Design and Implementation of Academic Search Aggregation Engine

Jun Yan\textsuperscript{a}, Peng Xu\textsuperscript{b}, Yu He\textsuperscript{c}, Zhi Wang\textsuperscript{d}

Beijing University of Posts and Telecommunications, Beijing, China

\textsuperscript{a}yanjun@bupt.edu.cn, \textsuperscript{b}xupeng0726@163.com, \textsuperscript{c}heyu_751@bupt.edu.cn, \textsuperscript{d}wangzhibupt@foxmail.com

Abstract. With the rapid development of the global Internet, academic search engines have become indispensable tools for academic research like literature, papers, books, preprints, and abstracts. The total amount of network information is growing rapidly today, and the global paper output doubles every five years. For academic researchers, it is very difficult to find papers that meet the needs in a large number of articles. Additionally, as the quality of journal papers varied, finding high-quality papers and accurately locating them is getting harder and harder. Many researchers find it tiring and inefficient that when looking for research fields, they need to read through a large number of articles in search results, and manually classify them. Through the clustering algorithm and data visualization and other related technologies, the systematic integration and classification of the search papers will help the researchers to accurately locate the required articles, and can greatly reduce the time for searching the documents. Regarding the previously stated problems, this thesis designs and implements an academic search aggregation engine (referred to as "aggregation engine") based on cloud computing and some clustering algorithms. On the one hand, the aggregation engine distributes the user's search request to multiple academic search engines, and then collects and summarizes the paper results, thereby uniformly displaying the search results; on the other hand, from the large number of papers obtained from academic search engines, the aggregation engine analyzes their titles, abstracts, and keywords, clusters papers with similar research points, and visualize clustering results to users, so that the result can be macroscopic and intuitive, grasping all the papers and the research points they involve. This paper first introduces the research background of the subject, through the investigation of the existing search engine and the existing aggregate search engine, it analyzes the needs of the researcher and other customers for the paper search; according to the demand analysis, we proposes a system design that distribute the search request to multiple academic search engines and then aggregated results. Finally, we realised the implementation of the aggregate search engine, and verified the effectiveness of the system through tests

1. Introduction

While providing great convenience for researchers, the massive amount of information data collected by academic search engine also brings a lot of trouble to academic researchers who try to find papers that meet the needs. A large number of articles obtained for search results often require users to read through to understand the content of the article. And the varied quality of the papers, users need to spend a lot of time and energy to locate the required papers. Existence of multiple engines also force users to have to traverse them to fully understand the overview of that field.

Aggregating the search results of academic search engines is a viable solution to existing problems, and several academic aggregate search engine products are available today. The light academic
aggregate search engine is a large third-party aggregate search engine in China, integrating the resources of more than ten academic search engines such as Baidu Research, Google Scholar, Bing Academic, and China National Knowledge Infrastructure into a unified user interface. Search engines select and utilize appropriate search engines to implement retrieval operations. However, light academics simply aggregates the addresses of search results from major search engines on one page, and does not realize the integration of search resources. Although it saves users the time from repeated input, it does not solve the fundamental problem. MeZW Academic Search is an aggregated search engine website whose search results are displayed by calling, controlling and optimizing the search results of several other independent search engines and displaying them in the same interface in a unified format. Although the search results have been processed to a certain extent, they are still listed one by one, and the result content is not classified. It still takes a lot of time for the user to filter the required resources. By investigating the related products of the industry's aggregate search engines, these search engines have the disadvantages of high information repetition rate, single search function, and lack of cluster analysis, which increases the online remaining time for users to obtain effective information.

In order to solve these problems, this paper designs and implements an academic aggregate search system, which can integrate the resources of multiple professional academic paper platforms (Baidu, Bing), realize parallel search, search more widely, obtain more resources, and integrate and classify resources. It bringing together papers with similar research content or research hotspots in the field, and present the intelligently processed search results to users intuitively and comprehensively, and adopt a friendly image front-end presentation method such as visual bubbles. In this manner, users have a global understanding of search results, and quickly focus on their areas of interest, saving them a lot of time.

