questions and 27 million answers Stackoverflow has been established as the most popular online library for programmers. [2]

2 Architecture

At this point the motivation of building a search system for the platform Stackoverflow should be clear. In order to present our search system called Searchflow this report choses a top-down approach which means we introduce our overall architecture first. To describe the retrieval process of Searchflow, we considered a simple and generic software architecture as shown in Fig. 2. The depicted system contains of a visual search system interface that allows the user to interact with the system by specifying a user need which is parsed and transformed by text operations. These text operations are the same which are performed on the documents that the search system retrieves. Before the actual query is generated in the generic software architecture, query operations are applied, which provide a system representation for the user need. The query is then processed to obtain the retrieved documents.

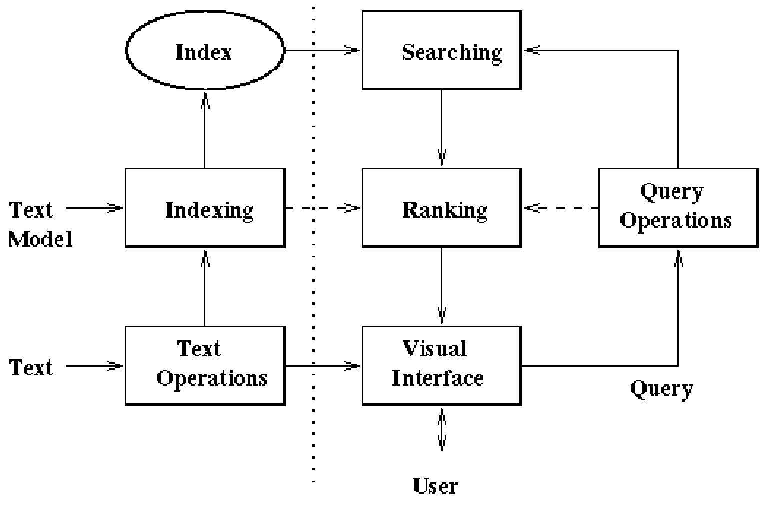


Fig. 2. The process of retrieving information.

In order to build the search system, it is necessary to define the text database. This is done by specifying the documents to be used, the operations to be performed and the text model (i.e., the text structure and what elements can be retrieved). The text operations transform the original documents and generate a logical view of them. The index structure (that is built before the user inputs his user need) allows fast query processing. Eventually, before being sent to the user, the retrieved documents are ranked according to a likelihood of relevance. The user then examines the set of ranked documents in the search for useful information. [3]

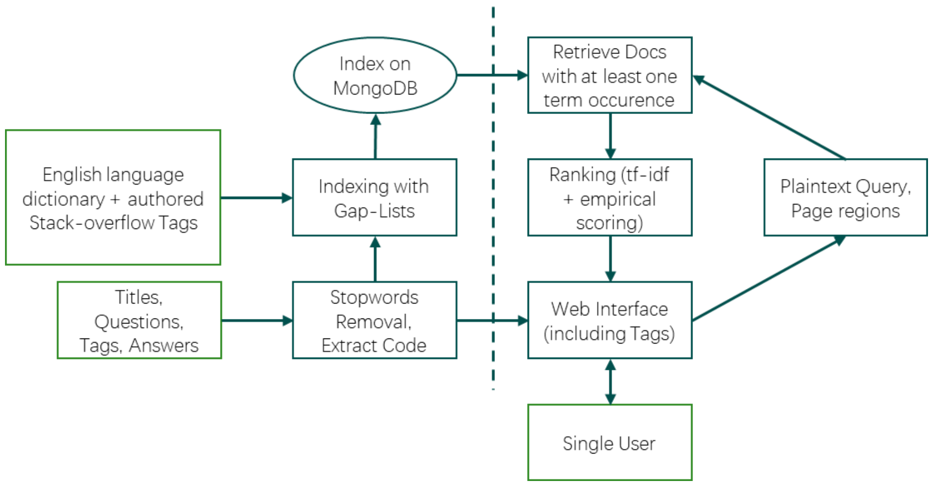

Index on MongoDB
Retrieve Docs with at least one term occurence
English language dictionary + authored Stack-overflow Tags
Indexing with Gap-Lists
Ranking (tf-idf + empirical scoring)
Plaintext Query, Page regions
Titles, Questions, Tags, Answers
Stopwords Removal, Extract Code
Web Interface (including Tags)
Single User

Fig. 3. Searchflow Architecture

The actual software architecture of Searchflow has been adapted from the generic architecture shown in **Fig. 2**. However, the generic components have been instantiated for this report because our search system serves the specific purpose of retrieving documents from Stackoverflow.com.

