A Novel Indexing Method for Improving Timeliness of High-Dimensional Data

Completed Research Paper

Jian Lu
Department of Computer Science
University of Massachusetts Lowell
jlu1@cs.uml.edu

Huong Pham
Manning School of Business
University of Massachusetts Lowell
huong_pham@student.uml.edu

Hongwei Zhu
Manning School of Business
University of Massachusetts Lowell
hongwei_zhu@uml.edu

Cindy Chen
Department of Computer Science
University of Massachusetts Lowell
cchen@cs.uml.edu

Abstract

Investment in information technology (IT) has been growing rapidly and one key reason for investing in IT is to improve information quality (IQ). Timeliness is an important IQ dimension that often needs to be improved for decision making. Especially in the era of big data, timeliness becomes more valued because of challenges of massive data size and high dimensionality. Many financial analyses require timely data to support time-critical decision making. In this paper, we develop a novel index method and effective query algorithms to reduce latency of querying high-dimensional data. The effectiveness of point, range, and similarity queries implemented using our methods is evaluated using a high-dimensional testbed conducted using real world financial data. Results show that our method outperforms existing methods in query speed of three types of queries frequently used in financial decision making.

Keywords

Information quality, data timeliness, delivery of high-dimensional data, database indexing

Introduction

Investment in information technology (IT) has been growing rapidly in the past few decades. One of the main objectives of investing in IT is to provide "the right information to the right people at the right time", i.e., to improve information quality (IQ) in organizations. This is particularly true given that information quality has been recognized as a multi-dimensional concept with dimensions such as accuracy, timeliness, and relevancy (Wang and Strong 1996).

For the timeliness dimension, we distinguish between two aspects. One is about the *currency* of data, which can be defined as "the extent to which the age of the data is appropriate for the task at hand" (Wang and Strong 1996). The other concerns the *latency* of data delivery, which can be measured by the time it takes to deliver the requested information. Latency leads to the poor timeliness, and then reduced utility of the information, i.e., it takes too long to deliver the data so that by the time the data eventually arrives it is already outdated to the time-critical decision making.

In the era of big data, being able to find and deliver the requested information quickly has become increasingly important. Many financial decision making tasks are time-critical and latency in data delivery will reduce the value of information to decision makers. In this research, we will focus on this latency issue and develop a novel method to improve the query processing speed for high dimensional financial data in relational databases. Indexing has been the primary approach to speed up query processing.

Numerous indexing techniques have been developed. However, most of these existing techniques perform poorly with high dimensional data. We addressed this particular issue by developing a **P**artition-**L**abeling technique to create a tree-based index. We called this technique the *PL-tree* indexing method.

This research makes the following contributions. First, we adopt the PL-tree indexing method that we have previously developed (Wang et al. 2013) for financial data; second, we develop a scalable querying system of point, range, and similarity queries for financial data; third, we evaluate the effectiveness of our system using a database of real world financial data of 26,649 firms from 1993 to 2012.

The structure of the paper is as follows. Section 2 gives a brief review of literature, including IQ issues, financial analysis methods and query processing techniques used in database and data mining. Section 3 presents our query processing system for financial data based on the PL-Tree index. Section 4 describes the empirical experimental results showing that our method can index and retrieve the data efficiently. The final section summarizes the contributions of the paper and points out areas for future research.

Literature Review

Timeliness and Financial Decision Making

Timeliness is one of the important IQ dimensions that organizations try to improve with IT investment. In this research, we will focus on a specific technique for improving timeliness. Several other factors contribute to timeliness issues. For example, when there is a large amount of data, delays will occur if communications and data processing technologies are inadequate (Strong et al. 1997). In large Enterprise Resource Planning (ERP) systems, users may withhold data entry for as long as possible when they fear potential adverse effects to downstream users and difficulties in data cleaning afterwards (Strong and Volkoff 2005).

Timeliness can affect, and be affected by, other IQ dimensions. For example, a lack of timeliness can lead to reduced accuracy of related data in large ERP systems because of tight-coupling of such systems (Cao and Zhu 2013). Spending time on cleaning data can improve accuracy at the price of reduced timeliness. In this case, a decision maker can optimize the tradeoff using techniques such as those described in (Ballou and Pazer 1995). The impacts of timeliness on accuracy and completeness with several architectural choices of distributed systems have been studied to help choose optimal system configurations (Cappiello et al. 2003). The dependency among quality dimensions (including timeliness) has also been investigated conceptually (Amicis et al. 2006; Barone et al. 2010) and empirically (Strong et al. 1997).

