Attn: Dr. Sun Aixin

NANYANG TECHNOLOGICAL UNIVERSITY

AI6122 Text Data Management and Processing

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below. We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work. We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work.

Name	Signature / Date
Liu Yaohong	Liu Yaohong Oct.30
Wang Yuhang	Wang Yuhang 30/10
Gao Han	10.30
Qin Xiaokai	Oct.30
Tong Xindi	10.30

Important note:

Name must **EXACTLY MATCH** the one printed on your Matriculation Card. Any mismatch leads to **THREE (3)** marks deduction.

AI6122-Text Data Management & Processing Assignment

Liu Yaohong
Nanyang Technological University
G2203319C, Singapore, Singapore
liuy0220@e.ntu.edu.sg

Wang Yuhang
Nanyang Technological University
G2201250C, Singapore, Singapore
ywang145@e.ntu.edu.sg

Gao Han
Nanyang Technological University
G2302986C, Singapore, Singapore
hgao005@e.ntu.edu.sg

Qin Xiaokai
Nanyang Technological University
G2302770H, Singapore, Singapore
qi0002ai@e.ntu.edu.sg

TONG Xindi
Nanyang Technological University
G2303905C, Singapore, Singapore
to0001di@e.ntu.edu.sg

Abstract

Similar to the instructions provided in the assignment handout, the purpose of this group project is to familiarize ourselves with the fundamental components of end-to-end text management and processing applications. This comprehensive task comprises several components: analyzing datasets from three different domains, developing a basic search engine and research trend explorer, and creating a simple application that utilizes the engine we construct.

Keywords: Text management, Tokenization, Stemming, Sentence Segmentation, POS Tagging

1 Domain Specific Dataset Analysis

In the initial section of our report, we focus on the Analysis of Domain-Specific Datasets. Here, we will provide detailed information about the datasets we employed and present the analysis results including the the **Tokenization and Stemming**, **Sentence Segmentation** and **POS Taggingas** just like the requirements outlined in the assignment handout.

1.1 Datasets Selecetion

We have chosen three distinct domain-specific datasets, to effectively simulate diverse text environments.
The first dataset consists of reviews related to British Airways. each review mainly describes the flight experience corresponding to different customers and routes.
The second dataset comprises healthcare documents primarily describing patients' diseases.
The third dataset is centered around research papers, mainly focus on how many highly mathematical modeling and physical theory to solve complex physical and cosmological problems.

1.2 Tokenization and Stemming

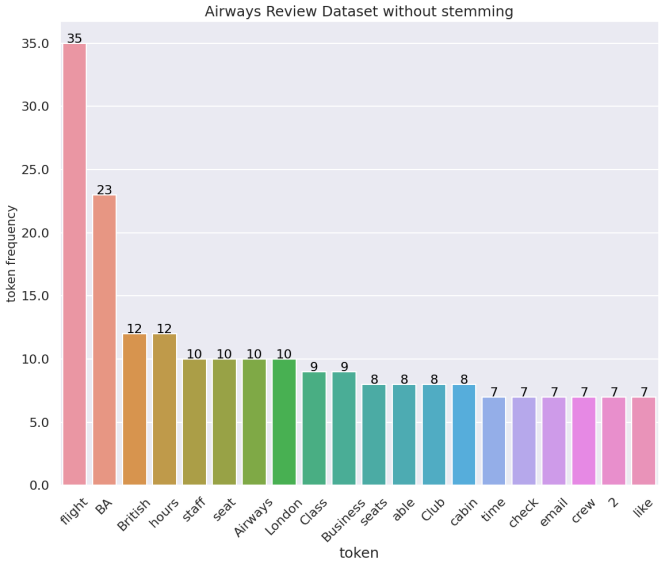
Tokenization is a major method to drill down the documents to works level which machine is able to learn. After tokenization, those individual pieces are called as token. However, the grammar of different languages are struggle for machine to differential. For example, in English, there are different tenses, like is and was; singular or plural etc. All these are the challenges for computer to understand neutral language. To solve the problem, stemming method is introduced. It helps to get the root of the words and reduce the variation of words. Hence, stemming is implemented universally in NLP domain. To unitized tokenization and stemming, python library *nltk & spaCy* has been used. We will introduce each steps respectively:

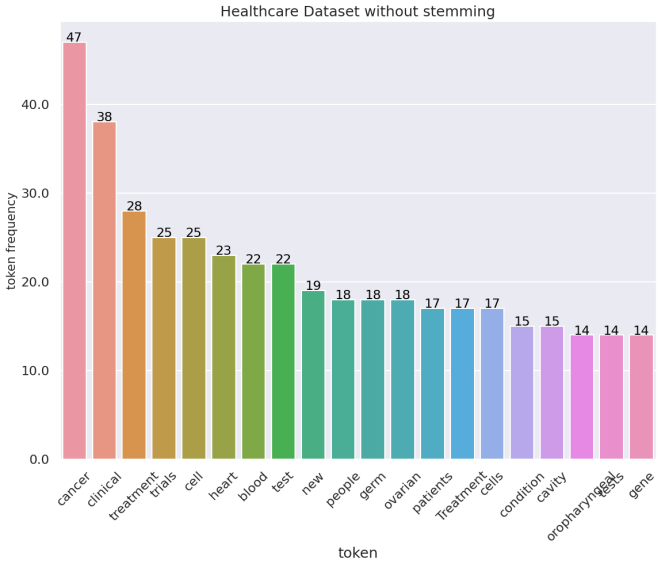
1. Tokenize text by using $nltk.word\_tokenize()$ and flatten the list of tokenized documents.
2. Stop words are a list of commonly used words, for example, I, is, a etc. They are usually be filtered out before or after text data processing. Therefore, stop words and punctuation has to be removed from token.
3. Implement the stemming method.
4. Counting and plot the top 20 tokens.

