Extended Experiment Report

December 2018

1 Context-Driven Distance Functions

Jaccard similarity and cosine similarity are two very common measurements while comparing item similarities. Similarity measures are used in various ways, examples include in plagiarism, collaborative filtering in recommendation systems, and etc.

Jaccard similarity is given by

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A \cap B| + |A - B| + |B - A|}$$

Cosine similarity is given by

$$C(A,B) = \frac{|A \cap B|}{\sqrt{|A|\,|B|}} = \frac{|A \cap B|}{\sqrt{(|A \cap B| + |A - B|)(|A \cap B| + |B - A|)}}$$

Simply put, in cosine similarity, the number of common attributes is divided by the total number of possible attributes. Whereas in Jaccard Similarity, the number of common attributes is divided by the number of attributes that exists in at least one of the two objects.

In our experiment, we implemented both distances. Based on our result, the Jaccard distance can perform better compare to Cosine distance in terms of pruning performance. Here why Jaccard has better performance in our work, for instance, given two views which are A = ['chest_pain_types', 'oldpeak', 'avg'] and B = ['chest_pain_types', 'thal', 'max']. Similarity score of A and B is 0.3333 by Cosine similarity and 0.166666 by Jaccard similarity. Our work focus on dissimilarity between views. If disimilarity can be calcualte by $1 - \textit{Similarity Score}$, then the distance score of A and B is 0.667 by Cosine dissimilarity and 0.833 by Jaccard dissimilarity.

The result shows that Jaccard distance has higher dissimilarity score compare to Cosine distance. Hence, using Jaccard distance function makes the schemes has better performance in terms of pruning rather than using Cosine distance function.

Note: 1 - Jaccard Coefficient can be used as a dissimilarity or distance measure, whereas the cosine similarity has no such constructs. I am not sure how to calculate dissimilarity score using Cosine similarity.

2 Prediction Interval

We decide to rely on non-parametric predictive interval models to determine maximum value with certain level of confidence. However, nonparametric methods are most appropriate when the sample sizes are small. When the dataset is large (e.g., N > 1000) it often makes no sense to use nonparametric method.

In case of the large dataset, we can use parametric method with the equation as follows:

$s = (X^2 NP(1-P))/(d^2(N-1) + X^2 P(1-P))$,

Where:

s = required sample size.

X^2 = the table value of chi-square for 1 degree of freedom at the desired confidence level (3.841).

N = the population size.

P = the population proportion (assumed to be 0.5 since this would provide the maximum sample size).

d = the degree of accuracy expressed as a proportion (0.5)

The table detail of determining sample size from a given population can be seen on the link below [1].

3 Hierarchical Attributes

The dataset contains such as location dimensions (country, state, province, city), time dimensions (Year, Quarter, Week, Days), product dimensions (product categories) as shown in Figure 1 commonly can be processed by OLAP. We need to find a way to calculate the similarity and dissimilarity between two views

[1] https://journals.sagepub.com/doi/abs/10.1177/001316447003000308

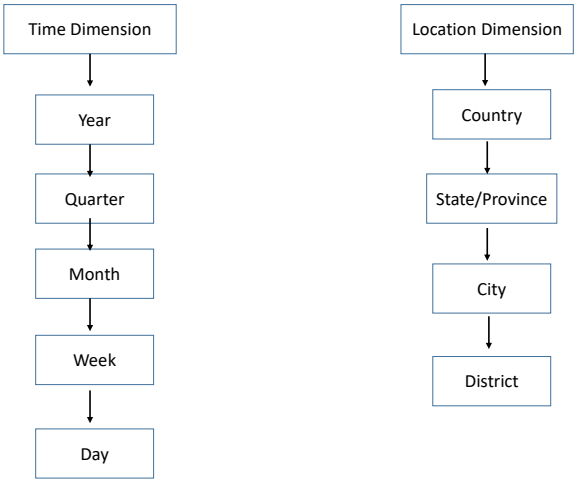

Figure 1: An example of hierarchical attribute

which have hierarchical attributes.

