Levine et al., J Comput Eng Inf Technol 2016, S1
DOI: 10.4172/2324-9307.S1-003

Journal of Computer Engineering & Information Technology

Research Article

A SCITECHNOL JOURNAL

Performance Comparison between Two Solutions for Filtering Data Sets with Hierarchical Structures

Levine M[1], Osei D[2], Cimino JJ[3], Liu C[1], Phillips BO[4], Shubrook JH[5] and Jing X[4]*

Abstract

Controlled terminologies with hierarchical structures are utilized widely to code diagnoses (e.g., International Classification of Diseases, ICD) and other medical concepts (e.g., Medical Subject Headings, MeSH) in healthcare data sets. The coded data sets can be useful for advanced statistical analysis or to explore aggregated effects by using multiple data sets across institutions. The analysis of results can be evidence for administrative decisions (e.g., resources allocation) or to validate hypotheses. However, publicly accessible analytic tools for such data sets are lacking. Our research team has developed and published the methods for filtering, analysing and visualizing such data sets. Current work focuses on the development of an online tool to assist other researchers with applying our methods. We report on a comparison of two approaches to developing the tool in order to provide evidence about the selection of tools and programming language.

Solution A uses MySQL14.14 and Python 2.7.6 and solution B uses MongoDB 3.04 and C++ (g++ 4.84) for data storage and algorithms implementation. Both solutions are Web applications. A virtual private server was set up on the cloud with the following specifications: 2 CPUs (2.4 GHzX2), 4GB RAM, and Ubuntu 14.04. At every stage, results were cross-verified until both implementations produced the same output. We compare the time to run the algorithm for a test data set (2011 MeSH data). The algorithms use class count (CC), ratio, and node count (NC) as filters. Solution A is faster at filtering NC and ratio while solution B is faster at filtering CC. However whether these differences are significant to human users needs further study.

Keywords

Filtering data sets; Python 2.7.6 scripting language; MySQL 14.14 database; RESTful API; Web application Controlled terminologies; Data analytic tools

Introduction

Controlled terminologies with hierarchical structures are utilized widely to code diagnoses (e.g., International Classification of Diseases, ICD [1]) and other medical concepts (e.g., Medical Subject Headings, MeSH [2]) in healthcare data sets. The coded data sets can be useful for advanced statistical analysis or to explore aggregated effects by using multiple data sets across institutions. Analysis of these data sets can provide evidence for administrative decisions (e.g., resources allocation) or to validate hypotheses. However, analytic software tools for such data sets are lacking. Our research team has developed and published the methods for filtering, analysing and visualizing such data sets [3,4]. Current work focuses on the development of an online tool to assist other researchers with applying our methods. This rapid communication reports on a comparison of two approaches to developing the tool in order to provide evidence about the selection of tools and programming language.

We compare the execution time for a test data set (2011 MeSH data). The algorithms use class counts (CC), ratio values, node counts (NC) and both CC and ratio values as filters [3]. At every stage, results were cross-verified until both implementations produced the same output. The ancestor-descendant table and parent-child table for the test data set are available for usage. The same test work flow for both solutions is: 1) upload the test data set; 2) display summary page (general statistics about the test data set, such as valid nodes, invalid nodes); 3) select filter methods (CC, ratio values, NC, CC+ ratio values); 4) display the preview of the results by using the selected filter; 5) set threshold for the selected filter; 6) execute and record results. The executive time is recorded in step 6. Every test point for both solutions shown in Figures 1-4 is an average executive time of five tests, i.e., the average of five tests per same filter, same threshold and same solution.

Solution A

Solution A employs a MySQL 14.14 database and Python 2.7.6 scripting language. The popular Python web micro-framework Flask serves up the web application. Flask is a lightweight server framework that provides the RESTful API to the client side web application. Angular JS is a model view controller framework that creates the user interface where the user interacts with the web application. NetworkX and Matplotlib are used for data analysis and results plotting.

Data modelling: Ancestor-descendant and parent-child tables are loaded directly into the MySQL database with minimal pre-processing required. The ancestor-descendant and parent-child tables are queried by the Python application and the graph model is built.

Data access and initial manipulation: The design of the Python application attempts to minimize the required amount of transactions with the MySQL database. Relevant data are pulled from the database when a user decides to perform filtering or requests a filtering preview.

Filtering: Filtering of the overall graph is achieved by querying the MySQL database for nodes that meet filtering requirements. The query results are used to create a graph model in memory using the Python library NetworkX. The resulting graph model can be further manipulated and filtered as required by the user.

Solution B

This alternative employed MongoDB 3.04 database and the C++ (g++ 4.84) programming language. The popular Boost C++ libraries and Wt framework were used to develop the web application. Wt framework is a server-side programming framework and was chosen because of its ability to use fast C++ libraries in the web application. It also supports several browsers (such as Firefox/Gecko, Internet Explorer, Safari, Chrome, Konqueror, and Opera).