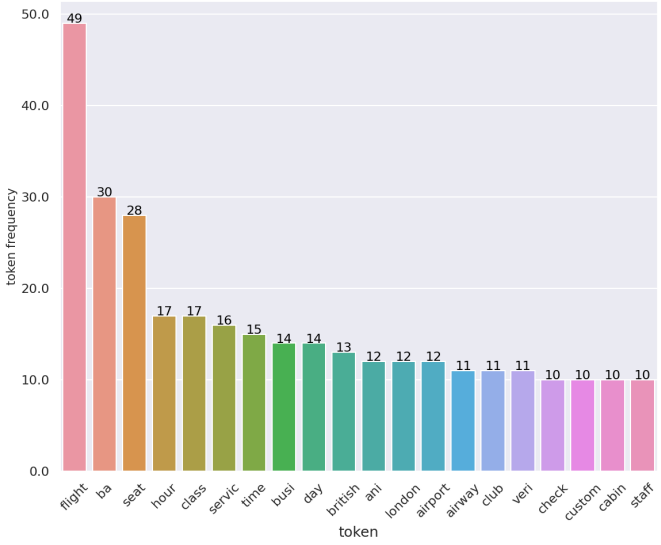
In order to analyst the dataset tokenization result, we plotted top 20 tokens which has been frequently used from 3 different scenarios:

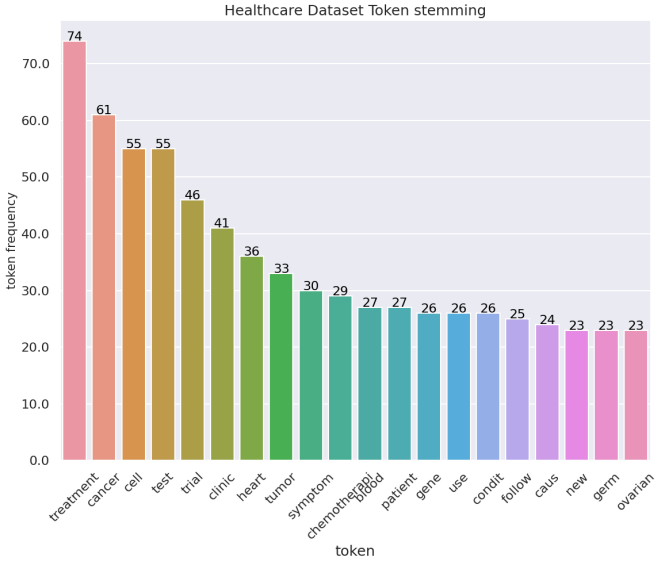
1. Tokenization without stemming mothed
2. Tokenization with stemming method
3. Tokenization with stemming method and stopwords with stemming method

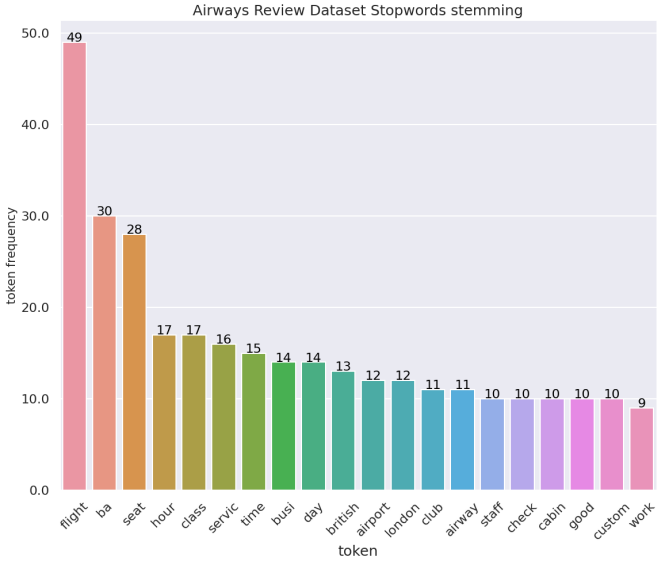
From scenarios 1&2, we found the sequence of most frequently using works change after consolidate all the roots of each word. For example, from Figure2, after implemented stemming function, the word 'treatment' jump to the first from third place. And this makes sences, comparing with the word 'cancer', the word 'treatment' is widely used in any health care news or paper. Same as Arxiv abstract dataset(see Figure3), the work 'result' is more popular than the words 'forms' in paper related area. After the stemming function applied, we found there are some words are frequently used in our life but did not be filtered out by stop words, such as 'any','use','obtain' etc(see Figure1 11^{th} token, Figure2 12^{th} token and Figure 3 19^{th} token). Those works supposed to be filtered but appears. the word 'used' is in stop words list, however, it dose not match with the word use that after stemming. Therefore, the 3^{rd} scenarios is utilized.