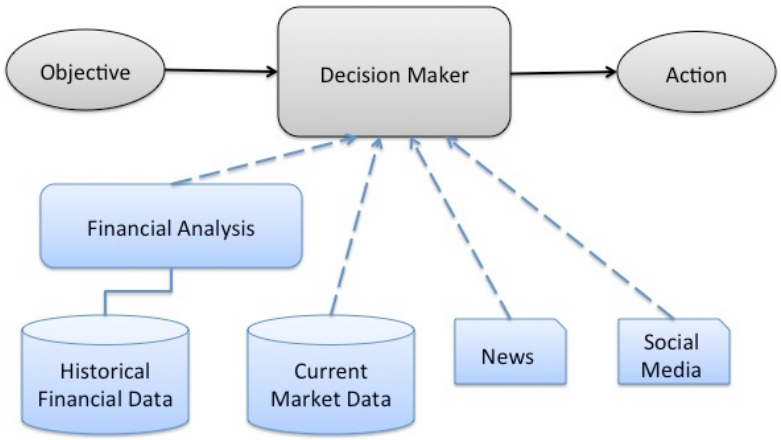

Figure 1. Financial Analysis and Decision Making

There has been an increasing need for better understanding the financial market since the recent financial crisis. With increasing amount of data and volatility of the market, it is important to provide the desired information in a timely manner. Figure 1 illustrates a typical scenario where decision makers draw information from multiple sources to produce insights for time-critical decisions. The latency of delivery for each information source will affect the response speed of the final action. Some financial analysis

could be prepared off-line, but in many cases the historical data should be analyzed on-the-fly. In such cases, the latency of delivery for the massive historical data could be the bottleneck in a time-critical decision making process.

For example, when a significant event is detected in Twitter feed and market begins to react to it, analysts need to quickly evaluate the situation by running a number complex models, which require relevant information from historical data bases. Each data source presents its unique challenges. Our research will focus on delivering historical data. Over time, data from the other sources will be part of historical data. The main challenges of historical data are in its large volume and high dimension. In terms of volume, a typical historical database has information about more than 30,000 companies spanning multiple decades. In terms of dimension, hundreds or thousands of attributes are used to describe each firm. A taxonomy recently adopted by the U.S. Securities and Exchange Commission to companies to describe their financial has more than 15,000 concepts (Zhu and Wu 2014).

Many financial analyses need not only base financial values (e.g., total assets and sales) but also financial ratios (Groppelli 2000) which are derived from base values. Financial ratios have been widely used in economics, finance, and accounting research to benchmark bank performance (Ho and Yun-Shan 2006), identify patterns in industry sectors (Gombola and Ketz 1983), and predict bankruptcy (Altman 1968; Atiya 2001; Ravi Kumar and Ravi 2007). The effectiveness of different financial ratios in producing useful insights has been studied in (Chen and Shimerda 1981). Among the identified financial ratios, 10~30 of them (Ho and Yun-Shan 2006; Serrano Cinca et al. 2005) are frequently used to sufficiently evaluate the performance of a firm.

Historical financial data is often used as input to analytics and data mining methods (Wu et al. 2008) to evaluate and predict financial performance. For instance, bankruptcy prediction studies published during 1968-2005 have used artificial neural networks (ANN), case-based reasoning, decision trees, and several other methods (Ravi Kumar and Ravi 2007). ANN has been used with 16 financial variables and 11 macroeconomic variables to predict firm performance (Lam 2004). Genetic algorithms have been applied to optimize financial ratio set and predict financial distress (Sun and Hui 2006). A model using *decision tree* and five financial ratios to have been used to predict the one-year-ahead changes of the stock earnings (Wang et al. 2009). Clustering can be used to eliminate the use of redundant financial rations in financial analyses (Wang and Lee 2008).

Despite the importance of historical data in financial analysis, there is no testbed for evaluating query efficiency. This research will develop a testbed using real world financial data.

Techniques for Efficient Queries

To implement the financial analyses in practice to deliver timely results, the input data must be obtained quickly. As noted earlier, financial data for a given analysis usually has 10-30 dimensions, which can be considered as high-dimensional data. High dimensionality is a result of trying to describe the objects via a collection of features, compared to traditional data (e.g. 2-D image, 3-D spatial data). We define the data of more than 3 dimensions as *high-dimensional data*. Sometimes, the term *ultrahigh dimensionality* is used to refer to hundreds or thousands of dimensions. In a high dimensional space, data is sparse and distance calculation is computationally intensive, which makes it difficult to find desired information. These problems are known as the "*curse of dimensionality*" (Bellman 1961). In the past several decades, many indexing methods for multidimensional data have been proposed to solve these problems. The R-tree family is the most popular indexing structure including many indexing methods. Among these methods, R*-trees (Beckmann et al. 1990) improved the R-trees (Guttman 1984) by minimizing the coverage and overlap between rectangles, which work well in many applications. Surveys of common indexing methods can be found in (Böhm et al. 2001). In this paper, we build the index of financial data by using a new indexing algorithm introduced in (Wang et al. 2013), which is developed for indexing high-dimensional data and outperforms the existing indexing methods on different queries.