Data modelling: Ancestor-descendant and parent-child tables (usually called collection in MongoDB) were modelled into a

***Corresponding author:** Xia Jing, Assistant Professor, College of Health Sciences and Professions, Ohio University, Athens, Ohio, USA, Tel: 740 593 0750; E-mail: jingx@ohio.edu

Received: March 15, 2016 **Accepted:** July 05, 2016 **Published:** July 12, 2016

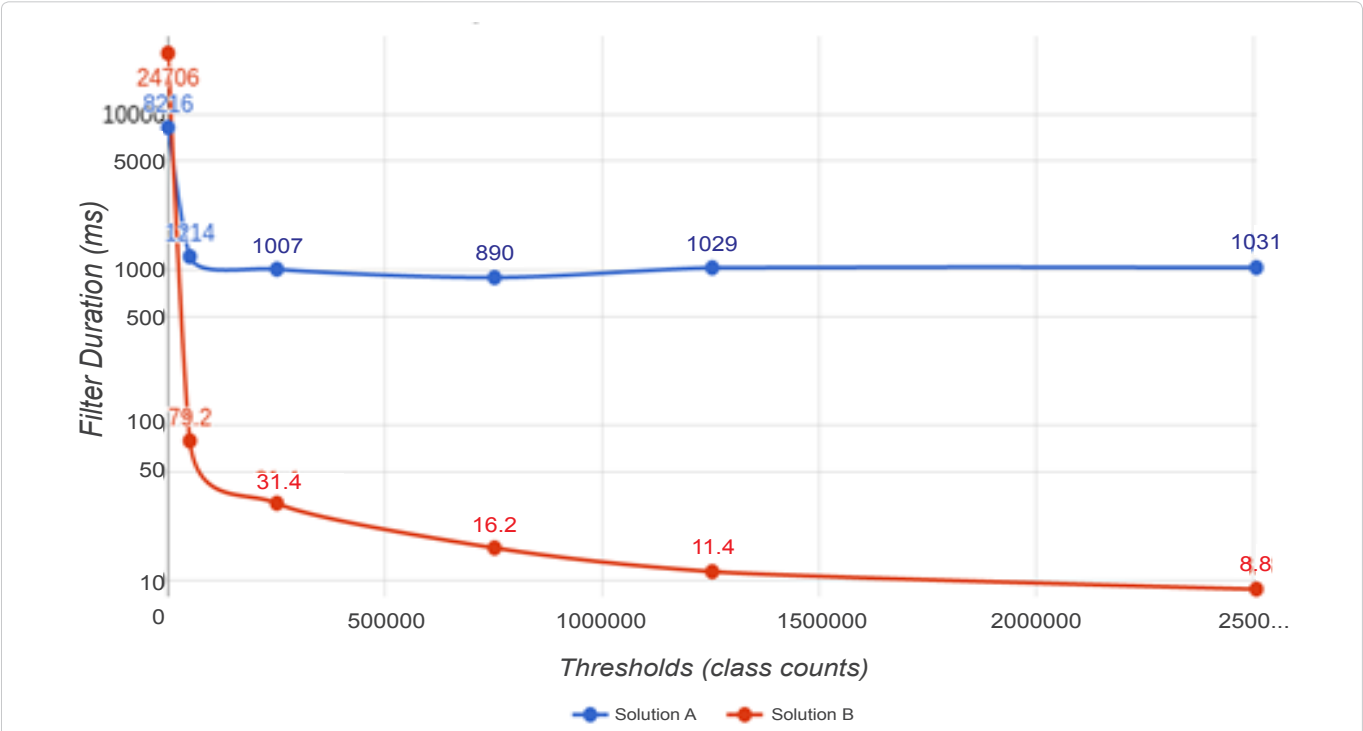

Figure 1: Executive time comparison for two solutions by using class counts as filters.

