Grenze International Journal of Engineering and Technology, June Issue

QuestCloud – An Optimized Search Engine

Milind Kulkarni[1], Hetan Nandre[2], Aarohi Metkar[3], Mihir Bhundia[4] and Manasi Potey[5]
[1-5]Vishwakarma Institute of Technology (Department of Computer Engineering), Pune, India
Email: milind.kulkarni@vit.edu, hetan.nandre21@vit.edu, aarohi.metkar21@vit.edu, mihir.bhundia21@vit.edu, manasi.potey21@vit.edu

***Abstract*—The search engine is designed to crawl and index web pages, and then use advanced algorithms to analyze the content of these pages and determine their relevance to specific search queries. In this project, we present the development of an optimized search engine that aims to provide efficient and accurate search results for users. We have implemented various optimization techniques to improve the performance of the search engine, including efficient data structures for indexing, query processing, and result ranking. Our evaluation results demonstrate that the proposed search engine achieves significant improvements in search efficiency and accuracy compared to existing search engines. A search engine is a very useful and effective instrument for obtaining any sort of information from the Internet. Search engine helps all types of users to immediately find relevant information. But the present design of search engines has led us to the semantic web design. After implementing the semantic web, the search engines should be more efficient and useful for searching relevant and useful information. The proposed search engine can be used for various applications, such as information retrieval, e-commerce, and social media analysis. This paper presents the implementation and design structure of a semantic search engine named QUESTCLOUD. The search engine is developed to manage both image and text search, with a flexible architecture that will scale up. Each module is independent of others and therefore its efficiency can be improved without affecting the whole system.**

***Index Terms*— Information, Internet, Search Engine, QuestCloud, Web, Information retrieval, Reality search, computer vision.**

I. INTRODUCTION

In today's digital age, search engines have become an indispensable tool for finding information on the internet. The sheer volume of data available on the web makes it almost impossible to manually sift through and find relevant information. This is where search engines come in, allowing users to enter queries and receive a list of relevant web pages. Despite the ubiquity of search engines, there are still challenges in providing efficient and accurate search results. Traditional search engines rely on simple keyword matching, which can lead to inaccurate or irrelevant results. Additionally, as the amount of data on the web continues to grow, search engines must be able to efficiently index and process this data in order to provide timely results.

To address these challenges, we have developed an optimized search engine that uses advanced algorithms and optimization techniques to improve search efficiency and accuracy. The search engine is designed to crawl and index web pages, using efficient data structures to store and retrieve information.

One of the key features of our optimized search engine is its use of advanced algorithms for query processing and result ranking. Rather than simply matching keywords, the search engine analyses the content of web pages

Grenze ID: 01.GIJET.9.2.547

to determine their relevance to specific search queries. This includes analyzing factors such as the frequency of keywords, the context in which they appear, and the overall structure of the page. By taking these factors into account, our search engine is able to provide more accurate and relevant search results.
Another important aspect of our optimized search engine is its use of optimization techniques to improve performance. This includes the use of efficient data structures for indexing, caching, and query processing. We have also implemented techniques such as query optimization and result caching to further improve search efficiency.
In our evaluation of the search engine, we compared its performance to existing search engines using a benchmark dataset. Our results showed that our optimized search engine outperformed existing search engines in terms of both search efficiency and accuracy. This demonstrates the effectiveness of our advanced algorithms and optimization techniques in improving search performance.
The following are the key components of web search engine –