Given an example OLAP query as follows:

```
SELECT QUARTER, REGION, SUM(SALES)
FROM SALESTABLE
GROUP BY (QUARTER)
```

According to Figure 1, this query has two hierarchical attributes which are $QUARTER$ and $REGION$. There are several previous works which focus on measuring similairity between OLAP's queries. Such as in the example above, OLAP query usually has more than one hierarchical attributes. However, in our work, the view will never has more than one hierarchical attributes. A view in our works consists only three components (one categorical attribute, one numerical attribute, and one aggregate function). To the best of our knowledge, only categorical attribute that can possibly has a hierarchy.

The main issue in our work is how to compute the distance between two views which have hierarchical attributes. For instance, given two views: $A[year, sales, avg]$ and $B[month, sales, avg]$. Without considering hierarchy the distance value between $year$ and $month$ is 1 (0 if same and 1 if different). However, by considering hierarchy we can have different distance score.

A simple way is to use height parameter. For instance, as shown in Figure 1, time dimension has 5 levels (height). Given two views: $A[year, sales, avg]$ and $B[month, sales, avg]$, the distance between $year$ and $month$ is $\frac{|H(V_i)-H(V_j)|}{Len(H)} = \frac{|1-3|}{5} = \frac{2}{5}$. While the categorical attribute from both view is exactly same, the distance is 0, e.g., $year$ vs. $year = \frac{|1-1|}{5} = 0$.

4 Costs Reduction

4.1 SeeDB vs. DiVE

4.1.1 Naive Search

We implemented a SeeDB naive exhaustive search by looping over all possible combinations of attributes, measure attributes, and aggregation functions where all queries are executed and no pruning required. DiVE naive search is implemented in the same way as SeeDB in terms of query execution. In this experiment, the only different between SeeDB and DiVE is DiVE has a diversification.

Cont.

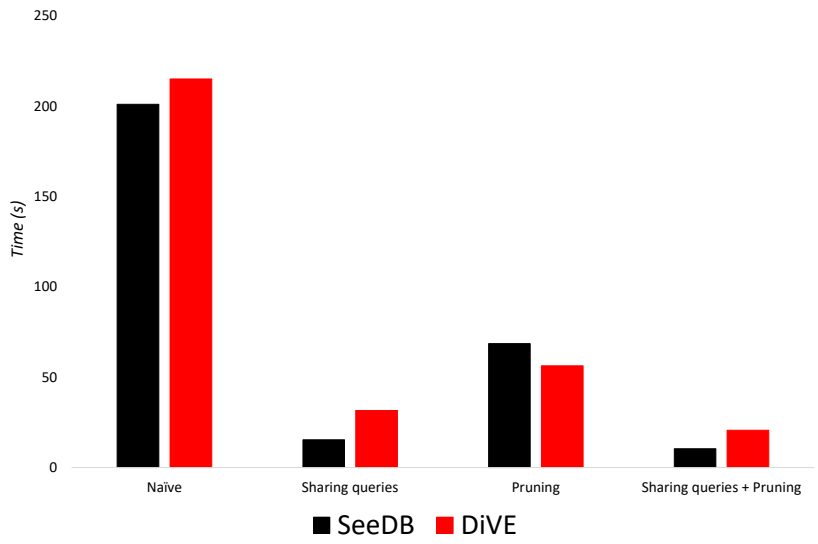

Figure 2: Performance comparision of SeeDB and DiVE with sharing queries optimization and pruning

4.1.2 Pruning optimizations

One way to further speed up the processing to find the top-k views, is to not evaluate queries with a low likelihood of being in the list of top-k most interesting views. We implemented pruning optimizations based on the paper. We splited the data into N phases (for example: N=15), then for each view currently being considered, we compute the importance score of that view on that partition. We then find the mean of the partition importance score thus far evaluated, and compute a confidence interval around that mean. If the upper-bound of the confidence interval for a view is lower than the lower-bound of the top-k views, then we drop that view from consideration. On the other hand, DiVE utilizes importance and diversity score to prune low-quality views. In this experiment, we used $\lambda = 0.6$ and $PI90$.

4.1.3 Query sharing optimizations

We followed the SeeDB paper that the autors applied sharing optimization by combining the queries with the same group-by attributes to perform multiple aggregations. This results in the execution of just two queries for every new group by attribute.

