Efficiently Answering Top-k Typicality Queries on Large Databases *

Ming Hua† Jian Pei† Ada W. C. Fu‡ Xuemin Lin¶ Ho-Fung Leung‡
† Simon Fraser University, Canada ‡ The Chinese University of Hong Kong, China
¶ The University of New South Wales & NICTA, Australia
{mhua, jpei}@cs.sfu.ca, {adafu, lhf}@cse.cuhk.edu.hk, lxue@cse.unsw.edu.au

ABSTRACT

Finding typical instances is an effective approach to understand and analyze large data sets. In this paper, we apply the idea of typicality analysis from psychology and cognition science to database query answering, and study the novel problem of answering top-k typicality queries. We model typicality in large data sets systematically. To answer questions like "*Who are the top-k most typical NBA players?*", the measure of *simple typicality* is developed. To answer questions like "*Who are the top-k most typical guards distinguishing guards from other players?*", the notion of *discriminative typicality* is proposed.

Computing the exact answer to a top-k typicality query requires quadratic time which is often too costly for online query answering on large databases. We develop a series of approximation methods for various situations. (1) The randomized tournament algorithm has linear complexity though it does not provide a theoretical guarantee on the quality of the answers. (2) The direct local typicality approximation using VP-trees provides an approximation quality guarantee. (3) A VP-tree can be exploited to index a large set of objects. Then, typicality queries can be answered efficiently with quality guarantees by a tournament method based on a Local Typicality Tree data structure. An extensive performance study using two real data sets and a series of synthetic data sets clearly show that top-k typicality queries are meaningful and our methods are practical.

*The research of Ming Hua and Jian Pei is supported in part by an NSERC Discovery Grant. The research of Ada Wai-Chee Fu is supported in part by the RGC Earmarked Research Grant of HKSAR CUHK 4120/05E. The research of Xuemin Lin is supported in part by the Australian Research Council Discovery Grant DP0666428 and the UNSW Faculty Research Grant Program. All opinions, findings, conclusions and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.
We thank the anonymous reviewers for their constructive comments, particularly, for pointing out some important related work.

VLDB '07, September 23-28, 2007, Vienna, Austria.

1. INTRODUCTION

Learning from examples is an effective strategy extensively adopted in practice. For instance, what should a teacher do to teach a kid the concept of mammal? A helpful approach is to show her/him some *typical* examples of mammals, such as leopard and lions as indicated in Section 6.1.1. Good examples are often more effective than feature descriptions as the first step to understand a concept or a large set of objects.

How should good examples be chosen? First of all, those examples must be typical. Using platypuses (which lay eggs instead of giving birth to live young) as examples of mammals may mislead kids in learning the concept. In addition, real examples are strongly preferred. Some virtual objects made up by assembling perfect features of the concept may not be easy to understand. More often than not, an ideal case may not be possible. For example, there exists no a single bird having all the expected features of birds.

Typicality analysis has been studied extensively in psychology and cognition science (see Section 3 for a brief review). Interestingly, the strategy of using typical examples to summarize and analyze a large set of objects can be applied to effective query answering. When a user asks a query which may return a large number of answers, instead of examining those answers one by one, the user may want to obtain a small number of typical answers for digesting.

1.1 Motivating Examples

Jeff is a junior basketball player. His dream is to play in NBA. As NBA has more than 400 active players, they are quite diverse. Jeff may want to know some representative examples of NBA players. Then, top-k typicality queries can help.

1.1.1 Simple Top-k Typicality Queries

Jeff asks, "*Who are the top-3 most typical NBA players?*"

We can imagine that, in the space of technical statistics, there exists a density function of the probability of being NBA players. The player who has the largest probability density is the most typical. This leads to our first typicality measure – the *simple typicality*. A top-k typicality query finds the k most typical objects in a set of objects.

1.1.2 Top-k Discriminative Typicality Queries

Jeff is particularly interested in becoming a guard. "*Who are the top-2 most typical guards distinguishing guards from other players?*"

Simple typicality on the set of guards is insufficient to

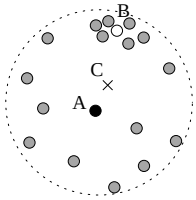


Figure 1: Medians, means and typical objects.

answer the question, since it is possible that a typical guard may also be typical among other players. Instead, players that are typical among all guards but are not typical among all non-guard players should be found.

In order to address this demand, we need the notion of *discriminative typicality*, which measures how an object is typical in one set but not typical in the another set. Given a set of objects and a target subset, a top-k discriminative typicality query finds the k objects with the highest discriminative typicality values.

1.1.3 Medians, Means and Typical Objects

In many previous studies, medians and means are often used to represent the aggregate of a set of objects. The mean of a set of points is the geometric center of the set, while the median is the point in the set that is closest to the mean. Can medians and means approximate typical objects well?

Consider a set of points in Figure 1. Points A and C are the median and the mean of the set, respectively, and point B (the white point) is the most typical point. Clearly, the most typical point B is quite different from the median and the mean in this example.

As the geometric centers, medians and means are not necessarily related to probability distribution which is captured by the typical objects.

In Section 6 (Table 4), we will use a real data set to further elaborate the differences between medians, means, and typical objects, and their different roles in data analysis.

1.1.4 Efficient Query Answering

Top-k typicality queries can be used to provide good examples for effective learning and understanding. The above scenarios may happen in many other applications, such as students learning concepts, medical analysts investigating medical cases, and city administration analyzing traffic accidents and crimes. In our empirical study (Section 6), we will illustrate the effectiveness of typicality queries using two real data sets.

As more and more data are accumulated, top-k typicality queries may be applied on large data sets. It is practically desirable that the queries can be answered online. However, However, we will show that, with a proper notion of top-k typicality, computing the exact answer to a top-k typicality query needs quadratic time which is often too costly for online query answering on large data sets. On the other hand, typical enough examples are often sufficient for understanding and analyzing large data sets. Thus, there are practical demands to have approximation methods to answer top-k typicality queries.

1.2 Our Contributions

In this paper, we formulate and tackle the problem of efficiently answering top-k typicality queries, and make the following contributions.

First, we extend the idea of typicality analysis from psychology and cognition science to database query answering and search, and identify the novel problem of top-k typicality query answering on large databases. We develop two effective measures for typicality.

Second, as computing exact answers to top-k typicality queries on large databases can be costly, we develop a series of approximation query answering algorithms for various situations. (1) The randomized tournament algorithm has linear complexity though it does not provide a theoretical guarantee on the quality of the answers. (2) The direct local typicality approximation using VP-trees provides an approximation quality guarantee. (3) A VP-tree can be exploited to index a large set of objects. Then, typicality queries can be answered efficiently with quality guarantees by an LT-tree tournament method.

Last, we conduct an extensive performance study on both real data sets and synthetic data sets to verify the effectiveness of top-k typicality query answering and the efficiency of our methods.

1.2.1 How Is This Study Different from Others?

To the best of our knowledge, this paper is the first study to systematically use typicality in ranking query answers from large databases. We also address the efficient query answering issue.

Top-k queries (also as known as ranking queries) have been heavily employed in many applications, such as searching web databases, similarity search, recommendation systems, etc. According to a user-specified preference function, a top-k query returns the best k results. Many previous studies investigated efficient approaches to answer top-k queries using aggregate scoring functions. Such an aggregate function maps an object to a certain preference score. The score depends on only the attribute values of the object, and is often independent to other objects in the database.

Top-k typicality queries are a special type of top-k queries, however, the typicality of an object depends on the distribution of the other objects in question. This poses a major challenge for efficient top-k typicality query answering. To the best of our knowledge, how to answer top-k queries efficiently using such "relative" score functions has not been studied well.

The rest of the paper is organized as follows. We propose the typicality measurements in Section 2. We systematically review the related work and point out the differences between this paper and the state-of-the-art studies in Section 3. We develop a randomized tournament query answering algorithm in Section 4, and the local typicality approximation methods in Section 5. We report a systematic performance study in Section 6. The paper is concluded in Section 7.

2. TOP-K TYPICALITY QUERIES

In this section, we define the two types of top-k typicality queries that will be addressed in this paper.