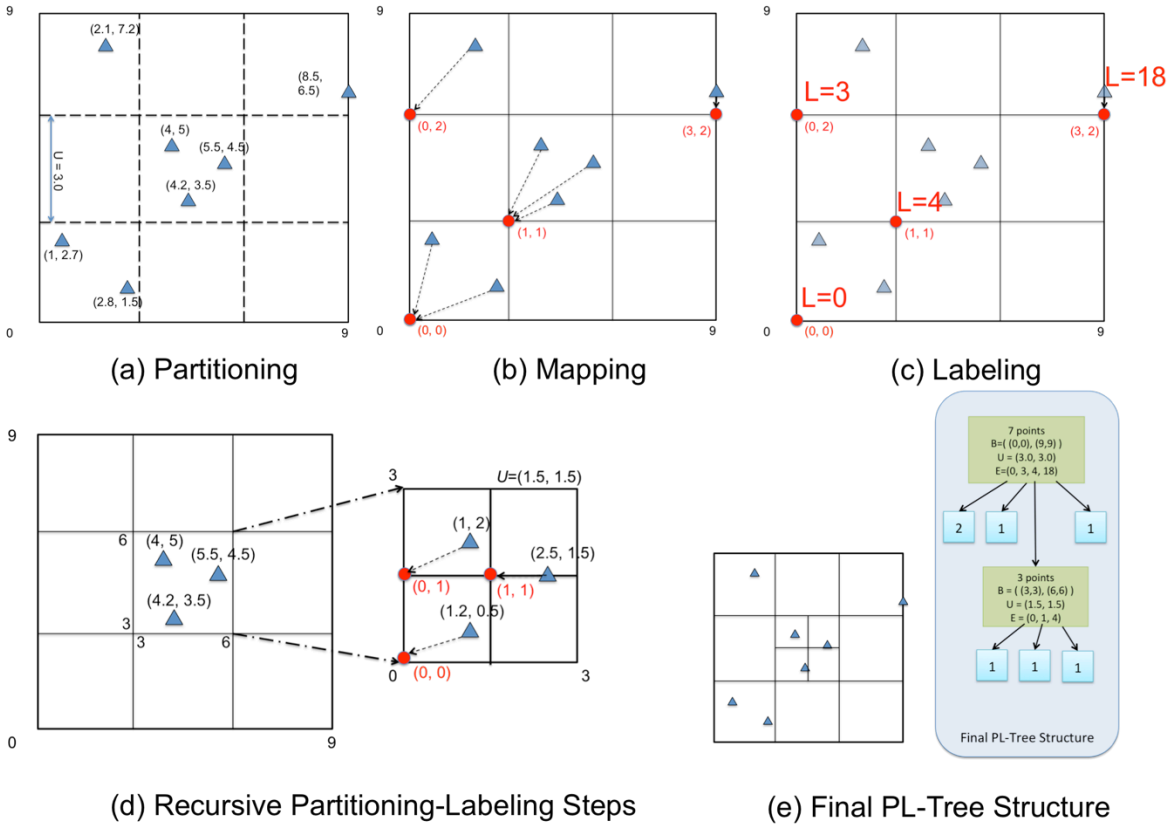

Figure 2: Partition-and-Label Process of PL-Tree

Methodology

In this research, we have designed a scalable query processing system for financial data. The system contains several components: data collection, index creation, and query processing. In particular, we focus on using index high dimensional data to ensure the timeliness, accuracy and completeness of the query result.

PL-Tree Algorithm

Our method is based on an indexing technique called *PL-Tree* that we have previously developed (Wang et al. 2013). The PL-Tree is a novel indexing algorithm for processing high-dimensional data. It uses algebraic techniques to reduce the negative impacts of the "curse of dimensionality". The PL-Tree supports efficient point queries, range queries, and *k*-nearest neighbor (*k*NN) queries, with the following additional properties: 1) overlap free; 2) strong scalability; 3) distribution insensitivity. These properties make it appropriate for indexing financial data, which are unevenly distributed and of large volume.

The main idea of the PL-Tree is to partition the multi-dimensional space into hypercubes recursively and map each hypercube into a unique scalar label. The algorithm includes the following steps:

1 Partition the space into hypercubes using $U = \{u_1, \ldots, u_d\}$, where u_i is calculated as the optimal bin-width according to the data distribution on the i-th dimension.
2 Map data points in each hypercube to the minimum point of the hypercube.
3 Label each hypercube with a scalar value generated by applying *Cantor pairing function* hypercube (Dovgoshey et. al. 2006) on the minimum point of the hypercube. The Cantor pairing function is a bijection between vectors and scalars, the bijective property ensures that the labels of the hypercubes are unique.
4 Repeat the partition-label procedure if a hypercube contains more data objects than a pre-determined bound (e.g. the size of a database page). That is, the hypercube is considered as a new

space and partitioned into sub-hypercubes, data objects in this new space are mapped and labeled recursively.

If we use a node to represent a hypercube and the sub-hypercubes as its children identified by a unique label value, the final structure of the data space can be represented as a tree of labels where the root is the node representing the whole data space and the leaves contain the data objects.

The PL-Tree index is constructed based on a recursive **P**artition-and-**L**abel procedure. Figure 2 shows an example of constructing a PL-Tree from 7 data points in the 2-dimensional space. Suppose we have 7 data points and each node of the PL-Tree contains no more than 2 points. Figure 2(a) is the partitioning step where the points are partitioned into different sub-hypercubes. For a hypercube, we calculate a vector $U = \{u_1, \ldots, u_d\}$ where u_i is the optimal bin-width for the data value projected to the i-th dimension ($U = \{3.0, 3.0\}$ in the example). The hypercube is partitioned by dividing the coordinates of i-th dimension by u_i. Figure 2(b) is the mapping step, where each data point is mapped to the minimum point of the hypercube, a minimum point is the integer result of the division where the divisor is u_i in the i-th dimension. Figure 2(c) is the labeling step. Every point is assigned with a label calculated by the Cantor pairing function on its mapped minimum point. In Figure 2(d), since there are more than 2 points in the middle sub-hypercube, a recursive procedure is needed to partition and label the three points. Figure 2(e) shows the final PL-Tree indexing these 7 points, as well as the hierarchical partitions of the data space.