Firstly, this paper introduces the relevant background of the academic search aggregation engine and related product research, and then introduces other scholars' work in the field of web document collection result aggregation, unsupervised text classification, etc. Then we introduce the academic search aggregation engine system proposed in this paper. The design is introduced, and the suffix tree clustering algorithm for academic papers in Data Analysis Subsystem is introduced in detail. Finally, we implement the k-means algorithm and the suffix tree clustering algorithm for academic papers in the academic search. We then evaluate the search engine in terms of ease of use and accuracy, and finally prove that the academic search aggregation engine using the suffix tree clustering algorithm for academic papers can better help researchers accurately locate the required articles.

2. Related Works

Many researchers have conducted in-depth research on the direction of web document search results aggregation: Salve Bhagyasri, Girdhar et al. proposed a new algorithm for reconstructing web search results by clustering pseudo-documents[1]. The feedback session is used to display the click order of the user to specific keywords. These sequences are mapped to the TF-IDF vector for generating pseudo-documents, and the K-Means clustering algorithm is used to generate the user search target from the pseudo-document, and search results are constructed according to the user search target to the network. And a new algorithm is proposed to reconstruct the network search results; Zhi Huang et al. managed to cluster the search results of the ambiguous query into query subjects related to the Wikipedia Disambiguation Elimination Page (WDP) build[2]. To improve the clustering results, a concept filtering method is proposed to filter the semantically unrelated concepts in each topic. In addition, a pre-K full relationship (TKFR) algorithm is proposed, which assigns search results to related topics based on the similarity between the results and the concepts in the subject; Marek Kozlowski et al. proposed a new method for clustering network search results based on frequent term set mining is proposed[3]. Firstly, the meaning of the query is obtained by word meaning induction, and the meaning is used to identify the tree of frequent closed term sets, and then cluster according to the vocabulary and semantic intersection of the search results and the sense of induction; Carlos Cobos, Andrea Duque and so on. For clustering of Web search results (Web Document Clusters (WDC)), a hyper-heuristic framework called WDC-HH was proposed[4], which allowed the definition of the best algorithm for WDC. The hyper-heuristic framework uses four advanced heuristics (performance-based level selection, taboo
selection, random selection, and performance-based roulette selection) to select low-level heuristics (to solve specific problems with WDC). Bhagyashri Girdhar Salve et al proposed to generate a feedback session from the user clicking the log[5]. A pseudo-document is generated by calculating a TF-IDF (term frequency inverse data frequency) vector for each URL in the post-clicked log using a feedback session. Then, using the K-means clustering algorithm, these pseudo-documents are clustered to generate a user search target, and the user search target is reconstructed, so that the user can obtain specific information quickly and correctly. Then, use the CAP indicator to calculate the effect of each user's search target. These indicators show how to reorganize them correctly.

For unsupervised text classification algorithms, some researchers have proposed adaptive heuristic algorithms: Shafiabady, Niusha et al. aimed at the problem that the expert knowledge used to organize and mark the training document dataset correctly is very limited, expensive, or even unavailable. They proposed a method of automatic clustering using self-organizing map (SOM) and correlation coefficient (CorrCoef)[6]. The automatically clustered data serves as a training label for support vector machine (SVM), and tested on a standard text dataset. The results show that the proposed classification method is better than the expert knowledge training classifier in terms of accuracy. In the case of unsupervised learning, some researchers have proposed using unsupervised learning to extract the subject or subject word of the document: Yashodhara Haribhakta et al. extracted the keywords by automatically identifying the relationship between words in a set of unstructured data without any training data set[7]. Keyword extraction is based on the assumption of a word decomposition, assuming that the word in the bigram or trigram word vector is the potential distribution of words in the unigram word vector. After extracting the keywords, use an unsupervised method to find the most appropriate topic for each document.

3. System design

3.1. Overall framework

Academic search aggregation engine is made up of three subsystems: data acquisition subsystem, data analysis subsystem and data visualization subsystem, which is shown in 0. The workflow of Aggregate Academic Search is shown as 0. To be more specific: After the front-end of system obtains the keywords searched by the user, it inputs the data to obtain the data acquisition subsystem.
The data acquisition subsystem is responsible for data collection and information acquisition. It crawls the Bing academic and Baidu academic according to the received input keywords and puts the crawled papers into the queue, and then it sends them to the data analysis subsystem for clustering analysis.

