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

Levine M1, Osei D1, Cimino JJ3, Liu C1, Phillips BO4, Shubrook JH and Jing X4*

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

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. AngularJS 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
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.
Filtering: Boost Graph Library (BGL) was the main library used for graph modeling 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].

Submit your next manuscript and get advantages of SciTechnol submissions

- 50 Journals
- 21 Day rapid review process
- 1000 Editorial team
- 2 Million readers
- Publication immediately after acceptance
- Quality and quick editorial, review processing

Submit your next manuscript at www.scitechnol.com/submission