Query algorithms

We use PL-Tree indexes to support the efficient implementation of three types of queries: point query, range query, and similarity query.

Point query

A point query is the simplest searching operation in a database system. Given a query point (a vector of attribute values), a point query asks for all the records with the exactly same values to the query point. The point query algorithm of our method is straightforward which starts from the root of the PL-Tree and search down to a leaf node, then search the same points in the leaf node.

Range query

A range query is a common database operation that retrieves all records where some value is between an upper and lower boundary. For example, suppose we have companies with their liquidity ratios, a range query may asks for the companies that have current ratio between 1.5 and 1.8 and quick ratio between 1.0 and 2.0.

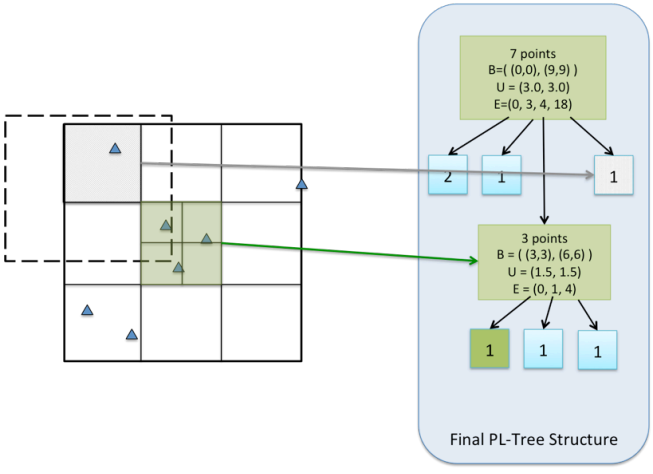

Figure 3: Range Query Illustration

The range query algorithm in a PL-Tree works as follows. Given by a range Q, it starts from the root of the tree. For each node, the algorithm examines its hypercube with Q and identifies the *innerblocks* (the sub-

hypercubes contained in *Q*) and *outerblocks* (the sub-hypercubes intersect with *Q*) by checking the labels that indicate the information of the coordinates of sub-hypercubes. For each *innerblock*, all data points (records) in it are returned; for each *outerblock*, we launch another range query on the sub-hypercube recursively. Figure 3 is an example of such range query algorithm, it firstly check the four sub-hypercubes of the root, and get 1 innerblock (left-top shadowed cube), 1 outer block (the center dark cube), and other 2 hypercubes outside the range. Then the data items under the node of the innerblock would be all returned, and the node of the outerblock would be checked with the query range recursively.

Similarity query

Identification of similar firms is important to both research and practice. Investors often benchmark a firm against similar ones. Target firms for merger and acquisition are usually identified using a similarity search. In our method, we implement a *similarity query* to find the firms with similar performance according to a set of financial items. The similarity query is based on the *k*NN query: given a query point *P* and a value *k*, a *k*NN query returns *k* data points which are closest to *P*.

Suppose we have some firms in the index and each firm consists of d values. Given a value k and a firm $P = (r_1, r_d, \dots, r_d)$, the similarity query is to find k firms that are most similar to the given firm. The similarity is measured by the *weighted Manhattan distance* function $dist(X, P) = \sum_{i=1}^{d} w_i |x_i - r_i|$ where $X = (x_1, x_2, \dots, x_d)$ could be any firm. We use the algorithm described in the paper (Hjaltason and Samet 1999). In the similarity query, the algorithm maintains a priority queue containing objects (node or data entry) sorted by their MINDIST to the query point *P*. The hypercube of a node is defined as a pair of the lower boundary $L = (a_1, a_2, \dots, a_d)$ and the upper boundary $U = (b_1, b_2, \dots, b_d)$, and the MINDIST of a node to *P* is defined as:

$$\text{MINDIST}\big((L, U), P\big) = \sum_{i=1}^{d} w_i (r_i - s_i), \text{ where } s_i = \begin{cases} a_i, & if\ r_i < a_i \\ b_i, & if\ r_i > b_i \\ r_i, & otherwise \end{cases}$$

And the MINDIST of a data entry to *P* is just the distance between these two points.

Initially, there is only the root node in the priority queue. Each time the object with the smallest MINDIST is retrieved from the queue. If the object is a node, we expand it by pushing its children into the queue, using the MINDIST as the key. If the object is a data entry, then it would be reported as the next similar data point. From the definition of MINDIST, we know that, if all weights are non-negative ($w_i \geq 0\ \forall i = 1,2, \dots, d$), the value of MINDIST of a hypercube must be smaller than that of any point contained in it.

Performance Evaluation

We evaluated the performance of the method using two real world datasets with 6 and 14 dimensions, respectively. We compare the performance of our method against the R*-tree (Beckmann et al. 1990) and sequential scan for the three types of queries. R*-tree is a commonly used indexing mechanism for multi-dimensional data in existing database applications (e.g. Oracle spatial index, and RTREE in MySQL). Sequential scan, which scans through the entire dataset to obtain query result, serves as the base line. We use number of page accesses and query execution time as the metrics for performance measurement.

Experiment Database

We construct the experiment database to store all necessary base values and the financial ratios derived from these base values. The raw data is the COMPUSTAT annual data of 26,649 industrial companies between 1993 and 2012, of a large number of financial items. From the massive data items, we pick 19 useful items, including six firm's identifications (e.g. CUSIP), the date information (data year – fiscal), and twelve commonly used financial variables (e.g. Assets – Total). Table 1 shows the details of the data items in the database.

