The Development of search engine service for official academic documents

A A G Y Paramartha¹ and L J E Dewi²

¹²Universitas Pendidikan Ganesha, Singaraja, Indonesia
yudhi.paramartha@undiksha.ac.id, joni.erawati@undiksha.ac.id

Abstract. Document search engines are necessary in educational institutions, where search engines are expected to find the right document quickly. Academic official documents are generally private so that public search engines such as Google cannot index and search these documents. Therefore, we need a specific search engine developed for educational institutions’ needs, in this case, Universitas Pendidikan Ganesha. This article discussed how to develop an academic official document search engine, starting from data retrieval and extraction, query parser development, and search engine evaluation. The query parser developed utilized the field annotation mapping mechanism and n-gram keyword mapping. The evaluation results showed a significant increase in performance compared to only using Solr’s Standard Query Parser.

1. Introduction

In an educational institution, official academic documents will grow with the development of the institution itself. The documents can be any academic rules and policies of an educational institution's entire academic process that added up thousands of documents each year. These documents also have unique characteristics such as repetitive information, long texts, time-sensitive, and the people involved in an activity that occurs on a seasonal basis. Search engine like Google generally does not index these documents because official documents’ private nature. Although several documents can be accessed by the public, these search engines generally do not specialize in official documents specific to a particular institution [1]. Over time, official academic documents will become increasingly difficult to be sorted and searched due to their increasing sizes. If it is done manually, searching for these documents will take a long time, and the accuracy of the search result will also be very low.

The need to find official academic documents that are fast and precise is very important because many educational processes are based on certain rules or policies. Policy officials often have difficulty in finding the relevant documents needed, such as institutional audits, analyzing previous policies, or the basis for explaining a policy to the public. Therefore, we need a specific search engine to search official academic documents, in a particular, a case study at Ganesha Universitas of Education. In this article, it discusses how the researchers developed a search engine with data sources in the form of official academic documents from the Legal and Administrative Information System (HTL). The search engine was developed based on Apache Solr with adjustments to the query parser by utilizing the annotation field and n-gram keyword mapping. Annotation mapping could improve search accuracy by matching keywords to the appropriate fields. Meanwhile, the n-gram keywords mapping was used to maintain compound words and word order from keywords to improve the search results.
2. Literature Review
The search engine requires a keyword from the user, and the search engine recommends search results that are relevant to that keyword. However, the typed keywords may be ambiguous, so search engines must have a mechanism to perform searches that are close to what the user wants[2]. This has become an interesting research topic for many researchers, where the accuracy of the results and efforts to reduce bias in search results are the main things in search engine development[3]. One important module in a search engine is the query parser. The query parser functions as a keyword adjustment to be used more accurately by search engines.

Search engines store metadata for each document called index. An index is a summary of important objects from a relevant document for searching. The more indexes, the more documents a search engine can search for, but the greater the need for computation and storage space[4]. The search engine development platform has been extensively developed. The two most widely used platforms today are Apache Solr and Elastic search [5]. Apache Solr and Elastic search have their respective advantages and disadvantages but have the same goal: an efficient search platform for rapid search engine development.

3. Method
The search engine development began with data collection from HTL. The data collected were in the form of HTL database and documents that were mainly in PDF format. PDF files generally not only store text, they also store images that contain text. This is caused by several things, such as scanned images in the form of a PDF file. Therefore, OCR (Optical Character Recognition) is also needed for text extraction from the image. The OCR engine used was based on the open-source Tesseract OCR library. The data collected were extracted and indexed by Apache Solr via the Solr REST Client.

Furthermore, the query parser was developed to function as a keyword adjuster to adjust the data characteristics and search engine needs. Users could enter a keyword, and the query parser would adjust the keyword. The parsed keywords were used by Solr to perform a search for the relevant document. Solr provided search results and the level of similarity based on their relevance to keywords. The method used to calculate the similarity was BM25 (k1 = 1.2, b = 0.75), the default similarity method used by Apache Solr[6].