2.1 Simple Typicality

In a set of objects S, which object is the most typical? To answer this query, we introduce simple typicality.

By intuition and as also suggested by the previous research

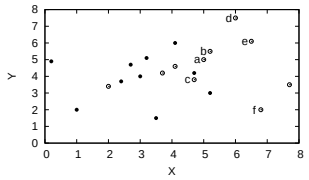


Figure 2: A set of points.

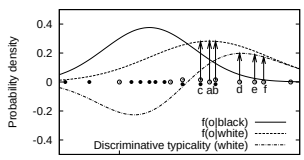


Figure 3: Typicality.

in psychology and cognition science (as will be reviewed in section 3), an object o in S is more typical than the others if o is more likely to appear in S. The probability density of membership of an object provides a measure of the confidence that an object may appear compared to other objects in the same data set. Thus, the probability density of o in S can be used to measure the typicality of o.

Definition 1 (Simple typicality). Given a set of objects S, we treat S as an independently and identically distributed sample of a continuous random variable $\mathcal{S}$. The **simple typicality** of an object $o \in S$ is defined as $T(o, S) = f(o)$ where f is the probability density function of $\mathcal{S}$. ∎

We will discuss how to estimate the probability density function in Section 4.1. In practice, typicality relies on a set of attributes as captured by the typicality queries as follows.

A **top-k simple typicality query** is in the following form.

```
SELECT    A_1, ..., A_n   FROM S   WHERE P
ORDER BY  SIMPLE TYPICALITY ON (A_{i_1}, ..., A_{i_l})
LIMIT     k
```

where $A_1, \ldots, A_n$ are attributes in table S, P is a predicate on tuples from S, $1 \leq i_j \leq n$ $(1 \leq j \leq l)$, and k is a positive integer. From the subset of tuples in S satisfying predicate P, the query returns the top k tuples having the largest simple typicality values that are computed on attributes $A_{i_1}, \ldots, A_{i_l}$.

Example 1 (Top-k simple typicality queries). Consider the following query on the set of points in Figure 2.

```
SELECT    X, Y   FROM S
WHERE     COLOR = white
ORDER BY  SIMPLE TYPICALITY ON (X)
LIMIT     3
```

The query asks for the top-3 typical points in the set of white points, where the typicality should be computed using attribute X only.

Figure 3 projects the points in S to attribute X. The probability density function of the white points and that of the black points are also plotted, while we will discuss how to compute the probability density functions in Section 4.1. Points a, b and c have the highest probability density values among all white points, and thus should be returned as the answer to the query. ∎

2.2 Discriminative Typicality

Simple typicality works when a set of objects are considered independently from other objects not in the set. In some scenarios, a user may have multiple classes of objects, such as the guards and all the other NBA players as illustrated in Section 1. To understand the discriminative features of one class against the others, one may want to find the objects that are typical in the target class, but not in the others.

Given a set of objects S and a target subset C, which object is the most typical in C but not in $(S - C)$? We use the discriminative typicality to answer such a question.

By intuition, for an object o, the difference between the probability density that o is a member of C and the probability density that o is a member of $(S - C)$ measures the discriminative typicality.

Definition 2 (Discriminative typicality). Given a set of objects S and a **target subset** $C \subset S$, the **discriminative typicality** of an object $o \in C$ is $DT(o, C, S) = [f(C|o) - f((S - C)|o)] \times f(o)$, where $f(C|o)$, $f((S - C)|o)$ and $f(o)$ are the conditional probability densities. ∎

In the definition, $f(C|o)$ is the posterior probability density that o is a member of target subset C. $f(C|o) - f((S - C)|o)$ is between -1 and 1. If $f(C|o) - f((S - C)|o) > 0$, then the density of object o in subset C is larger than the density of o in subset $(S - C)$. Moreover, a larger value of $f(C|o) - f((S - C)|o)$ indicates the better discriminability of o in distinguishing subset C from the other objects in S.

The rationale of the definition of discriminative typicality can be justified using the Bayesian theory.

Theorem 1 (Discriminative typicality).

$$DT(o, C, S) = f(o|C)f(C) - f(o|(S - C))f(S - C)$$

Proof sketch. Using Bayesian theorem, we have $f(C|o) = \frac{f(C)f(o|C)}{f(o)}$, and $f((S - C)|o) = \frac{f((S-C))f(o|(S-C))}{f(o)}$. The theorem follows with Definition 2 and the two equations. ∎

A **top-k discriminative typicality query** is in the form as follows.

```
SELECT    A_1, ..., A_n   FROM S   WHERE P
ORDER BY  DISC TYPICALITY ON (A_{i_1}, ..., A_{i_l})
LIMIT     k
```

where $A_1, \ldots, A_n$ are attributes in table S, P is a predicate on tuples in S, $1 \leq i_j \leq n$ $(1 \leq j \leq l)$, and k is a positive integer. The query treats the subset of tuples in S that satisfying P as the target subset, and returns the top-k tuples in the target subset having the largest discriminative typicality values computed using attributes $A_{i_1}, \ldots, A_{i_l}$.

Example 2 (Top-k discriminative typicality queries). Consider the set S of points in Figure 2 again and the following query.

```
SELECT    X, Y   FROM S
WHERE     COLOR = white
ORDER BY  DISC TYPICALITY ON (X)
LIMIT     3
```

In Figure 3, we also plot $DT(o, C, S)$, where C is the set of white points, and S is the set of all points.

To see the difference between discriminative typicality and simple typicality, consider objects a, b, c, which have a large simple typicality value among all white points. However, they also have a relatively high probability density value as a member in the subset of black points comparing to other white points. Therefore, they are not discriminative.

White points d, e, f are the answer to the query, since they are discriminative. ∎

3. RELATED WORK

Our study is mainly related to the previous work in the following aspects: typicality in psychology and cognition science, top-k queries (also as known as ranking queries) in databases, the 1-median problem, typicality probability and spatially-decaying aggregation.

3.1 Typicality Analysis in Psychology and Cognition Science

Typicality of objects has been widely discussed in psychology and cognition science [19, 35].

People judge some objects to be "better examples" of a concept than others. This is known as the *graded structure* [32] of a category. It is reported in [32] that every category observed so far has been found to have graded structure. Generally, the graded structure is a continuum of category representativeness, beginning with the most typical members of a category and continuing through less typical members to its atypical members.

There are several determinants of graded structure. One determinant is the *central tendency* [5] of a category. Central tendency is either one or several very representative exemplar(s), either existing in the category or not. An exemplar's similarities to the central tendency determine its typicality in this category. Another determinant of typicality is the *stimulus similarity* [28]. Generally, the more similar an instance is to the other members of its category, and the less similar it is to members of the contrast categories, the higher the typicality rating it has.

The prototype view [31] suggests that a concept be represented by a prototype, such that objects "closer to" or "more similar to" the prototype are considered to be better examples of the associated concept. The exemplar view [10] is an alternative to the prototype view that proposes using real objects as exemplars instead of abstract prototypes that might not exist in real life. Finally, the schema view [13] improves the prototype view by model concepts in schema theory and AI knowledge representation.

Feature-frequency model defines typicality from a different scope [29]. A typical member of a category will share many attributes with other members of the category and few attributes with members of other categories. An attribute can be scored based on how many members possess that attribute. A family resemblance score for each member sums up the numerical scores of all attributes possessed by that member. A category member with a higher family resemblance score is considered more typical.

Although typicality has not been used before in query answering on large databases, the idea of typicality was recently introduced into ontology design and conceptual modeling [4, 3], which are generally related to database design.

How Is Our Study Related?

Our typicality measures resemble the general spirit of typicality measures used in psychology and cognition science. As suggested by the previous studies in psychology and cognition science, typicality measures may vary in different applications. In our study, we propose simple typicality and discriminative typicality for different application requirements.

Studies on typicality in psychology and cognition science often do not address the concerns about efficient query answering from large databases. Complementary to those studies, we focus on efficient query answering.

