# EFFICIENTLY PROCESSING OF TOP-K TYPICALITY QUERY FOR STRUCTURED DATA

Jaehui Park[1] and Sang-goo Lee[2]

[1]Electronics and Telecommunications Research Institute, Daejeon, Korea
jaehui@etri.re.kr
[2]School of Computer Science and Engineering, Seoul National University
sglee@snu.ac.kr

***ABSTRACT***

*This work presents a novel ranking scheme for structured data. We show how to apply the notion of typicality analysis from cognitive science and how to use this notion to formulate the problem of ranking data with categorical attributes. First, we formalize the typicality query model for relational databases. We adopt Pearson correlation coefficient to quantify the extent of the typicality of an object. The correlation coefficient estimates the extent of statistical relationships between two variables based on the patterns of occurrences and absences of their values. Second, we develop a top-k query processing method for efficient computation. TPFilter prunes unpromising objects based on tight upper bounds and selectively joins tuples of highest typicality score. Our methods efficiently prune unpromising objects based on upper bounds. Experimental results show our approach is promising for real data.*

***KEYWORDS***

*Typicality, Top-k query processing, Correlation, Lazy join, Upper bound*

## 1. INTRODUCTION

Analyzing typical characteristics of objects is an effective method to understand the semantics of the objects in real-world data sets. Traditional studies in cognitive science [1, 2] have noted that a measure of typicality generally improves people's judgment, whether some objects to be "better examples" for a given concept (or a category). For example, consider a user who wants to learn a concept, mammals, using a zoology data set. Based on typicality analysis, lions may be more useful example than whales because lions have typical attributes of mammals, such as quadruped (four legs). Finding typical instance is a useful application for reflecting semantics of whole data set by only using a limited set of objects. Therefore, lions and bears are better examples than whales and platypuses when we introduce a conceptual knowledge of mammals to children. Following general understandings in cognitive science, we adopt intuitions from typicality analysis to information retrieval tasks, especially, rankings. In this paper, we focus on a ranking model for objects with categorical attributes in a large database using the concept of typicality. Moreover, several processing techniques are proposed to improve the efficiency of retrieval in large scale data sets.

David C. Wyld et al. (Eds) : CST, ITCS, JSE, SIP, ARIA, DMS - 2014
pp. 391–400, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4136

More precisely, we first investigate the problem of applying the notion of typicality analysis into ranking of database query results. Motivated by [3], we propose a novel model, typicality query model, for relational databases. From the definition [3], a typical object shares many attribute values with other objects of the same category, and few attribute values with objects of other categories. Given a query, which determines a specific category, computing common attribute values of objects is crucial for typicality query. In this paper, statistical relationships based on correlation analysis [4, 5] are adopted to specify the amount of the common attribute values for queries. Furthermore, the correlation analysis naturally provides for quantification of common attribute values of objects in not only a set of a single category but also multiple categories. However, constructing comprehensive dependency model for every correlation yields unreasonably high computational costs. Therefore, we develop the typicality query model by introducing limited independence assumption on attribute values for efficient computation. Previous studies [6, 7] have proved that the assumption reduces a significant amount of computations without deteriorating the quality of rankings over structured data.

Secondly, we propose a method to find top-k typical objects efficiently. Despite the significance of the topic that users are more interested in the most important, that is, top-k query results is emphasized recently, little attention has been paid to aggregating scores of an individual object that are dependent (or, correlated) to each other. Previous studies, such as [3], have proposed approximation methods to provide fast answers for top-k typicality query. Despite existing studies have focused on approximation or new measures of association, our model mainly concerns efficient computation for top-k results without approximate solutions. Basically, we perform a prune-and-test method for a large number of objects 1) before aggregating exact scores by investigating an upper bound property of the correlation coefficient, and 2) by predicting unnecessary joins to avoid beforehand. We can check whether candidate objects have a potential to become top-k answers for a typicality query without computing their exact typicality scores. We further save a lot of join query processing cost to predict the typicality score by estimating the cardinality of tuples that directly matched to queries. Our methods significantly reduce unnecessary join processing time. To our knowledge, our work is first approach to compute top-k objects over relational databases on typicality measures, which are based on the correlation of individual objects.

We have conducted and performed performance study on a real data set. Extensive sets of evaluation tests are not provided in this paper because this work is still in progress. As a summary, our method, TPFilter yields average execution time that are much smaller than that of the competitive work [3] on zoology data sets.

