Noogle: An Intelligent Search Engine

Sanjay Ramachandran
University of Illinois at Chicago
Chicago, Illinois
sramac22@uic.edu

ACM Reference format:
Sanjay Ramachandran. 2018. Noogle: An Intelligent Search Engine. In *Proceedings of Information Retrieval Search Engine Project, Chicago, Illinois USA, Fall 2018,* 4 pages.
https://doi.org/

1 INTRODUCTION

From buying stuffs to finding answers to our never ending questions to discovering the meaning of life, Search Engines play a vital role in our daily life. It combines a wide variety of computer science concepts to understand the user query, know where to look for answers and present them in the right sequence to satisfy the information needs of the user. Google's Search has risen to be this undefeatable titan in the field of search engines. This is due to a combination of complex systems that try to do what we see when we search for anything in Google in a matter of seconds. This report presents a newborn search engine aspiring to be Google's Search one day. The report has the following sections (letters inside parenthesis indicate the topics in the project description pdf, that will be discussed in the sections listed): 1) Components(a), 2) Evaluation(d, c), 3) Challenges(b), 4) Intelligent Component and Evaluation(e), 5) Error Analysis(f), 6) Future Work(h), 7) Conclusion

2 COMPONENTS

Noogle is a search engine built for the UIC domain. It includes the following components working together to find the answers for the given queries.

- **User Interface**: Noogle has a command line interface, where users can input a search query and view the results as well. It has options to paginate the results to avoid information overload
- **Web Crawler**: Noogle's web crawler, nicknamed NoSpidey, crawls all the webpages in the given domain in a breadth-first fashion. It first checks the rules in the domain's robots.txt and if allowed, gets the URL and passes it to the HTML parser to parse and extract the required details. It gets back a list of outgoing links from the page, does link canonicalization, removes duplicate links from the list and adds all the remaining links to the back of the link queue. It then creates a node for the current web page in the web graph data structure and includes all the outgoing connections in the adjacency list. After this, it writes the content of the web page to the disk and commits it
- **HTML Parser**: Noogle's HTML parser gets a URL as the input. It then downloads the resource and only if it's a html file, it tries to parse the document. The parser looks for anchor tags and retrieves the outgoing url and converts it to an absolute URL. It also extracts the anchor text which can be a good description of the target page. All outgoing links are then returned by the parser. The parser excludes tags like *'script', 'style', 'link', 'meta', 'header', 'svg', 'path', 'nav', 'form', 'input', 'img', 'footer'*. For all the other tags, it extracts the contents of the tag and writes it to a data structure to be returned to the crawler
- **PreProcessor**: The preprocessor component does functions like tokenization, stemming, stop words removal and returns the input document as a list of tokens
- **Indexer**: The indexer has a hash based key-value store to hold the inverted index for Noogle's retrieval model. It also has another similar data structure to store the document length for all the crawled documents to be used in the similarity calculation. These two stores will later help in representing the documents and queries as vectors for Noogle's vector space retrieval model. It takes a tokenized document as an input and updates the inverted index by storing the term frequency of all the token in the document
- **Retrieval Model**: Noogle implements a Vector based retrieval model. In this model, documents and queries are represented as vectors. The vector components are the tf-idf weights of the tokens in the document. Four different similarity measures are available to calculate the similarity between the query and the documents. Noogle provides, cosine similarity, inner product measure, dice coefficient and jaccard coefficient similarity
- **Ranking Component**: Noogle follows the footsteps of it's master, Google, in that it uses the famous PageRank algorithm for ranking the results returned by the retrieval model. Since, it's an offline algorithm, the module is triggered by NoSpidey, once it completes crawling the web pages. Once the pagerank scores converge, it is available for the Query Processor to use in ranking the results
- **Query Expansion**: This component is used in query expansion to get words that are related to the given query terms. It takes the entire corpus and creates clusters of co-occurring words also called Topics using the LDA algorithm. The Query Processor uses this component internally to expand the search span for related documents
- **Query Processor**: The query processor receives the user query as an input. It then preprocesses the query using the same component that was used to preprocess the crawled documents. Once the processed tokens are returned, it uses the query expansion component to get additional related terms, calls the retrieval model to get relevant documents and then ranks them based on the harmonic mean between the