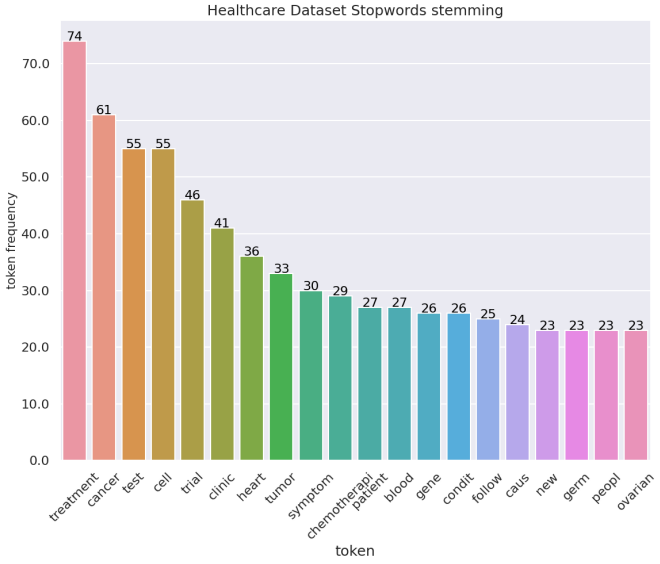


Figure 1. Airways Review Dataset Tokenization&Stemming Result

Figure 2. Healthcare Dataset Tokenization&Stemming Result

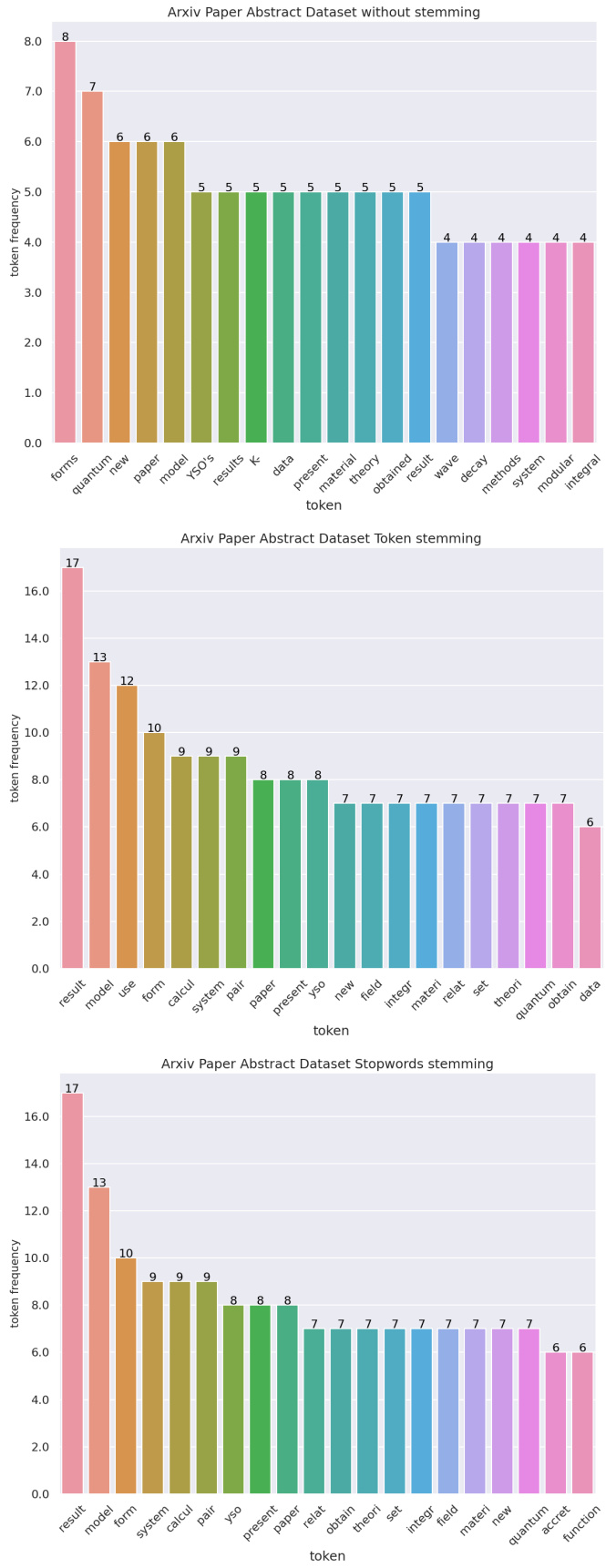


Figure 3. Arxiv Paper Abstract Dataset Tokenization&Stemming Result

And we can see from 3^{rd} subplot from Figure 123, those words have been filtered. Now the plotting is providing more meaningful information of those dataset. furthermore, by performing those preprocessing techniques,machine can have better understanding when it is trying to learn from those token rather than learning from the data without preprocessing.

1.3 Sentence Segmentation

Sentence segmentation is a fundamental task in natural language processing that involves breaking down text into distinct sentences. This process enhances text comprehension and facilitates information retrieval. It is commonly achieved through the analysis of sentence structure and punctuation usage. Sentence segmentation holds significant importance in applications like automatic summarization, text analysis, and information retrieval.

In this subsection, we primarily utilize the $sent_tokenize()$ function from the *nltk.tokenize* (NLTK library) to automatically segment the sentences within each dataset's documents.

As illustrated in Figure 4,5 and 6 it is evident that on the all three datasaets, the length of sentences distributions are mainly Gaussian distribution. The length of words in a sentence is mainly concentrated on 20 for British Airways review dataset, but for the healthcare and paper abstract datasets, this number has been increased to roughly 25. These length differences may reflect the nature of texts in different fields. For example, the airline evaluations may be more concise and straightforward which results the final sencentes could be shorter. While the disease descriptions and paper abstracts may require more details and explanations to provide enough information for the doctors to diagnose and readers to grasp the main contribution of this academic paper , which caused sentences will be relatively longer.

