Bidirectional Pattern Matching in Yearbook Data Preprocessing Platform: Postprint

Authors: Shi Liting, Zhang Qian, Zhong Yongheng, Hu Sisi, Li Zhenzhen

Date: 2017-11-08T00:00:00+00:00

Abstract

【Objective】 To achieve structured storage of yearbook indicator data and complete the update and entry of yearbook data. 【Application Background】 The yearbook preprocessing platform is a C/S tool platform that uniformly organizes, reviews, and uploads yearbook data, uses VC++ as the primary programming language, and provides a data foundation for yearbook database construction. 【Method】The bidirectional pattern matching processing is an improvement based on the WM pattern algorithm, utilizing word segmentation technology to extract information elements from entered indicators, employing stored procedures to achieve pattern set reduction, and using information bidirectional matching to ensure accurate and efficient matching. 【Results】 Through analysis of matching results from experimental data entry, it was found that bidirectional pattern matching has a high indicator matching rate and accuracy. 【Conclusion】 The bidirectional matching algorithm can meet the requirements of yearbook entry and improves the efficiency of yearbook data preprocessing work.

Full Text

Using Bidirectional Pattern Matching Model to Pre-Process Yearbook Data

Shi Liting[1], **Zhang Qian**[2], **Zhong Yongheng**[1], **Hu Sisi**[1], **Li Zhenzhen**[1]
[1](Wuhan Library, Chinese Academy of Sciences, Wuhan 430071, China)
[2](The 9th Designing Institute, China Aerospace Science & Industry Corporation, Wuhan 430040, China)

Abstract

[**Objective**] To achieve structured storage of yearbook indicator data and complete the update and entry of yearbook data.
[**Context**] The yearbook data preprocessing platform is a C/S tool platform for

unified yearbook data organization, auditing, and uploading, developed primarily using VC++ to provide a data foundation for yearbook database construction.
[**Methods**] The bidirectional pattern matching process improves upon the WM pattern algorithm by utilizing word segmentation technology to extract information elements from entered indicators, employing stored procedures to reduce pattern sets, and implementing bidirectional information matching to ensure accuracy and efficiency.
[**Results**] Analysis of matching results from experimental data entry demonstrates that bidirectional pattern matching achieves high indicator matching rates and accuracy.
[**Conclusions**] The bidirectional matching algorithm satisfies yearbook entry requirements and improves the efficiency of yearbook data preprocessing.

Keywords: Bidirectional pattern matching; Yearbook data; WM algorithm
Classification Number: G350

Yearbook data represents comprehensive, systematic, and accurate factual compilations that document the movements and development status of entities during previous time periods, encompassing comprehensive yearbooks, specialized yearbooks, statistical yearbooks, and regional yearbooks. Analysis of such data helps researchers understand current conditions and study development trends, serving as a valuable reference for summarization, statistical analysis, and comparative studies. Consequently, yearbook data is widely applied across various research fields and holds significant intrinsic value [1].

Currently, yearbook data lacks unified formatting standards and is predominantly stored in unstructured formats as data tables. Moreover, substantial differences exist in storage formats between national and regional data, which hinders data querying, processing, and analysis. No publicly available yearbook data entry platforms exist domestically or internationally. Manual data organization and entry would require substantial time and labor costs. To ensure data authenticity and accuracy while fully utilizing existing yearbook resources, we have developed a yearbook data preprocessing platform that processes massive volumes of yearbook data. This platform employs computers to uniformly enter indicators in diverse formats, batch-storing unstructured yearbook data into databases as structured data. This standardizes operational procedures, ensures data quality at the source, and provides industrial technology intelligence workers with a solid foundation for data analysis, thereby possessing important practical significance.

The platform' s automatic matching module can merge data describing identical indicators across tables from different years and store them in normalized form, facilitating indicator data queries and resolving data update issues. This module represents a critical step for successful automatic yearbook data entry. This paper presents exploratory research on automatic indicator data matching, designing and implementing bidirectional indicator data matching based on the characteristics of indicator data. This approach provides robust technical sup-

port for automatic yearbook data entry and enhances the efficiency of yearbook data preprocessing.