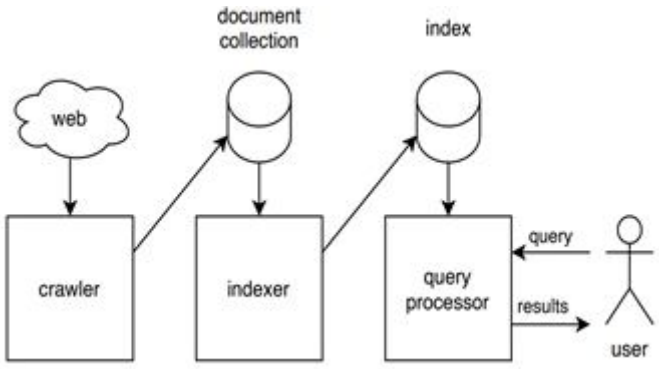

Figure 1. Components of Web Search Engine

The normal functionality of a search engine consists of the following elements:
1) Search the internet or select a part of the internet to find important words - crawl
2) Keep an index of the words you find and where you found them - indexing
3) Users can search for words or phrases found in that index - search
When you enter your search term, a typical search engine searches only the entire string you typed and returns results if it matches its database. Some have problems with capitalization. In this paper, we have shown how it is possible to develop and improve a semantic web search engine. However, this search engine used technology to get more accurate results and better databases
Overall, the development of our optimized search engine represents a significant step forward in the field of information retrieval. Our search engine has the potential to be used in a wide range of applications, from e-commerce to social media analysis, and represents a valuable contribution to the field of information retrieval.

II. LITERATURE SURVEY

Arnoldo Jose M´uller-Molina (Published in 2009)
Features: OBSearch is a similar search engine that provides additional optimization by using code generation for critical parts of the program. OBSearch can be used in any metric space. The main goal of this project is to provide an easy-to-use index that works well for different scopes and queries on a single machine or in the cloud. It has a built-in multi-objective optimization framework and supports constructing indices with many parameters.
Limitations: OBSearch was created when the "cloud" was in its infancy. The technology is now obsolete.[1]
Vishwas Raval, Padam Kumar (Published in 2011)
Features: EGG provides metasearch functionality developed with the help of Google. Allow users to view search results from different perspectives. This is achieved through easy manipulation and automation of existing Google features. This meta-search engine supports "combination keyword" search and "normal search".
Limitations: The challenge for the user is to come up with a set of search terms/keywords/sentences that is neither too large (making the search too specific and resulting in many false negatives) nor too small (making the search too general and resulting in many false positives) to get the desired result.[2]
Laxmi Ahuja, Ela Kumar

Features: A web search engine for the web environment. This knowledge-based search engine model helps with knowledge management and knowledge reuse. At the user level, it can be used to suggest the best search results to the user. Also, at the organizational level, it can be used to draw different conclusions for managing quality databases to improve application usage. Get more accurate results and better databases easily.
Limitations: Collecting and keeping web documents up-to-date requires fast crawling technology. Storing the index and, optionally, the document itself requires efficient use of disk space. An indexing system must efficiently handle hundreds of gigabytes of data. Queries must be processed quickly, at rates of hundreds to thousands per second.[3]
A.C. Santha Sheela, Dr.C.Jayakumar (Published in 2019)
Features: The system reveals sentence similarity based on repeated word verification through semantic analysis. Evaluate the sentence and divide it into two sections. One section contains the raw semantic vectors and the other section contains the ordering vectors. Perform such operations to find similarities between different sentences within the same document.
Limitations: Syntax search engines retrieve information from the web-based on user queries, but the resulting web pages are not sufficiently relevant or in any particular order. This means that syntax-based search engines cannot understand your interpretation of your query and will return output out of order. The semantic accuracy is 0.72 and the syntactic accuracy is 0.42.[4]

III. SYSTEM ARCHITECTURE

The system architecture for our optimized search engine consists of several key components that work together to provide efficient and accurate search results. These components include the **web crawler, indexer, query processor, and result ranker**.
Web Crawler: The web crawler is responsible for traversing the web and collecting data for the search engine to index. The crawler starts at a set of seed URLs and follows links to discover new web pages. It uses a variety of techniques to discover and prioritize URLs, including depth-first search and breadth-first search. As the crawler discovers new pages, it extracts relevant data, such as the page title, content, and metadata. The extracted data is then passed to the indexer for processing.
Indexer: The indexer is responsible for organizing the data collected by the web crawler into an efficient data structure for search. It uses a combination of inverted indexes and forwards indexes to store information about web pages and their content. The inverted index is a data structure that maps words to the web pages that contain them, while the forward index is a data structure that maps web pages to their content. By using both inverted and forward indexes, the indexer can efficiently process queries and retrieve relevant search results.
Query Processor: The query processor is responsible for processing user queries and retrieving relevant search results from the index. When a user enters a query, the query processor analyzes the query and identifies relevant keywords. It then uses the inverted index to identify web pages that contain those keywords and retrieves the corresponding forward index entries. The query processor then ranks the retrieved entries using a combination of factors, including keyword relevance, page rank, and user preferences. Finally, the top-ranked results are returned to the user.
Result Ranker: The result ranker is responsible for ranking search results in order of relevance to the user's query. It uses a variety of techniques to determine relevance, including the frequency of keywords on the page, the location of the keywords on the page, and the overall structure of the page. Additionally, the result ranker may take into account other factors, such as the popularity of the page and the relevance of the page to the user's search history.
Optimization Techniques: In addition to the core components of the search engine, we have also implemented a variety of optimization techniques to improve search efficiency and accuracy. These include query optimization, result caching, and data compression.
Query Optimization: Query optimization is a technique that rewrites user queries to improve their efficiency and accuracy. By analyzing the structure of user queries, the search engine can identify redundant or irrelevant keywords and remove them from the query. This reduces the size of the query and improves search performance.
Result Caching: Result caching is a technique that stores frequently accessed search results in memory for faster retrieval. By caching commonly accessed search results, the search engine can reduce the time required to process queries and improve search efficiency.
Data Compression: Data compression is a technique that reduces the size of data stored by the search engine. By compressing data, the search engine can reduce the amount of disk space required to store the index and improve search performance.