There are missing values in the raw data. We mark missing values as infinite and rewrite range queries to properly handle this treatment. This is done by assuming that each attribute has a finite maximum value, and for certain range queries we add an upper boundary condition with the maximum values of the query attributes. For example, assume we have a range query on the attribute a_1 to find all records with $a_1 \geq$

1.5. This condition is rewritten as $1.5 \leq a_1 < a_1^{max}$ where a_1^{max} is the maximum value on the attribute a_1 of all data records.

Item	Description
SMBL	Ticker Symbol, used to uniquely identify publicly traded shares of a particular stock.
CUSIP	CUSIP, 9-character alphanumeric code to identify a North American financial security.
GIC	Global Industry Code, segment Global Industry Classification Standard (GICS) codes
NAICS	North American Industry Classification Code
CIK	Central Index Key, given by the United States Securities and Exchange Commission.
AT	Assets – Total, total assets restated up to 10 years.
CAT	Current Assets – Total, represents cash and other assets expected to be realized in cash or used in the production of revenue in the next 12 months.
CEQ	Common Equity – Total, the common shareholders' interest in a company.
CSHO	Common Shares Outstanding, represents the net number of all common shares outstanding at year-end.
CLT	Current Liabilities – Total, represents liabilities due within one year.
LT	Liabilities - Total
DCL	Debt in Current Liabilities
LTDT	Long-Term Debt – Total, Represents debt obligations due more than one year
IB	Income Before Extraordinary Items, the income of a company after all expenses but before provisions for common and/or preferred dividends.
SALES	Net Sales
RDX	Research and Development Expense
PC_F	Price Close - Fiscal Year

Table 1. Financial Variables in COMPUSTAT annual database

We also include a subset of frequently used ratios to construct our test database. The selected ratios are commonly used in research (Bhojraj et al. 2003; Hrazdil et al. 2013) and practice (such as popular stock screeners hosted by Yahoo and Google). We derive the following 11 ratios from the base variables described in Table 1:

1 *Current ratios*, measures whether or not a firm has enough resources to pay its debts over the next 12 months.
2 *Total Assets Turnover*, measures the efficiency of a company's use of its assets in generating sales revenue or sales income to the company.
3 *R&D-to-Sales Ratio*, a useful ratio for comparing the effectiveness of R&D expenditures between companies in the same industry.
4 *Net Profit Margin*, the percentage of selling price that turned into profit.
5 *Return-on-Assets*, indicates how profitable a company's assets are in generating revenue. It is always used for comparing competing companies in the same industry.
6 *Return-on-Equity*, the rate of return on the ownership interest of the common stock owners, or the efficiency at generating profits from every unit of the investments.
7 *Leverage Ratio*, a general term for any technique of multiply gains and losses.
8 *Long-term Debt-to-Assets Ratio*, indicates the proportion of the company's assets that are financed with long-term debt.
9 *Price-to-Book Ratio*, a ratio used to compare a company's current market price to its book value.

10 *Price-to-Earning Ratio*, the faster-growing or less-risky firms tend to have higher P/E ratios than slower growing or more-risky firms.
11 Enterprise-Value-to-Scales, a measurement reflecting the market value of a whole business.

Datasets and Experiment Configuration

Finally, there are 6 nominal attributes and 24 numeric attributes. In practice, we can create indexes on any sets of the attributes for different purposes. In this research, we create two datasets from the original database: a lower-dimensional dataset (including 6 basic financial items), and a higher-dimensional dataset (including 14 financial variables and ratios). Due to missing values, the number of indexed records in the higher dimensional dataset reduced to approximately 76,158.

We implement the PL-tree and R*-tree indexing algorithms in Python on a machine of 1.8GHz Intel Core i5 processor and 8GB 1600 MHz DDR3 memory, running Mac OS system using a SSD drive that has a faster disk I/O speed. We apply LRU for buffer management and set access pages of size 4KB. Since the query time is mainly determined by the time for disk I/O, our method would outperform existing methods even if a slower hard drive is used. Therefore, for each experiment, we compare the performances in term of both the number of pages accessed and the total elapsed time in seconds.