3.2 Top-k (Ranking) Queries

As pointed out in Section 1.2.1, *top-k typicality queries* can be viewed as one special type of *top-k (ranking)* queries. There are numerous query answering algorithms proposed for top-k queries in the literature. The threshold algorithm (TA) [27] is one of the best algorithms. TA first sorts the values in each attribute and stores them into a set of sorted lists. Then, TA scans the sorted lists in parallel. Each time a new tuple appears, TA looks up in all lists to calculate its score. Also, TA maintains a "stopping value", which acts as a threshold to prune the tuples in the rest of the lists if they cannot have better scores than the threshold. Several variants of TA have also been proposed, such as [20].

The recent development and extension of top-k query answering include using views to answer top-k queries efficiently [16], removing redundancy in top-k patterns [36], applying multidimensional analysis in top-k queries [37], continuous monitoring of top-k queries over a fixed-size window of the most recent data [26], and so forth.

How Is Our Study Related?

Our top-k typicality queries aim at providing a list of objects with the highest typicality scores according to the proposed typicality measures. This is similar to the existing ranking queries in general.

However, our scoring functions (typicality measures) are different from most of the scoring functions studied before in top-k queries. The typicality score for each object is based on its relationship with the other objects in question. This is much more complicated than scoring an individual object based on its own attributes independently, like most other scoring functions do. Therefore, some classic and efficient query answering algorithms for top-k queries cannot be directly applied to answer top-k simple typicality and discriminative typicality queries. We have to develop new efficient query answering algorithms.

3.3 The (Discrete) 1-Median Problem

Finding typical objects is also broadly related to the 1-median problem in computational geometry. Given a set S of n points, the 1-*median problem* is to find a point M minimizing the sum of distances from all points in S to M. Point M is called the median of the data set. Under the constraint that M belongs to S, it is known as the *discrete* 1-*median problem*. We can always find the exact median in $O(n^2)$ time.

Several approximation algorithms have been proposed to compute the approximate median efficiently. [7] proposes a quad-tree based data structure to support finding the approximate median with a constant approximation ratio and $O(n \log n)$ time complexity. A randomized algorithm was proposed in [12], which computes the approximate median in linear time. Although the approximation ratio cannot be bounded, it performs well in practice. [23] also provides a $(1 + \delta)$-approximation algorithm with runtime $O(n/\delta^5)$ which is based on sufficiently large sampling. [6] proposes an algorithm to solve the median problem in L_1 metric in $O(n \log n)$ time.

How Is Our Study Related?

The top-k simple typicality query is somewhat similar to the discrete 1-median problem in that they both want to find the objects in a data set optimizing the scores with respect to

their relationship to other objects. However, as will be clear in Section 4.1, the functions to optimize are different. The methods of the 1-median problem cannot be applied directly to answer top-k typicality queries.

3.4 Other Related Models

Typicality probability [11, 21] in statistic discriminant analysis is defined as the Mahalanobis distance between an object and the centroid of a specified group, which provides an absolute measure of the degree of membership to the specified group. Given an observation o, [11, 21] also define the probability of group membership as the posterior probability of o as a member of group g. o will then be classified into group g with maximal probability of group membership. This is also known as *maximal likelihood classification*.

Spatially-decaying aggregation [14, 15] is defined as the aggregation values influenced by the distance between data items. Generally, the contribution of a data item to the aggregation value at certain location decays as its distance to that location increases. Nearly linear time algorithms are proposed to compute the ϵ-approximate aggregation values when the metric space is defined on a graph or on the Euclidean plane.

How Is Our Study Related?

Although typicality probability also looks into the typicality of group members, it tackles a different problem. The use of Mahalanobis distance somewhat restricts the applications of this definition, while our definition of typicality can be applied to data sets in generic metric spaces.

Discriminant analysis using probability of group membership mainly focuses on how to correctly classify the objects. It does not consider the typicality of group members. Our definition of discriminative typicality combines both the discriminability and the typicality of the group members, which is more powerful in capturing the "important" instances in multi-class data sets.

Moreover, [11, 21] do not discuss how to answer those queries efficiently on large data sets.

Spatially-decaying sum with *exponential decay function* [14, 15] is similar to our definition of simple typicality. However, in [14, 15], the spatially-decaying aggregation problem is defined on graphs or Euclidean planes, while we assume only a generic metric space. On the one hand, the efficiency in [14, 15] may not be carried forward to the more general metric space. The techniques developed in this paper may be useful to compute spatially-decaying aggregation on a general metric space. On the other hand, when typicality queries are computed on graphs or Euclidean planes, some ideas in [14, 15] may be borrowed.

4. QUERY ANSWERING ALGORITHMS

In this section, we first discuss how to estimate probability density, then, we show that answering top-k typicality queries is quadratic in nature. Last, we present a randomized tournament approximation algorithm (RT).

4.1 Probability Density Estimation

Estimating probability density is essential in typicality computation. There are several model estimation techniques in the literature [18], including parametric and non-parametric density estimation. Parametric density estimation requires a certain distribution assumption, while non-parametric estimation does not. Among the various techniques proposed for non-parametric density estimation [17], histogram estimation [24], kernel estimation [1, 9] and nearest neighbor estimation [25] are the most popular. In this paper, we use kernel estimation, because it can estimate unknown data distribution effectively [22].

Input: a set of n objects $o_1, \ldots, o_n$ and positive integer k;
Output: the top-k objects with the highest simple typicality values;
Method:
1: FOR each object o, set $T(o) = 0$;
2: FOR $i = 0$ TO $(n-1)$
3: FOR $j = i+1$ TO n
4: $w = \frac{1}{(n-1)\sqrt{2\pi}} e^{-\frac{d(o_i,o_j)^2}{2h^2}}$;
5: $T(o_i) = T(o_i) + w$; $T(o_j) = T(o_j) + w$;
END-FOR
END-FOR
6: return the top-k objects according to $T(o)$;

Figure 4: An exact algorithm to answer top-k simple typicality queries.

Kernel estimator is a generalization of sampling. In random sampling, each sample point carries a unit weight. However, an observation of the sample point increases the chance of observing other points nearby. Therefore, kernel estimator distributes the weight of each point in the nearby space around according to a *kernel function* K. Moreover, the bandwidth parameter (also as known as the smoothing parameter) h is introduced to control the distribution among the neighborhood of the sample. As shown in [34], the accuracy of the kernel estimation depends mostly on the bandwidth h and lightly on the choice of the kernel K. In this paper, we choose the commonly used Gaussian kernels. Our approach can also be adapted using other kernel functions.

We want to address the top-k typicality problem in a generic metric space. The only parameter we use in density estimation is the distance (or similarity) between two objects. Formally, given a set of objects $S = (o_1, o_2, \ldots, o_n)$ in a generic metric space, the underlying probability density function $\hat{f}(x)$ is approximated as follows:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} G_h(x, o_i) = \frac{1}{n\sqrt{2\pi}}\sum_{i=1}^{n} e^{-\frac{d(x,o_i)^2}{2h^2}}$$

where $d(x, o_i)$ is the distance between x and o_i in the metric space, and $G_h(x, o_i) = \frac{1}{\sqrt{2\pi}} e^{-\frac{d(x,o_i)^2}{2h^2}}$ is a **Gaussian kernel**.

4.2 An Exact Algorithm and Complexity

Theoretically, given a set of objects S, if the probability density of an object $o \in S$ $f(o) \propto \frac{1}{\sum_{o' \in S} d(o,o')}$, then the discrete 1-median problem can be reduced to a special case of the top-1 simple typicality query problem. As so far no better than quadratic algorithm has been found for exact solutions to the general discrete 1-median problem (except in L_1 metric space), it is challenging to find a better than quadratic algorithm for computing exact answers to general top-k similarity queries.

Figure 4 gives a straightforward quadratic algorithm that computes the exact answer to a top-k simple typicality query.

Quadratic algorithms are often too costly for online queries

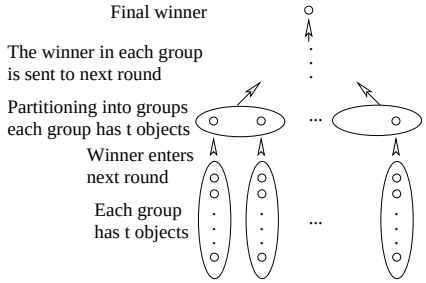


Figure 5: The illustration of the randomized algorithm.

on large databases. On the other hand, good approximations of the exact answers are often good enough for typicality analysis. This motivates our development of approximation algorithms.

