Building an intelligent search engine

Sivaraman Lakshmipathy
University of Illinois at Chicago
Chicago, Illinois
slaksh5@uic.edu

ABSTRACT

Search engines are ubiquitous in the current web-based environment. They serve the purpose of identifying the best possible responses to user queries based on numerous factors. This report discusses the construction of an intelligent search engine confined to the UIC domain. The different components of the search engine are discussed in the following sections, along with the challenges involved in the process. The search engine is also evaluated using quantitative measures.

KEYWORDS

search engine, retrieval, pagerank

ACM Reference Format:
Sivaraman Lakshmipathy. 2020. Building an intelligent search engine. In *CS 582 - Information Retrieval Project.* , 6 pages.

1 INTRODUCTION

The advent of digitization brought upon a need to efficiently store the information and create efficient techniques to search the data. When the World Wide Web provided a connected view of the world's information in the form of web pages, the domain of search and retrieval encountered a new playing field. While multiple techniques were developed and deployed, Google made giant strides by designing a search engine that would evolve into being synonymous with the act of searching the web. While it remains the pinnacle of search engines, different domains have specific search engines build for different purposes.

This report summarizes the construction of a simple search engine that encapsulates a snapshot of the UIC web domain. The work involved studying the concepts behind the construction of the engine, designing the engine, and identifying components to be enhanced with smartness. This resulted in the construction of Wizard: An Intelligent Search Engine. Wizard crawls the web pages in the UIC domain, indexes the contents of the web pages, and generates a reverse lookup mechanism to identify the relevant documents for the search queries. The process of construction, the challenges involved, and the evaluation are discussed in the following sections.

2 COMPONENTS

Wizard, the search engine, is built to crawl through the web pages and index information to generate results for user search queries. At this point, the search engine is configured to enumerate information in the UIC domain only. The major components of Wizard are discussed below.

2.1 Crawler

The crawler encapsulates the act of crawling the web pages, downloading the contents of the pages, and adding the linked web pages to the web graph. It traverses the web pages in the UIC domain using a breadth-first strategy, beginning with a configured start page. The configuration also allows the modification of the maximum limit of pages to be crawled. The engine was evaluated by crawling 10,000 valid pages.

During the traversal, the outbound links in the current web page are recorded in the web graph as the destinations for the source web page. The web page URL is hashed to generate unique hash codes as identifiers. This would allow the crawler to identify and ignore duplicate URLs and links. The URLs are converted to their canonical versions before being hashed. Furthermore, the crawler also seeks out the crawler rules available in 'robots.txt' and abides by the specifications.

The crawler also persists the contents of the web pages in order to be utilized by the other components.

2.2 HTML parser

Once the contents of a page are retrieved, they are processed to extract the contents. This component handles the extraction operations. The title, contents, and the hyperlinks are extracted from the HTML contents using the BeautifulSoup library.

2.3 Text Processor

This component handles all the text processing operations. All the documents and search queries undergo the following transformation in the given order: *tokenization, punctuation, and stop words removal, stemming*.

2.4 Indexer

This component is responsible for building the inverted index retrieval model for the engine. It processes all the persisted web page contents individually, extracts the tokens using the Text Processor component, and builds up the inverted index data structure for the unique terms in the vocabulary. This data structure contains the list of related documents for each term along with the corresponding term frequencies. A separate data structure holds the document length. These data structures are persisted in memory to be used while retrieving results based on user queries.

2.5 PageRank

This component utilizes the ideas from the original PageRank[1] paper to build a ranking scheme. The PageRank algorithm calculates the importance of a web page by modeling a random surfer and identifying the probability of the surfer visiting the web page.

The web graph is built during the crawling process. The nodes and the links are added to the graph and represented using an adjacency list representation. As outlined in the paper, each node in the graph is assigned an initial score equivalent to the inverse of the total number of nodes in the graph. The PageRank algorithm is run for a fixed number of iterations, dictated by the number of edges in the graph, to generate the updated scores. These scores are used to rank the retrieved web pages in the retrieval phase. The final graph is persisted in the memory to be used during document retrieval.