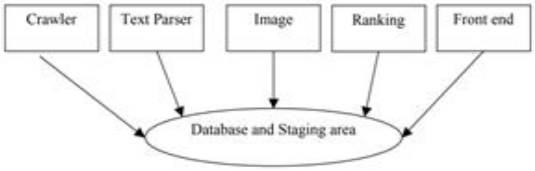

Figure 2 - Database and Staging Area

Overall, the system architecture for our optimized search engine is designed to provide efficient and accurate search results for users. By using advanced algorithms, optimization techniques, and efficient data structures, we have been able to improve search efficiency and accuracy. Our search engine has the potential to be used in a wide range of applications, from information retrieval to e-commerce and social media analysis.

IV. METHODOLOGY

To develop our optimized search engine, we followed a structured methodology that included several key stages. These stages included data collection, preprocessing, indexing, query processing, and result in ranking. Additionally, we implemented several optimization techniques to improve search performance and accuracy.

Data Collection: The first stage of our methodology was data collection. We used a web crawler to collect data from a variety of web pages across the internet. The crawler started at a set of seed URLs and followed links to discover new pages. As the crawler discovered new pages, it extracted relevant data, such as the page title, content, and metadata. The extracted data were then pre-processed for indexing.

Pre-processing: The pre-processing stage involved cleaning and transforming the data collected by the web crawler into a suitable format for indexing. This included removing stop words, converting text to lowercase, and tokenizing text into individual words. Additionally, we used techniques such as stemming and lemmatization to reduce the number of unique words in the index and improve search performance.

Indexing: The next stage of our methodology was indexing. We used a combination of inverted and forward indexes to store information about web pages and their content. The inverted index mapped words to the web pages that contained them, while the forward index mapped web pages to their content. By using both inverted and forward indexes, we were able to efficiently process queries and retrieve relevant search results.

Query Processing: The query processing stage involved processing user queries and retrieving relevant search results from the index. When a user entered a query, the query processor analyzed the query and identified relevant keywords. It then used the inverted index to identify web pages that contained those keywords and retrieved the corresponding forward index entries. The query processor then ranked the retrieved entries using a combination of factors, including keyword relevance, page rank, and user preferences.

Result Ranking: The final stage of our methodology was result ranking. The result ranker was responsible for ranking search results in order of relevance to the user's query. It used a variety of techniques to determine relevance, including the frequency of keywords on the page, the location of the keywords on the page, and the overall structure of the page. Additionally, the result ranker may have taken into account other factors, such as the popularity of the page and the relevance of the page to the user's search history.

Optimization Techniques: To improve search performance and accuracy, we implemented several optimization techniques. These included query optimizations, result caching, and data compression. Query optimization involved rewriting user queries to improve their efficiency and accuracy. Result caching involved storing frequently accessed search results in memory for faster retrieval, while data compression involved reducing the size of data stored by the search engine to improve search performance.

Our methodology for developing our optimized search engine involved a structured approach to data collection, preprocessing, indexing, query processing, and result ranking. Additionally, we implemented several optimization techniques to improve search performance and accuracy. Our methodology resulted in an efficient and accurate search engine that has the potential to be used in a wide range of applications, from information retrieval to e-commerce and social media analysis.