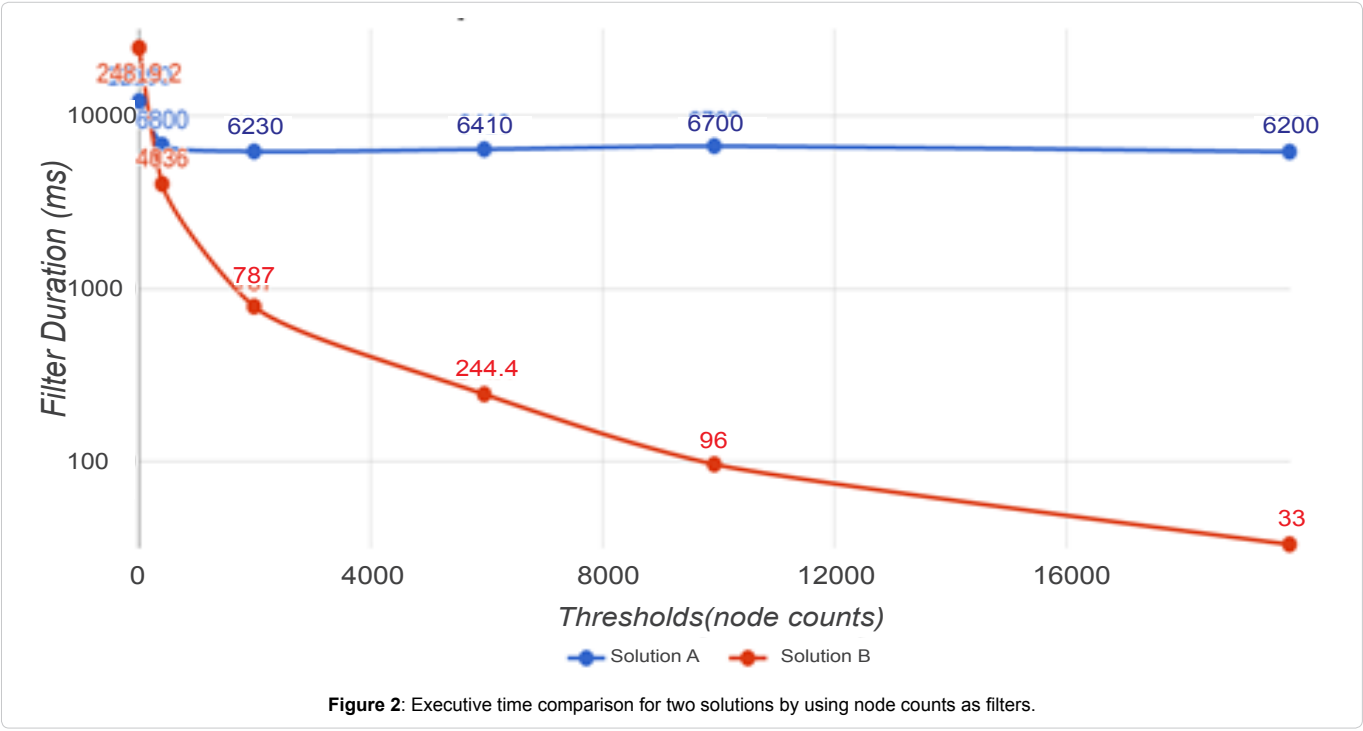

Figure 2: Executive time comparison for two solutions by using node counts as filters.

structured data type using a program written as part of the project. This stage was to leverage how MongoDB stores data, to save time and network capacity when an item is queried. The data are organized in such a way that when a node is queried all the relevant information about it is readily available such as, all of a node's ancestors and descendants or parents and children. This eliminates excessive memory usage or CPU consumption.

Data access and initial manipulation: The objective was to minimize the number of requests and responses between the web application and the database. Therefore filters such as CC and ratio values were pre-calculated on the database side and sent in bulk to the client web application.

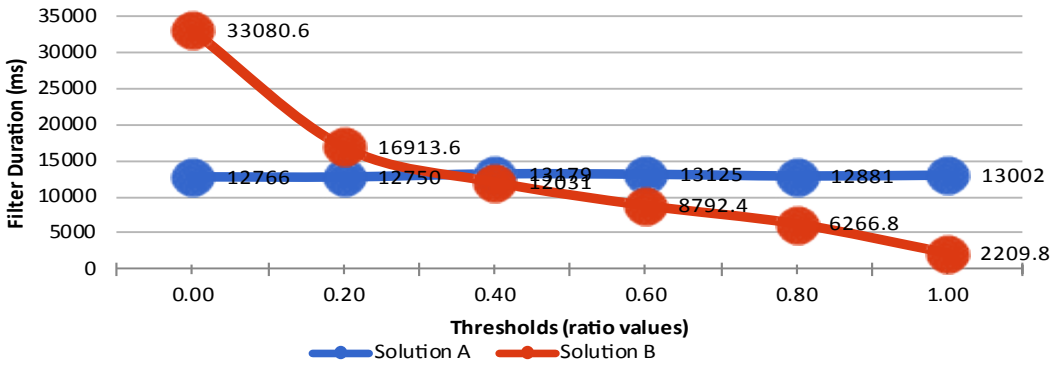

Figure 3: Executive time comparison for two solutions by using ratio values as filters.