2.1 Platform Functional Design

The yearbook data preprocessing platform processes yearbook table files. Considering the relative independence of processing procedures for different yearbook types, the system employs a Client/Server architecture and is provided as an application tool for data entry personnel. This architecture reasonably distributes computation and data between client and server ends, fully leveraging the processing capabilities of client PCs while effectively reducing network traffic and server computational load [2].

To achieve normalized yearbook data storage, the system must implement automatic indicator system construction, automatic yearbook document data recognition, yearbook data matching and update entry, data auditing, and data uploading modules, as illustrated in [Figure 1: see original paper]. This paper focuses on the implementation of the matching and update module.

2.2 Research Status of Pattern Matching

The yearbook data matching and update module constitutes the core and most challenging aspect of system construction, with algorithm performance directly affecting data sustainability and entry efficiency. Multi-year indicator data is often stored across different indicator table files. When entering the most recent year' s indicator data, it is necessary to compare Chinese character strings describing indicators to locate corresponding original indicators for data updates. Achieving accurate and efficient Chinese character string matching thus becomes the key problem.

Chinese string matching, also known as string pattern matching, has a development history of over 40 years and represents a crucial research topic in text processing. Due to rapid growth in network information, pattern matching technology has been widely applied across various domains, including firewall filtering, web search engines, intrusion detection systems, and biomedicine [3-4], and has been implemented in most operating systems and application software.

The BF (Brute Force) algorithm represents the earliest string pattern matching algorithm and is the most fundamental and simple approach, also known as the brute-force algorithm. The KMP (Knuth-Morris-Pratt) algorithm, introduced in 1977, was the first algorithm to achieve O(n) time complexity [5]. Another renowned algorithm is the BM (Boyer-Moore) algorithm [6]. Although its worst-case time complexity is O(mn), its performance in most practical scenarios surpasses even the KMP algorithm. Under the assumption that characters appear with equal probability and are independent, Yao proved in 1979 that the algorithm' s average time complexity lower bound is O(nlog|Σm/m|) [7].

Among existing multi-pattern matching algorithms, classic approaches include

the AC algorithm and WM algorithm. In recent years, the string matching research field has continuously evolved, with numerous new algorithms emerging. According to incomplete statistics, nearly 40 classic algorithms were published before 2000, with over 50 new algorithms published subsequently [8].

The academic community has reached a consensus regarding string matching problems: often, the simpler the algorithm concept, the better its practical performance. Some researchers' work on pattern matching suffers from a disconnect between theory and practice, with algorithmic complexity continuously increasing but practical results proving unsatisfactory—often less efficient than classic AC and WM algorithms [9]. On the other hand, as network information continues to grow, the volume of data requiring processing has become increasingly large while demands for matching speed have intensified, presenting researchers with significant challenges [4]. Therefore, pattern matching research should focus on practical application effects while improving classic algorithms, combining theory with practice to develop methods suitable for large-scale dataset pattern matching while ensuring efficiency.

2.3 WM Algorithm Principle

Among multi-pattern algorithms, the AC algorithm and WM algorithm are classics, both requiring pattern set preprocessing. The AC algorithm [10] maintains a state machine, consuming more resources in construction time and space complexity than the WM algorithm. Moreover, if the pattern set is dynamically variable, the cost of dynamically adjusting the AC automaton is much higher than that of the WM algorithm. The WM algorithm [11] employs a jumping concept that eliminates the need to match some characters while using Hash hashing methods to improve processing speed, achieving high efficiency in practical applications. Therefore, this paper adopts the higher-performance WM algorithm for indicator matching processing.

The WM algorithm borrows the bad-character shift rule from the BM single-pattern matching algorithm. In multi-pattern matching problems, however, the pattern string set is relatively large, covering a correspondingly larger character set, which reduces the probability of bad characters appearing and decreases the sliding window jump distance, significantly reducing the efficiency of the aforementioned rule. Consequently, the WM algorithm introduces the concept of character blocks, treating several consecutive characters as a matching unit to increase the sliding distance after each matching attempt. The character block length B is typically 2 to 3 bytes [11-12].