4.3 RT: A Randomized Tournament Algorithm

Inspired by the randomized tournament method [12] for the discrete 1-median problem, we propose a randomized tournament algorithm for answering top-k simple typicality queries as follows.

Let t be a small integer, called the *tournament group size*. To find the most typical object in a set S of n objects, we partition the objects into $\lceil \frac{n}{t} \rceil$ groups randomly such that each group has t objects. For each group, we use the exact algorithm to find the object that has the largest simple typicality value in the group. Those winner objects in the groups are selected to the next round, as illustrated in Figure 5.

The winners of the previous round are again partitioned randomly into groups such that each group contains t objects. The most typical object in each group is selected and sent to the next round. The tournament terminates until only one object is the winner. The final winner is an approximation of the most typical object.

To approximate the second most typical object, we run the randomized tournament again with the following constraint: the most typical object already chosen in the previous tournament cannot be selected as the winner in this tournament. The final winner in the second tournament is the approximation of the second most typical object. So forth we can find the approximations of the top-k typical objects in k times of tournaments.

In order to achieve a higher accuracy, we can run this randomized tournament several times for selecting the approximation of the i-th most typical object ($1 \leq i \leq k$), and pick the object with the largest simple typicality among all the final winners. The algorithm is shown in Figure 6.

The typicality computation within one group has the complexity of $O(t^2)$. There are $\lceil \log_t n \rceil$ tournament rounds in total. Without loss of generality, let us assume $n = t^m$. Then, the first round has $\frac{n}{t}$ groups, the second round has $\frac{\frac{n}{t}}{t} = \frac{n}{t^2}$ groups, and so forth. The total number of groups is $\sum_{1 \leq i \leq \log_t n} \frac{n}{t^i} = \frac{n}{t-1}(1 - \frac{1}{t^m}) = O(\frac{n}{t})$. The complexity of selecting the final winner is $O(t^2 \cdot \frac{n}{t}) = O(tn)$. If we run each tournament v times for better accuracy, and run tournaments to choose top-k typical objects, the overall complexity is $O(kvtn)$.

The randomized algorithm runs in linear time with respect to the number of objects. However, the accuracy of the approximation to the answer is not guaranteed in theory, though in practice it often has reasonable performance.

Input: a data set S, positive integer k, tournament size t and number of validations v;
Output: approximation to the answer to a top-k simple typicality query;
Method:

```
1:  FOR i = 1 TO k
2:    set the candidate set to empty;
3:    FOR j = 1 TO v
4:      REPEAT
5:        randomly partition the objects in S into groups, each
          group has t objects;
6:        FOR EACH group g
7:          compute the typicality of each points in g within
            the group;
8:          select one object other than any already output
            object with the largest typicality in g and remove
            the other objects;
          END FOR-EACH
9:      UNTIL there is only one object left in S;
10:     add the winner to the candidate set;
      END FOR
11:   compute the simple typicality of all the points in the
      candidate set over the whole data set;
12:   output the object with the largest simple typicality as the
      approximation of the i-th most typical object;
    END FOR
```

Figure 6: The randomized tournament algorithm (RT) for answering top-k simple typicality queries.

Discriminative typicality can be calculated using Theorem 1. The randomized tournament algorithm can also be used to answer top-k discriminative typicality queries if the discriminative typicality measure is applied. Limited by space, we omit the details here.

5. LOCAL TYPICALITY APPROXIMATION

While the randomized tournament method is efficient, it cannot guarantee the approximation quality. Can we provide some quality guarantee and at the same time largely retain the efficiency? In this section, we develop several heuristic local typicality approximation methods. We use simple typicality as the example most of the time. It can be extended to discriminative typicality straightforwardly.

5.1 Locality of Typicality Approximation

In Gaussian kernel estimation, given two points a and p, the contribution from p to $\hat{f}(a)$ is $\frac{1}{n\sqrt{2\pi}} e^{-\frac{d(a,p)^2}{2h^2}}$, where n is the size of the data set. The contribution of p decays exponentially as the distance between a and p increases. Therefore, if p is remote from a, p contributes very little to the density of a.

Moreover, in a metric space, given three points a, b and p, the triangle inequality $|d(a,p) - d(b,p)| < d(a,b)$ holds. If $d(a,p) \gg d(a,b)$, then $d(a,p) \approx d(b,p)$. Therefore, the objects far away from a and b will have similar contributions to the probability density values $\hat{f}(a)$ and $\hat{f}(b)$.

Based on the above observations, given a set of objects S and a subset $C \subseteq S$, can we use the locality to approximate the object having the largest simple typicality value in C?

Definition 3 (Local neighborhood). Given a set of objects S, a neighborhood threshold σ, and a subset $C \subseteq S$,

The σ-**local neighborhood** of C

$$LN(C,\sigma) = \{o | o \in S, \min_{o' \in C}\{d(o,o')\} \leq \sigma\}$$

is the set of objects whose distance to at least one object in C is at most σ. ∎

We can use the local neighborhood to compute the local typicality defined as follows.

Definition 4 (Local simple typicality). Given a set of objects S, a neighborhood threshold σ, and a subset $C \subseteq S$, the **local simple typicality** of an object $o \in C$ is defined as $LT(o,C,\sigma) = f(o|LN(C,\sigma))$. ∎

The following result uses local simple typicality to approximate the simple typicality with a quality guarantee.

Theorem 2 (Local typicality approximation). *Given a set of objects S, neighborhood threshold σ, and a subset $C \subseteq S$, let $\widetilde{o} = \arg\max_{o_1 \in C}\{LT(o_1,C,\sigma)\}$ be the object in C having the largest local simple typicality value, and $o = \arg\max_{o_2 \in C}\{T(o,S)\}$ be the object in C having the largest simple typicality value. Then,*

$$T(o,S) - T(\widetilde{o},S) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \quad (1)$$

Moreover, for any object $x \in C$,

$$T(x,S) - LT(x,C,\sigma) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \quad (2)$$

Proof. For any object $x \in C$,

$$T(x,S) = \frac{1}{|S|}\Big(\sum_{y \in LN(C,\sigma)} G_h(x,y) + \sum_{z \in (S - LN(C,\sigma))} G_h(x,z)\Big)$$

Since $LT(x,C,\sigma) = \frac{1}{|LN(C,\sigma)|}\sum_{y \in LN(C,\sigma)} G_h(x,y)$,

$$T(x,S) = \frac{1}{|S|}\Big(|LN(C,\sigma)| \cdot LT(x,C,\sigma) + \sum_{z \in (S - LN(C,\sigma))} G_h(x,z)\Big) \quad (3)$$

Because $LN(C,\sigma) \subseteq S$, $\frac{|LN(C,\sigma)|}{|S|} \leq 1$. Thus,

$$T(x,S) \leq LT(x,C,\sigma) + \frac{1}{|S|}\sum_{z \in (S - LN(C,\sigma))} G_h(x,z)) \quad (4)$$

According to the definition of local neighborhood, $d(x,z) > \sigma$ for any $z \in (S - LN(C,\sigma))$. Thus,

$$\frac{1}{|S|}\sum_{y \in (S - LN(C,\sigma))} G_h(x,y) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \quad (5)$$

Inequality 2 follows with Inequalities 4 and 5 immediately.

Applying Equation 3 to o and $\widetilde{o}$, respectively, we have

$$\begin{aligned} & T(o,S) - T(\widetilde{o},S) \\ = \; & \tfrac{|LN(C,\sigma)|}{|S|}(LT(o,C,\sigma) - LT(\widetilde{o},C,\sigma)) + \\ & \tfrac{1}{|S|}\textstyle\sum_{z \in (S - LN(C,\sigma))}(G_h(o,z) - G_h(\widetilde{o},z)) \end{aligned}$$

Using Inequality 5, we have $\frac{1}{|S|}\sum_{z \in (S - LN(C,\sigma))}(G_h(o,z) - G_h(\widetilde{o},z)) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$.