According to **Fig. 3**, Searchflow has been designed for a single user. The system provides the user with a web interface. For advanced search queries the interface suggests tags from Stackoverflow. When the user submits her query, the system generates a plaintext query regarding an optional page region that can be specified by the user. The available page regions are further explained in section 3.2. If documents contain at least one term of the query they are retrieved from the database. Before they are displayed to the use, the search results are ranked by our Searchflow scoring functions which are further explained in section 3.1. Searchflow relies on a MongoDB where all questions and answers are indexed. This index has been built with gap-lists in order to save memory. Our text model comprises of an English language dictionary and a list of Stackoverflow tags. The tags are useful in addition to the dictionary because they are authored by Stackoverflow and they consist of unique computer science-related terms. We consider the computer science-related terms to be especially useful. As another text processing step, stop words are removed after data retrieval. Moreover, code that has been posted by users on Stackoverflow is extracted and handled separately.

The following sections of this work will focus on the aspects of the user interface design, the data retrieval process, indexing methods and searching and scoring functions.

3 The Implementation

3.1 User Interface Design

In order to interact with the user Searchflow boasts a web interface that is as easy to use as possible. By taking inspiration from other well-established search engines the user can feel familiar with SearchFlow without having previously used it or having to read a manual. That is why the design follows a minimalistic design rule with as few buttons as possible. This involves that no redundant or unnecessary information is displayed to the user.

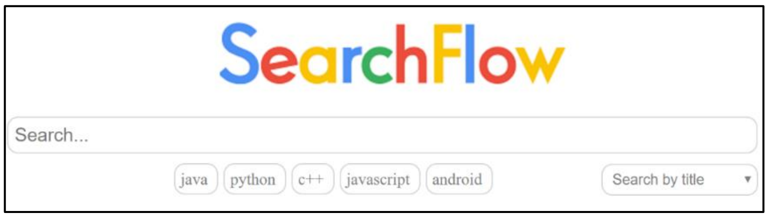
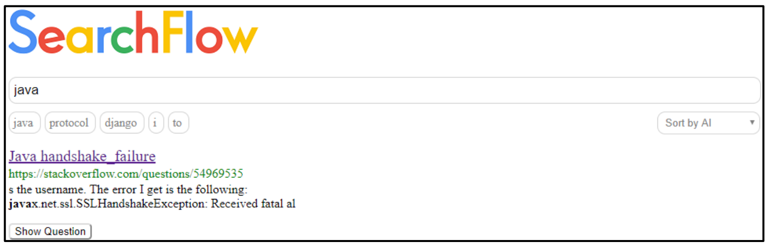


Fig. 4. The frontpage consist of a search bar and a drop down menu where the user can choose which region to search in. An additional feature are the tags buttons which show the most frequently used tags which may assist the user when doing a search.

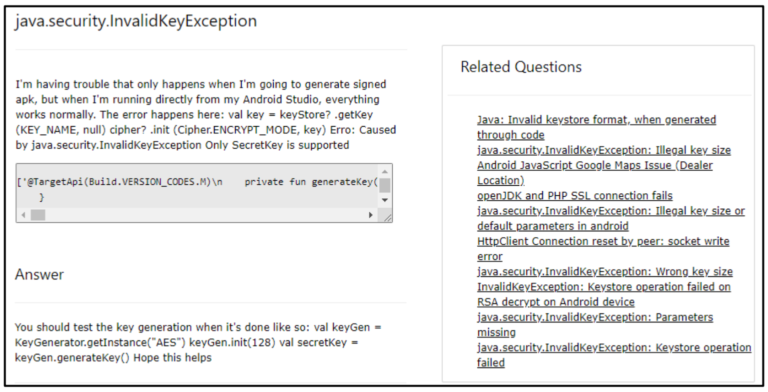
The user interface is intended to obtain results overview only. These results can include questions and answers. For additional information needs, we provide the user with a link to the relevant question on Stackoverflow. She can jump to Stackoverflow by simply clicking this link. An additional button (Show Question) is also provided if the user wishes to see the question and answers directly in the user interface.

Another paradigm of our design is the single user functionality. The user of the graphical interface does not need any user authentication in order to obtain search results. This involves that there is no customization available (except cookie-based search suggestions in the search field). To adhere with data privacy rights, this also means that the output of results is not personalized (neither the number nor the rank of results).