The data analysis system subsystem does the cluster aggregation for the collected papers. After the aggregation, the result is sent to the data visualization subsystem.

The data visualization subsystem presents the result to the user.

3.2. Design of core module

Data Acquisition System

The data acquisition subsystem is responsible for data collection and information acquisition. This system supports the paper search of Baidu academic and Bing academic. According to the keywords entered in the user searching box, the data acquisition subsystem crawls all the paper information which is searched through HTTP request. Each paper is a separate entity with the title, URL, abstract, author, source, reference book, publication date, keywords and other information of the paper. The papers are loaded into the queue they are sent to the data analysis subsystem for clustering analysis.

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

3.3. Data Analysis Subsystem

The data analysis subsystem is responsible for the cluster aggregation of the collected papers. Users get a large number of searching results from Baidu academic and Bing academic through keyword searching, which often contain different academic papers of with both good and bad quality. However, users are not satisfied with the results for the most time. In order to find the articles they need, users need to browse each paper in turn to understand the content of the article. Therefore, the data analysis subsystem should do word segmentation and remove words that are not used from the title, abstract and other information of each paper, for the received paper queue from the data acquisition subsystem. The result of segmentation is used as the input of cluster analysis.

In the process of clustering analysis of article queue, suffix tree clustering algorithm is used. Suffix tree is a data structure used to deal with string problems. It describes all suffixes of a given string, and many important string operations can be implemented quickly on the suffix tree. Suffix tree is first proposed to support effective string matching and query, while suffix tree clustering algorithm, which regards text as a phrase string rather than a subset, can make full use of the approximate information between words to achieve the ideal clustering effect.

3.3.1. Suffix tree

Suffix tree is defined as[8]: A suffix tree T of string S with m words is a directional tree with m leaf nodes. These leaf nodes are labeled from 1 to m in separately. The internal nodes except the root node have at least two child nodes. Each edge in the tree is identified by a non-empty string of S.

For any leaf node i of suffix tree, the edge on the path from root node to i is the suffix substring of s from 1 to m, that is $S[i ... m]$. We add the character $\$, which does not belong to the character set, at the end of the suffix of the non-leaf node, to indicate the end at the leaf node.

3.3.2. Suffix Tree Construction Algorithm. The Suffix Tree Construction Algorithm is shown in Table 1.
Table 1. Suffix Tree Construction Algorithm

1. Initialize a suffix tree \( T = \{ \} \).
2. Create a root node \( T = \{ \text{root} \} \).
3. Add the first leaf node which is noted by 1 while the edge is noted by \( S[1 \ldots N] \).
4. For \( i=2 \) to \( N \) do
5. Create a sub-node \( B \), which is noted as \( i \).
6. Traverse \( T \) to find the longest prefix \( S[i, r] \) of \( S[i, N] \).
7. If we don’t find, we add a sub-node \( B \) at the root node. The edge is noted as \( S[1 \ldots N] \).
8. Else if \( r < N \), we split node \( r \) and add a node \( B \) under the split node. The edge is noted as \( S[r, N] \).
9. Else if \( r = N \), we add a node \( B \) under node \( r \), the edge is noted as $.$.

3.3.3. Suffix Tree Cluster Algorithm. The basic idea of Suffix Tree Cluster Algorithm is to realize clustering based on document public phrases, which is shown as Table 2.

Table 2. Suffix Tree Cluster Algorithm

1. Text preprocessing
2. Do word segmentation for the text
3. Remove the stop words
4. Calculate TF-IDF to find the keywords of the text.
5. Construct suffix tree model of text set for keywords of each text.
6. Using suffix tree model to find base class nodes.
7. Merge the base class nodes according to the similarity between the base classes.
8. Calculate the similarity of base classes.
9. Take the base class as the vertex and connect the base class with similarity of 1 to construct the base classes association graph.
10. Cluster based on connectivity between base class nodes.