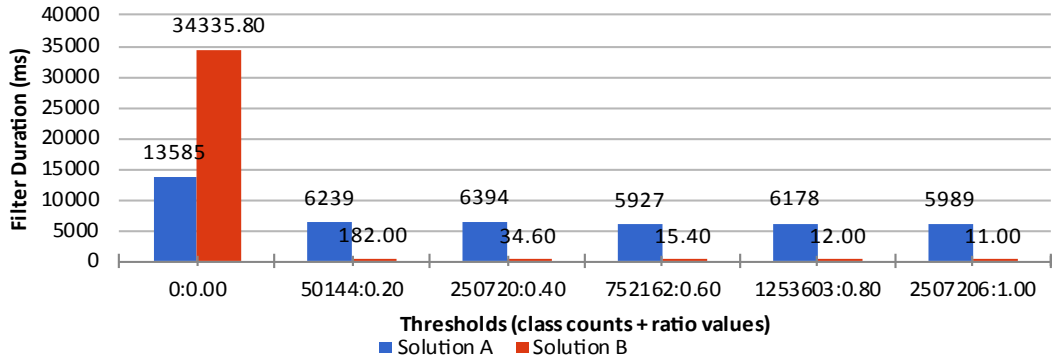

Figure 4: Executive time comparison for two solutions by using both class counts and ratio values as filters.

Filtering: Boost Graph Library (BGL) was the main library used for graph modelling and filtering. A base graph model is first built. This model sits in memory for fast access and manipulation. Filtering is implemented by creating a filtered graph for each algorithm using the base graph model. This way memory is conserved and data manipulation is accelerated.

Test

A VPS (Virtual Private Server) was employed to ensure an even base reading while testing our solutions. One solution was tested at a time to ensure a fair distribution of resources. The VPS had the following specifications:

Number of CPUs: 2

CPU: Intel(R) Xeon(R) CPU E5-2630L v2 @ 2.40 GHz

RAM: 4GB

Each scenario (i.e., a specific threshold for a specific filter) was tested five times and the average value was calculated for the shown figures. Filter time was determined as the time it took the solution to generate results after the user sets threshold and clicks the filter button (i.e., step 6). Time calculation was embedded in the solutions. Thus it was calculated within the source code and the output was displayed.

Results

Figures 1-4 summarize the testing results by using class counts (Figure 1), node counts (Figure 2), ratio values (Figure 3) and combination of class counts and ratio values (Figure 4) as filters.

Discussion

From the results show in Figures 1-4, it is a clear trend that solution B starts slow when there is no filtering (i.e., threshold=0). However with the threshold values increase, the execution time of solution B continues to decrease. On the other hand solution A reaches to a relative stable execution time in almost all filters or their combination eventually. The two solutions reach to the same execution time at some points when we implement all filters with different threshold values.

MySQL is a relational database management system and MongoDB provides dynamic schema, which can help store the hierarchical relationships, such as the path files we will use to generate graphs. However in terms of calculation of different filters (NC, CC, ratio, and CC plus ratio), the advantages are not obvious for either one. MongoDB may perform better with larger data sets, such as SNOMED CT (Systematized Nomenclature of Medicine-Clinical Terms) structure. For our test data set, it is inconclusive about which solution is faster.

Conclusion

Both solutions showed different strengths and weaknesses in different situations. Solution A was generally superior at handling lower filtering values. However its performance degrades with higher threshold values. Solution B started slow, however its execution time continued to decrease when threshold values increased. Whether these differences are significant to human users in real life scenarios needs further study. Using MongoDB for the project may have an advantage in generating and storing path files at later stage.

Acknowledgement

This project is supported by Ohio University Baker Fund and partially through Ohio University College of Health Sciences & Professions new faculty start up fund. Levine M and Osei D contribute to this work equally and their family names are listed in alphabetic order.

References

1. CDC. International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM) [Internet].
2. NLM NIH. MeSH [Internet].
3. Jing X, Cimino JJ (2011) Graphical methods for reducing, visualizing and analyzing large data sets using hierarchical terminologies. AMIA Annu Symp Proc 2011: 635–643.
4. Jing X, Cimino JJ (2014) A complementary graphical method for reducing and analyzing large data sets: Case studies demonstrating thresholds setting and selection. Methods Inf Med 53: 173-185.

Author Affiliation

Top

[1]*Russ College of Engineering and Technology, Ohio University, USA*

[2]*Scripps College of Communication, Ohio University, USA*

[3]*Informatics Institute, School of Medicine, University of Alabama, Birmingham, Alabama, USA*

[4]*College of Health Sciences and Professions, Ohio University, Athens, Ohio, USA*

[5]*College of Osteopathic Medicine, Touro University, Vallejo, California, USA*

This article is published in the special issue **Current Trends in Applications for Software/Hardware Integration** and has been edited by Dr. J. Todd McDonald, University of South Alabama, USA