Java handshake_failure
https://stackoverflow.com/questions/54969535
s the username. The error I get is the following:
javax.net.ssl.SSLHandshakeException: Received fatal al

Fig. 5. The result page reminds one of the frontpage with the addition of a list showing the results. This simplifies additional searches while ensuring that the user is familiar with the user interface.


java.security.InvalidKeyException

I'm having trouble that only happens when I'm going to generate signed apk, but when I'm running directly from my Android Studio, everything works normally. The error happens here: val key = keyStore? .getKey (KEY_NAME, null) cipher? .init (Cipher.ENCRYPT_MODE, key) Erro: Caused by java.security.InvalidKeyException Only SecretKey is supported

```
['@TargetApi(Build.VERSION_CODES.M)\n    private fun generateKey(
    }
```

Answer

You should test the key generation when it's done like so: val keyGen = KeyGenerator.getInstance("AES") keyGen.init(128) val secretKey = keyGen.generateKey() Hope this helps

Related Questions

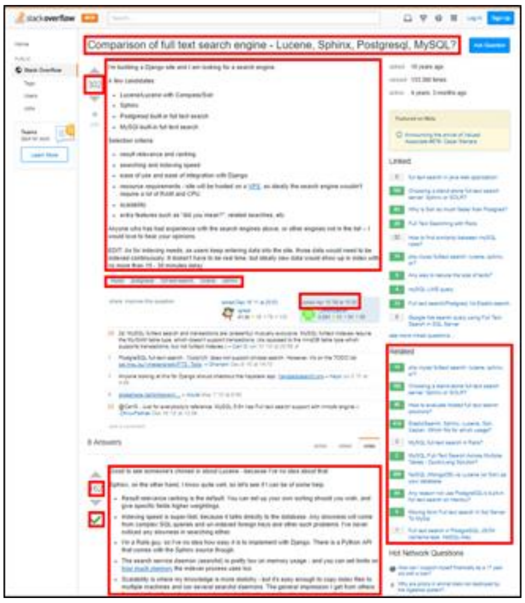
Java: Invalid keystore format, when generated through code
java.security.InvalidKeyException: Illegal key size
Android JavaScript Google Maps Issue (Dealer Location)
openJDK and PHP SSL connection fails
java.security.InvalidKeyException: Illegal key size or default parameters in android
HttpClient Connection reset by peer: socket write error
java.security.InvalidKeyException: Wrong key size
InvalidKeyException: Keystore operation failed on RSA decrypt on Android device
java.security.InvalidKeyException: Parameters missing
java.security.InvalidKeyException: Keystore operation failed

Fig. 6. The question page displays the question with its answers. A related question list is also shown to assist the user.

3.2 Data Retrieval Process

In order to provides the user with search results, documents from Stackoverflow.com had to be retrieved. As mentioned in the introduction 17.7 million questions and 27 million answers cannot be retrieved in a reasonable amount of time. The limitations are the application programming interface (API) of Stackoverflow allowing 300 calls per day and the internet protocol address (IP) based blocking of incoming requests to 10 requests per day. Consequently, Searchflow aimed at building a search system comprising two years' worth of questions and their corresponding answers from Stack Overflow. However, the data still comprises of 4.7 million question and corresponding 5.1 million answers. With our hardware running for three weeks with only four hours of daily downtime this amount of data still could not be retrieved although a dynamic proxy change program has been added. The final version of Searchflow consists of 471,901 questions (another 2.9 million Question ID was in queue) and 512,304 answers which have been asked and answered between May 2017 and May 2019.[1] These figures equal 2.67 % (and 1.80 % respectively) of overall data and 2.5 GB of disk space on MongoDB. [4]

The data does not only comprise of questions and answers from Stackoverflow.com. In order to improve search results Searchflow retrieved titles, dates, tags, votes and views of questions. Stackoverflow provides the user with related questions that have been retrieved as well. Related to answers Searchflow obtains the binary value for an accepted answer. Answers were sorted according to "upvote" and "accepted" and inserted into databases. Codes of the asked question were also separated.



Comparison of full text search engine - Lucene, Sphinx, Postgresql, MySQL?

Fig. 7. Retrieved document fields

[1] Query: select count(*) as count from posts where postTypeId = 1 and creationDate >= '05/31/2017 0:00' select count(*) as count from posts where postTypeId = 2 and creationDate >= '05/31/2017 0:00

3.3 Indexing Methods

The tokens were indexed as posting lists. In order to save on memory, the number of terms were limited to the words in the nltk library and extended by computer science terms. The computer science terms were taken from all tags available on Stack Overflow. Stop words, as defined by nltk, were then removed. This resulted in an index with a total of 286 736 terms. In addition to this, the postings lists are gap lists. This was also to save on memory. The underlying implementation is a doubly linked list with each node recording the gap to the previous node as well as the frequency of the term in the document region. The reason for the linked list structure is due to that we cannot know the size of the array beforehand. This ensures that we do not have to create new arrays as the posting list is growing. It also should result in much faster insert theoretically. Due to that our crawled data was sorted, it likely was not much faster if at all, since all additions were likely at the end. Due to this, the indexing was not as fast as desired. To add a document to a posting list, the complexity of insertion is O(N) where N is the size of the posting list. For an average size of N/2 and N insertions, the complexity for insertion of the whole posting list is $O(N^2/2)$. For a document size of K, and words per document of Z the complexity grows to $O(K•Z•N^2/2)$. With K being close of 500 000 in our case, and Z likely averaging above a few hundred words, it is easy to see why the indexing was slow.