Database Indexes and Queries

Point Queries

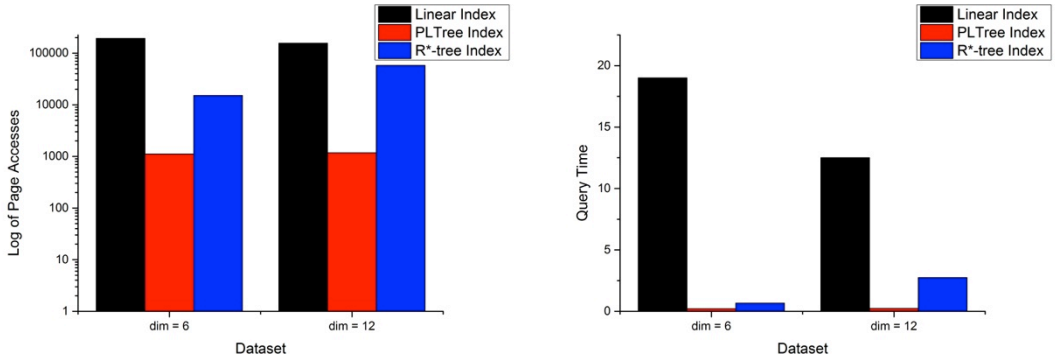

Figure 4: Point Query Performances on Lower- and Higher-Dimensional Datasets

We evaluate point queries by searching 100 randomly selected points. For point queries on a hierarchical tree index, the query cost directly corresponds to the height of the tree. As shown in Figure 4, both PL-Trees and R*-trees outperform the sequential scan regardless of the data dimensionality. PL-Trees have a better performance than R*-trees because the overlap-free property of the PL-Tree guarantees that there is only one path from root to leaf for a single point query. As dimensionality increases, PL-Trees maintain a constant efficiency but R*-trees' performance deteriorates.

Range Queries

We evaluate the performance of range queries with varying *selectivity factors*. A selectivity factor is the ratio of the numbers of records in the query range to the number of all records in the database. We vary the selectivity factor among 0.1%, 1%, 5%, 10%, 20%, and 30%, a reasonably wide range covering most scenarios in real-world applications. For each selectivity factor, we execute 100 random range queries and compare the performance for different selectivity factors among all indexing methods.

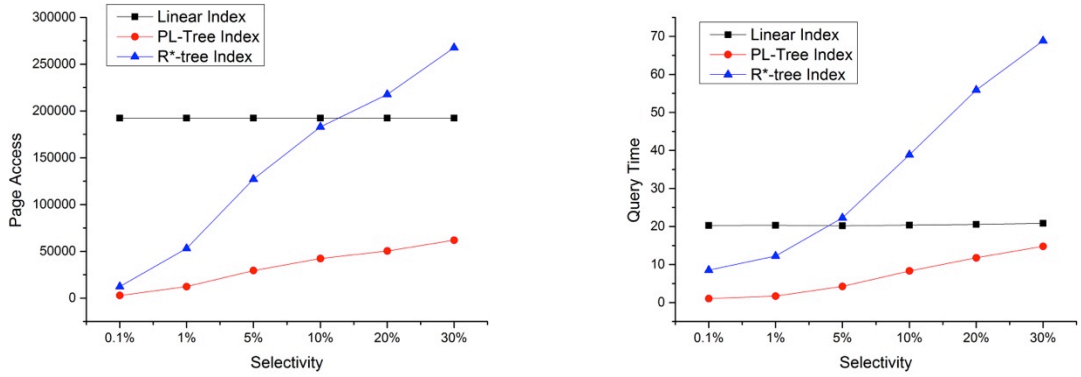

Figure 5: Range Query Performances on Lower-Dimensional Dataset

Figure 5 shows performances of range queries on the lower-dimensional data. For both PL-tree and the R*-tree, the number of page accesses and elapsed time increase with increasing selectivity. The increases are much more slowly for PL-tree. We know that R*-tree indexing produces nodes with many overlapping areas. This weakness worsens with increasing selectivity (and dimensionality, as shown in Figure 6). In contrast, PL-tree does not have this weakness so that it scales well as range increases in range queries.

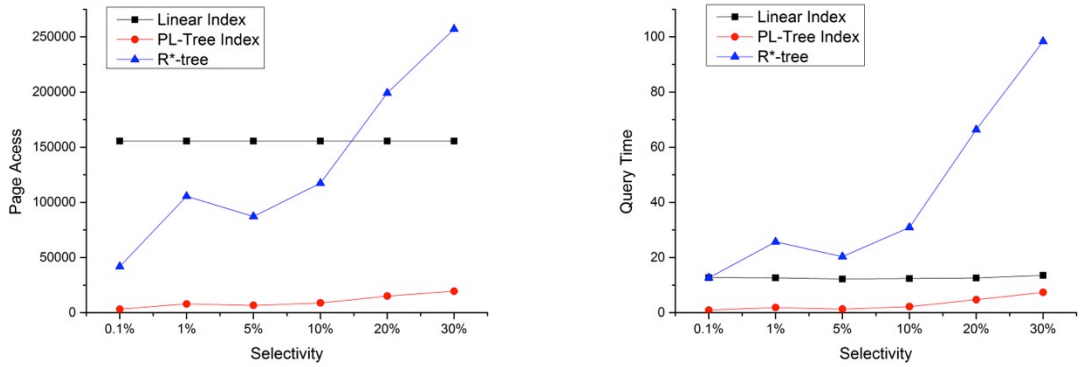

Figure 6: Range Query Performances on Higher-Dimensional Dataset

Figure 6 shows the performances for the higher-dimensional data. Note that the number records in the higher-dimensional dataset is approximately 1/3 of the lower-dimensional dataset. Thus the absolute values for PL-tree and the baseline case are actually lower with the higher-dimensional dataset.

Similarity Queries

For each dataset, we randomly choose 1,000 companies as the query points for the similarity queries and carry out 1,000 similarity queries with different values of *K*. The similarity query (*P*, *K*) will return the first *K* similar points to the query point *P*. In this experiment, we vary the values of *K* in 1, 10, 100, 200, 500, and 1,000.