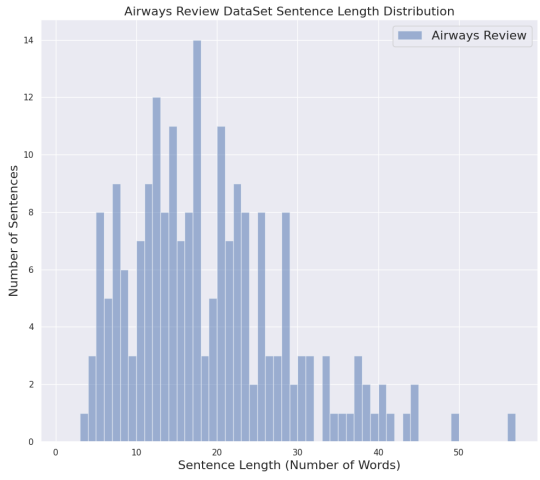


Figure 4. Sentences length distribution on Airways Review

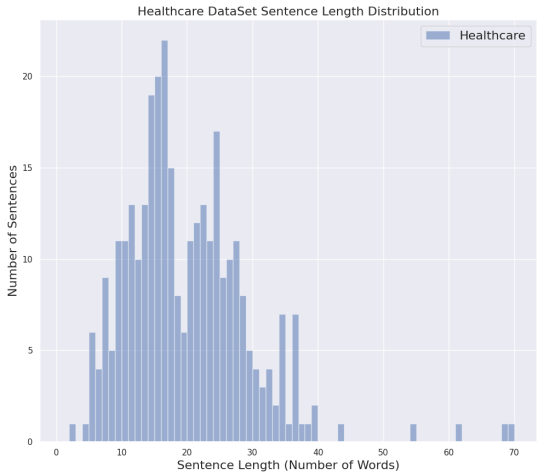


Figure 5. Sentences length distribution on paper abstract

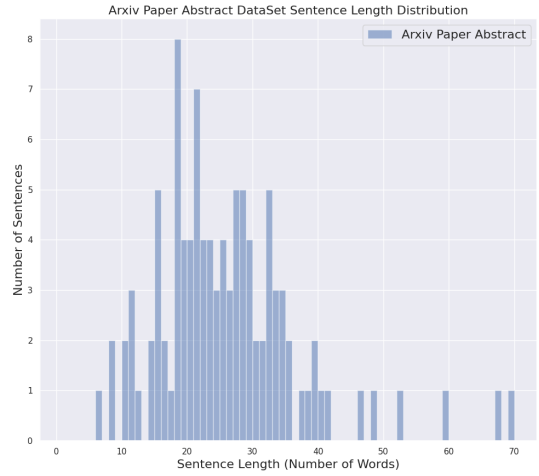


Figure 6. Sentences length distribution on paper abstract

1.4 POS Tagging

Part-of-speech tagging (POS) is a crucial linguistic task that assigns grammatical labels (e.g., noun, verb, adjective) to each word in a text, aiding in syntactic and semantic analysis. It enables machines to understand the role of words in sentences, improving tasks like language understanding, sentiment analysis, and information extraction. POS tagging enhances the accuracy of natural language processing applications by providing valuable linguistic context.

Conducting POS analysis on every sentence in all files across datasets lacks depth. To align with the manual's guidelines, we will adopt a random sentence selection approach for analysis. In order to more accurately simulate the random selection process, we primarily utilize the following code to randomly choose three sentences from each domain dataset:

After we randomly select the three sentences, to facilitate this POS test, we employ the *pos_tag* function from the NLTK library to assign a grammatical role, such as noun, verb, adjective, adverb, etc., to each word in the selected sentences. The result of POS-tage is show on Table123.

2 Development of a Simple Search Engine

In this section of the report, we will focus on the building of our simple search engine, including the details of parsing data, index building, key words searching and ranking system.

2.1 dblp data preprocess

DBLP provides open bibliographic information on major computer science journals, proceedings, and other form of publications. The dataset use XML and the total size of the XML is about 4GB. We choose to use Lucene to index and search the data, because Lucene is widely recognized for its high-performance full-text search capabilities, allowing fast retrieval of large volumes of text data.

We opt for the SAX library for XML parsing due to its low memory consumption as it doesn't retain the entire XML document in memory. This resource efficiency is especially crucial when handling large XML files. By referencing the dblp.dtd file, we gain precise knowledge of tag structures. During parsing, each publication (or paper) is treated as a "document" and added to the index database.

When processing document data containing author names that may not always be in English, it is not appropriate to apply stemming and stopwords removal to tokenize names. Therefore, when handling <author> and <editor> tags, we exclusively employ the standard analyzer with lowercase filtering. For other tags, we utilize a combination of stopwords removal, English stemming, and lowercase filtering.

Based on the number of "documents" to be indexed in the dataset, we collect the time needed to index every 10% of the documents, the figure is shown in figure 15. We can see the indexing is relatively time-consuming at the beginning, then the time consumed decreases, and after that the time consumed slowly increases again. This can be attributed to several reasons:

- **Memory Cache**: Lucene uses memory caching to expedite the indexing process. In the initial phases, these caches may need to be populated, which can slow down the process. Over time, as the index structure and data become more stable, the memory caches become more effective. However, this performance boost might diminish once memory caches reach a certain threshold.

All	flights	were	late	.					
DT	NNS	VBD	RB	.					
On	flight	BA621	,	the	ravioli	pasta	had	dried	out
IN	NN	NNP	,	DT	NN	NN	VBD	VBN	RP
along	the	edges	.						
IN	DT	NNS	.						
There	is	no	where	to	put	anything	except	one	small
EX	VBZ	DT	WRB	TO	VB	NN	IN	CD	JJ
pull	out	draw	at	ground	level	that	can	not	be
VBZ	RP	NN	IN	NN	NN	WDT	MD	RB	VB
reached	when	seated	nor	accessed	when	sleeping	.		
VBN	WRB	VBN	CC	VBN	WRB	VBG	.		

Table 1. POS Tagging of the Airline Reviews

If	the	uterus	and	cervix	are	taken	out	through	a
IN	DT	NN	CC	NN	VBP	VBN	RP	IN	DT
small	incision	(	cut	)	in	the	abdomen	using	a
JJ	NN	(	VBN	)	IN	DT	NNS	VBG	DT
laparoscope	,	the	operation	is	called	a	total	laparoscopic	hysterectomy
NN	,	DT	NN	VBZ	VBN	DT	JJ	NN	NN
.									
.									
Its	possible	that	controlling	these	conditions	will	reduce	the	risk
PRP$	JJ	IN	VBG	DT	NNS	MD	VB	DT	NN
of	developing	Alzheimers	.						
IN	VBG	NNP	.						
This	test	measures	blood	glucose	in	people	who	have	not
DT	NN	VBZ	NN	NN	IN	NNS	WP	VBP	RB
eaten	anything	for	at	least	8	hours	.		
VBN	NN	IN	IN	JJS	CD	NNS	.		

Table 2. POS Tagging of the Healthcare

The	numerical	methods	were	found	to	be	flexible	and	robust
DT	JJ	NNS	VBD	VBN	TO	VB	JJ	CC	JJ
,	and	matched	analytic	results	to	a	high	accuracy	.
,	CC	VBD	JJ	NNS	TO	DT	JJ	NN	.
The	average	dark	matter	field	fluid	constant	derived	from	Earth-Moon
DT	JJ	JJ	NN	NN	NN	JJ	VBN	IN	NNP
system	data	is	4.39×10^{-22}	s^{-1}	m^{-1}	.			
NN	NNS	VBZ	CD	NN	NN	.			
We	also	study	an	analogue	of	Newman's	conjecture	for	overpartitions
PRP	RB	VBP	DT	NN	IN	NNP	NN	IN	NNS
.									
.									