Since $LT(\widetilde{o},C,\sigma) \geq LT(o,C,\sigma)$, $LT(o,C,\sigma) - LT(\widetilde{o},C,\sigma) \leq 0$. Thus,

$$T(o,S) - T(\widetilde{o},S) \leq \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$$

Inequality 1 is shown. ∎

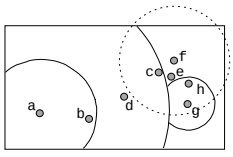


Figure 7: Decomposing a set of objects in a VP-tree.

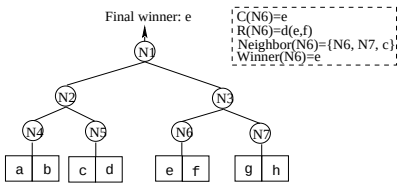


Figure 8: The VP-tree.

5.2 DLTA: Direct Local Typicality Approximation Using VP-trees

Inequality 2 in Theorem 2 can be used immediately to approximate simple typicality computation with quality guarantee. Given a neighborhood threshold σ, for each object $x \in S$, we compute the σ-local neighborhood of $\{x\}$, i.e., $LN(\{x\},\sigma)$, and the local simple typicality $LT(x,\{x\},\sigma)$. We use the local simple typicality $LT(x,\{x\},\sigma)$ as the approximation of the simple typicality $T(x)$, and pick the k objects with the highest local simple typicality values as the approximation to the answer of the top-k simple typicality query. As ensured by Theorem 2, the error in simple typicality is less than $\frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$.

Searching the σ-neighborhood (i.e., the neighbors with distance up to σ) for each object can be very costly. To implement the direct local typicality approximation efficiently, we can use a VP-tree which can support σ-neighborhood searches effectively.

A VP-tree [38] is a binary space partitioning (BSP) tree. Given a set of objects S, a VP-tree T indexes the objects in S. Each node in a VP-tree represents a subset of S. Roughly speaking, for each non-leaf node N and the set of nodes S_N at N, a *vantage point* is selected to divide the set S_N into two exclusive subsets S_{N_1} and S_{N_2} ($S_N = S_{N_1} \cup S_{N_2}$) such that, to search the nearest neighbor within distance σ for an object $p \in N$, likely we only need to search either S_{N_1} or S_{N_2} but not both. S_{N_1} and S_{N_2} are used to construct the two children of N. For example, consider the objects in Figure 7. A VP-tree in Figure 8 indexes the objects.

A VP-tree can be constructed top-down starting from the root which represents the whole set of objects. A sampling method is given in [38] to select vantage points for internal nodes. Then, the first half subset of objects that are close to the vantage point form the left child of the root, and the second half subset of objects that are far away from the root form the right child. The left and the right children are further divided recursively until a node contains only one object (a leaf node). A VP-tree can be constructed with cost $O(|S| \log |S|)$.

Searching a VP-tree for the σ-neighborhood of a query point is straightforward using the recursive tree search. Once an internal node in the tree can be determined in the σ-neighborhood of the query point, all descendant objects of the internal node are in the neighborhood and no subtrees need to be searched.

The cost of computing the local simple typicality of an object x is $O(|LN(x,\sigma)|)$. Then, the cost of computing the local simple typicality of all objects is $O(\sum_{x \in S}|LN(\{x\},\sigma)|)$. Although the local neighborhood search can be sped up using a VP-tree (i.e., reducing V), the σ-neighborhoods may still contain many objects. In the worst case where σ is

larger than the diameter (i.e., the largest pairwise distance) of the data set, the σ-neighborhood of each object contains all other objects, and thus the method of direct local typicality approximation, even using a VP-tree, is $O(|S|^2)$.

5.3 LT3: Local Typicality Approximation Using Tournaments

Can we reduce the cost of computing local simple typicality further?

5.3.1 Randomized Tournaments

Straightforwardly, Inequality 1 in Theorem 2 can be applied to the randomized tournament algorithm to guarantee the answer quality immediately.

In a tournament selecting the most typical object in a group C, instead of computing the simple typicality in C, we can find the object $\widetilde{o}$ in C having the largest local typicality as the winner of the group, and send it to the next round of tournament. We have the following quality guarantee.

THEOREM 3. *In a set S, let o be the object of the largest simple typicality and $\widetilde{o}$ be an object computed by a randomized tournament method using local typicality approximation and the tournament group size t. Then,*

$$T(o,S) - T(\widetilde{o},S) < \frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}} \cdot \lceil \log_t |S| \rceil$$

Proof sketch. We only need to notice that because of the triangle inequality in metric spaces, by each level of tournament, an error up to $\frac{1}{\sqrt{2\pi}} e^{-\frac{\sigma^2}{2h^2}}$ in terms of the difference of simple typicality is introduced, as indicated by Inequality 1 in Theorem 2. For a set S of objects, there are $\lceil \log_t |S| \rceil$ levels of tournaments. ∎

The quality guarantee depends on the group size t in the tournaments. By setting values of σ and t properly, we can ensure the required approximation quality.

To compute the local simple typicality, we need to consider all objects in the local neighborhood. However, in the randomized tournament algorithm, objects are partitioned into groups randomly. If the objects in a group are not local to each other, then the σ-local neighborhood may include many objects. In the worst case, each local neighborhood includes all the other objects in the data set. Then, computing the local simple typicality values of the objects in the group takes the cost of $O(nt + t^2) = O(nt)$. As discussed in Section 4.3, there are $O(\frac{n}{t})$ groups, the cost of the algorithm can still be as large as $O(nt) \cdot O(\frac{n}{t}) = O(n^2)$.

5.3.2 Local Typicality Trees (LT-trees)

Can we partition the objects systematically so that the groups tend to be local? Moreover, as a VP-tree can facilitate neighborhood search, can we use a VP-tree? Here, we propose a sampling method employing these tactics.

Given a set S of objects. We construct a VP-tree as described in [38]. Since a VP-tree is a binary tree, to reduce the number of rounds of tournaments, we can also use an MVP-tree [8], a t-nary VP-tree that uses more than one vantage point to partition the space. Without loss of generality, let us assume $t = 2^l$ and the data set contains t^m objects.

We assign a layer number to each node in the VP-tree. The root node has layer number 0, and a node is assigned layer number $(i + 1)$ if its parent has layer number i. We remove all those nodes in the VP-tree whose layer number is not a multiple of t. For a node N of layer number jt $(j \geq 1)$, we connect N to its ancestor in the VP-tree of layer $(j-1)t$. We call the resulting tree the *LT-tree* (for local typicality tree).

In addition, each node in the LT-tree records the following three pieces of information: the approximate center, the radius, and the σ-neighborhood.

For a node N in the LT-tree, let S_N be the set of objects at N. To compute the approximate center at a node N in the LT-tree, we draw a sample R of $\sqrt{|S_N|}$ objects from S_N, and compute the pairwise distance between every two objects in R. Then, for each object $x \in R$, the center-score of x is the maximal distance from x to another point in R. The object in R of the minimum center-score is chosen as the center. This center approximation procedure is popularly used in computational geometry. It takes $O(|S_N|)$ time for each node N, and $O(|S| \log_t |S|)$ time for all nodes in the LT-tree.

Once the center c of a node N is chosen, the radius is given by the maximum distance between c and the other objects at N. This can be computed in time $O(|S_N|)$ for each node N, and $O(|S| \log_t |S|)$ for all nodes in the LT-tree.

We use a range query in the LT-tree to compute a superset of the σ-neighborhood of S_N for every node N in the LT-tree, which always achieves a better typicality approximation than using the σ-neighborhood. To compute the superset, we start from the root and iteratively search for the nodes that completely lies in the σ-neighborhood of N, using the approximate center and radius of N. Once all objects at a node N' in the VP-tree are in the σ-neighborhood of N, we use N' to represent them and do not search any subtree of N'.

5.3.3 Query Answering

To answer a top-k simple typicality query, we run tournaments on the LT-tree bottom-up. First, a tournament is run for each leaf node in the LT-tree. The winner enters the tournament at the parent node. The winner o_1 at the root node is the approximation of the most typical object. The approximation quality is guaranteed by Theorem 3.