The WM algorithm requires constructing SHIFT, HASH, and PREFIX tables, primarily utilizing hash table concepts to enhance performance. The SHIFT table records the sliding window movement distance for all character blocks in the character set when they appear in text T. The HASH table records the positions of all feature strings where the SHIFT entry is 0 and the suffix character block hash values are identical. The PREFIX table records the positions of feature

strings with identical prefix character block hash values in the feature string set. The specific matching process is as follows [11-12]:

1. Pointer p points to the suffix of text T' s sliding window. If p > Tend, exit; otherwise, calculate the hash value of the character block pointed to by p.
2. Query the SHIFT table to obtain the SHIFT[hash] value. If equal to 0, it indicates that the character block pointed to by p matches a suffix character block of some feature string; proceed to step (3). If not 0, execute p = p + SHIFT[hash] and return to step (1).
3. Query the HASH table; HASH[hash] points to the starting position of all feature strings that may match the current substring within text T' s sliding window.
4. Calculate the hash value prefix of the prefix character block of the current substring within text T' s sliding window.
5. For each feature string pointed to by HASH[hash], query the PREFIX table. If its PREFIX[hash] matches the prefix value, perform exact matching—conducting complete matching from the first character of the feature string and reporting results. After the entire matching process concludes, execute p = p + 1 and return to step (1).

3.1 Indicator Matching Design

The indicator data update module must complete the aggregation of historical data. Considering data rigor, matching result accuracy represents the primary criterion for determining update success. Under this premise, processing efficiency must be maximized. Direct application of the WM algorithm to indicator matching is infeasible primarily because the indicator system is massive, causing WM preprocessing time and space costs to increase dramatically and matching performance to decline sharply. Addressing the accuracy requirements of indicator matching and the characteristics of the indicator set itself, this paper processes the indicator set in the following ways to adapt the WM algorithm for indicator matching and improve matching performance:

1. Pre-screen the indicator pattern set to a certain degree, eliminating large numbers of irrelevant indicator sets.
2. Clean indicator text by removing irrelevant characters and extracting information element sets describing indicators for subsequent matching.
3. Incorporate reverse matching in the matching phase to further improve matching accuracy and efficiency.

Based on the above analysis, the flow design of the bidirectional pattern matching in this paper is shown in [Figure 2: see original paper].

3.2 Indicator Text Cleaning and Information Element Extraction

The first step in bidirectional matching processing is indicator text cleaning and information element extraction, with results directly affecting the accuracy of

subsequent Chinese character matching. Yearbook data is stored as Excel text files, requiring extraction of key information describing indicators from tables. Critical fields for indicator names—such as table names, row and column headers, and units—often contain Chinese character errors. These erroneous symbols or useless information directly impact indicator matching. For example, in "固定资产投资 2014," the "2014" represents year information rather than the indicator name and must be removed. Similarly, in "固定资产投资 (," the "(" is an erroneous symbol requiring removal. Therefore, before data matching, character strings extracted from Excel must be cleaned, and key information describing indicators must be extracted for subsequent matching to ensure character matching accuracy.

Based on the characteristics of yearbook indicator names, this paper designs and implements an operational flow for extracting key indicator information, as shown in [Figure 3: see original paper], including Chinese character recognition, Chinese word segmentation, data cleaning, and useful information extraction.

1. **Chinese Character Recognition**
   Indicator files contain numerous English translations corresponding to indicators, which constitute irrelevant matching information and should be removed. Therefore, it is necessary to distinguish between Chinese and English character blocks, extracting Chinese character blocks for subsequent operations. The platform adopts Unicode as the Chinese character encoding method, where Chinese characters range from 4E00 to 9FBF. Whether a character is Chinese can be determined by examining the display range of its ASCII code. However, proprietary English vocabulary within indicators, such as GDP, belongs to indicator information elements and should be retained.

2. **Chinese Word Segmentation and Erroneous Symbol Removal**
   Indicator description Chinese character strings contain useless information such as punctuation marks, auxiliary words, conjunctions, etc. This useless information interferes with matching results and affects matching efficiency. Analysis of useless characters in indicators reveals that removing irrelevant information based on part-of-speech achieves good results. Therefore, this paper employs a word segmentation system to segment Chinese indicator character strings, analyzes segmentation results, and removes irrelevant information based on part-of-speech to obtain key information sets describing indicators.