Table 3. POS Tagging of the Arxiv Paper Abstract

- **Indexing Strategy**: Lucene may execute certain time-consuming operations during the initial phases, such as building and sorting the vocabulary. These operations are typically performed only once, so indexing speed increases once they are completed. But as

the number of documents increases, the size of the inverted index (a data structure containing words in documents and their positions) also grows. Each time a new document is indexed, updating and maintaining these data structures takes additional time.

- **Merge Operations**: Lucene regularly performs segment merging operations, which combine multiple small index segments into a larger one. This helps in maintaining index performance but also requires time. With an increasing number of documents, more merge operations are needed.
- **Resource Constraints**: Indexing more documents requires additional memory and processor resources to effectively build the index. If system resources are insufficient, it can limit indexing speed.

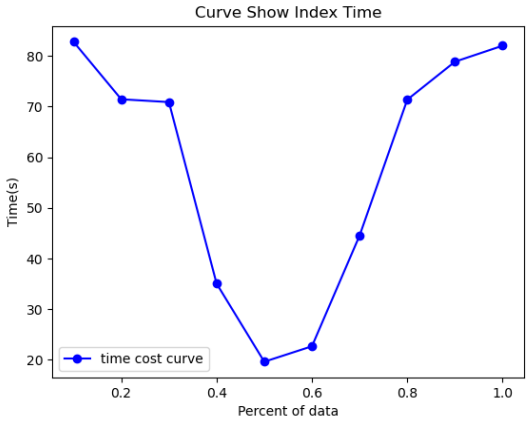


Figure 7. The time cost curve of index dblp data.

2.2 Searching

2.2.1 Search mode. Lucene has developed plenty of Query implementations, including Boolean Query, Phrases, and others. The searching process includes query, weight, scoring, and BulkScorer. It supports users to build their own query search implementation.

The query depends on the content that users input and weight is to count the weights for queries. The scores obtained from weights will be returned based on the provided segments and BulkScorer is calculated from scores for various documents. [2]

Our search engine is capable of accommodating term-based searches, encompassing both single terms and phrases. Furthermore, it supports field-based searching across categories like "title," "author," "year," and "source." Additionally, this search engine has the capability to perform searches across multiple fields, each with corresponding input. The searching time will also be calculated when returning results. Users can choose their preferred search mode in this search engine. By invoking and applying the Lucene package, we are able to implement fuzzy queries in each search mode. This project offers a total of four search modes:

```
The total number of documents: 10245088
Please specify the search mode number:
1. Strict Term Search
2. Flexible Term Search
3. Specific Field Search
4. Advanced Specific Field Search
```

Figure 8. 4 modes of the searching engine

- **Strict Term Search**: When a phrase is entered, it is treated as a unified entity without segmentation. The program will attempt to match the phrase across all fields.
- **Flexible Term Search**: When entering a phrase, the phrase is segmented, and a match is considered if the distance between two words is within 10 characters. The program will attempt to match across all fields.
- **Specific Field Search**: The program will prompt the user to specify a single field. The entered phrase will be segmented, and a match will be considered if the distance between two words is within 10 characters.
- **Advanced Field Search**: The program will require the user to specify all fields. The entered phrase will be segmented, and a match will be considered if the distance between two words is within 10 characters.

```
Please specify the search mode number:
1. Strict Term Search
2. Flexible Term Search
3. Specific Field Search
4. Advanced Specific Field Search
1
Please input your search: cluster
Searching for: cluster
title:cluster author:cluster year:cluster
Function took 59 milliseconds to search.
rank:1, score:3.713406, doc_id: 8295005
<B>Clustering</B> of <B>clusters</B>.

rank:2, score:3.5872278, doc_id: 8298029
To <B>cluster</B>, or not to <B>cluster</B>: An analysis of <B>clusterability</B> methods.
```

Figure 9. Searching Result of Strict Term Search

```
Please specify the search mode number:
1. Strict Term Search
2. Flexible Term Search
3. Specific Field Search
4. Advanced Specific Field Search
2
Please input your search: clustering Cycles
Searching for: clustering Cycles
title:"cluster cycl"~10 author:"cluster cycl"~10 year:"cluster cycl"~10
Function took 122 milliseconds to search.
rank:1 score:6.550342 doc_id: 597745
<B>Clustering</B> <B>Cycles</B> into <B>Cycles</B> of <B>Clusters</B>.

rank:2 score:6.550342 doc_id: 9493815
<B>Clustering</B> <B>Cycles</B> into <B>Cycles</B> of <B>Clusters</B>.

rank:3 score:4.5483913 doc_id: 1369833
 <B>Cluster</B>-<B>Cycling</B>.

rank:4 score:4.5483913 doc_id: 8020712
FedCluster: Boosting the Convergence of Federated Learning via <B>Cluster</B>-<B>Cycling</B>.
```