Information Retrieval Search Engine Project, Fall 2018, Chicago, Illinois USA
2018. ACM ISBN .
https://doi.org/

similarity score and PageRank score of each document. The component provides options to select the similarity measure and also outputs the results got by using only the similarity measure, similarity with pagerank scores and similarity, pagerank along with query expansion
- **Loader**: This is a small component that loads the pre-trained models, page rank statistics and the inverted index from binary files. Thus, there is no need to run all the steps every time Noogle is executed

3 EVALUATION

Various analysis of the query results returned by Noogle were performed. The queries used for evaluation are as follows:

(1) uic fall 2017 cs graduate requirements
(2) computer science uic
(3) uic cs job fair
(4) professor cornelia caragea uic
(5) uic research labs

The following results are obtained by calculating the precision at rank 10. The relevant documents are obtained by searching with the same query and selecting the results that are in both the Google's search results and Noogle's crawled documents.

A quick manual evaluation of the results indicate that there are multiple results that point to the same page. Query number 5 had a pretty good precision since all results were unique.

Query Number	Precision@10
1	0.5
2	0.4
3	0.6
4	0.4
5	0.8

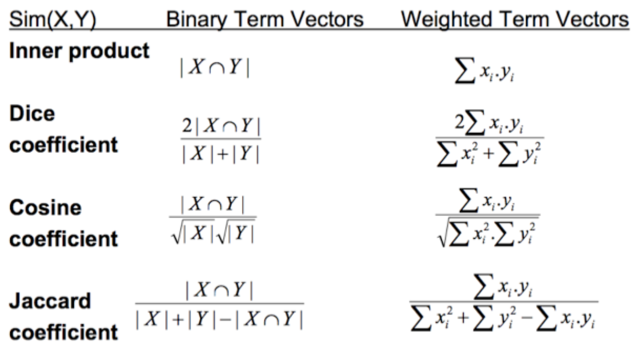
Table 1: Precision@10 for the queries

The weighting scheme used in Noogle is the tf-idf weighting scheme. The formula is as follows:

$$tfidf_{ij} = tf_{ij} * log_2(N/df_i)$$

where tf_{ij} is the frequency of term i in document j, N is the corpus size and df_i is the document frequency of term i.

The different similarity measures implemented in Noogle are described in the below image.



Sim(X,Y)	Binary Term Vectors	Weighted Term Vectors
Inner product	$\|X \cap Y\|$	$\sum x_i \cdot y_i$
Dice coefficient	$\frac{2\|X \cap Y\|}{\|X\|+\|Y\|}$	$\frac{2\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2}$
Cosine coefficient	$\frac{\|X \cap Y\|}{\sqrt{\|X\|}\sqrt{\|Y\|}}$	$\frac{\sum x_i \cdot y_i}{\sqrt{\sum x_i^2 \cdot \sum y_i^2}}$
Jaccard coefficient	$\frac{\|X \cap Y\|}{\|X\|+\|Y\|-\|X \cap Y\|}$	$\frac{\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i \cdot y_i}$

Among these cosine similarity outperforms inner product and it performs slightly better than dice and jaccard coefficients. The table-2 shows the difference in the results obtained using the different similarity measures.

Measure	Results
cosine	https://www.cs.uic.edu/courses https://www.cs.uic.edu/Main/Courses http://logos.cs.uic.edu/reed https://www.cs.uic.edu/undergraduate-programs https://www.cs.uic.edu/Main/UndergraduatePrograms https://www.cs.uic.edu/cs-minor https://www.cs.uic.edu/Main/ComputerScienceMinor http://www.cs.uic.edu/Theys http://www.cs.uic.edu/Troy https://www.cs.uic.edu/graduate-programs
Inner Product	https://www.cs.uic.edu/graduate-programs/cs-graduate-program-degree-requirements https://catalog.uic.edu/gcat/graduate-study/graduate-study https://www.cs.uic.edu/graduate-programs https://catalog.uic.edu/ucat/colleges-depts/liberal-arts-sciences https://www.cs.uic.edu/undergraduate-programs https://www.cs.uic.edu/Main/UndergraduatePrograms https://www.cs.uic.edu/bachelor-of-science-in-cs-human-centered-computing-concentration http://www.cs.uic.edu/Theys https://catalog.uic.edu/ucat/colleges-depts/business-administration http://registrar.uic.edu/current$_{s}tudents/calendars/final - exams$
dice and jaccard	The 3 unique results are https://today.uic.edu/uic-named-a-green-ribbon-school-sustainability 20170928-img_1209 http://nlp.cs.uic.edu/component/content/article/4-news/55-rachel-harsley-defends-her-phd-march-1-2017 https://www.cs.uic.edu/registration-restrictions-for-current-cs-gradu