The rest of the paper is organized as follows. In Section 2, we define the typicality query in relational databases and the typicality score based on descriptive statistics, namely correlation. Section 3, we introduce the top-k typicality query processing method, TPFilter. In Section 4, we show a brief set of evaluation results. Finally, we present concluding remarks and further study in Section 5.

## 2. QUERY MODEL

In this section, we formally define a typicality query model in relational databases. In Section 2.1, we introduce the notion of the typicality query. In Section 2.2, we develop a probabilistic ranking function based on a statistical model from classical statistics.

## 2.1. Typicality Query

We consider a set of relations $R = \{r_1, r_2, \ldots, r_N\}$ and each relation $r_i$ as a set of $n$ tuples $\{t_{i1}, t_{i2}, \ldots, t_{in}\}$. For simplicity, we use tuple $t_j$ to represent $t_{ij}$ when $r_i$ is clear in the context. Given a keyword query $Q = \{k_1, k_2, \ldots, k_q\}$, we would like to assign a ranking score $S(I)$ for an object $I$ of a certain relational schema $H(R)$ defined on the relations $R$. The relational schema $H(R)$ contains referential relationships between relations. Figure 1(a) illustrate an example relational schema as a directed graph that has *7* vertices, corresponding to relations $R = \{r_1, \ldots, r_7\}$. Directed edges represent the referential relationships between the relations. Colored vertices, $r_4$ and $r_6$, represent relations that contain query keywords $Q = \{k_1, k_2\}$ in their tuples. We restrict our attention in this work to acyclic relational schema, which are common in database contexts. In our query model, the logical unit of the retrieval may be multiple tuples joined together based on primary key-foreign key relationships. In the example above, joining tuples of schema $H(R')$ (Figure 1(b)) represent a set of result given keyword query $Q = \{k_1, k_2\}$. It corresponds to join query expression that produce joining network of tuple set for the keyword query $Q$. We assign the ranking score $S(I)$ to each answer $I$, which is a joining network of tuple set. We define basic requirements for $I$ as follows:

1) Every keyword in query $Q$ is contained in at least one relation $r_i$ in $H(R')$
2) Let $t$ and $t'$ be any two adjacent tuples, and assume that they are in relations $r$ and $r'$, respectively. r and r' must be connected in the relational schema H(R'), and joining tuples, $t \bowtie t'$, must belong to $r \bowtie r'$.
3) No adjacent tuple can be removed if it fulfills the above requirements.


*(a) H(R): relational schema on R* *(b) A retrieval unit of H(R')*

Figure 1. Directed graph of relational schema

From the requirement (2), $H(R')$ may contain the set of relations that do not include any keyword but connects others. We call tuple sets from those relations as *free tuple sets*. On the other hand, the set of tuples that satisfy requirement (1) is denoted as a *non-free tuple set*. Finding optimal answers satisfying above requirements in arbitrary queries is NP-hard problem. The focus of this paper is not on developing algorithms to efficiently compute near-optimal (or approximate) answers of relational schema $H(R')$. Rather, the objective of this paper is to introduce an effective ranking model in relational databases – that of computing a typicality measure $S(I)$ efficiently for top-$k$ objects $I$. We assume that all possible $H(R')$s for the query $Q$ are generated.

Our typicality query model retrieves a list of objects ordered by their typicality scores. Now the typicality query is defined as follows:

***Definition 1.*** (Typicality query) Given a keyword query $Q = \{k_1, k_2, \ldots, k_q\}$ and a database $R = \{r_1, r_2, \ldots, r_N\}$ with a schema $H(R)$, a *typicality query* is defined as following form.

SELECT *
FROM $\forall r^K = \{r | \exists k_i \in t \wedge t \in r\}$ JOIN $\exists r^F = \{r | r^K \leftarrow r \rightarrow r^K\} \in H(R')$
WHERE $r^F.\mathtt{a} - k_1$ AND ... AND $r^K.\mathtt{a} - k_q$
ORDER BY $S(I \text{ of } H(R'))$

where the arrows denote the primary key-foreign key relationship, and $I$ is an object of a relational schema $H(R')$, which produce the joining network of tuples in $r^K$ and $r^F$. $r^K$ corresponds non-free tuple sets, and $r^K$ corresponds to free tuple sets. We call the score $S(I)$ as *typicality score* of an object $I$.