The search engine evaluation was done by measuring the precision, recall, mean average precision (MAP). Precision is used to measure the ratio between the relevant documents and the number of all documents retrieved. Recall is a comparison between the relevant documents retrieved and all relevant documents [7]. Average Precision is a calculation used to see whether a relevant search result is ranked in the top order. This calculation considers the precision of a series of search results compared to the ranking of each relevant document at position k. The following formula is an average precision calculation:

$$AveP = \frac{\sum_{k=1}^{n}(Precision(k) \times rel(k))}{number \ of \ relevant \ documents}$$  \hspace{1cm} (1)

Mean average precision (MAP) is the average of average precision based on the number of tests performed. Calculation of MAP can be seen in the following formula:

$$MAP = \frac{\sum_{i=1}^{Q} AveP(i)}{Q}$$  \hspace{1cm} (2)
4. Result and Discussion

4.1. Data Extraction and Indexing

Data extraction began by reading the HTL database and retrieved all the required information. Information retrieved was in the form of document title, document number, year, date authorized, document type, list of people involved, and document URL. Then the PDF document was retrieved via the document URL. All text in the PDF document was retrieved and saved. If the PDF document contained images, then text extraction was done by OCR.

The search engine was developed based on Apache Solr. Before the document could be indexed, the extracted documents were saved into a JSON document before being indexed by Solr. The data were taken from HTL database and HTL documents in PDF form. The number of documents used was 6,335 documents taken from HTL. The following table is the schema used in JSON documents as well as those used in Solr.

| Field          | Data Source                  | Index/Query Analyzer                              |
|----------------|------------------------------|---------------------------------------------------|
| Id             | Generated ID                 | -                                                 |
| doc_id_number  | Database tuple               | -                                                 |
| Year           | Database tuple               | -                                                 |
| date_authorized| Database tuple               | -                                                 |
| url            | Database tuple               | -                                                 |
| doc_title      | Database tuple               | Standard tokenizer, lower case filter, stop word filter, Indonesian stemmer. |
| doc_type       | Database tuple               |                                                   |
| people_involve | Database tuple               |                                                   |
| doc_text       | Extracted from PDF document  |                                                   |

Based on the fields used, advanced users may be able to manually filter the data by fields and find the relevant documents. However, ordinary users generally want to enter only one keyword in the search field. Therefore, search engines should be able to predict what users want based on the entered keywords. That being said, we need the query parser which is discussed in the next section.

4.2. Query Parsing

Solr can accept REST requests with several parameters. Keywords were processed by utilizing the existing Standard Query Parser in Solr. However, initial tests using Solr’s Standard Query Parser by directly entering keywords could not produce satisfactory search results, especially on complex keywords. The parser must know the annotations of the document and Solr’s Standard Query Parser does not appear to be suitable for various document complexities[8]. Therefore, adjustments were made to the query parser based on the document characteristics.

Based on the analysis, there were several things that the query parser had to be able to adjust to get better results. First, the query parser had to be able to figure out that the entered keywords were specific keywords such as year and document type. Second, the query parser had to be able to know that the keywords entered were a person’s name. Therefore, a keywords annotation mechanism was developed, as shown below.
Algorithm 1 fields annotation mapping

```
1: function parseQuery(query)
2:     for each term in query
3:         if term exist in doc_type from index then
4:             parsedQuery.add('doc_type', term)
5:         end if
6:         if term exist in people_involve from index then
7:             parsedQuery.add('people_involve', term)
8:         end if
9:         if term is year formatted then
10:            parsedQuery.add('year', term)
11:         end for
12:     parsedQuery.add('doc_title', query)
13:     parsedQuery.add('doc_text', query)
14:     return parsedQuery
15: end function
```

Annotation mapping was done by taking each word in the keywords. Each word would be searched for in a specific field via Solr’s Standard Query Parser. If search results were obtained, the word was considered as a keyword that matched the field. Annotation mapping was used to identify the doc_type and people_involve fields. Meanwhile, the year field could be mapped by seeing whether the word used the year format or not. The following table is an example of an annotation mapping that has been done.

| Keywords:                               | Parsed Query:          |
|----------------------------------------|------------------------|
| John Doe odd semester academic supervisor 2019 | year : 2019            |
|                                        | doc_type : decree of the academic supervisor |
|                                        | people_involve : John Doe |
|                                        | doc_title : odd semester academic supervisor |
|                                        | doc_text : odd semester academic supervisor |