The platform utilizes the ICTCLAS word segmentation system, which supports custom dictionaries, delivers good segmentation results, and provides powerful data processing efficiency [13]. By analyzing segmentation results, miscellaneous items irrelevant to matching are removed, including punctuation (w), auxiliary words (u), conjunctions (c), prepositions (p), and quantifiers (q).

3. **Year and Unit Extraction**
   Year and unit are important parameters for describing indicator data and must be extracted. Year information extraction from tables is relatively

simple, as years are generally numeric and typically appear in table names or row/column headers, allowing identification based on numeric character ranges. For unit extraction, it is necessary to establish a unit vocabulary database to identify unit information in yearbook tables while also identifying cells explicitly marked with units to expand the unit information database.

For critical information describing indicators—such as table names, row headers, column headers, and units—comprehensive processing through the above steps removes irrelevant Chinese character miscellaneous items to obtain keyword sets describing individual indicators, i.e., indicator information element sets. Special attention must be paid to the fact that table names contribute significantly less to indicator description than row and column headers. Therefore, this paper extracts indicator names in the order of row headers, column headers, and table names, sorting the result set of this step by information weight to improve indicator matching efficiency. Indicator year and unit information are essential elements for normalized merging of indicator data.

3.3 Pattern Set Reduction

The indicator system is a massive data collection. Reducing it can decrease the number of pattern matching sets, eliminating large amounts of irrelevant indicator information and ensuring indicator matching processing efficiency. The indicator system is stored in an MSSQL database. Considering the large volume of indicator text, this step employs database queries, using indicator information elements as keywords for SQL conditional queries in the database. While the Like operator is commonly used for fuzzy queries in keyword searches, Like clauses force the database system to perform linear scans of text fields, reducing system performance [14]. To improve the efficiency of the entire indicator screening process, this paper utilizes database full-text retrieval technology for data queries, employing FullText to construct word indexes in text strings, thereby shortening retrieval time [14].

As this process requires frequent database operations, this paper implements the step using stored procedures. Indicator information elements are passed as parameters to the stored procedure, enabling large collections of T-SQL statements to be pre-compiled and optimized in advance, directly returning the filtered data set and improving execution speed.

The specific reduction process is as follows: Let the text requiring matching be A1. Perform information element extraction on A1 to obtain set A{a1, a2, a3, a4, a5···an}. Conduct conditional queries in the database using set A, setting a threshold of 10,000. Sequentially add a1, a2, a3, a4, a5···an to the query conditions within a loop. If no indicator data is found, remove invalid keywords. If query results exceed the threshold, continue the loop. If query results fall below the threshold, exit the loop and filter out the relevant indicator set B. Set B then represents all indicator name sets related to the table containing A1.

Simultaneously, to avoid subsequent repeated queries, it is necessary to record the condition set C from this query and return it in dataset form.

3.4 Forward Matching Processing

Forward matching processing is a string matching process that compares indicator information element set A{a1, a2, a3, a4, a5···an} with the screened indicator set B{b1, b2, b3, b4, b5···bm}, setting a threshold of 100 to identify the smallest qualifying character set B1 for reverse matching. Its main processing logic is similar to the indicator screening process. When adding a1, a2, a3, a4, a5··· an to comparison condition set C, if the condition already exists within set C, continue the loop; if not, add it to query condition set C. Due to the presence of invalid information elements in the information element set, after performing string matching between condition set C and indicator set B, if no relevant matching indicators are returned, the query condition must be removed. If the number of matched indicators is less than the threshold, proceed directly to reverse matching processing. The matching flow is implemented in pseudocode as follows:

```
for (i=0; i< A; i++) {
    char* sFilter= A[i];
    // If query condition is empty or already exists in query set
    if (sFilter.IsEmpty()||sFilter in C){
        continue;
    }
    C.Add(sFilter);    // Add to query conditions
    // WM algorithm implementation function
    nRes=FindIndWM (C ,B);
    if (nRes ==0){
        // If no values found, the query condition is invalid
        // Remove the query condition
        C.Remove(sFilter);
        continue;
    }else if(nRes>0&& nRes<100){
        // Within threshold range
        // Reverse matching
```