To find the approximation of the second most typical object, we do not need to completely run the tournaments again. Instead, we can reuse most of the results in the tournaments finding the most typical object o_1. The only tournaments we need to run are on the nodes containing o_1. We run those tournaments bottom-up, too. First, we run a new tournament at the leaf node N_1 containing o_1, but do not include o_1 in the new tournament. Then, the winner o_1' is sent to N_2, the parent of N_1, and a new tournament is run there by replacing o_1 by o_1'. A series of m tournaments are needed to find a new winner o_2 in the root node, which is the approximation of the second most typical object. At each level of the LT-tree, only one node needs to run a tournament.

As shown in our experiments, using the LT-tree we can improve both the quality and the efficiency of local typicality approximation. The approximation of the most typical objects is very close to the exact ones.

By using the LT-tree, the objects in a group are relatively local to each other. Thus, the neighborhood may likely be local, too.

5.3.4 A Sampling Method for Bounding Runtime

Category	# tuples	Most typical	Most discriminative typical	Most atypical
Mammal	40	Boar, Cheetah, Leopard, Lion, Lynx, Mongoose, Polecat, Puma, Raccoon, Wolf ($T = 0.16$)	Boar, Cheetah, Leopard, Lion, Lynx, Mongoose, Polecat, Puma, Raccoon, Wolf ($DT = 0.08$)	Platypus ($T = 0.01$)
Bird	20	Lark, Pheasant, Sparrow, Wren ($T = 0.15$)	Lark, Pheasant, Sparrow, Wren ($DT = 0.04$)	Penguin ($T = 0.04$)
Fish	14	Bass, Catfish, Chub, Herring, Piranha ($T = 0.15$)	Bass, Catfish, Chub, Herring, Piranha ($DT = 0.03$)	Carp ($T = 0.03$)
Invertebrate	10	Crayfish, Lobster ($T = 0.16$)	Crayfish, Lobster ($DT = 0.01$)	Scorpion ($T = 0.08$)
Insect	8	Moth, Housefly ($T = 0.13$)	Gnat ($DT = 0.02$)	Honeybee ($T = 0.06$)
Reptile	5	Slowworm ($T = 0.17$)	Pitviper ($DT = 0.007$)	Seasnake ($T = 0.08$)
Amphibian	3	Frog ($T = 0.2$)	Frog ($DT = 0.008$)	Newt, Toad ($T = 0.16$)

Table 1: The most typical, the most discriminatively typical, and the most atypical animals (T =simple typicality value, DT =discriminative typicality value).

To make the analysis complete, here we provide a sampling method which provides an upper bound on the cost of local typicality computation with quality guarantee.

Suppose we want to compute the local simple typicality $LT(p, C, \sigma)$. We consider the contribution of an object $o \in LN(C, \sigma)$ to $LT(p, C, \sigma)$, denoted by

$$\eta(o) = \frac{1}{|LN(C,\sigma)|} G_h(p, o) = \frac{e^{-\frac{d(p,o)^2}{2h^2}}}{|LN(C,\sigma)|\sqrt{2\pi}}$$

We can draw a sample of $LN(C, \sigma)$ to estimate the expectation of $\eta(o)$. Please note that $LT(p, C, \sigma) = |LN(C, \sigma)| \cdot E[\eta(o)]$, where $E[\eta(o)]$ is the expectation of $\eta(o)$ for $o \in LN(C, \sigma)$.

THEOREM 4. *For any δ $(0 < \delta < 1)$ and ϵ $(\epsilon > 0)$ and a sample R of $LN(C, \sigma)$, if*

$$|R| > \frac{3\sqrt{2\pi} \cdot e^{\frac{\sigma^2}{2h^2}} \cdot \ln \frac{2}{\delta}}{\epsilon^2}$$

then

$$f\{|\frac{|LN(C,\sigma)|}{|R|} \sum_{o \in R} \eta(o) - LT(p, C, \sigma)| > \epsilon \cdot LT(p, C, \sigma)\} < \delta$$

Proof sketch. The theorem can be proved using a special form of the Chernoff-Hoeffding bound due to Angluin and Valiant [2]. Limited by space, we omit the details here. ∎

Theorem 4 provides an upper bound of the sample size, which is independent of the size of data sets. For example, the sample size is $1,171$ when $\epsilon = 0.3$, $\delta = 0.3$, $\sigma = 2h$, and is $202,735$ when $\epsilon = 0.1$, $\delta = 0.1$, $\sigma = 3h$. The larger ϵ and δ, the smaller the sample size. The larger σ, the larger the sample size.

Using the sampling method suggested by Theorem 4, we can have a tournament algorithm using an LT-tree of cost $O(n \log n)$. However, we observe from the experiments that, since the LT-tree already exploits the locality of objects nicely, the sampling method does not bring in too much gain in efficiency, but may loose much in quality. Limited by space, we do not include the experimental results on this sampling method.

6. EMPIRICAL EVALUATION

In this section, we report a systematic empirical study using real data sets and synthetic data sets. All the experiments were conducted on a PC computer with a 3.0 GHz Pentium 4 CPU, 1.0 GB main memory, and a 160 GB hard disk, running the Microsoft Windows XP Professional Edition operating system. Our algorithms were implemented in Microsoft Visual C++ V6.0.

By default we set the bandwidth of the Gaussian kernel estimator $h = \frac{1.06s}{\sqrt[5]{n}}$ as suggested in [33], where n is the size of the data set and s is the standard deviation of the data set which can be estimated by sampling.

6.1 Typicality Queries on Real Data Sets

In this section, we use two real data sets to illustrate the effectiveness of typicality queries on real applications.

6.1.1 Typicality Queries on the Zoo Data Set

We use the Zoo Database from the UCI Machine Learning Database Repository[1]. It is a small data set of 100 tuples on 15 Boolean attributes and 2 numerical attributes. All tuples are classified into 7 categories (*mammals*, *birds*, *reptiles*, *fish*, *amphibians*, *insects* and *invertebrates*).

We compute the simple typicality and the discriminative typicality for each animal in the data set. Table 1 shows the most typical, the most discriminatively typical, and the most atypical animals of each category. Since some tuples, such as those 10 most typical animals in category *mammals*, have the same values on all attributes, they have the same typicality value.

The results match our common sense of typicality. The most typical animals in each category can serve as good exemplars of the category. For example, in category *mammals*, the most typical animals are more likely to be referred to as a mammal than the most atypical one, *platypuses*, which are one of the very few mammal species that lay eggs instead of giving birth to live young. As another (possibly arguable) justification, searching "lion, mammal" in Google returns $881,000$ results, but searching "platypus, mammal" returns only $141,000$ results.

Sometimes, animals from different categories are also similar to each other. For example, in the Zoo Database, *slugs* from category *invertebrates* and *termites* from category *insects* are different only in attribute *number of legs*. Moreover, *slowworm*, the most typical *reptiles* in the Zoo Database, is also similar to *newts* in category *amphibians*, as well as *basses* and *catfish* in category *fish*. The differences are only on two or three attributes. Therefore, a typical instance in one category may also resemble some instances in other

[1]http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Name	T	Position	Minuts	Points per game	3 point throw	Rebounds	Assists	Blocks	Personal Fouls
Danny Granger	0.0383	Forwards	22.6	7.5	1.6	4.9	1.2	0.8	2.7
Devean George	0.0382	Forwards	21.7	6.3	3	3.9	1.0	0.5	2.2
Michael Finley	0.0378	Guards	26.5	10.1	5	3.2	1.5	0.1	1.3

Table 2: The most typical NBA players in 2005-2006 Season (T for simple typicality values).

	Name	T	DT	3 point throw	Rebounds	Assists	Blocks
Top-2 discriminative typicality query	Delonte West	0.021	0.0095	4.3	4.1	4.6	0.7
	David Wesley	0.021	0.0092	5.2	2.5	2.9	0.1
Top-2 simple typicality query	Ronald Murray	0.076	0.0059	2.4	2	2.6	0.1
	Marko Jaric	0.075	0.0046	2.3	3.1	3.9	0.3

Table 3: The most discriminatively typical guards (T for simple typicality values, DT for discriminative typicality values).