Table 2: Results obtained for query (1) using different similarity measures

4 CHALLENGES

Building Noogle wasn't an easy task given the complexity of each component and the irregularities in the modern web structure. Some of the challenges faced are listed below:

- Some web pages are invalid, taken down or redirected resulting in junk service messages being written to the corpus
- Few domains had certificate issues while trying to access them resulting in a SSL error
- Some web servers were unresponsive resulting in a Timeout error
- robots.txt was not available for certain domains or resulted in a timeout or ssl error when trying to download it. This resulted in a situation where the crawler had to completely

ignore domains for which robots.txt couldn't be accessed even though the link was valid

- Running a search engine on a single machine proved to be very slow. Therefore, Noogle had to be handicapped to only have a partial view of the web structure of UIC domain. With a threshold of 5000 pages, around 4100 pages were only downloaded excluding pdfs, xmls, inaccessible pages and etc.
- Across the web pages, the usage of anchor <a> tags were inconsistent. Some of them had the href attribute, thus being a link to another page, whereas some <a> tags were used just for the clickable action. An explicit check was required to avoid adding empty links to the crawler's queue
- For the same web page, multiple urls were being used. For eg. https://www.cs.uic.edu/Main/Courses and https://www.cs.uic .edu/courses points to the same web page. In addition to this, the url for a page is not used in a consistent way across pages. For eg., few pages had https://cs.uic.edu and others had https://www.cs.uic.edu
- Some pages had repetitive information for the desktop and mobile layouts due to poor design. This messed up the weights in the inverted index

Most of the above challenges had to be handled using explicit conditional statements.

5 INTELLIGENT COMPONENT AND EVALUATION

Noogle contains two intelligent component viz. the PageRank component and the Document Clustering component.

Page Rank provides us with an option to assign a score to a page based on how authoritative its contents are, using the link structure of the web. A page will have a high score if many pages, which are in turn authoritative, point to it. Once NoSpidey finishes crawling the web and creates the web graph, it triggers the page rank power iteration method to execute. The power iteration method initializes the page rank scores of all pages to 1/(of nodes). It then creates an adjacency matrix, normalizes it, makes the matrix stochastic, irreducible and aperiodic and takes the transpose of the same. The power iteration uses the following expression to calculate page rank scores.

$$P = (1 - d) * e/n + d * A^T * P$$

where e is a column vector of 1s and d=0.85. P is initialized to a column vector of 1/n, n = number of pages/nodes. Noogle first uses its retrieval model to get a list of relevant documents for a given query. It then ranks the pages in the list based on their page rank scores.

Document clustering using unsupervised approaches like Topic modeling will help Noogle in expanding the search results by retrieving documents that are related to the user query's topic. Noogle in the background creates a topic model using the corpus, the corpus is initially converted to a tf-idf vector based dataset. It then gets the query and finds the topic that it is closely related to. It then uses the topic to get closely occurring terms to the query's terms and expands the search results.

The table-3 shows the results for query (1) when only the similarity measure was used, when page rank and similarity were used and when similarity, page rank and clustering were used.

Features used	Results
cosine similarity	https://www.cs.uic.edu/courses https://www.cs.uic.edu/Main/Courses http://logos.cs.uic.edu/reed https://www.cs.uic.edu/undergraduate-programs https://www.cs.uic.edu/Main/UndergraduatePrograms https://www.cs.uic.edu/cs-minor https://www.cs.uic.edu/Main/ComputerScienceMinor http://www.cs.uic.edu/Theys http://www.cs.uic.edu/Troy https://www.cs.uic.edu/graduate-programs
cosine similarity, PageRank	https://www.cs.uic.edu/graduate-programs https://www.cs.uic.edu/undergraduate-programs https://www.cs.uic.edu/courses https://www.cs.uic.edu/cs-minor http://logos.cs.uic.edu/reed https://www.cs.uic.edu/Main/Courses https://www.cs.uic.edu/Main/UndergraduatePrograms https://www.cs.uic.edu/Main/ComputerScienceMinor http://www.cs.uic.edu/Theys http://www.cs.uic.edu/Troy
cosine similarity, PageRank, Clustering	https://www.ois.uic.edu/cms/One.aspx?portalId=1581&pageId=836301 https://www.ois.uic.edu/cms/One.aspx?portalId=1581&pageId=969720 https://www.ois.uic.edu/news https://www.ois.uic.edu/students/admitted https://www.ois.uic.edu/students/admitted/travel___visas https://www.ois.uic.edu/news/o_i_s_news_-_11_07_2017 https://www.ois.uic.edu/news/o_i_s_news_-6_28_2018 https://www.ois.uic.edu/news/o_i_s_n_e_w_s_-_05_24_2017 https://www.ois.uic.edu/faculty__staff__and_scholars https://www.ois.uic.edu/news/o_i_s_news_-_2_22_2018