Character comparison employs the WM algorithm, with implementation details described in Section 2.3. The matching process pseudocode is as follows:

```
while(text<textend) {
    hashVal=hashBlock(text); // Calculate current block's hash value
    // Find block's bad character shift table (SHIFT) to get next match
    shift_{distance}=SHIFT[hashval];
    // Calculate current position's hash value
    if(shift_{distance}==0) {
        // Current block appears at end of some pattern
```

```
shift_{distance}=1;
p=HASH[hashval]; // Get starting position of all patterns possibly matching current
while(p) // Check if patterns in subset match;
text+=shift_{distance}; // Select next possible match entry
```

3.5 Reverse Matching Processing

Forward matching alone is insufficient to solve practical matching problems. Forward matching typically yields multiple indicators completely matching A. Even if set B1 contains only a single indicator text matching A, considering the rigor of matching data and to improve matching accuracy, matching results must be verified through reverse matching. Reverse matching represents further screening and verification of matching results based on forward matching. It operates inversely to forward matching, representing a character matching process from indicator set B1 to the entry indicator string A.

Forward matching results for indicators often yield one or multiple indicator data corresponding to the indicator requiring matching. For example, for the indicator in the national economic accounting module: "International Balance of Payments-Current Account-Income," the following corresponding indicators can be found: "International Balance of Payments-Current Account-Income," "International Balance of Payments-Current Account-Employee Compensation Income," and "International Balance of Payments-Current Account-Investment Income." In this case, reverse matching can remove incorrectly matched items such as "International Balance of Payments-Current Account-Employee Compensation Income" and "International Balance of Payments-Current Account-Investment Income," as shown in .

Table 1. Example of Reverse Matching Process

Indicator Requiring Matching	Forward Matched Indicator Set	Unmatched Characters
International Balance of Payments-Current Account-Income	International Balance of Payments-Current Account-Income	
	International Balance of Payments-Current Account-Employee Compensation Income	Employee Compensation
	International Balance of Payments-Current Account-Investment Income	Investment

The implementation approach is as follows: Extract information element sets from dataset B1 obtained through forward matching processing, and use them

to perform character matching against the indicator name A requiring matching. If a unique matching indicator can be found, the match is successful; if multiple matching indicators are found, the match fails. In Table 1, "Employee Compensation" and "Investment" do not appear in the indicator "International Balance of Payments-Current Account-Income" requiring matching. The reverse matching result is "no match," thus enabling identification of the unique matching item and successful matching.

4 Experimental Analysis and Conclusions

The platform was developed using Microsoft Visual C++ 2010, employing OLE/COM to operate Excel tables. The database server side uses Microsoft SQL Server, with client-side support for both Microsoft SQL Server and Access.

To verify the application effect of the bidirectional matching algorithm, entry and matching tests were conducted on the 2010 China Population and Employment Statistics Yearbook files, with the operation interface shown in [Figure 4: see original paper].

The 2010 China Population and Employment Statistics Yearbook contains 122 yearbook files. Matching and entry operations were performed on indicator data from all tables, with 110 files succeeding and 12 failing. The average processing time per table was 97 seconds. Among the 12 failed files, 8 were special files with discontinuous data records (e.g., data for the period from November 1, 2008, to October 31, 2009) that had limited value and did not require collection and entry. The remaining 4 files had table indicator reading issues; analysis revealed these were special tables requiring format adjustment before data entry.

Among the 110 successfully entered files, there were 4,892 indicators, with 4,278 indicators matched successfully. To verify the correctness of matching results, an error detection program was designed and implemented based on indicator data characteristics to select potentially incorrectly matched indicators, as shown in [Figure 5: see original paper]. Through automatic error detection by the program and manual batch auditing, 678 incorrectly matched indicators were identified. Incorrectly matched indicators can be corrected using the platform' s function for finding associated indicators, as shown in [Figure 6: see original paper]. For matched indicator data, batch entry is required after auditing; for unmatched indicator data, the platform has designed and implemented a function to find potentially associated indicators for semi-automatic manual association.