***Proposition 1.*** (Typical instance) Given objects $I$ enumerated from all possible relational schema $H(R')$ over $H(R)$, $Q = \{k_1, k_2, \ldots, k_q\}$ and user specified threshold $t$, an instance whose score $S(I)$ is over the threshold $t$ ($S(I) > t$) is denoted as a *typical instance*.

In a straightforward way, typicality query model process all the joins in every $H(R')$ for given queries, compute typicality score $S$, and then selects the most typical objects according to user specified threshold. With large databases, the total cost of query processing may be prohibitive. The computation method will be presented in Section 3.

## 2.2. Typicality Score

Assuming a keyword query $Q = \{k_1, k_2, \ldots, k_q\}$ and relational schema $H(R')$ are given, we note that typicality query selects all objects $I = \{I_1, I_2, \ldots, I_{|I|}\}$ having identical attributes $A = \{a_1, a_2, \ldots, a_m\}$. We aim to assign a *typicality score* for each object $I_i$ to order them by its occurrence distribution in database $D$; it follows the general notion of typicality measure used in cognitive science. Based on the perception in [3], a typical object shares many attribute values with other objects of the same category, and few attribute values with objects of other categories. Intuitively, we can estimate the typicality score by counting common attribute values of objects given queries. Figure 2 illustrates a simple data set to compute typicality scores for eight objects, and objects $I_1$~$I_4$ are in same category.

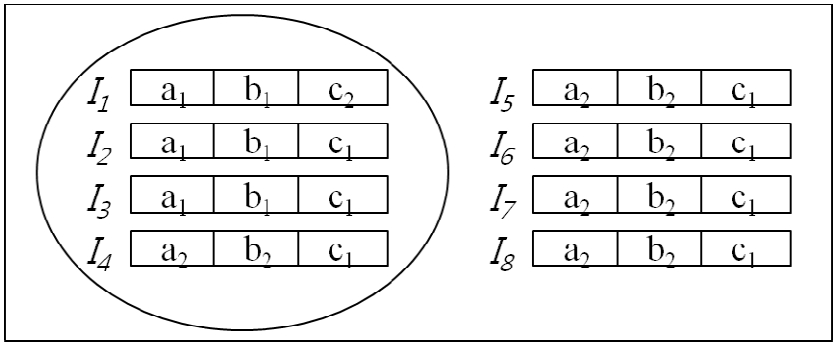


Figure 2. A single category selects four objects over a set of eight objects

Assuming the category is identified by given query $Q$, we can estimate each typicality score as the ratio of the number of common attribute values within given category to the number of attribute values shared with objects of other categories.

$$S(I_1) = \frac{2 \times 3}{1} = 6$$

$$S(I_2) = S(I_3) = \frac{3 \times 3}{1 \times 4} = 2.25$$

$$S(I_4) = \frac{1}{2 \times 4} = 0.125$$

Above scores are calculated by naively counting the number of occurrences to quantify the typicality of an object. The object $I_1$ is most typical because it shares two attribute values with the objects in the same category, but also no attribute is shared with objects in other categories. On the other hand, the objects $I_2$ and $I_3$ share an attribute value $c_1$ with the objects in other categories. This is a simplified notion of typicality score. To define typicality score in a principled way, mutual implications on the occurrences or absences of attribute values $I.a_j$ with $Q$ should be derived effectively. We note that the intuition is closely linked to the notion of correlation from classical descriptive statistics; correlation has been recognized as an interesting and useful type of patterns due to its ability to reveal the underlying occurrence dependency between data objects [9].

Any existing statistical measures [10] can be used to represent the extent of relationship (dependency) between elements. In this paper, we adopt *Pearson correlation coefficient* to model the interpretation from previous paragraph; but we remark that other measurements [10] can also be applied in a similar way. In our model, a binary random variable represents the absence and the presence of an attribute value given a query. In this context, the *Pearson correlation coefficient* for two random variables *X and Y* (as $E(XY) - E(X)(Y)/(\sqrt{E(X^2) - (E(X)^2)}\sqrt{E(Y^2) - (E(Y)^2)})$ is reduced to computational form as follows. We omit the proof due to limited space.