The version of published DiVE scheme used pruning as the way to reduce the costs. In this experiment, we applied query sharing on DiVE. Instead of executing query views start from the top directly, DiVE collected some views with same group-by and performing multiple aggregations.

4.1.4 Cost comparison result

SeeDB. The naive exhaustive search method of SeeDB has a runtime of 201 seconds. When we implement the sharing-based multiple aggregation optimizations, we get a runtime of 15.375 seconds, for a speedup of 13x. By using pruning optimizations on the naive method, we get a runtime of 68.6 seconds for a 2.9x speedup. Implementing both optimizations gives us a runtime of 10.4 seconds for a speedup of 19x over exhaustive search.
DiVE. The total cost of DiVE naive search is 215 seconds. DiVE with adaptive pruning has a runtime of 56.26 seconds for a 3.8x speedup. Implement the sharing-based multiple aggregation optimizations on DiVE,

Cont.

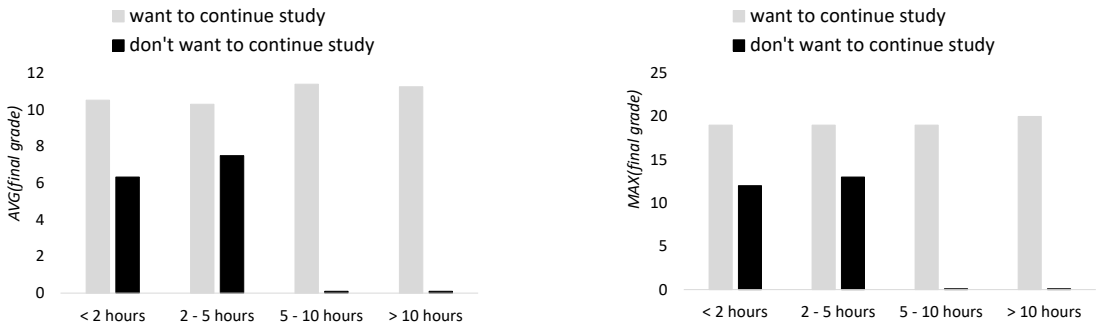

Figure 3: Study time and final grade of students who want to countinue their study to higher education vs. students who do not want to continue their study to higher education

we get speedup of 6.8x (37.57 seconds). While we implementing both optimizations gives us a runtime of 20.65 seconds for a speedup of 10.5x over exhaustive search.

These experiments were conducted on a Windows Dell with 3.6 GHz Intel Core i7-7700 and using Flights dataset. The detail comparison can be seen in Figure 2. Overall, the performance of DiVE is below SeeDB due to DiVE has diversity function that need to be calculated. Moreover, combining both optimizations (pruning and query sharing) give us optimum performance for both SeeDB and DiVE.

5 New Dataset

The student performance dataset is used in this experiment as the motivation example dataset. This dataset can be downloaded from Kaggle [2] and UCI ML website [3]. This dataset has 28 attributes, 5 measures, and four aggregate functions are used in this experiments. The data were obtained in a survey of students math and Portuguese language courses in secondary school. It contains a lot of interesting social, gender and study information about students. In this experiment, only math dataset is used.

For instance, the analyst wants to compare between students who want to continue their study to the higher education and students who do not want to continue their study to higher education. Figure 3 shows two views that have the highest importance score: 1) A = studytime, M = final grade, and F = AVG, the importance score is 0.812831582477977; 2) A = studytime, M = final grade, and F = MAX, the importance score is 0.797221012363291. This Figure shows without diversity, recommended views may suffer from redundancy.

Figure 3 also describes that students who want to continue to higher education relatively have better final grade compared to students who do not want to continue to higher education. Interestingly, some students who want to continue their education, they spend more time to study (5 to 10 hours, even more than 10 hours) per week. To contrary, students who do not want to continue their study, all of them only spend less than 5 hours a week for study.

[2] https://www.kaggle.com/uciml/student-alcohol-consumption
[3] https://archive.ics.uci.edu/ml/datasets/student+performance

The End.