2.6 Topic Modelling

The second smart component is built using the idea of topic modeling. Topic Modelling is the task of discovering and extracting topics from a data corpus. This would allow the corpus to be grouped under the relevant topics and, as such, help in the retrieval process by identifying the topic relevant to the user queries.

Latent Dirichlet Allocation is a modeling technique that is used in this component. The contents of the corpus are converted to a bag of words representation and trained for a fixed number of iterations to generate the final model. During this process, various values were experimented upon to identify the ideal number of topics to be generated from the corpus. A value of 20 topics was chosen based on the coherence score. During the information retrieval process, the user query is provided as an input to this component. The persisted model identifies the relevant topic for the search query and expands it by including the top 5 words in the topic. This allows the engine to expand the search into relevant documents based on the current context.

2.7 Query Processor

This component handles all the user queries and fetches relevant results. Each new query is processed by each of the following smaller components to fetch the final results.

2.7.1 *Query expansion*.

Using the Latent Dirichlet Allocation model generated earlier, each query is expanded upon by identifying words that are similar in context to the search terms.

2.7.2 *Retrieval*.

Each new user query is compared against the retrieval model generated earlier using one of the following similarity measures: Inner Product, Dice similarity, Cosine similarity, and Jaccard similarity. This is a user-configurable parameter, and the default similarity measure utilized by the engine is Cosine similarity. Two sets of results are generated for the original query and the expanded query.

The results are stored in a local cache to enable pagination in the user interface. A subset of the results, bounded by a count of 10, is available to the user at a given time, and the user can navigate to the previous or next set of results for the same query.

2.7.3 *Ranking*.

The retrieved documents are ranked using the harmonic mean of the similarity measure and the PageRank scores, in a decreasing manner.

2.8 Loader

This component loads all the persisted models to perform the search and retrieval process. The web graph with PageRank scores, inverted index retrieval model, and the Latent Dirichlet Allocation model is loaded from memory. If they are unavailable, this component prompts the user to regenerate the required models from scratch.

2.9 Interface

The engine interacts with the end-user via an interface. Two options are available: text-based interface or Graphical User Interface. When the user enters a new query, the retrieved results are populated for viewing. The Graphical User Interface also features pagination to navigate between the results.

3 CHALLENGES

The design and construction of components of the search engine encountered significant challenges. They contributed to the evolution of the design process significantly. Significant challenges are discussed below.

- Resources: The crawler component performs a traversal through the web pages in a breadth-first manner. The crawling was resource-intensive in terms of space and time. For instance, crawling the 10,000 valid pages utilized three hours. Thus, it was critical that the resources weren't utilized in crawling and processing the same web page multiple times. This was handled by detecting cycles in the web graph and avoiding them.
- Crawling rules: One of the major design decisions was to identify the crawling rules by looking for robots.txt for the URLs being crawled. However, it quickly became apparent that there may be instances where the rules were unavailable or inaccessible during crawling. Thus, the crawler had to skip entire batches of URLs if the rules were unavailable.
- URL normalization and hashing: It was observed during crawling that the same URL could be represented in multiple forms with differences in the URL scheme and anchors within the same web page. To avoid processing the same URL twice, the URL is normalized to extract the root version, and a unique hash value is generated as its identifier.
- File types: Each web page is processed to extract all the outgoing links. These links may lead to downloadable content and media files. To handle them, only the outgoing links that were HTML pages were crawled.
- URL accessibility: A significant number of outgoing links were unavailable due to their absence or other network-related issues. Few web pages required authentication to allow access. Such URLs were ignored by the crawler, based on specific status messages received on accessing them. The major accessibility issues recorded were related to HTTP errors, Timeout errors, SSL, and certificate related errors.
- Persistence design choices: When crawling the web pages, the contents are downloaded for indexing purposes. Since the indexer works only with the text content in the file, a design choice was made to extract only the text content from the downloaded pages for persistence purposes. This would lead to losing the structure of the web page but leads to reduced space requirements for persistence.
- Topic modeling: To extract the topics in the corpus, LDA modeling is utilized. The number of topics to be identified