We divided the Stack Overflow page into four areas; title, text, code, and tags. These were then indexed separately. As such, it is possible to search in a separate region. The indices were saved using mongodb. The documents originally had long document ids of size 10^7, they were given an extra id ranging from 1 to len(documents) in order to save space in the posting lists.

3.4 Searching & Scoring Functions

Searchflow uses String matching methods to retrieve relevant documents. Therefore, the search query string and the document string are compared. However, relevant documents must be ranked because the user has a limited amount of time to obtain information about his information need. The input parameters for the ranking function have been retrieved from Stackoverflow and are listed below.

Scoring Parameter	**Reason**	**Desirable Value**
Views	If a question has more views, more users have clicked on it because it seemed relevant for them.	High
Number of Tags	If a user used more tags, the quality of the question is higher.	high
Votes	If a question received more votes, more users find it useful.	high
Accepted Answer	If an answer has been accepted, the problem has been solved for the user who asked the question.	true

In order to improve performance, the scoring is calculated in two steps. First, documents are retrieved by doing intersection of the posting list with the query terms. Thereafter TF-IDF is calculated which we sum to get a score. The top scored documents are picked to calculate the full TF-IDF and then compared with the query using cosine similarity. Depending on the region we also take the scoring parameters mentioned above in consideration.

4 Performance Evaluation

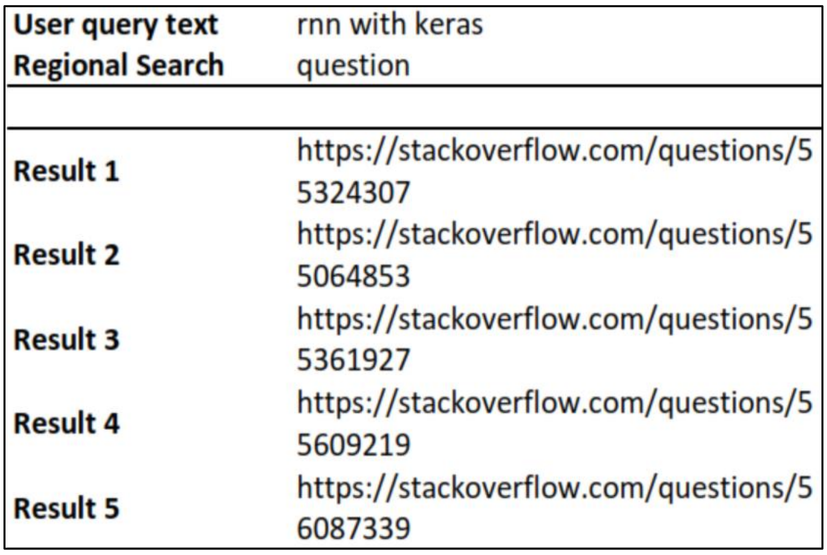
As the architecture and implementation of Searchflow are clear by this point of the report, the performance has to be evaluated in order to prove the contribution of our system towards a performant Stackoverflow index. Here fore, Recall and precision metrics are not sufficient to fully evaluate the performance of Searchflow. Due to the ranking of search results as described in section 3.4 these metrics do not reflect the relevance of the retrieved documents compared to a ground truth. Precision equals the fraction of relevant documents among the retrieved document, while recall (also known as sensitivity) is the fraction of relevant documents that have been retrieved over the total amount of relevant documents.

As a next step, in order to measure the relevance, one would flag all retrieved documents as relevant or not relevant. With this information, one could calculate measures of relevance such as Precision, Recall and the F-Score. However, this labelling is not feasible since almost half a million of document would need to be labelled. Moreover, the ranking of result could still not be described adequately. Due to these insufficiencies a new performance metric must be calculated.

We developed a new metric to evaluate Searchflow that takes ranked results into account and scales from 0 to 1. A score of 1 means that all results match with the benchmark. A score of 0 means that documents with a high relevance have not been returned with a high rank. By this metric each user scenario can be evaluated and compared.