Figure 10. Searching Result of Flexible Term Search

2.2.2 Ranking. The ranking process in Lucene is included in the searching process, with the output of the ranking score

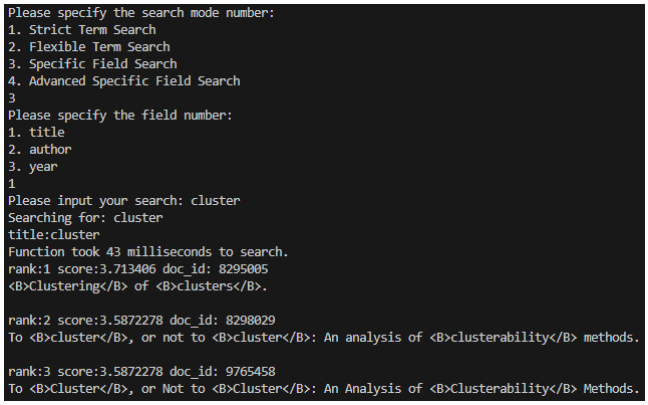


Figure 11. Searching Result of Specific Field Search

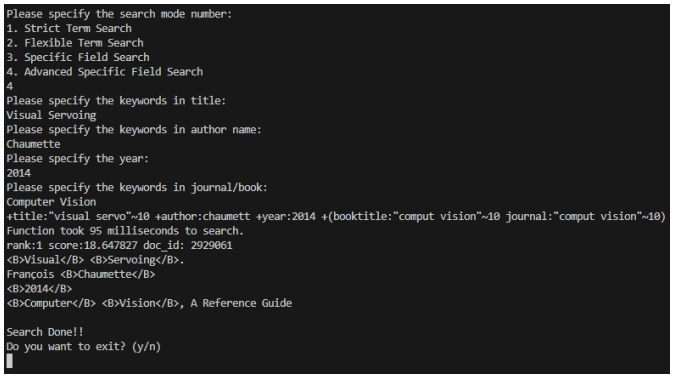


Figure 12. Searching Result of Advanced Field Search

in descending order. The score is calculated on the field and the scoring results will be combined and then returned to documents.

- **Analyzer**: Analyzer is used to analyze the text for one search engine. StandardAnalyzer is the most widely used one. In this search engine, the StandardAnalyzer is also applied with authors' and editors' names not considering stop words. [2]
- **Scoring Formula**: The document score in Lucene is determined by a similarity metric, which is determined by the Similarity class. Lucene combines the Boolean model (BM) of Information Retrieval with the Vector Space Model (VSM) of Information Retrieval. The Boolean model is applied to identify potential target documents, while the Vector Space Model is used to assess each selected document. This assessment is performed using a Tf-Idf scoring model.[4]

2.2.3 Highlight. The best sections of text based on the query will be bolded in the return snippets, which use the highlighter package in Lucene. One highlighter object is created based on Formatter and Scorer. The formatter determines the highlight display method, and the score for the highlighted part is determined by Scorer.[1]

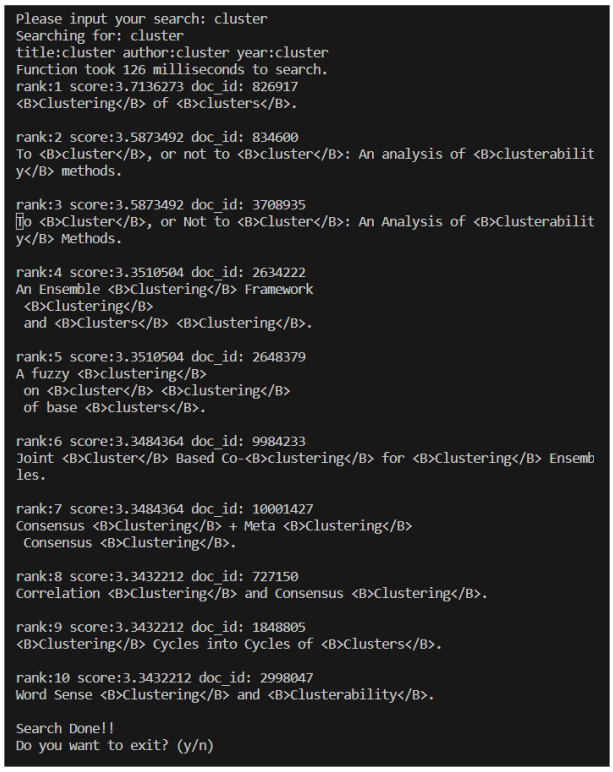


Figure 13. The returning result will first show the searching time and then the top N best matched documents sorted by their ranking with key words highlighted in the snippets.

- **Scorer**: Score will get the weight of each document based on the score value of the term and then obtain the appearance times and position of the term in corresponding documents based on its content.
- **Formatter**: Based on the text and the score calculated by Scorer, the query content exists if the score is larger than 0. Then the formatter will rewrite the term by adding pre-tag and post-tags on both sides of the search query.
- **Highlighter**: First The scorer first initializes the term occurrence position and then adds the recording corresponding location to the token stream. The formatter then will be called to reconstruct the snippets. It will finally output the most related text in the corresponding search fields. [1]

3 Research Trend Explorer

In this section, we need to choose one conference that have publications for more than 15 years, then we should get the research trend over the years.

3.1 Conference Data Preparation

Here we select the data from the SIGIR conference as the data for studying the research development trends. SIGIR (Special

1993	Lessons from PANGLOSS.
2007	Making mind and machine meet: a study of combining cognitive and algorithmic relevance
2018	Design and Evaluation of Query Auto Completion Mechanisms.
2023	MoocRadar: A Fine-grained and Multi-aspect Knowledge Repository for Improving Cognitive
2018	Identify Shifts of Word Semantics through Bayesian Surprise.
2019	A New Perspective on Score Standardization.

Figure 14. The extracted SIGIR conference paper titles.

Interest Group on Information Retrieval) conference is a renowned academic conference in the field of information retrieval and search engines.

We have obtained the title data of articles published at SIGIR from 1971 to 2023. This data was primarily extracted from the Lucene index database we obtained earlier. We retrieved all documents with "sigir" in their booktitle, extracted the year and title data, and stored it in CSV format. The extracted data is shown like the figure14. We have collected a total of over 6,000 data entries, and we plan to use this data for analyzing the research trends at SIGIR over the years.

3.2 Split Sentence to Phrases

The research trend can be represented by some key phrases that were popular among the papers published in a specific year and a key phrase may contain one or more words. In this task, we should not only limit our analysis to sentence-level word tokens but also consider phrases.

There are many python libraries that can help us split sentence to phrases and they have different characteristics and methods when dealing with text data:

- **RAKE** is a keyword extraction algorithm used to quickly extract keyword phrases from text. It primarily relies on word phrase frequency and word distribution in the text to determine keywords. RAKE does not consider POS or syntax, it extracts keywords based on phrase frequency.
- **SpaCy** provides comprehensive language processing capabilities, including POS tagging and syntactic analysis. SpaCy is widely used in various natural language processing tasks to analyze text structure and content.
- **TextRank** is an automatic summarization and keyword extraction algorithm based on graph theory. It constructs a graph of relationships between words and applies an algorithm similar to PageRank to determine keyword importance, providing a more comprehensive understanding of text.
- **YAKE** is a keyword extraction algorithm designed to extract key phrases from text. It combines word frequency and weighted information, considering both syntax and context of words. YAKE allows customization of keyword weighting methods based on specific task requirements.