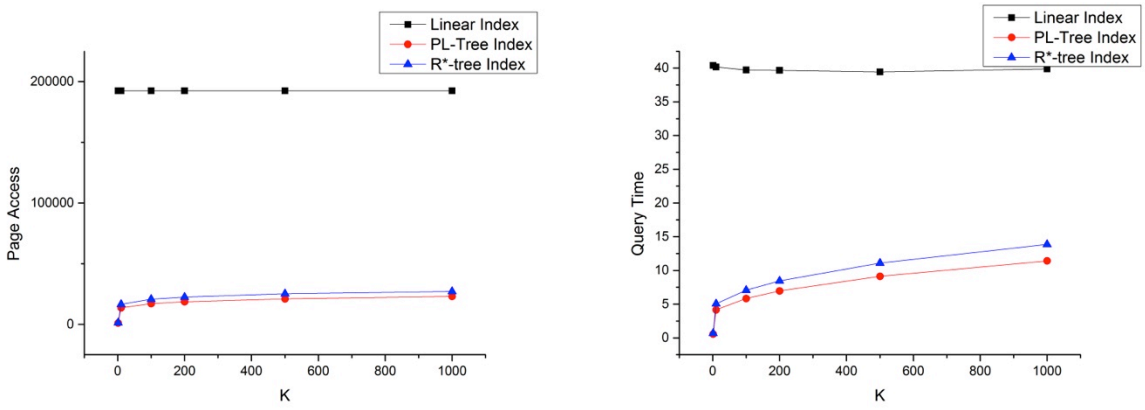

Figure 7: Similarity Query Performances on Lower-Dimensional Dataset

As shown in Figure 7, both PL-tree and R*-tree have very good performance for similarity queries. That is because these two indexes have the following property: the records in the same page are close to each other. Again, our method slightly outperforms the R*-tree due to the overlapping-free property of our method.

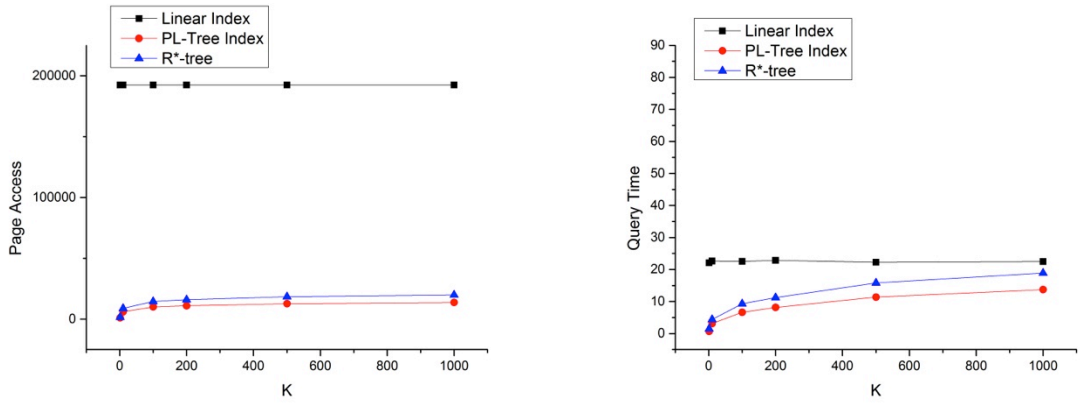

Figure 8: Similarity Query Performances on Higher-Dimensional Dataset

We also carry out similarity queries on the indexes of higher dimensionality. Figure 8 shows that, similar to results of lower dimensionality, tree-like indexes still have much fewer page accesses. However, we also find that tree-like indexes need extra time to compute MINDIST and maintain priority queue. As a result R*-trees have similar performance to the sequential scan for larger *k* values, but PL-Trees still outperform other methods for similarity queries. Our method will outperform even more when the dataset size is larger because the distance calculation in our method only depends on the dimensionality, whereas in the case of R*-tree, it will increase with both dimensionality and data size as the overlap tends to grow with data size.

Conclusion

It is important to address the "curse of dimensionality" for financial analyses since they make extensive use of high-dimensional data extensively. In this paper, we focus on the timeliness challenge in querying high-dimensional data. We have developed a querying system for financial data using the PL-tree indexing method. The PL-Tree ensures that the nodes in the tree have no overlap. This property gives the method excellent performances for querying large volumes of high-dimensional data. We also evaluated the performance of our system on a testbed using real world financial data. The testbed is used to evaluate

the performance of the method for point, range, and similarity queries. The evaluation empirically confirm the superior performance and scalability of the method over the existing R*-tree method.

For future research, we will evaluate the method for other commonly used query operations. The testbed will also be enhanced with more financial values and financial ratios. We will make the testbed available to other researchers who want to perform benchmark testing. We expect that this research will enable the development of financial applications to support the time-critical decision-making.

Acknowledgements

We thank all authors, committee members, and volunteers for their hard work and contributions to the conference.

References

Altman, E. I. 1968. "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy," *The Journal of Finance* (23:4), pp. pp. 589–609.

Atiya, A. F. 2001. "Bankruptcy prediction for credit risk using neural networks: A survey and new results," *Neural Networks, IEEE Transactions on* (12:4), pp. 929–935.

Ballou, D. P., and Pazer, H. L. 1995. "Designing Information Systems to Optimize the Accuracy-Timeliness Tradeoff," *Information Systems Research* (6:1), pp. 51–72.