Given two binary random variables *X* and *Y*, the *Pearson correlation coefficient* $\rho$ is:

$$\rho(X,Y) = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 0}n_{\cdot 1}}} \quad (1)$$

where $n_{XY}$, (for $X = 0, 1$ and $Y = 0, 1$), is the number of attribute value observations in a set of $n$ objects, which are specified in Table 1.

Table 1. A two-way table of binary random variables X and Y

| | Y=1 | Y=0 | Total |
|---|---|---|---|
| X=1 | $n_{11}$ | $n_{10}$ | $n_{1\cdot}$ |
| X=0 | $n_{01}$ | $n_{00}$ | $n_{0\cdot}$ |
| Total | $n_{\cdot 1}$ | $n_{\cdot 0}$ | n |

Two binary random variables are considered positively associated if most of the observations fall along the right diagonal cells. In contrast, negative implication between variables is determined based on values in the left cells. Based on the correlation $\rho$, we can estimate the mutual implications of the occurrences of attribute values given a keyword query *Q*. We can specify the implication for each given query keyword $\rho(X, Y|k \in Q)$ as an aggregated score for an object *I*.

***Definition 2.*** (Typicality score) Given a keyword query $Q = \{k_1, k_2, \ldots, k_q\}$ and an object $I$ with attributes $A = \{a_1, a_2, \ldots, a_m\}$, a *typicality score S* of an object $I$ is defined as following equation

$$S(I) = \sum_{j=1}^{q} \sum_{a_x, a_y \in A} \rho(I.a_x, I.a_y | k_j) \quad (2)$$

where $I.a_x$ and $I.a_y$ denote a pair of arbitrary attribute values of the object $I$. In order to estimate the typicality score of an object $I$, we aggregate every correlation $\rho$ between pairs of attribute values $I.a_x$ and $I.a_y$ given query $Q$.

However, computing all combinations of attribute values is very expensive due to the complexity of relational databases with many attributes. In practice, it is necessary to define a practical assumption to avoid computing the correlation coefficients for an exponential number of attribute value combinations. We propose a limited independent assumption as in binary independence model as follows:

***Definition 3.*** (Limited Independence Assumption) Given a keyword query $Q = \{k_1, k_2, \ldots, k_q\}$ and an object $I$ with attributes $A = \{a_1, a_2, \ldots, a_m\}$, we assume dependence only between two specified sets of attribute values ($I.A_q$ and $I.A_{nq}$). The two sets of attributes are defined as follows: $A_q = \{a | I.a \cap Q \neq \emptyset\}$ and $A_{nq} = A\text{-}A_q$. The attribute values $I.a_i$ ($a_i \in A_q$) are assumed to be mutually independent. Analogously, the attribute values $I.a_j$ ($a_j \in A_{nq}$) are assumed to be mutually independent. We allow dependencies between $I.a_i$ ($a_i \in A_q$) and $I.a_j$ ($a_j \in A_{nq}$). Therefore, $\rho(I.a_i, I.a_j | k_l)$ is considered for typicality score.

Like most successful retrieval model (e.g., TF-IDF and BM25), our assumption between elementary values has empirically shown to be practical. Although our model defines the limited dependencies among values for our purpose, this assumption is patently significant for ranking relational data [6]. From our previous work [7], the assumption is validated to improve the retrieval performance. The assumption reduces the expression (Equation 2) to a following function, which is a simplified form:

$$S(I) = \sum_{j}^{q} \sum_{a_x \in A_q, a_y \in A_{nq}} \rho(I.a_x, I.a_y | k_j) \quad (3)$$

## 3. TOP-K PROCESSING OF TYPICALITY QUERY

In this section, we introduce a pruning method to efficiently remove the unpromising candidate objects before computing the actual typicality scores. By analyzing the mathematical properties of the correlation coefficients, we can derive upper bounds of typicality scores to test false positive candidates. Also, to compute top-k scores of objects, we don't need to join all the candidate tuples, but aggregate only the correlation values to calculate typicality scores. In this area, a number of top-k query processing techniques have already been proposed. However, top-k typicality query processing has crucial difference from the previous studies. Although most previous studies have focused on the ranking scores of individual objects with sorted access, our typicality score is quantified by its relationship with other objects. Therefore, classical algorithm cannot be adopted in a straightforward way. Moreover, our method is represented in a feasible form as compared to computational approaches in cognitive science.