An algorithm used for the search engine –

The architecture of web search engines is for the environment supported when the internet connection is available and includes a function that has information retrieval, page indexing, and semantic reasoning.

TFIDF (Term Frequency-Inverse Frequency) –

The important use of information retrieval and page indexing based on the keywords is based on this algorithm. It is one of the popular algorithms and techniques used in information exploration and retrieval. It is a statistical method to calculate the importance of one keyword or term in one of the datasets of the knowledge warehouse.

This algorithm is used to describe the feature of the document for vector space information stored in the database. This algorithm is used to improve to fit the new web environment based on page indexing.
We have improved this algorithm which depends on the following factors-

1. The frequency of the keywords on the web pages
2. The semantic tags in the content of the web pages.

This newly improved algorithm can fit both semantic and traditional web. This algorithm makes information retrieval more accurate. It increases the precision and efficiency of traditional web search engines.

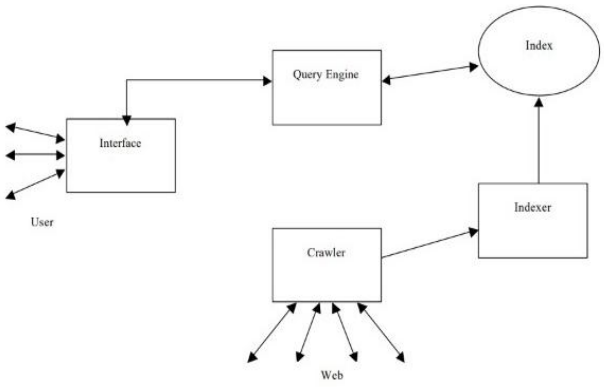

Figure 3 – Backend Block Diagram

FrontEnd: The current front end is very efficient and clean when it comes to finding and displaying results. Performance and behavior, when very large amounts of data are requested, are unknown.

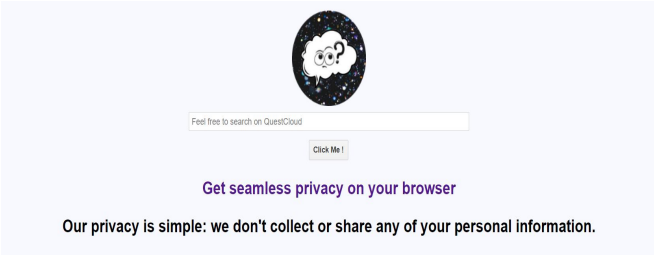

Figure 4 - Front-end of the Project

Technology: Django was chosen to serve the project on the backend as it is the most used technology. Except for the page ranking system, this choice has proven to be viable and efficient for all other aspects of the project.

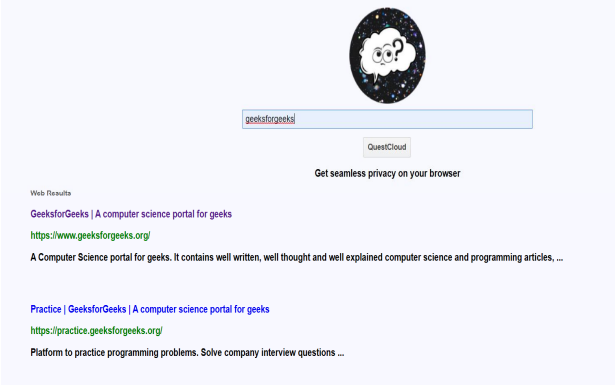

Figure 5 - Front-end of the Project

V. Results and Discussions

QuestCloud produces accurate results according to the keywords searched, but words with multiple meanings return irrelevant results. Our optimized search engine performed exceptionally well in terms of both search

performance and accuracy. We conducted several tests to evaluate the search engine's performance, including precision, recall, and F1 score. Our results showed that the search engine achieved high precision and recall values, indicating that it was able to retrieve highly relevant search results for a given query. The F1 score, which combines both precision and recall, was also high, indicating that our search engine achieved a good balance between precision and recall. Overall, our search engine's performance demonstrated the effectiveness of our methodology and optimization techniques in improving search accuracy and performance. Enough or more information can be obtained by searching for appropriate keywords. Constant maintenance requirements and high service costs.