Similarity measure	Search results
`Inner Product`	https://www.evl.uic.edu/category.php?id=3&related=3 https://www.evl.uic.edu/category.php?id=27\&related=3 https://www.evl.uic.edu/category.php?id=19&related=3 https://www.evl.uic.edu/category.php?id=9&related=3 https://www.evl.uic.edu/category.php?id=3&related=4 https://www.evl.uic.edu/category.php?id=27&related=4 https://www.evl.uic.edu/category.php?id=9&related=4 https://www.evl.uic.edu/related.php?id=21&related=3 https://www.evl.uic.edu/category.php?id=19&related=4 https://accc.drupal.uic.edu/classroom/all
`Dice coefficient`	https://www.cs.uic.edu/~brents/cs594-dcn https://housing.uic.edu/apply https://housing.uic.edu/halls https://www.evl.uic.edu/entry.php?id=723 https://today.uic.edu/from-retail-space-to-learning-place http://ocle.uic.edu https://housing.uic.edu/halls/tbh https://www.evl.uic.edu/entry.php?id=365 https://dining.uic.edu/locations/uh https://businessconnect.uic.edu/undergraduate
`Cosine coefficient`	https://www.cs.uic.edu/~brents/cs594-dcn https://housing.uic.edu/apply https://accc.uic.edu/services/infrastructure/network/residence-hall-network https://accc.uic.edu/service/resident-hall-network-res-net http://m.uic.edu/map/list https://housing.uic.edu/halls https://today.uic.edu/from-retail-space-to-learning-place http://ocle.uic.edu https://housing.uic.edu/halls/tbh https://dining.uic.edu/locations/uh
`Jaccard coefficient`	https://www.cs.uic.edu/~brents/cs594-dcn https://housing.uic.edu/apply https://housing.uic.edu/halls https://www.evl.uic.edu/entry.php?id=723 https://today.uic.edu/from-retail-space-to-learning-place http://ocle.uic.edu https://housing.uic.edu/halls/tbh https://www.evl.uic.edu/entry.php?id=365 https://dining.uic.edu/locations/uh https://businessconnect.uic.edu/undergraduate

Table 1: A comparison of search results based on similarity measures for the query: `thomas beckham hall`

is a parameter that had to be identified. Since the corpus is smaller and limited to the UIC domain, a larger number of topics resulted in significant terms like 'science' being exclusive to a single topic. This lead to issues during query expansion where the search terms were significantly smaller than the document lengths. Thus, the LDA model was generated with different values, and the effect on query expansion was experimented upon. Finally, a threshold of 20 topics was identified to provide satisfactory results.

4 RETRIEVAL MODEL DISCUSSION

4.1 Weighting scheme

The search engine utilizes a vector space retrieval model. For each web page content, the indexer component generates an inverted index. This index would contain the term frequency and the inverted document frequency for each word in the vocabulary. The TF-IDF weighting scheme is used as per the standard definition.

$$tf_{ij}idf_i = tf_{ij}\ log_2(N/df_i)$$

Using the TF-IDF weighting scheme works well since this would allow words frequently occurring in the document but rarely in the rest of the collection to have higher weights.

4.2 Similarity measures

Similarity measures help to compute the degree of similarity between two vectors. Since the extract contents are converted to a vector space model, similarity measures can be used to match the user query and the relevant documents. The Wizard search engine supports four similarity measures, namely, Inner Product, Dice coefficient, Cosine coefficient, and Jaccard coefficient.

- Inner Product

$$\sum x_i . y_i$$

- Dice coefficient

$$\frac{2 \sum x_i . y_i}{\sum x_i^2 + \sum y_i^2}$$

- Cosine coefficient

$$\frac{\sum x_i . y_i}{\sqrt{\sum x_i^2 . \sum y_i^2}}$$

- Jaccard coefficient

$$\frac{\sum x_i . y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i . y_i}$$

where x_i and y_i are the weights of the terms in the query and the documents.

It was observed from the results during the evaluation phase that the Cosine coefficient similarity measure performed better and is thus configured as the default similarity measure in the search engine. The similarity measures can also be configured by the end-user from the interface. Table 1 shows the effect of the utilization of the different similarity measures for a sample user query. It can be observed that the cosine coefficient similarity measure results in more relevant results.