Category (position)	Median/mean/most typical	Name	Simple typicality	# games	Avg. min. per game
All players	median	Ryan Gomes	0.0271	61	22.6
	mean	N/A	0.0307	54.4	20.51
	most typical	Danny Granger	0.3830	78	22.6
Centers	median	Jake Voskuhl	0.0903	51	16
	mean	N/A	0.0679	52.42	17.36
	most typical	Francisco Elson	0.1041	72	21.9
Forwards	median	Al Jefferson	0.0747	59	18
	mean	N/A	0.0509	54.83	19.97
	most typical	Maurice Taylor	0.0910	67	18.1
Guards	median	Charlie Bell	0.0488	59	21.7
	mean	N/A	0.0230	54.54	21.73
	most typical	Ronald Murray	0.0756	76	27.8

Table 4: Comparison among medians, means, and typical players in the NBA data set.

categories, which makes this typical instance not so distinguishing.

Thus, we apply *discriminative typicality analysis* on the Zoo Database to find the discriminative typical animals for each category that can best distinguish the category from other categories. In some categories, the tuples having the largest simple typicality value also have the highest discriminative typicality value, such as categories *mammals*, *birds*, *fish*, *invertebrates*, and *amphibians*.

In some categories such as *insects* and *reptiles*, the most typical animals are not the most discriminatively typical. For example, in category *reptiles*, the most discriminatively typical animal is *pitvipers* in stead of *slowworm*, because *slowworm* are also similar to some animals in other categories besides *reptiles*, such as *newts* in category *amphibians*. On the other hand, *pitvipers* are venomous. Very few animals in the other categories are venomous. The result matches the analysis above.

By the typicality queries on the Zoo Database, we can identify typical cases which cannot be easily captured by other queries.

6.1.2 Typicality Queries on the NBA Data Set

We also apply typicality queries on the NBA 2005-2006 Season Statistics[2]. The data set contains the technical statistics of 458 NBA players, including 221 guards, 182 forwards and 55 centers, on 16 numerical attributes.

Table 2 shows the top-3 most typical players, and some of the attribute values. The results answer Jeff's question in Section 1.1.1.

[2]http://sports.yahoo.com/nba/stats/.

To answer Jeff's question in Section 1.1.2, we conduct a top-2 discriminative typicality query on position guards. The results are shown in Table 3. For comparison, in the same table we also list the answer to the top-2 simple typicality query on position guards. To explain the results, we list some selected attributes as well. The most discriminatively typical guards have better performance in 3 point throw than those of the highest simple typicality. 3 point throw is a skill popular in guards, but may not be common in other players.

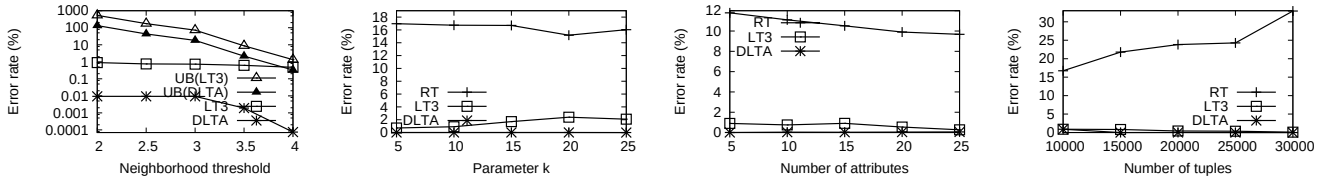
6.1.3 Medians, Means and Typical Objects

Figure 1 elaborates the differences among medians, means, and typical objects in a data set. To further understand the differences in the real data sets, we compute the medians, the means and the typical objects in the NBA data set. The results are shown in Table 4.

We can clearly see that, in this data set, the simple typicality scores of the medians and the means are often substantially lower than the most typical players. This clearly justifies that the geometric centers may not reflect the probability density distribution.

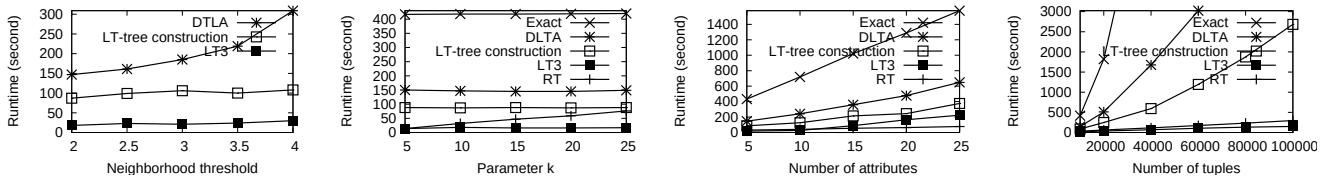
In the table, we also list the values on attributes `number of games played` and `average number of minutes played in each game` to illustrate the differences among medians, means and the most typical players. A typical player can be very different from the median player and the mean. For example, Ronald Murray is identified as the most typical guard, but Charlie Bell is the median guard. Murray makes fewer rebounds than Bell, but contributes more assists. To this extent, Murray is more typical than Bell as a guard.

In principle, typicality analysis is based on the probabil-



(a) Error rate vs. neighborhood. (b) Error rate vs. k. (c) Error rate vs. dimensionality. (d) Error rate vs. cardinality.

Figure 9: Approximation quality.



(a) Runtime vs. neighborhood. (b) Runtime vs. k. (c) Runtime vs. dimensionality. (d) Runtime vs. cardinality.

Figure 10: Efficiency and scalability.

ity density distribution, and is essentially different from the geometric aggregate analysis using medians and means.

6.2 Approximation Quality

To evaluate the query answering quality on large data sets, we use the Quadraped Animal Data Generator also from the UCI Machine Learning Database Repository to generate synthetic data sets with 72 attributes. Limited by space, we only report the results on top-k simple typicality queries here. We use up to 25 numeric attributes, and compute the typicality of a tuple in the whole data set.

To measure the error made by an approximation algorithm, we use the following *error rate* measure. For a top-k typicality query Q, let A be the set of k objects returned by the exact algorithm, and $\widetilde{A}$ be the set of k objects returned by an approximation algorithm. Then,

$$\text{Error rate} = \frac{\sum_{o\in A} T(o) - \sum_{o\in \widetilde{A}} T(o)}{\sum_{o\in A} T(o)} \times 100\%$$

The error rate of the exact algorithm (Figure 4) is always 0. We compare three approximation algorithms: the randomized tournament method (RT, Figure 6), the direct local typicality approximation method (DLTA, Section 5.2), and the LT-tree tournament method (LT3, Section 5.3). By default, we set the number of tuples to $10,000$, the dimensionality to 5 attributes, and conduct top-10 simple typicality queries. When local typicality is computed, by default we set the neighborhood threshold to $2h$, where h is the bandwidth of the Gaussian kernel. In the randomized tournament method, by default the tournament group size is 10 and 4 times validation are conducted. We observe that although with more rounds of validations, the quality of randomized tournament may increase, but after 3 rounds, the quality improvement is very small.

Figure 9(a) shows the change of approximation quality with respect to the neighborhood threshold. In the figure, the error bounds given by Theorems 2 and 3 are also plotted. That is, $UB(DLTA) = \frac{|A|\cdot\frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}}}{\Sigma_{o\in A}T(o)} \times 100\%$ and $UB(LT3) = \frac{|A|\cdot\frac{1}{\sqrt{2\pi}}e^{-\frac{\sigma^2}{2h^2}\cdot\lceil\log_t|S|\rceil}}{\Sigma_{o\in A}T(o)} \times 100\%$. To make the curves legible, the error rates are in the logarithmic scale. Clearly, the larger the neighborhood threshold, the more accurate the local typicality approximation. Our methods perform much better than the error bounds. This clearly shows that our methods are effective in practice.