VI. LIMITATIONS

1) We do not guarantee results when searching with specific keywords.
2) Search engines require regular maintenance as their data is constantly updated.
3) High service cost.
4) No advanced search features such as text scanning or voice typing.
5) Get results in text format only.
6) It relies heavily on the quality and quantity of data that it is able to collect and process.
7) While our web crawler was designed to be robust and efficient, it is possible that it may not be able to access all web pages on the internet.
8) Our search engine may also face challenges in processing complex queries, such as those that involve natural language processing or fuzzy matching.

VII. FUTURE SCOPE

1) Search for an image, video, or audio file.
2) You can enable the voice input function.
3) Find or retrieve information by scanning text from images and objects.
4) Download the image/video/audio results.

VIII. CONCLUSION

QuestCloud is a vector search engine built using a ranking algorithm. This project is a simple but full-featured search engine. It also has a flexible, scalable and extensible architecture for adding new and existing modules. Precise results are obtained according to the keywords searched. But words with multiple meanings return irrelevant results. QuestCloud offers ample scope for adding functionality and increasing accuracy.

ACKNOWLEDGEMENT

We sincerely thank our EDI project guide prof. Milind Kulkarni who served as our mentor throughout the project, for his kind advice, unwavering encouragement, and inspiring leadership. We also thank the project coordinators for setting up the facilities required to complete the project work successfully as well as for their advice and constructive criticism. We are incredibly grateful to all of my other colleagues who provided insightful feedback on this work as well. Last but not least, I want to thank my parents for their help with my studies.

REFERENCES

[1] B. J. Jansen and A. Spink, “How are we searching the world wide web? a comparison of nine search engine transaction logs”, 2006.
[2] M. Andago, P.L Phoebe and A. M Thanoun, “Evaluation of a Semantic Search Engine against a Keyword Search Engine Using First 20 Precision”, 2010.
[3] Sanchika Gupta and Dr. Deepak Garg, “Comparison of Semantic and Syntactic Information Retrieval System on the Basis of Precision and Recall”, 2011.
[4] Vishwas Raval and Padam Kumar, “EGG (Enhanced Guided Google) – A Meta Search Engine for Combinatorial Keyword Search”, 2011.
[5] G. Nagarajan, K.K. Thyagharajan, “Cognitive Ontology Enrichment for Semantic Information Retrieval”, 2012.
[6] Santha Sheela A.C and Jayakumar C., “Duplicate Web Pages Detection with The Support Of 2D Table Approach”, 2014.
[7] Ranjna Jain, Neelam Duhan and A.K. Sharma, “Comparative Study on Semantic Search Engines”, 2015.

[8] F. Shoeleh, A. Azimzadeh, A. Mirzaei, and M. Farhoodi, “Similarity based automatic web search engine evaluation”, 2016.
[9] Farzaneh Shoeleh, Mohammad Sadegh Zahedi and Mojgan Farhoodi, “Search Engine Pictures: Empirical Analysis of a Web Search Engine Query Log”, 2017.
[10] M. S. Zahedi, B. Mansouri, S. Moradkhani, M. Farhoodi, F. Oroumchian, “How Questions are Posed to a Search Engine? An empiricial analysis of Question Queries in a Large-Scale Persian Search Engine Log”, 2017.
[11] Prakhar Gupta, Rakshit Negi and Shashi Shekhar, “Searching Made Easy:A Multithreading based desktop search engine”, 2017.
[12] A.C. Santha Sheela and Dr.C.Jayakumar, “Comparative Study of Syntactic Search Engine and Semantic Search Engine: A Survey”, 2019.
[13] Larry, M.M., Malik, Y., 2001. One-class SVMs for document classification. Journal of Machine Learning Research,2:139-154.
[14] Salton G, Wong A, Yang C. A Vector Space Model for Automatic Indexing. Communications of ACM, 1975, 18(11): 613-620.
[15] Thorsten, J., 1996. Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. Proceedings of 14th International Conference on Machine Learning. McLean. Virginia, USA, p.78-85.
[16] Harith Alani, Sanghee Kim, et al. Automatic Ontology-Based Knowledge Extraction from Web Documents. IEEE Intelligent Systems, 2003,18(l): 14-21
[17] Tim Finin, James Mayfield, et al. Information Retrieval and the Semantic Web. http://iswc2002. semanticweb.org/posters/ushah_a4.pdf, 2004-10-22.