In Section 3.1, we introduce a candidate pruning method, TPFilter, to efficiently prune the unpromising objects before computing the typicality scores. By analyzing the mathematical properties of the correlation coefficients, we derive upper bounds of typicality scores to test false positive candidate objects. In Section 3.2, we propose an efficient join query processing method, Lazy Join, to reduce the cost of join operations on multiple relations. To compute top-k scores of

objects, we don't need to join all the candidate tuples, but aggregate only the correlation values to calculate typicality.

### 3.1. TPFilter

Let $Pr(I_i.a_j)$ denote the ratio of the cardinality of the attribute value $I_i.a_j$ to the size of the database subset $I$ of ($H(R')$), which has same schema with object $I_i$. From section 2.2, we can transform Equation 1 by adopting observable variables $\Pr(I_i.a_j)$ to Equation 3 if we specify the binary random variable X as $I_i.a_j$ (also, Y by $I_i.a_k$). For simple presentation, we use $X$ and $Y$ to represent $I_i.a_j$ and $I_i.a_k$, respectively and $A_q = \{I_i.a_j\}$. This is not an unusual constraint since we assume that keywords in $Q$ are independent to each other.

$$\rho(X,Y) = \frac{n_{11}n_{00} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 0}n_{\cdot 1}}} = \frac{(n - n_{10} - n_{01} - n_{11})n_{11} - n_{10}n_{01}}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 0}n_{\cdot 1}}} = \frac{nn_{11} - (n_{10} + n_{11})(n_{01} + n_{11})}{\sqrt{n_{1\cdot}n_{0\cdot}n_{\cdot 0}n_{\cdot 1}}}$$
$$= \frac{\frac{n_{11}}{n} - \frac{n_{1\cdot}}{n}\frac{n_{\cdot 1}}{n}}{\sqrt{\frac{n_{1\cdot}}{n}\frac{n_{0\cdot}}{n}\frac{n_{\cdot 0}}{n}\frac{n_{\cdot 1}}{n}}} = \frac{Pr(X,Y) \quad Pr(X)\,Pr(Y)}{\sqrt{Pr(X)\,Pr(Y)\,(1 - Pr(X))(1 - Pr(Y))}} \tag{3}$$

We propose an upper bound $\bar{\rho}(X,Y)$ for the bivariate correlation coefficient as a filter of unpromising objects.

***Definition 4.*** (Typicality score upper bound $\bar{S}(I)$) Given an object $I$ ($I = \{I_i\}$) with attributes $A = \{a_1, a_2, \ldots, a_m\}$, let $A_q = \{a_1, \ldots a_i\}$ for a keyword query $Q$. The upper bound of typicality score of object $I$ is defined as follows:

$$\bar{S}(I) = \sqrt{\frac{1 - Pr(I.A_q)}{Pr(I.A_q)}} \sum_{a_y \in A - A_q} \sqrt{\frac{Pr(I.a_y)}{1 - Pr(I.a_y)}} \tag{4}$$

**Proof sketch.** Without loss of generality, we assume $Pr(X) \leq Pr(Y)$. Then, following inequalities are to be true.

$$\Pr(X,Y) \leq P(X) \leq P(Y). \tag{5}$$

$$\rho(X,Y) = \frac{\Pr(X,Y) - Pr(X)\Pr(Y)}{\sqrt{Pr(X)\,Pr(Y)\,(1 - Pr(X))(1 - Pr(Y))}}$$
$$\leq \frac{\Pr(X) - Pr(X)\Pr(Y)}{\sqrt{Pr(X)\,Pr(Y)\,(1 - Pr(X))(1 - Pr(Y))}} \tag{6}$$
$$\leq \sqrt{\frac{Pr(Y)}{Pr(X)}}\sqrt{\frac{1 \quad Pr(X)}{1 - Pr(Y)}} = \bar{\rho}(X,Y)$$

Therefore, for all attribute values Y = I.$a_i$ ($\in$A), we can aggregate each correlation upper bounds $\bar{\rho}(X,Y)$Y). Then we can derive typicality score upper bound $\bar{S}(I)$ (Equation 4).