By using the query parser above, search performance could indeed be improved. However, it could still be improved by using n-grams keyword searching. This was necessary because some words were compound words and uncommon words such as brands, location names, and people’s names. For example, the compound word “academic supervisor” was translated by Solr’s Standard Query Parser into two words, namely “academic” and “supervisor”. This caused Solr to retrieve all documents with the two words separately, whereas the user might want a document containing the compound word “academic supervisor”. Some researchers have previously found that n-grams keyword searching (mostly using bigram and trigram) on document searches could improve search results compared to using only unigram[9]. This was also applied not limited in the searching for natural language texts[10]. Although n-gram mapping produces permutation of keywords, the use of n-gram can also maintain the order of keywords[11] so that it is expected to improve search accuracy. Therefore, keywords were also formed into bigram and trigram for the doc_title and doc_text fields. The following is an example of n-gram mapping in the query parser.
Query Parser Model B: Query Parsing with Fields Annotation and N-Grams Keywords Mapping

| Keywords                          | Parsed Query                                                                 |
|----------------------------------|-----------------------------------------------------------------------------|
| John Doe odd semester academic supervisor 2019 | year: 2019, doc_type: decree of the academic supervisor, people_involve: John Doe, doc_title: "odd semester" "semester academic" "academic supervisor" "odd semester academic supervisor" odd semester academic supervisor, doc_text: "odd semester" "semester academic" "academic supervisor" "odd semester academic supervisor" odd semester academic supervisor |

The example above is a simplification of what the query parser actually produced. Please note that n-gram mapping also produces n-gram words that do not match the original order. For example, the word “academic supervisor” produces 2 keywords, namely “academic supervisor” and “supervisor academic”.

4.3. Evaluation

Three models were evaluated, namely the base model, which used Solr’s Standard Query Parser, model A using the fields annotation mapping, and model B using the fields annotation and n-grams keywords mapping. The search engine was tested with 12 keywords: a combination of several keywords based on the document's topic, type of document, names of people involved, and when the document was approved. The following were the test cases used:

| Keywords                                      | Number of Relevant Document |
|----------------------------------------------|----------------------------|
| informatics management                       | 10                         |
| student activity unit                        | 10                         |
| outstanding award for associate degree student | 5                          |
| study program evaluation report              | 10                         |
| computer science study program evaluation report | 2                          |
| thesis supervisor odd semester 2019          | 10                         |
| thesis supervisor John Doe                   | 8                          |
| thesis supervisor John Doe 2019              | 6                          |
| ICIRAD committee                             | 7                          |
| ICIRAD committee 2019                        | 5                          |
| ICIRAD committee John Doe 2019               | 2                          |
| international conference speakers            | 2                          |

Notes: The keywords used were actually in Indonesian. However, for the convenience of the reader, the keywords were translated into English. The names of the persons were also made anonymously for privacy.
Figure 1 is the results of the evaluation using precision and recall on search results ranked 1 to 10. As for the recall, this study used maximum number of retrieved true-positives documents from all model tested as a ground-truth relevant document for each query[12]. Figure 1 below is the Precision-Recall chart.

![Precision-Recall Chart](image)

**Figure 1.** Precision-Recall Evaluation.

Using a query parser with field annotation mapping (model A) appeared to have a significant impact compared to using Solr’s Standard Query Parser. This can be seen from the precision value of 0.50 in document rank 1 (P@1), 0.46 in the top-5 retrieved document (P@5), and 0.33 in the top-10 retrieved document (P@10), compared to 0.00, 0.17, and 0.13 on the base model respectively. Model A also showed a significant recall increase with a value of 0.10 in document rank 1 (R@1), 0.39 in top-5 retrieved documents (R@5), and 0.53 in top-10 retrieved documents (R@10), compared with 0.00, 0.13, and 0.20 on the base model respectively. By adding n-gram keyword mapping to the query parser (model B), a significant improvement also occurred compared to model A and the base model. Precision model B values were P@1: 0.91, P@5: 0.80, and P@10: 0.62 with a recall of R@1: 0.10, R@5: 0.73, and R@10: 0.97.