In our case, we utilize all four libraries to tokenize our conference title data and we show the top five key phrase of the year 2001 in table 4. From this table, it can be seen that using different tokenization libraries results in some differences. Among them, the final results from spaCy and TextRank are relatively close.

3.3 Key Phrases Selection

By selecting key phrase, PRE(Pointwise relative entropy)[3] and TF-IDF(Term Frequency-Inverse Document Frequency) has been implemented respectively.

PRE(Pointwise relative entropy). is a potent criterion for key phrase selection, serving as an effective tool for gauging the divergence between two distributions. This makes it particularly adept at measuring the disparities between a key phrase and the entirety of a document. Here's a step-by-step guide to harnessing the capabilities of PRE:

1. Arrange the collected paper titles year by year, crafting a distinct document for each annum.
2. Implement crucial preprocessing tasks on these documents. This encompasses word segmentation, purging of stop words, and converting all text to lowercase.
3. For every annual text data, deduce the probability distribution of every term contained within.
4. Calculate the divergence for every term in a given year's document in comparison to the identical term from other years. This can be done using the formula: $P(w|D_1) \cdot \log \frac{P(w|D_1)}{P(w|D_2)}$.
5. Pinpoint the top N terms boasting the loftiest divergence values from each year's text data. These terms, determined by their divergence metrics, are earmarked as key phrases.

Moreover, the innate strength of Pointwise Relative Entropy lies in its ability to detect nuanced differences between terms, paving the way for the identification of profoundly insightful key phrases. It meticulously assesses the shifts in the distribution of key phrases across document sets from varying years. Consequently, PRE furnishes a more granular and precise snapshot of the distinctiveness embedded in the data.

We can see the results in table 5, the selected key phrases for different year is pretty good. However, it's pivotal to note that its precision comes at the cost of demanding greater computational resources.

TF-IDF. is a common method to select key phrases, which helps to determine which phrases have higher importance in a certain document or time period.The major step of implementing is following:

1. Organize the collection of paper titles for each year into separate text data, creating a distinct document for each year.
2. Perform essential preprocessing tasks on these documents, including word segmentation, removal of stop words, and conversion of all text to lowercase.

	top1	top2	top3	top4	top5
RAKE	speech	cite	xml documents	text categorization	search engines
SpaCy	xml documents	text categorization	speech	question answering	indexing
TextRank	xml documents	text categorization	speech	indexing	question answering
YAKE	speech	text	segmentation	topic	text summarization

Table 4. The top five phrases in 2001 using different libraries (using the PRE algorithm).

	top1	top2	top3	top4	top5
2008	a study	score regularization	affective feedback	the information	unstructured query
2009	mapreduce	boston	wikipedia	july	usa
2010	web search	blog retrieval	search trails	people	experimental design

Table 5. The top five phrases in 2008, 2009 and 2010 using SpaCy (using the PRE algorithm).

	top1	top2	top3	top4	top5
2008	information retrieval	retrieval	a study	research	development
2009	information retrieval	retrieval	research	web search	search
2010	information retrieval	retrieval	web search	research	text

Table 6. The top five phrases in 2008, 2009 and 2010 using SpaCy (using the TF-IDF algorithm).

3. Calculate the term frequency ($tf_{t,d}$) for each term in the document.
4. Determine the document frequency (df_t) for each term.
5. Compute the TF-IDF weight using the formula:
   $w_{t,d} = (1 + \log_{10} tf_{t,d}) \cdot \log_{10} \left(\frac{N}{df_t}\right)$
   A higher score indicates that the term holds greater importance within the document.
6. Based on the TF-IDF scores assigned to each term, select the top N terms with the highest scores as representative key phrases for each year's document.

TF-IDF is a straightforward and efficient method for assessing the importance of key phrases. In comparison to PRE (Pointwise Relative Entropy), TF-IDF demands considerably fewer computational resources and offers simplicity in terms of understanding and explanation.

However, unlike PRE, TF-IDF primarily relies on term frequencies within documents and doesn't capture the relationships or contextual information between terms in different documents. Furthermore, it does not consider variations in the distribution of terms across different document collections, which may limit its ability to reflect the distinctiveness of key phrases across different years. We can see the results from table 6, the selected key phrases for different years are quite similar, this algorithm is not a good choice.

3.4 Summary

Based on the results above, it is evident that the PRE algorithm performs better. However, the final outcomes are also significantly influenced by the choice of tokenization libraries. To achieve even better results, incorporating techniques like n-grams for analysis is advisable. Furthermore, when extracting research trend terms, we have solely focused on extracting data from article titles. Since article titles contain limited content, they may not adequately reflect the actual research trends. If feasible, it would be beneficial to perform phrase extraction from the actual content of the articles, thereby expanding the vocabulary sample. This approach would provide a more accurate representation of the real research trends.

4 Application

We have developed a simple search engine with a user-friendly interface based on the algorithm. This application is built using Java and merges the 'main' branch file into the search engine in part 2.

Users can choose four types of searching by clicking the corresponding button and typing in the search bar. The search results will be displayed in the lower section of the interface, with the search query highlighted. Users can change the search mode after each search.

Users can perform strict or flexible searches by clicking on the corresponding button and entering your search query. The results will be researched across all fields and displayed in the text area. Users can also select the target search field by clicking on the 'Field Search' button, which provides options such as "title," "year," and "author" for searching. Multiple field searches are supported as well by entering search queries in the respective fields. The returned results will consist of document snippets that match the search query.