Table 3: Results obtained for query (1) w and w/o intelligent components. The last row only has the results for the expansion terms

6 ERROR ANALYSIS

Building Noogle resulted in lots of learning of what will and will not work. **What worked:-**

- Stemming definitely was beneficial for the system. It reduced the index size by 10k entries and also helped in grouping words that might mean or inclined in the same direction
- The combination of stemming, cosine similarity and page rank resulted in better query results than the results obtained by using a subset of the three
- Saving the state of the web graph, crawler queue, page rank scores and the models proved beneficial when incrementally developing the system. Some of the components tend to run for hours and creating snapshots really helped during crucial times during the project

- A well tested duplicate removal code helped NoSpidey from being stuck in cycles as many pages have links to parent sites like uic.edu, etc.

Limitations:-

- The UIC domain has lots of non-textual format documents like pdfs, images, xml files and so on. These had to be excluded from the corpus, thus resulting in a reduction in precision. Some of these documents contain valuable and direct answers to few of the queries
- Noogle's crawler NoSpidey currently runs on a single thread. This avoids concurrency issues but crawling 1000s of links and writing to disk resulted in a slow run of the crawling process (took 2hrs to crawl 5000 pages)
- No metadata like last modified, etc are captured. This can help in providing with the latest documents in the results. But this did not have a major impact for the UIC domain
- Since Noogle runs on a single machine, it couldn't capture the entire perspective of the UIC domain. This decreased the authority score for lots of pages and some pages weren't reached at any point during the crawl. Crawling 5000 pages resulted in creating around 30000 nodes in the graph. This would take around 6GB of memory while trying to calculate page rank. So for page rank only the pages that were downloaded and links from these to other pages in the reduced set were only considered
- The inverted index only stores unigrams from the corpus. The preprocessor, query processor, retrieval model and query expansion, all work using unigrams only. In the course of evaluation, some highly frequent yet important unigrams affected the result decreasing the precision. For eg. terms like fee have a different meaning when used as unigrams, affecting what we are looking for in the original query like out-of-state fee. Query expansion terms did not mean anything related to the original query without being combined with words from it
- Harmonic mean does not always help in combining the different relevance scores. Because of the above problem, unnecessary documents are ranked higher. For eg. looking for 'cs graduate' or 'graduate requirements' is better than looking for 'cs' and 'graduate' or 'graduate' and 'requirements' separately since the latter appears in lots of other pages too. So Noogle had to sort the results by similarity score and take only top n documents for ranking
- link based duplicate removal did not always work well because of the inconsistencies in the way URL is used by different pages and also some pages have multiple URLs

7 FUTURE WORK

Noogle has a lots of room for improvements and has lots to learn to become like Google one day. Some future work considerations include:

- A parallel implementation of NoSpidey with concurrent accesses to the link queue, web graph and file io
- Use of ngrams instead of just unigrams in indexing, document and query processing
- An interesting way to combine multiple relevance scores like similarity, page rank, etc. would be a supervised learning approach, where we have a set of training queries and relevant documents. Now the system can build a model and learn the best way to linearly combine the different relevance scores by optimizing the weights over some loss functions, so that it retrieves the relevant documents for the training queries. This can be a better alternative to harmonic mean and other averages and it is very scalable and efficient
- Implement a content based filtering system to remove duplicates

8 CONCLUSION

Working on Noogle created a whole new perspective of how IR systems work in real life. It is very much different from what one can understand from reading the theoretical concepts from the book. Each component in itself is complex and when they are integrated, multitude of issues come into the picture. Complex nature of human language and the internet poses a lot of challenges for simpler solutions involving just the similarities and authority scores. In addition, it requires a lot of correlated features working together to effectively retrieve documents. But that being said, tf-idf vectors with cosine similarity is an highly effective strategy to retrieve documents. Along with pagerank, Noogle tries to be like Google at certain times. Noogle will not stop evolving after this project. Reverse Engineering complex systems and improvising is a crucial skill for today's engineers and this project surely helped me in kick starting a journey down the road. Thank you for this opportunity.