Apart from precision and recall, the search engines were also evaluated with mean average precision (MAP), where MAP produces a value within a certain threshold by considering the order in which the relevant documents appear[13]. The MAP evaluation in this study used document threshold 1, 3, 5, and 10. This refers to the Google Organic CTR History data, which states that more than 50% of users will choose search results at rank 1 to 3 and 60% users will choose the search result at rank 1 to 5 [14][15]. We also used a threshold of 10 documents, as users tend to see at least 10 search results before they enter new keywords or go to the next search page. Figure 2 is a summary of the results of the MAP evaluation. As seen from these data, all MAP values in model A were above 0.5. Meanwhile, in model B, the MAP values were above 0.9. This indicates that more than 90% of the retrieved relevant documents were at the top results.
5. Conclusion

The official academic document search engine is a necessity to be owned by educational institutions, including ours, where there are so many documents that must be searched quickly and accurately. One of the challenges in developing a good search engine is how the query parser can identify key pieces of a specific data set. The use of field annotation mapping and n-gram keyword mapping in the query parser that we developed has a significant impact on search engine performance. This could be seen from higher precision (P @ 10: 0.62) and recall (R @ 10: 0.73) compared to the other two models tested. The MAP results also indicated that the relevant search results were in the top order with MAP values above 0.9. Although this research was limited to the specific topic and data set, the result of the search engine is sufficient for our needs.

References

[1] Li Z, He J, Chen D, Fang X, Song Y and Li Z 2018 A hybrid approach for measuring similarity between government documents of China ACM Int. Conf. Proceeding Ser. 431–435
[2] Senthilkumar N C and Pradeep Reddy C 2019 Collaborative Search Engine for Enhancing Personalized User Search Based on Domain Knowledge J. Med. Syst. 43
[3] Gao R and Shah C 2020 Toward creating a fairer ranking in search engine results Inf. Process. Manag. 57 102–138
[4] van den Bosch A, Bogers T and de Kunder M 2016 Estimating search engine index size variability: a 9-year longitudinal study Scientometrics 107 839–856
[5] Luburić N and Ivanović D 2016 Comparing Apache Solr and Elasticsearch search servers 6th Int. Conf. Inf. Soc. Technol. ICIST 2016 287–291
[6] Balabantaray R C and Ghosh S 2018 A case study for ranking of relevant search results Adv. Intell. Syst. Comput. 564 231–9
[7] Arora M, Kanjilal U and Varshney D 2016 Evaluation of information retrieval: precision and recall Int. J. Indian Cult. Bus. Manag. 12 224
[8] Brouwer M and Kemps-snijders M 2017 MTAS : A Solr / Lucene based Multi Tier Annotation Search solution CLARIN Annual Conference 2016 pp 19–37
[9] Laclavík M, Steingold S and Ciglan M 2016 Detecting Brands in User Search Queries he 9th ACM International Conference on Web Search and Data Mining
[10] Rigaux P and Travers N 2019 Scalable searching and ranking for melodic pattern queries Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019 pp 801–808
[11] Moen H, Peltonen L-M, Suhonen H, Matinolli H-M, Mieronkoski R, Telen K, Terho K, Salakoski T and Salanterä S 2019 An Unsupervised Query Rewriting Approach Using N-
gram Co-occurrence Statistics to Find Similar Phrases in Large Text Corpora Proceedings of the 22nd Nordic Conference on Computational Linguistics pp 131–139

[12] Henderson P and Ferrari V 2017 End-to-end training of object class detectors for mean average precision Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)10115 LNCS 198–213

[13] Hast A 2020 Consensus Ranking for Increasing Mean Average Precision in Keyword Spotting Proceedings of the 2nd International Workshop on Visual Pattern Extraction and Recognition for Cultural Heritage Understanding pp 46–57

[14] Ziakis C, Vlachoupoulou M, Kyrkoudis T and Karagkiozidou M 2019 Important factors for improving Google search rank Futur. Internet11

[15] Google Google Organic CTR History