TF-IDF in the text preprocessing is actually \( TF \times IDF \), where TF stands for word frequency which is the frequency of the word in the document. The formula is shown as following:

\[
TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}
\]

where \( i \) represents the word and \( j \) represents the text. \( n_{i,j} \) represents the frequency of word \( i \) appears in text \( j \). \( \sum_k n_{k,j} \) represents the sum of frequency of all words appear in text \( j \). IDF represents the inverted text frequency. The fewer documents contain the word \( i \), the larger IDF is, which means the word \( i \) is more representative. The formula is shown as following:
\[ IDF_i = \log_{10} \frac{|D|}{1 + |\{j: t_i \in d_j\}|} \]  \hspace{1cm} (2)

where \(|D|\) represents the number of documents in the text set, \(|\{j: t_i \in d_j\}|\) represents the number of documents which contain word \(i\). Because this number could be zero, we usually use \(1 + |\{j: t_i \in d_j\}|\) as denominator. Then we calculate the product of TF and IDF, which is shown as following:

\[ TFIDF_{i,j} = TF_{i,j} \times IDF_i \]  \hspace{1cm} (3)

The base classes mentioned in 0 means the middle node except the root node in the suffix tree. The general basic class similarity calculation is mainly based on the following formula. There are two basic classes \(B_m\) and \(B_n\), the similarity between them is:

\[ \text{sim}(B_m, B_n) = \begin{cases} 1, & \frac{|B_m \cap B_n|}{|B_m|} > \alpha, \\ 0, & \text{otherwise}, \end{cases} \]  \hspace{1cm} (4)

where \(\alpha\) is combining coefficient whose value range is \((0,1)\). It is a human defined threshold and we usually take \(\alpha=0.5\).

3.3.4. Optimized Suffix Tree Clustering Algorithm for Academic Papers. In this paper, we propose an optimized suffix tree clustering algorithm for the designed academic aggregate search engine. More specifically, we mainly optimize three parts —— the input of suffix tree, text preprocessing, base class nodes score.

Optimization of the input of suffix tree

Title, abstract and keywords are three parts that effectively summarize the main idea and algorithm for an academic paper. Therefore, they work as input to our clustering algorithm, which avoids the branch out of the subject caused by unrelated sentences in the body part of the paper.

Optimization of text preprocessing for Chinese academic paper

We optimize the text preprocessing procedure, in terms of title, abstract and keywords. Academic-related dictionaries with plenty of technical terms, such as cloud manufacturing, key management and support vector machine, are used in the word segmentation process, which contributes to the effective and error-free recognition of technical terms.

Optimization of base class nodes score

To calculate the similarity of base class nodes, we firstly give each node a score. After that, we pick several nodes with the highest scores and perform the base class combination. The score is defined as follow:

\[ \text{Scores}_i = ps_i \times ds_i \times ss_i \]  \hspace{1cm} (5)

\(ps_i\) is defined:

\[ ps_i = e^{-\frac{(n_{w_i}-\alpha)^2}{\beta}} \]  \hspace{1cm} (6)

where \(w_i\) is the word vector of base class node \(i\), \(n_{w_i}\) is the total number of words represented by base class nodes, \(\alpha\) is the number of words represented by base class node \(i\), \(\beta\) is a hyperparameter denoted as an impact factor of the number of words. \(ps_i\) indicates the impact of the number of words represented by base class node \(i\).\(ds_i\) is defined:

\[ ds_i = \gamma \sum_{w_i} d_{w_i} \]  \hspace{1cm} (7)
where $\gamma$ is a weighted hyperparameter denoted the impact factor of $ds_i$, $dw_i$ is the number of documents containing $w_i$, $ds_i$ indicates the impact of the number of documents containing $w_i$. $ss_i$ is defined:

$$ss_i = \frac{\sum_{d \in D_{ki}} t_{ki,d} W_d}{\sum_{d \in D_{ki}} t_{ki,d}}$$

(8)

where $D_{ki}$ is the set of documents containing keyword $k_i$ represented by base class node $i$, $t_{ki,d}$ is the frequency of $k_i$ in document $d$, $W_d$ is the weight of document $d$. $D_{ki}$ indicated the impact of the frequency of keywords represented by base class node $i$. $W_d$ is defined:

$$W_d = \begin{cases} 
7, & d \in \text{Title} \\
6, & d \in \text{Keyword} \\
4, & d \in \text{Abstract} 
\end{cases}$$

(9)

data visualization subsystem

The data visualization subsystem uses the bubble chart in the front end. The size of a bubble is used to represent the number of articles in a category. The distance of bubbles reflects the similarity between different clusters. A bubble chart is essentially a scatter graph containing more information.