Basically, to calculate a typicality score of an object $I_i$, we have to compute the joint distribution $Pr(X, Y)$ of all attribute values in $I_i$. Computing all these pairs of attribute values in $R'$ is too costly for online queries on large databases. The typicality score upper bound is determined only

by the observable variables $Pr(X)$ and $Pr(Y)$ ($Y \in A_{nq}$). We note that calculating the upper bound is much cheaper than the computation of the exact typicality score, since the upper bound can be easily computed as a function of cardinality of the joining tuples without considering the joint distributions, e.g., $Pr(X,Y)$. Storing every pairs of attribute values is inefficient for online processing. Note that the $\bar{S}(I)$ has monotone property, which is useful to filter lower scores at early stage. If both $Pr(X)$ and $Pr(X,Y)$ are fixed, then the correlation value of $X$ and $Y$ is monotonically decreasing with $Pr(Y)$. Therefore, we can maintain a queue of current top-k typical objects discovered so far, which is denoted as $C$. The objects in $C$ are sorted in the descending order of their typicality scores. The typicality score of the k-th object in $C$ is also denoted as *typicality_min*. For each newly candidate object $I$ to be evaluated, its typicality score S($I$) should be at least *typicality_min*; otherwise, the object $I$ is immediately removed from the set of candidates.

### 3.2. Lazy Join

Typicality query model must view all relations in a holistic manner in order to aggregate the tuples joined for a keyword query. While a complete evaluation of all the joins for queries is necessary for conventional selection query, we are interested in only top-k results. We propose an algorithm *Lazyjoin* that perform joins without producing all the objects for relational schema $H(R')$.

We start by describing baseline method *Baseline* for top-k typicality query. *Baseline* issues a SQL expression equivalent to CN to retrieve result objects. Then, the objects from each CN are computed to derive typicality scores. We get the top-k typical objects with highest typicality scores. Candidate network generation algorithm reviewed in Section 2 cannot avoid unnecessary CN generation without evaluation on a large set of realtions.

*LazyJoin* computes a bound $\bar{S}(I)$before join operations are performed. If $\bar{S}(I)$ quarantees that the instance $I$ does not exceed the typicality scores already processed k-th instance, the instance I safely removed from further consideration. To derive $\bar{S}(I)$ without joins, we have to consider a hypothetical score of each tuple to be aggregated as $\bar{S}(I)$. Similarly, we can calculate a typicality score of each tuple set. However, joining tuples make redundant tuples. Typicality scores are multiplied by the number of tuple connections, that is, primary key-foreign key relationship. We estimate the number of connections to predict final typicality scores for joining network of tuples. Let $TS(t)$ denote a partial score of a participating tuple $t \in r^K$ in $H(r) \in H(R')$. We calculate $TS(t)$ by counting the number of join tuples determined by t. This can be easily retrieved by a single scan of database.

## 4. EXPERIMENTAL EVALUATION

In our experimental study, we use a zoology database from the UCI Machine Learning Database Repository. All tuples are classified into 7 categories (*mammals, birds, reptiles, fish, amphibians, insects and invertebrates*). All the experiments are conducted on a PC with MySQL Server 5.0 RDBMS, AMD Athlon 64 processor 3.2 GHz PC, and 2GB main memory. Our methods are implemented in JAVA, connected to the RDBMS through JDBC. Due to a lack of space, the algorithm codes of the database probing modules and the index construction are not provided in this paper. We proactively identify all of the correlations between attribute values using an SQL query interface. The interface computes all pair-wise correlation by single table scan and stores the results in the auxiliary tables.

We have computed the typicality scores to evaluate the correspondence of our typicality model for real-world semantics. Several measures in cognitive science are adapted to test the effectiveness of *categorization* and *specification*. However, the extensive set of the evaluation study on the quality of our model is incomplete and is still in progress. While computational studies in cognitive science rely on manual surveys, we perform the quality evaluation based on the classical measures in information science, e.g., precision and recall. The average precision is up to 0.715, which is a competitive result compared to [3]. As we consider every relation is identified at static time, the comparative study with [3] is feasible. To evaluate the performance of our top-k computation method, we measure the execution time of top-k results with various query sets ($Q_1$ ~ $Q_{10}$, fixed k=3) and various the parameter $k$ (1~6, fixed query $Q_4$). The parameter, typicality_min $t$ is determined as 0.4.Query sets are constructed by randomly selected keywords from the data sets. Our method greatly improves the *Baseline* (in Section 3) in query execution time, and reasonably yields better performance in time compared to the previous work [3]. From the above results, we find that our basic premise, that the prune-and-test method is very efficient for top-k retrieval. It is premature to conclude that our query model is effective for every context in structured data because this work is still in early stage. In the evaluation, we would like to introduce the potential impact of the topic, typicality analysis for ranking data.