In Figure 9(b), we vary the number of k in the top-k queries. The approximation quality of RT is not sensitive to k, since it runs k times to select the top-k answers. Both DLTA and LT3 see a larger error rate with a larger value of k. The error rate of DLTA increases from 0.004% to 0.015% when k increases from 5 to 25. When more objects are returned, those distant neighbors may get a better chance to play a role in typicality.

Figure 9(c) shows the impact of dimensionality on the error rate. DLTA achieves the best approximation quality, the error rate is up to 0.066%. LT3 has an accuracy close to DLTA, and is much better than RT. The error rate decreases as the dimensionality increases. As the dimensionality increases, the data set becomes sparse, and the average pair-wise distance in the data set also increases. The local typicality approximation becomes more effective.

Figure 9(d) tests the approximation quality versus the number of tuples in the data set. When the cardinality increases, the data set becomes dense, and the local typicality approximation is more accurate. That is why LT3 and DLTA perform better with larger data sets. However, the approximation quality of RT decreases in large data sets, since with a fixed tournament group size, the larger the data set, the more likely the most typical object in a random group biases.

In summary, DLTA and LT3 can achieve good approximation quality, and DLTA is a little bit better. This strongly justifies that our local typicality approximation technique is effective. Both DLTA and LT3 are much more accurate than RT. Moreover, DLTA and LT3 perform well on large and high-dimensional data sets.

6.3 Efficiency and Scalability

To test the efficiency and the scalability of our methods, we report the runtime in the experiments conducted in Figure 9. The results are shown in Figure 10.

As shown in Figure 10(a), the runtime of DLTA increases substantially when the neighborhood threshold increases, but the increase of LT3 is mild. LT3 conducts tournaments using an LT-tree, and the σ-neighborhoods are indexed.

Figure 10(b) shows that the runtime of DLTA and LT3 is insensitive to the increase of the number of answers to be computed. LT3 incrementally computes other top-k answers after the top-1 answer is computed. Thus, computing more answers only takes minor cost. RT has to run the tournaments k rounds, and thus the cost is linear to k.

As shown in Figure 10(c), among the four methods, RT is the fastest and the exact algorithm is the slowest. LT3 and DLTA are in between, and LT3 is faster than DLTA. All methods are linearly scalable with respect to dimensionality.

Figure 10(d) shows the scalability of the four algorithms with respect to database size. RT has a linear scalability, while the runtime of the exact algorithm increases dramatically on large data sets. LT3 clearly has the better performance and scalability than DLTA on large data sets.

In summary, as RT has linear complexity, when runtime is the only concern, RT should be used. While DLTA and LT3 are much more scalable than the exact algorithm and are much more accurate than RT, they are good for situations where both accuracy and efficiency matter. LT3 has the better efficiency and scalability than DLTA, and can achieve comparable accuracy to DLTA.

7. CONCLUSIONS

In this paper, we apply the idea of typicality analysis from psychology and cognition science to query answering, and study the novel problem of answering top-k typicality queries. The simple typicality and the discriminative typicality measures are proposed for different applications. As computing the exact answers to top-k typicality queries on large data sets can be too costly, we develop a series of approximation methods. By a systematic empirical evaluation using both real data sets and synthetic data sets, we illustrate the effectiveness of top-k typicality queries, and verify the accuracy and the efficiency of our methods.

Typicality can find other applications in databases. As future work, we would like to explore the potential of typicality analysis in data summarization, data warehousing and data mining.

8. REFERENCES

[1] I. S. Abramson. On bandwidth variation in kernel estimates-a square root law. *Annals of statistics*, 10(4):1217–1223, 1982.

[2] D. Angluin and L. G. Valiant. Fast probabilistic algorithms for hamiltonian circuits and matchings. In *STOC '77*.

[3] C. M. Au Yeung and H. F. Leung. Formalizing typicality of objects and context-sensitivity in ontologies. In AAMAS'06.

[4] C. M. Au Yeung and H. F. Leung. Ontology with likeliness and typicality of objects in concepts. In *ER'06*.

[5] L. W. Barsalou. The instability of graded structure: Implications for the nature of concepts. *Concepts and conceptual development*, pages 101–140, 1987.

[6] S. Bespamyatnikh et al. Optimal facility location under various distance functions. In WADS'99.

[7] P. Bose et al. Fast approximations for sums of distances, clustering and the fermat-weber problem. *Computational Geometry: Theory and Applications*, 24(3):135 – 146, April 2003.

[8] T. Bozkaya and M. Ozsoyoglu. Indexing large metric spaces for similarity search queries. In *TODS'99*.

[9] L. Breiman et al. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, May 1977.

[10] L. R. Brooks. Nonanalytic concept formation and memory for instances. In *Cognition and categorization*, 1973. Hillsdale.

[11] N. A. Campbell. Some aspects of allocation and discrimination. *Multivariate Statistical Methods in Physical Anthropology*, pages 177–192, 1984.

[12] D. Cantone et al. An efficient approximate algorithm for the 1-median problem in metric spaces. *SIAM Journal on Optimization*, 16(2):434–451, 2005.

[13] B. Cohen and G. L. Murphy. Models of concepts. *Cognitive Science*, 8:27–58, 1984.

[14] E. Cohen and H. Kaplan. Spatially-Decaying Aggregation Over a Network: Model and Algorithms. In *SIGMOD'04*.

[15] E. Cohen and H. Kaplan. Spatially-Decaying Aggregation Over a Network. In *JCSS'07*.

[16] G. Das et al. Answering top-k queries using views. In *VLDB'06*.

[17] L. Devroye. *A course in density estimation*. Birkhauser Boston Inc, 1987.

[18] L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer; 1st edition, 2001.

[19] D. Dubois et al. Vagueness, typicality, and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6:167–183, 1991.

[20] R. Fagin et al. Optimal aggregation algorithms for middleware. In *PODS'01*.

[21] G. M. Foody et al. Derivation and applications of probabilistic measures of class membership from the maximum likelihood classification. *Photogrammetric Engineering and Remote Sensing*, 58:1335-1341, 1992.

[22] D. Gunopoulos et al. Selectivity estimators for multi-dimensional range queries over real attributes. *VLDB Journal*, 14(2):137–154, April 2005.

[23] P. Indyk. Sublinear time algorithms for metric space problems. In *STOC'99*.

[24] Y. Kanazawa. An optimal variable cell histogram based on the sample spacings. *Annals of statistics*, 20(1):291–304, 1992.

[25] Y. Mack and M. Rosenblatt. Multivariate k-nearest neighbor density estimates. *Journal of Multivariate Analysis*, 9:1–15, 1979.

[26] K. Mouratidis et al. Continuous monitoring of top-k queries over sliding windows. In *SIGMOD'06*.

[27] S. Nepal and M. V. Ramakrishna. Query processing issues in image(multimedia) databases. In *ICDE'99*.

[28] R. M. Nosofsky. Similarity, frequency, and category representations. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 14(1):54–65, 1988.

[29] S. K. Reed. *Cognition: Theory and Applications*. Wadsworth Publishing. 6 edition, 2003.

[30] L. J. Rips and A. Collins. Categories and resemblance. *Journal of experimental psychology*, General 122(4):468 – 486, 1993.

[31] E. Rosch. Cognitive representations of semantic categories. *Journal of Experimental Psychology: General*, 104:192–233, 1975.

[32] E. Rosch. On the internal structure of perceptual and semantic categories. *Cognitive Development and Acquisition of Language*, pages 111–144, 1973.

[33] D. W. Scott and S. R. Sain. Multi-dimensional density estimation. *Handbook of Statistics*, 23: Data Mining and Computational Statistics, 2004.

[34] B. W. Silverman. *Density Estimation for Statistics and Data Analysis (Hardcover)*. Chapman and Hall, 1986.

[35] V. Tamma and T. Bench-Capon. An ontology model to facilitate knowledge-sharing in multi-agent systems. *Knowledge Engineering Review*, 17(1):41–60, 2002.

[36] D. Xin, H. Cheng, X. Yan, and J. Han. Extracting redundancy-aware top-k patterns. In *KDD'06*.

[37] D. Xin et al. Answering top-k queries with multi-dimensional selections: The ranking cube approach. In *VLDB'06*.

[38] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA'93*.