4. experiment and evaluation

Based on the above system design and the suffix tree clustering algorithm for academic papers, we used Java to implement each subsystem. What’s more, we used Restful interface to achieve remote calls between each subsystem. The system is deployed in a virtual machine. In the experiment, we use cloud computing of searching the Chinese to evaluate our system in terms of functionality, accuracy, and performance.

4.1. Functionality Analysis

We type “云计算” as input and start our academic search aggregation engine system. The result is shown in 0.

![Fig.3 The output of academic search aggregation engine](image)

The result shows that the academic search aggregation engine system can perform the expected functions, including acquisition, analysis and aggregation of papers in a given field.
4.2. Accuracy Analysis
We evaluate the accuracy of our system by clustering on different base class combination coefficient.

![Fig.4 Ablation on base class combination coefficient (y-axis stands for the number of cluster categories)](image)

The overall number of crawled papers is set to 144. It can be shown in Fig.4 that the number of cluster categories increases with the increase of the combination coefficient. When the combination coefficient is small, papers with different ground truth categories are aggregated into one category. When the combination coefficient is large, papers will be divided into excessive account of sub-categories, which leads to the crowded and confusing visualization front end. Therefore, we choose 0.4 as the optimal base class combination coefficient. The number of categories is 45. The result is shown in Fig.4.

4.3. Performance Analysis
Furthermore, we evaluate the performance of our system on the different number of crawled papers. System performance is measured by crawl time and cluster time.

![Fig.5 cluster result when the base class combination coefficient is 0.4](image)
As shown in figure 6, resulting from a multithreading crawler, the crawl time slightly grows when the number of clustered papers increases. However, cluster time cost by the suffix tree clustering algorithm increases when the number of clustered papers increase. Thus, we set the number of crawled papers to 182, avoiding too much time spent on clustering algorithm.

5. conclusion

In this paper, we design and implement an academic aggregate search engine based on an optimized suffix tree clustering algorithm to help researchers locate required research articles accurately and efficiently. Experiments show that our system can efficiently acquire, analysis and aggregate required research articles. What’s more, our visualization system elegantly presents the relationship between different categories in the form of bubble charts, which greatly facilitates the process of searching for documents.

The optimized suffix tree clustering algorithm proposed in this paper mainly focuses on the reasonability of the clustering in the academic scenario. In future, we attempt to optimize the time and space complexity of the system.

References

[1] Salve Bhagyashri Girdhar ; R. B. Wagh. A novel algorithm for restructuring web search results by clustering pseudo documents. 2015 International Conference on Information Processing (ICIP)

[2] Zhi Huang, Zhendong Niu, Donglei Liu, etc. A Novel Method for Clustering Web Search Results with Wikipedia Disambiguation Pages. Lecture Notes in Computer Science

[3] Marek Kozlowski. Web Search Results Clustering Using Frequent Termset Mining. Lecture Notes in Computer Science

[4] Carlos Cobos, Andrea Duque. Algorithm for Clustering of Web Search Results from a Hyper-heuristic Approach. Lecture Notes in Computer Science

[5] Bhagyashri Girdhar Salve; R. B. Wagh, etc. Restructuring web search results by generating feedback session and clustering pseudo documents. 2015 Conference on Power, Control, Communication and Computational Technologies for Sustainable Growth (PCCCTSG)

[6] Shafsiabady, Niusha (IMU, Malaysia); Lee, L.H.; Rajkumar, R.; Kallimani, V.P.; Akram, Nik Ahmad; Isa, Dino. Using unsupervised clustering approach to train the Support Vector Machine for text classification. Neurocomputing, v 211, p 4-10, October 26, 2016

[7] Haribhakta, Yashodhara (College of Engineering, Pune, Shivajinagar, Pune Maharashtra, India); Malgaonkar, Arti; Kulkarni, Parag. Unsupervised topic detection model and its application in text categorization.ACM International Conference Proceeding Series, p 314-319, 2012, Proceedings of the CUBE International Information Technology Conference, CUBE 2012

[8] Li Yanjun. High performance text document clustering.USA:Wright State University,2007.