Table 2. Query execution time (varying query sets) in msec

| | **Baseline** | **Hua et al. [3]** | **TPFilter** |
|---|---|---|---|
| **$Q_1$** | 1790 | 205 | 102 |
| **$Q_2$** | 2990 | 340 | 190 |
| **$Q_3$** | 5010 | 401 | 310 |
| **$Q_4$** | 8506 | 489 | 353 |
| **$Q_5$** | 10809 | 550 | 450 |
| **$Q_6$** | 17609 | 610 | 531 |
| **$Q_7$** | 21002 | 721 | 608 |
| **$Q_8$** | 30002 | 795 | 689 |
| **$Q_9$** | 59725 | 860 | 765 |
| **$Q_{10}$** | 96094 | 903 | 833 |

Table 3. Query execution time (varying $k$) in msec

| k | **Baseline** | **Hua et al. [3]** | **TPFilter** |
|---|---|---|---|
| 1 | 1702 | 1259 | 690 |
| 2 | 4420 | 1542 | 830 |
| 3 | 7520 | 1605 | 999 |
| 4 | 10290 | 1701 | 1480 |
| 5 | 28892 | 5020 | 3012 |
| 6 | 44205 | 10450 | 7895 |

## 5. CONCLUSIONS

In this paper, we introduced a novel ranking measure, *typicality*, based on the notions from cognitive science. We proposed the typicality query model and the typicality score based on the correlation measure, *Pearson correlation coefficient*. Then, we propose an efficient computation method, *TPFilter*, that efficiently prunes unpromising objects based on a tight upper bound, and avoid unnecessary joins. Experimental results show that our method works successfully for the real data set. Although the detail discussions of several parts are omitted, this paper proposed a promising tool for ranking structured data.

Further study is required to develop different types of typicality analysis in various applications. We would like to explore the potential of typicality analysis in data mining, data warehousing and other emerging application domains. For example, for social networks, it would be required to identify typical users in the network, which will represent certain communities or groups. Also, ranking user nodes and user groups considering typicality would be an interesting topic in social network analysis.

## ACKNOWLEDGEMENTS

This research was funded by the MSIP(Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013.

## REFERENCES

[1] Rein, J., Goldwater, M., Markman, A.: What is typical about the typicality effect in category-based induction?. Memory & Cognition, Vol. 38 (3), pp. 377--388. (2010).
[2] Yager, R.: A note on a fuzzy measure of typicality. International Journal of Intelligent Systems, Vol. 12 (3) pp. 233--249. (1997).
[3] Hua, M., Pei, J., Fu, A., Lin, X., Leung, H.: Efficiently answering top-k typicality queries on large databases. In: VLDB, pp. 890--901. (2007).
[4] Ilyas, I., Markl, V., Haas, P., Brown, P., Aboulnaga, A.: CORDS: automatic discovery of correlations and soft functional dependencies. In: SIGMOD, pp. 647--658. (2004).
[5] Xiong, H., Shekhar, S., Tan, P., Kumar, V.: TAPER: a two-Step approach for all-strong-pairs correlation query in large databases. TKDE VOl. 18(4), pp. 493--508. (2006).
[6] Chaudhuri, S., Das, G., Hristidis, V., Gerhard, W.: Probabilistic ranking of database query results. In VLDB, pp. 888--899. (2004).
[7] Park, J., Lee, S.: Probabilistic ranking for relational databases based on correlations. In PIKM, pp. 79--82. (2010).
[8] Hristidis, V. and Papakonstantinou, Y. 2002. DISCOVER: keyword search in relational databases. In VLDB, pp. 670-681. (2002).
[9] Ke, Y., Cheng, J., Yu, J.: Top-k Correlative Graph Mining. In SDM, pp 493--508 (2009).
[10] Tan, P, Kumar, V., Sririvastava, J.: Selecting the right interestingness measure for association patterns. In: SIGKDD, pp. 32--41, (2002)..

## AUTHORS

Jaehui Park received his Ph.D. degree in Department of Computer Science and Engineering from Seoul National University, Korea, in 2012 and his B.S. degree in Computer Science from KAIST, Korea, in 2005. Currently, he is a research engineer of Electronics and Telecommunications Research Institute, Korea. His research interests include keyword search in relational databases, information retrieval, semantic technology, and e-Business technologies.