Barone, D., Stella, F., and Batini, C. 2010. "Dependency Discovery in Data Quality," in *Advanced Information Systems Engineering*, Lecture Notes in Computer Science, B. Pernici (ed.), (Vol. 6051) Springer Berlin Heidelberg, pp. 53–67.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. 1990. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles," in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data*, SIGMOD '90, New York, NY, USA, pp. 322–331.

Bellman, R. E. 1961. *Adaptive Control Processes: A Guided Tour*, 'Rand Corporation. Research studies, Princeton University Press.

Bhojraj, S., Lee, C. M. C., and Oler, D. K. 2003. "What's My Line? A Comparison of Industry Classification Schemes for Capital Market Research," *Journal of Accounting Research* (41:5), pp. 745–774.

Böhm, C., Berchtold, S., and Keim, D. A. 2001. "Searching in High-dimensional Spaces: Index Structures for Improving the Performance of Multimedia Databases," *ACM Comput. Surv.* (33:3), pp. 322–373.

Cao, L., and Zhu, H. 2013. "Normal Accidents: Data Quality Problems in ERP-enabled Manufacturing," *J. Data and Information Quality* (4:3), pp. 11:1–11:26.

Cappiello, C., Francalanci, C., and Pernici, B. 2003. "Time-Related Factors of Data Quality in Multichannel Information Systems," *J. Manage. Inf. Syst.* (20:3), pp. 71–92.

Chen, K. H., and Shimerda, T. A. 1981. "An Empirical Analysis of Useful Financial Ratios," *Financial Management (pre-1986)* (10:1), p. 51.

Gombola, M. J., and Ketz, J. E. 1983. "Financial Ratio Patterns in Retail and Manufacturing Organizations.," *Financial Management (1972)* (12:2), pp. 45–56.

Groppelli, A. A. 2000. *Finance*, Business review books, (4th ed.) Hauppauge, N.Y: Barron's.

Guttman, A. 1984. "R-trees: A Dynamic Index Structure for Spatial Searching," in *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*, SIGMOD '84, New York, NY, USA, pp. 47–57.

Hjaltason, G. R., and Samet, H. 1999. "Distance Browsing in Spatial Databases," *ACM Trans. Database Syst.* (24:2), pp. 265–318.

Ho, C.-T., and Yun-Shan, W. 2006. "Benchmarking performance indicators for banks," *Benchmarking* (13:1), pp. 147–159.

Hrazdil, K., Trottier, K., and Zhang, R. 2013. "A comparison of industry classification schemes: A large sample study," *Economics Letters* (118:1), pp. 77–80.

Lam, M. 2004. "Neural network techniques for financial performance prediction: integrating fundamental and technical analysis," *Data mining for financial decision making* (37:4), pp. 567–581.

Ravi Kumar, P., and Ravi, V. 2007. "Bankruptcy prediction in banks and firms via statistical and intelligent techniques – A review," *European Journal of Operational Research* (180:1), pp. 1–28.

Serrano Cinca, C., Mar Molinero, C., and Gallizo Larraz, J. L. 2005. "Country and size effects in financial ratios: A European perspective," *Global Finance Journal* (16:1), pp. 26–47.

Strong, D. M., Lee, Y. W., and Wang, R. Y. 1997. "Data Quality in Context," *Commun. ACM* (40:5), pp. 103–110.

Strong, D. M., and Volkoff, O. 2005. "Data Quality Issues in Integrated Enterprise Systems," in *IQ*, .

Sun, J., and Hui, X. 2006. "An Application of Decision Tree and Genetic Algorithms for Financial Ratios' Dynamic Selection and Financial Distress Prediction," in *Machine Learning and Cybernetics, 2006 International Conference on*, , August, pp. 2413–2418.

Wang, H., Jiang, Y., and Wang, H. 2009. "Stock return prediction based on Bagging-decision tree," in *Grey Systems and Intelligent Services, 2009. GSIS 2009. IEEE International Conference on*, , November, pp. 1575–1580.

Wang, J., Lu, J., Fang, Z., Ge, T., and Chen, C. 2013. "PL-Tree: An Efficient Indexing Method for High-Dimensional Data," in *Advances in Spatial and Temporal Databases*, Lecture Notes in Computer Science, M. Nascimento, T. Sellis, R. Cheng, J. Sander, Y. Zheng, H.-P. Kriegel, M. Renz, and C. Sengstock (eds.), (Vol. 8098) Springer Berlin Heidelberg, pp. 183–200.

Wang, R. Y., and Strong, D. M. 1996. "Beyond Accuracy: What Data Quality Means to Data Consumers," *J. Manage. Inf. Syst.* (12:4), pp. 5–33.

Wang, Y.-J., and Lee, H.-S. 2008. "A clustering method to identify representative financial ratios," *Information Sciences* (178:4), pp. 1087–1097.

Wu, X., Kumar, V., Ross Quinlan, J., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G., Ng, A., Liu, B., Yu, P., Zhou, Z.-H., Steinbach, M., Hand, D., and Steinberg, D. 2008. "Top 10 algorithms in data mining," *Knowledge and Information Systems* (14:1), pp. 1–37.

Zhu, H., and Wu, H. 2014. "Assessing the quality of large-scale data standards: A case of XBRL GAAP Taxonomy," *Decision Support Systems* (59:0), pp. 351–360.