4.3 Ranking

The retrieved documents are initially ranked based on their similarity scores. The PageRank scores obtained are used to rank the retrieved documents in order to obtain finer ranked results. In order to calculate the PageRank scores, the web graph constructed during the crawling phase is utilized. The power iteration method[1] is utilized to calculate the scores.

$$Score(A) = (1 - \epsilon) \sum_{B \to A} \frac{Score(B)}{outdegree(B)} + \frac{\epsilon}{n}$$

where A and B are nodes in the graph, and n represents the total number of nodes in the graph. The value of ϵ is adapted from the original paper[1]. The scores of the nodes are updated for a fixed number of times proportional to the log of the total number of edges in the graph.

5 EVALUATION

Once the user enters the search query, the search engine returns two sets of results. The first set contains the results for the original search query. The second set of results is generated using the expanded query constructed by the engine. In order to evaluate the results provided by the search engine and to compare the effects of query expansion, the similarity measure is fixed as Cosine coefficient. A variety of query inputs were fed as input to the search engine, and the results were obtained. In order to evaluate the results, the results were verified manually and also compared against the search results provided by the Google search engine. Precision was calculated at rank 10 based on the manual evaluation. The recall value wasn't utilized since the set of relevant results is generally unbounded compared with the results from the Google search engine.

A set of 5 sample queries used for evaluation, along with the precision values are presented below:

(1) uic courses offered by mscs department
(2) covid-19 uic
(3) uic pop-up pantry
(4) professor carnelia caragea
(5) graduate student council uic

Query number	Precision (original query results)	Precision (expanded query results)
(1)	0.5	0.3
(2)	1.0	0.1
(3)	0.5	0.6
(4)	0.4	0.3
(5)	0.8	0.0

Table 2: Precision @ Rank 10 results

It can be observed that the precision values for the original queries are higher than the precision values for the expanded query for most scenarios. This observation is elaborated in the following section.

6 RESULTS

Observation of the results indicates that the search engine is limited by the content crawled by the search engine. For instance, search terms that belong to topics that have been crawled extensively returned better results.

One of the major observations from the results leads to the conclusion that the search engine's performance is significantly higher for the original query than the expanded query. However, this can be explained by the lack of unique topics generated from the crawled contents. On studying the web graph, which was generated using a breadth-first technique, many web pages are limited to specific subdomains. In turn, the query expansion tries to match the queries with the limited topics available. Increasing the number of topics does not work since the unique topics are limited for the corpus size of 10,000. Thus, it can be observed that queries with unique

Original query: `neural networks`	Expanded query: `neural network data mine learn confer cs`
https://www.evl.uic.edu/entry.php?id=157	https://www.cs.uic.edu/~liub
https://ecc.uic.edu/career-toolbox/networking	http://www.cs.uic.edu/~liub/WebMiningBook.html
https://careerservices.uic.edu/students/networking	http://www.cs.uic.edu/~liub/WebContentMining.html
https://bioe.uic.edu/undergraduate/bioengineering-major	http://www.cs.uic.edu/~liub/diagnosticDM/diagnosticDM.html
http://bpl.ahslabs.uic.edu	http://www.cs.uic.edu/~liub/professionalAct.html
https://www.evl.uic.edu/entry.php?id=480	http://www.cs.uic.edu/~liub/pastTeaching.html
https://www.evl.uic.edu/entry.php?id=560	https://www.cs.uic.edu/~liub/Bing-Liu-short-CV.html
https://www.evl.uic.edu/entry.php?id=852	http://www.cs.uic.edu/~liub/teach/cs583-fall-07/cs583.html
https://accc.uic.edu/services/infrastructure/network	http://www.cs.uic.edu/~liub/teach/cs583-spring-08/cs583.html
http://networking.accc.uic.edu	http://www.cs.uic.edu/~liub/teach/cs583-spring-07/cs583.html

Table 3: Studying the effects of query expansion

search terms like query (1), (3), and (4) have a comparable performance for both versions of the query. Meanwhile, queries with common or more frequently used words suffer when generalization is introduced due to the mismatch in the relevant topics.

The combined effect of the available crawled pages, and query expansion is presented using an example in table 3. The original set of results retrieved the pages related to networking since the web pages related to the actual coursework were unavailable in the network graph. Here, query expansion helps by adding context to the word 'network' by adding related terms to the query.