The yearbook preprocessing platform has achieved automatic entry of indicator data, standardizing the entry process and avoiding errors that may occur in manual entry, thereby saving considerable labor and time costs and playing an important role in yearbook data entry work. As one of the core steps in the yearbook entry platform, bidirectional matching processing has achieved high data matching accuracy, meeting expected goals and saving time and labor costs

for yearbook entry while ensuring the accuracy and security of yearbook data entry.

However, to achieve high matching rates, the algorithm' s processing speed has been somewhat sacrificed. Future work will further analyze the inherent patterns of indicators to research and implement methods for improving the processing speed of the bidirectional matching algorithm, thereby enhancing its practicality.

References

[1] Song Lili. Consideration and Exploration on Informatics in Yearbook [J]. Lantai World, 2013(02): 11-12.

[2] Fan Sheng. The Comparison Between C/S Structure and B/S Structure and the Ways to Access Web Database [J]. Information Science, 2001, 19(4): 443-445.

[3] Alomari O, Othman Z. Bees Algorithm for Feature Selection in Network Anomaly Detection [J]. Journal of Applied Sciences Research, 2012(8): 1748-1756.

[4] Wang Chunyu. The String Pattern Matching Algorithm Based on Edit Distance [D]. Qinhuangdao: Yanshan University, 2015.

[5] Knuth D E, Morris Jr J H, Pratt V R. Fast Pattern Matching in String [J]. SIAM Journal on Computing, 1977, 6(2): 323-350.

[6] Boyer R S, Moore J S. A Fast String Searching Algorithm [J]. Communications of the ACM, 1977, 20(10): 762-772.

[7] Yao A C. The Complexity of Pattern Matching for a Random String [J]. SIAM Journal on Computing, 1979, 8(3): 368-387.

[8] Faro S, Lecroq T. The Exact Online String Matching Problem: A Review of the Most Recent Results [J]. ACM Computing Surveys (CSUR), 2013, 45(2): Article No.13.

[9] Hou Miao. Research of Parallel String Matching Algorithm [D]. Harbin: Harbin Institute of Technology, 2014.

[10] Aho A V, Corasick M J. Efficient String Matching: An Aid to Bibliographic Search [J]. Communication of the ACM, 1975, 18(6): 333-340.

[11] Wu S, Manber U. A Fast Algorithm for Multi-Pattern Searching [R]. Report TR-94-17. Tucson, AZ: Department of Computer Science, University of Arizona, 1994.

[12] Wang Yipei, Shi Chun, Dai Shangjing, et al. An Improved Wu-Manber Multi-pattern Matching Algorithm for Chinese Encoding [J]. Journal of Chinese Computer Systems, 2015, 36(4): 779-781.

[13] Zhang Huaping. ICTCLAS2011 API Document [K]. Beijing Institute of Technology, 2011.

[14] Song Min. Research and Realization of Key Techniques of Library' s Digital Resource Integration Platform Based on SOA [J]. New Technology of Library and Information Services, 2009(9): 22-27.

Author Contribution Statement:
Shi Liting, Zhong Yongheng, Zhang Qian: Conceived the research idea and designed the research plan.
Hu Sisi, Li Zhenzhen: Conducted the experiments.
Shi Liting, Hu Sisi, Zhang Qian: Cleaned and analyzed the data.
Shi Liting: Drafted the manuscript.
Shi Liting, Zhong Yongheng: Revised the final version of the paper.

Conflict of Interest Statement:
All authors declare no conflict of interest.

Supporting Data:
Supporting data is self-archived by the authors, E-mail: shilt@mail.whlib.ac.cn.
[1] Shi Liting. Yearbook2010.zip. China Population and Employment Statistics Yearbook 2010.
[2] Shi Liting. Inds2010.xlsx. China Population and Employment Indicator Classification System.
[3] Shi Liting. yearbook_{result2010}.xlsx. China Population and Employment Statistics Yearbook 2010 Indicator Matching Results.

Received Date: 2016-03-09
Revised Date: 2016-05-27

Note: Figure translations are in progress. See original paper for figures.