As per UGC guidelines an electronic bar code is provided to seure your paper

International Journal for Modern Trends in Science and Technology, 7(12): 134-136, 2021
Copyright © 2021 International Journal for Modern Trends in Science and Technology
ISSN: 2455-3778 online
DOI: https://doi.org/10.46501/IJMTST0712023
Available online at: http://www.ijmtst.com/vol7issue12.html

Search Engine for Wikipedia Corpus

Anshuman | Vibhor Sharma

Department of Information Technology, Maharaja Agrasen Institute of Technology, New Delhi, India.

To Cite this Article
Anshuman and Vibhor Sharma. Search Engine for Wikipedia Corpus. *International Journal for Modern Trends in Science and Technology* 2021, 7 pp. 134-136. https://doi.org/10.46501/IJMTST0712023

Article Info
Received: 31 October 2021; Accepted: 05 December 2021; Published: 08 December 2021

ABSTRACT

The growing popularity of Knowledge engine project that is google like search engine under Wikipedia grabbed my attention towards this project. The main objective of this project to provide a better search result on Wikipedia using Best algorithms to search and retrieve the file and pages from the Wikipedia Corpus. The aim of the project is to build a scalable and efficient search engine using Wikipedia Corpus. This project will inculcate a depth knowledge of working of search engine, various page ranking algorithms, implantation of multiword and multifield search on Wikipedia corpus, thus resulting in interactive interface for Wikipedia search result and efficient search result so that the user need not to wonder over more than 5 – 20 references for his searches. This project will contribute to application of search engine and its working in the improved way. The internal search is also able to search all pages for project purposes, whereas external search engines cannot be used on any talk page, a large part of project space, and any page.

KEYWORDS: Search Engine, Wikipedia Corpus, searching query

INTRODUCTION

Thisproject involves building of scalable and efficient search engine on the Wikipedia Data Dump without using any external index. For this project data dump of Wikipedia is used. The search engine returns in real time. **Multiword and multi field search on Wikipedia corpus is implemented**.

METHODOLOGY

The growingpopularity of Knowledge engine project that is search engine for Wikipedia has grabbed the attention to achieve a better search result for the Wikipedia feed. This project is based on the Working of search engine, creation of effective inverted index, implementation of multiword and multi-field search on Wikipedia corpus, implementation of better ranking algorithm , better sorting for algorithm for merging indices , Use *of better NLP algorithm to for language processing and bringing out meaningful words* .

Thus creation of scalable and efficient search engine on data dump of Wikipedia without any external index creation requires several steps.

1. **XML Parsing** over Wikipedia page/es is the primary task in-order to retrieve data and create inverted index. The library that can be used for this purpose is SAX parser. This will help in parsing over the corpus and bring out the content of the Wikipedia page and segregate the page wise in the form of title, body, Link, references etc. This parsing over the page needs to be efficient in order to fetch the content properly thus enabling the further NLP algorithms to run on it.