Thus, it can be concluded that query expansion would be helpful when the search engine is scaled to include a larger corpus of information.

6.1 Error analysis

6.1.1 *Ideas that worked.*

- Text processing: The text processing component performed stemming of the tokenized words, and removed stop words. This greatly helped during retrieval since the queries underwent the same transformation and could be compared effectively with the documents.
- Efficient crawling: Since the crawling process might encounter cycles in the web graph, it was ensured that the same page wasn't processed more than once. This ensured that the limited resources weren't overused.
- Persistence: The crawled web pages were persisted in the local persistence individually after the contents and outgoing links were extracted. This allowed the indexer and the LDA components to generate the required models without storing the entire data in the execution memory. This ensured that the search engine could scale upwards in terms of the number of pages crawled by the search engine.
- Logger: The search engine is a complex application that utilized multiple components to perform the search and retrieve the relevant results. This resulted in several background operations that occur for each search. It was observed that recording the transformations allowed a better understanding of the design changes during the development phase. Thus, a customized logger was introduced to record them. This also allowed the search engine to record the queries encountered by the engine. This form of logging could help in debugging when building upon the existing engine.
- PageRank: The utilization of the PageRank scores for ranking enhanced the quality of the results returned by the search engine for the query terms.

6.1.2 *Shortcomings.*

- Crawling limit: The search engine crawled a maximum of 10,000 web pages. This presented a limited view of the entire UIC domain, and thus entire sub-domains were missing from the search engine's view, leading to reduced performance for terms related to missing domains.
- Performance: The crawler is currently designed to be executed on a single thread. This lead to a sequential crawling scenario, which was time-intensive. For example, it took 3 hours to crawl the 10,000 pages. This could be handled by converting the crawler to be multi-threaded.
- Limited content: The crawler processes only HTML pages. The different media types are excluded, thus leading to a loss of information. Support for media files could enhance the results of the engine.
- Topic modeling limitations: The number of topics to be extracted from the corpus is limited due to the small size of the corpus. Additionally, further heuristics could be included to identify the topics.
- Ranking limitations: Harmonic mean is utilized to combine the similarity measure and the PageRank scores. This leads to higher weights for PageRank scores when compared to similarity measures in a few cases. This could be handled to generate better ranking results.
- Crawling limitations: The search engine doesn't provide support to check for updated contents in already downloaded documents. This could lead to scalability issues when the search engine needs to look for updated pages.

7 RELATED WORK

The introduction of intelligent components vastly increased the performance of the search engine. This could be expanded upon by studying and adapting similar components. An exciting area of exploration could be to introduce an auto-completion feature for the search engine. This would enable us to obtain user feedback based on whether the suggestions match with the actual user inputs. This could be used to improve the quality of the retrieved results.

Another area of work could be to utilize Part of Speech tagging in the topic modeling component. This could help distinguish the terms with their meaning.

8 CONCLUSION AND FUTURE WORK

Designing the search engine was an enriching experience. Building the separate components and integrating them lead to gaining a significant understanding of the working of information retrieval systems. Some of the future work that could be considered are:

- Multi-thread crawler: The crawler runs on a single thread at the moment, which could benefit from a multi-threaded approach to speed up the process.
- N-grams: The topic modeling and content processing components currently consider only the unigrams in the web pages. This could be extended to consider bigrams and trigrams to index and retrieve documents more efficiently.
- Persistence: With an increase in the number of web pages being crawled, the associated graph and inverted indexes also grow significantly. Currently, the entire data structures are stored as a whole. This could become an overhead with an increase in scale. Thus, a distributed version of the persistence of the data structures could help in the engine's performance.

The vision for the search engine evolved during the development phase. The design of the Topic Modelling component provided an opportunity to study the effects of modeling on the query expansion task. It would be beneficial to study and adopt more techniques to make the search engine more efficient. In conclusion, adapting and evolving Wizard, the search engine, is a long term goal that could lead to great results.

REFERENCES

[1] Sergey Brin and Lawrence Page. 1998. The Anatomy of a Large-scale Hypertextual Web Search Engine. *Comput. Netw. ISDN Syst.* 30, 1-7 (April 1998), 107–117. https://doi.org/10.1016/S0169-7552(98)00110-X