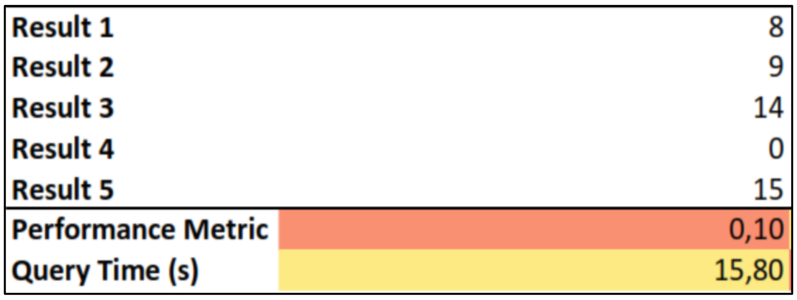
```
Function Performance Metric:
  For each result in benchmark:
  Get rank of result in Searchflow
    If not present: Set rank to (number of results + 1)
  Obtain absolute value of $rank_{benchmark} - rank_{Searchflow}$
  Sum up absolute values for every result
return e^(- sum of absolute values)
```

The benchmarks for our metric contains six scenarios in which a user expresses an information need. The ranked top 5 search results of these scenarios are compared with the actual top 20 results. Here, each scenario contains a query text and a search region. The latter parameter defaults to a search on all data. These benchmark results have been obtained by searching on Google.com and Stackoverflow.com with the built-in ElasticSearch instance. Important here is that the amount of results has been narrowed down to the documents that have been retrieved by Searchflow. Since cross-validation between the ranked outputs of Google and Stackoverflow has shown major deviations, the ranking obtained with this methodology cannot be taken into consideration for an overall performance evaluation. As a solution for this problem, we have ranked the obtained results manually. This method is qualified since we have been daily users of Stackoverflow throughout our academic careers and can empirically evaluate the degree of relevance of a question and answer.



User query text	rnn with keras
Regional Search	question
Result 1	https://stackoverflow.com/questions/55324307
Result 2	https://stackoverflow.com/questions/55064853
Result 3	https://stackoverflow.com/questions/55361927
Result 4	https://stackoverflow.com/questions/55609219
Result 5	https://stackoverflow.com/questions/56087339

Fig. 8. Each user scenario consists of five results from the database. These ranked results serve as a benchmark for the evaluation of Searchflow.



Result 1	8
Result 2	9
Result 3	14
Result 4	0
Result 5	15
Performance Metric	0,10
Query Time (s)	15,80

Fig. 9. Every result of each user scenario is evaluated towards the Searchflow results separately. For the first scenario the metric is 0.1 with a query time of 15.8 seconds.

As another performance metric query times have been measured. This time is measured from the submit of user query until the display of all results. It can be observed that queries consisting of more terms have a longer query time. The reason for this is most likely the intersection procedure that must be performed on different kinds of datasets.

5 Future Work & Conclusion

Concluding this work an outlook on future work is given. Improvements in later versions of Searchflow comprise of functional and non-functional improvements. Functional improvements can be observed by the user. As an example, a spellcheck could be done on the user input. The user can be prompted for similar keywords he might want to search for. On the other hand, non-functional improvements might be an additional computer science dictionary. This would improve the tokenization of texts because many terms in computer science are very specific and mostly not included in a standard English dictionary.

Another significant improvement will be the use of distributed computing to improve crawling the data. Our current version of crawler partially supports this feature but certainly we can make more improvements.

The user interface for showing the question and answers is not optimal since it is static with the text first and code after. This is due to our misunderstanding with the project since our initial thought was that we would only display the ranked results and link to StackOverflow. Since we separated the code from the text when crawling we lost the structure of the page. Without re-crawling we do not know the structure of the text and code in each page and therefore cannot create a dynamic structure for the user interface.

The TF-IDF is currently calculated during runtime which is not ideal for performance. In a future iteration the TF-IDF scores of the documents should be calculated beforehand and stored in the database.

With these improvements in mind, Searchflow could outperforms other well-established search systems due to more relevant search results and more precise search criteria (such as our page regions). While overall performance and document retrieval still remain a challenge, our overall architecture is well suited for a highly specialized search system based on data retrieved from Stackoverflow.com.

References

1. Stackoverflow, https://stackoverflow.com/tour, last accessed 2019/06/10.
2. Stackoverflow, https://data.stackexchange.com/stackoverflow/query, last accessed 2019/06/10.
3. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley Longman Publishing Co., Boston, USA (1999).
4. Stackoverflow, https://data.stackexchange.com/stackoverflow/query, last accessed 2019/06/10.