2. **Tokenization** It is the process of separating a piece of words into a token Here tokens can be either words, characters or sub-words. Hence tokenization can be broadly classified into 3 types – word character and sub-word (in gram characters) tokenization. This can help in separating contents that is in the form of sentence to be broken down into small words that is called tokens thus There are various ways for tokenization, the simplest way to tokenize the string is to use the whitespace in the string as a delimiter of words. There are various challenges to tokenize the word simply based on white spaces , so tokenizer implements a variety of rules to tokenize the English words. It separates the phrase terminating with punctuation like (!?.:,) from adjacent tokens besides this it contains rules for English contractions . This can be done by python library NLTK, genism.
3. **Case-Folding** It is a way of converting all letters to smaller letter. This conversion of tokens into smaller case reduces the index size as both capital and small letter word will be same. But this comes with the challenges that some of the words that are noun which is always in capital letter can not be separated from the its other use.
4. **Stop Word Removal** In natural Language processing the useless word(data) is referred as Stop words . These words has to be removed in order to reduce the size of inverted index that will help in reducing the search space. Stop words are present in abundance in English language , by removing these words , we remove the low level information from the text in order to give the more importance to the most significant words . So removal definitely reduces the dataset size thus reduces the training time of due to less number of token involved in the process.
5. **Stemming** It is the process of producing the morphological variant of root/base word . Stemming programs is commonly referred to as stemming algorithm or stemmers . The stemming algorithms reduces the words like “chocolates”, “chocolaty”, “choco” to the root word “chocolate” and “retrieve” , “retrieval” , and “retrieves” reduces to stem “retrieve” .It is a part of linguistic studies in morphology and artificial intelligent , information retrieval and extraction Stemming and AI knowledge extract the information from the vast sources like big data or Internet since the additional form of words related to the subjects may need to be searched to get the best result It is a part of query and search engine .
Lemmatization It is the process of reducing the number of the words into a single word by combining the common together . It is the process of transforming the dictionary base form .Its library contains the lexical database for English words .These words are linked together based on their semantic relationship. The python library available for this is NLTK .
6. **Inverted Index creation** : An inverted index is an index data structures storing a mapping from content , such as words or numbers , to its locations in a document or a set of documents . In simple words , it is a hash map like data structure that directs you from a word to a document or a web page . It consists of documents that stores the information about the word and its occurrences. These document are later parsed and used for processing the query. When inverted index is ready we need to merge these documents to get inorder to locate the occurrence of word in terms of document number , the field of document in which it is occurring in other words all the locations where this word has occurred need to be stored in a fashion that will be both space and time efficient while retrieving it . for example Sachin d1-t1b2d7|d5-t1|d6-t2b3c5
This type of encryption will be efficient in retrieving the content .
7. **Scalable Inverted Index creation** : The main challenge over here is to build huge index with the limited memory . Thus this can be achieved by
- Building a local inverted index
- Merge Local Inverted index
- Obtain Larger Inverted Index

The building of local inverted index and merging the can be achieved by merge sort create a hash-table like data structure .
This hash table will store the word corresponding to its locations and occurrences in the table.
8. **Ranking Algorithm :**Rather than returning the set of documents that satisfy a query expression , it is expected to return top K documents that satisfy the

query .To implement ranking , store the frequencies in the posting list . The more frequent the query term in the document the higher the ranking of that document should be . Since the field is being stored in the document where terms occur and have different weightage for term occurring in title, info box , body , references.The document with the query term is occurring is expected to be more important than the document with the term in body. Ranking algorithm needs to be efficient in terms of ranking the ranking the search result , for ranking based on frequency the term frequency count is used that is used to store the frequency of the element in its occurrences in the posting list .

9. **Search Query**The types of query supported is of both type either the field wise query like defined title , body , and info or the simple query of string containing related word .

FUTURE SCOPE

Its popularity can be explained by the wide use of various search engines like google and yahoo and its use in day to day life for information gathering. This will make the user to stay on the Wikipedia for searching and information gathering. The page rank algorithm and various optimization in the algorithm will help to give the best search results. Thus improved search result will increase user experience in a better way. Wikipedia uses a powerful search engine, with a search box on every page. The search box will navigate directly to a given page name upon an exact match. But, you can force it to show you other pages that include our search string by including a tilde character ~ anywhere in the query. The maximum search string is 300 characters long.However, search can instantly search all pages on the wiki when the search is kept to a simple word or two.Wikipedia's searches can be made domain-specific (i.e., search in desired namespaces). The search engine also supports special characters and parameters to extend the power of searches and allow users to make their search strings more specific.

CONCLUSION

This project facilitates the query in following manner , the query is processed parsed and is given to the respective query handler that is in form of simple or field . The word is searched in all the document using the inverted index and the results are ranked accordingly and appropriate results are shown after process the query . The documents are ranked on the basis of TF- IDF scores . The preferences of various categories are taken into consideration too , thus the result occur after the steps are highly similar to that of expected .

REFERENCES

[1] David Nikolas Milne "A Knowledge Based Search Engine powered by Wikipedia"

[2] W. Bruce, Croft ,Donald Metzler ,Trevor Strohman " Search engine and information retrieval in practice"

[3] WikipediaKnowledge engine source link https://en.wikipedia.org/wiki/Knowledge_engine

[4] "A Mathematical Modeling language". lpl.unifr.ch. Retrieved 2020-07-14.