In the text area, the total searching time and searching result will be listed according to their score in descending

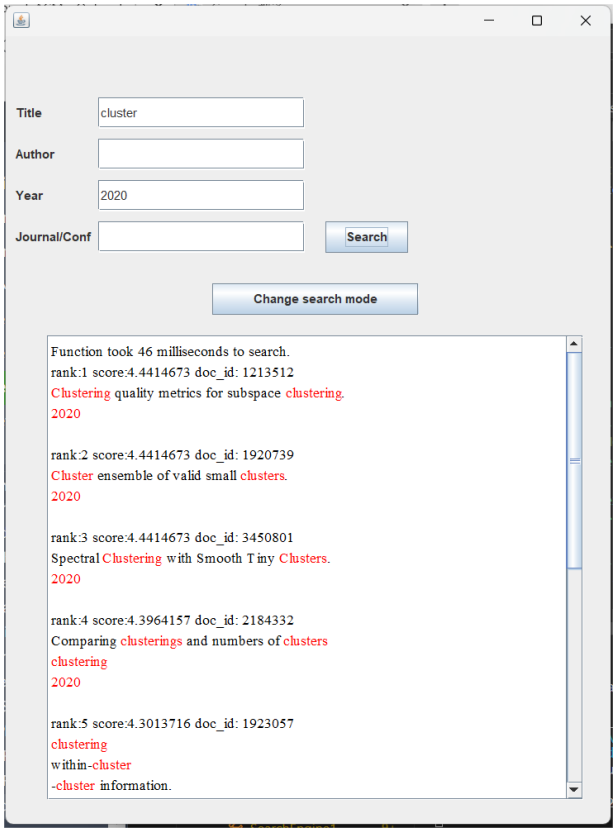


Figure 16. Multiple field search result

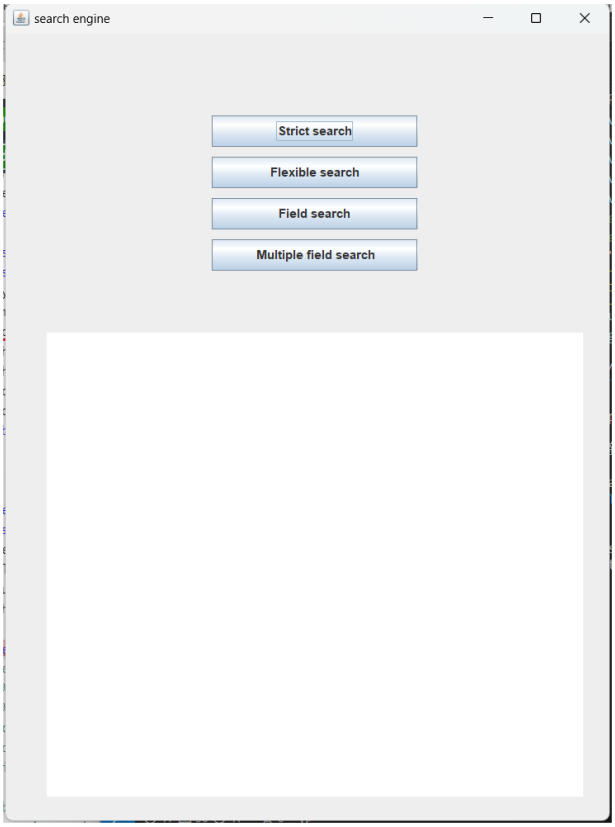


Figure 15. Pattern Chosen Interface

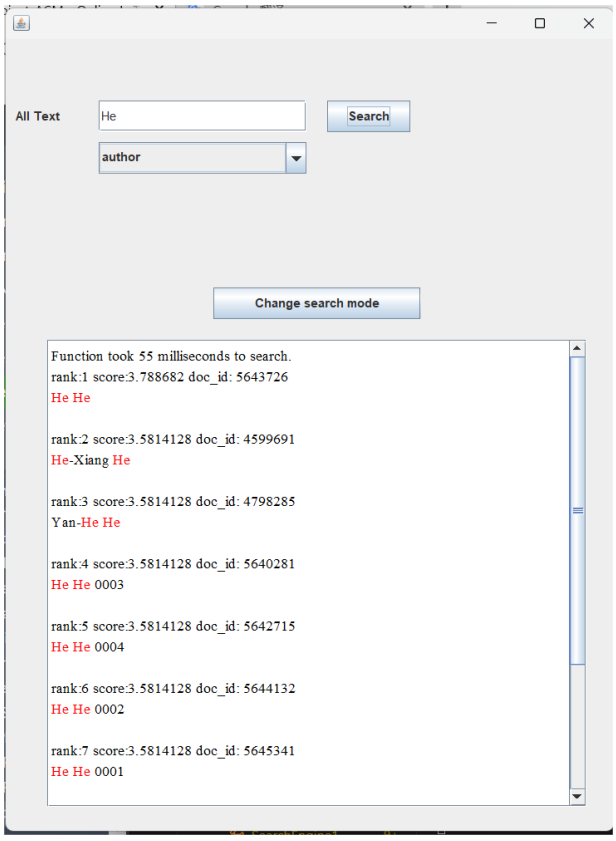


Figure 17. Multiple field search result

order, with search query words highlighted in red. Scroll bars will be provided if the displayed text is long.

Announcement

Liu Yaohong and Wang Yuhang completed the task of Domain Specific Dataset Analysis. Gao Han, Qin Xiaokai, and Tong Xindi accomplished the task of developing a Simple Search Engine. Additionally, Gao also completed the task of developing a Research Trend Explorer and contribute the task writing with Liu together. Finally, the application was developed by Tong. All individuals contributed to the writing and editing of this project.

References

[1] Andrzej Białecki, Robert Muir, Grant Ingersoll, and Lucid Imagination. 2012. Apache lucene 4. In *SIGIR 2012 workshop on open source information retrieval*. 17.

[2] Erik Hatcher and Otis Gospodnetic. 2004. *Lucene in action (in action series)*. Manning Publications Co.

[3] Solomon Kullback and Richard A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

[4] Ren Shuhuai. 2014. Analysis and Improvement of the Searching Algorithm on Lucene. *Libraly Journal* 33, 12 (2014), 17.