Applied Intelligence (2020) 50:3219–3238
https://doi.org/10.1007/s10489-020-01712-5

Predictive intelligence of reliable analytics in distributed computing environments

Yiannis Kathidjiotis[1] · Kostas Kolomvatsos[1] · Christos Anagnostopoulos[1]

Published online: 14 May 2020

Abstract
Lack of knowledge in the underlying data distribution in distributed large-scale data can be an obstacle when issuing analytics & predictive modelling queries. Analysts find themselves having a hard time finding analytics/exploration queries that satisfy their needs. In this paper, we study how exploration query results can be predicted in order to avoid the execution of 'bad'/non-informative queries that waste network, storage, financial resources, and time in a distributed computing environment. The proposed methodology involves clustering of a training set of exploration queries along with the cardinality of the results (score) they retrieved and then using query-centroid representatives to proceed with predictions. After the training phase, we propose a novel refinement process to increase the *reliability* of predicting the score of new unseen queries based on the refined query representatives. Comprehensive experimentation with real datasets shows that more *reliable predictions* are acquired after the proposed refinement method, which increases the reliability of the closest centroid and improves predictability under the right circumstances.

Keywords Predictive intelligence · Exploration query prediction · Centroid refinement · Machine learning

1 Introduction

Due to the importance and relevance of data in distributed computing environments, large-scale data analytics, predictive modelling, and exploration tasks, they have rightfully found their place in almost all, if not all, of today's industries. While having access to humongous amounts of data is very beneficial, it has introduced many new challenges. One of them is that they cannot be accessed directly (like a traditional data management systems would be accessed); instead, subsets of them can be acquired through *exploration querying*.

✉ Christos Anagnostopoulos
christos.anagnostopoulos@glasgow.ac.uk

Yiannis Kathidjiotis
yiannis.ka@windowslive.com

Kostas Kolomvatsos
kostas.kolomvatsos@glasgow.ac.uk; kostasks@uth.gr

[1] Essence: Pervasive & Distributed Intelligence Research Lab, School of Computing Science, University of Glasgow, Glasgow, G12 8QQ, UK

Although *exploration querying* acts as a solution for accessing distributed data, in most cases there is lack of knowledge about the underlying data distributions and their impact on the results. As a result, users/analysts may find it hard to come up with a query to execute and it can be even harder to find a query that will return a satisfying number of results. The number of results returned by a query, which for future convenience will be referred to as **score**, can vary from being to little to be significant, to being extremely high, which can be more than needed. Apart from the frustration that might be involved in finding the *correct* query, executing the aforementioned queries can lead to the waste of network and storage resources that are involved in transferring and storing query results among computing nodes in a distributed computing environment (including processed data or even raw data for analytics tasks).

As an illustration, we consider the scatter plot in Fig. 1, where the dots represent data points in a 2-dimensional space and the squares represent exploration queries over attributes in both dimensions. One can notice that exploration Query 1 is expected to return a relatively big amount of of data (high score), while Query 3 returns a smaller amount of data (medium score), and Query 2 returns no data (zero score) and thus is by any means *not worth executing*. Users may think though that Query 1 returns

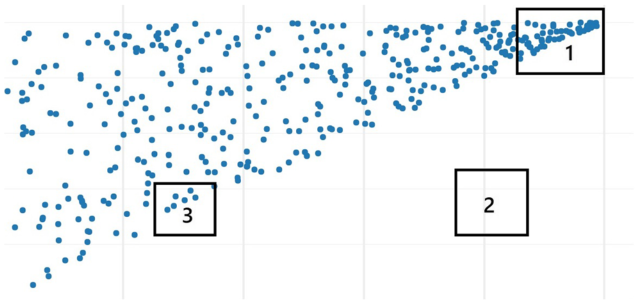


Fig. 1 An illustration example of a dataset and three exploration queries represented by rectangles over the 2-dimensional data space

more results than they require or that Query 3 does not return enough results, therefore in both cases they would have to redefine their exploration queries and exploration policy, in general. The queries would still need to be re-executed for their effect to be seen and they could still not satisfy the users' needs.

Query processing and exploration involves financial costs. The Google's BigQuery [11] service provides interactive exploratory analytics along with MapReduce [9]. This involves the issuing of many exploratory queries over this service with certain non-negligible cost depending on the complexity of the query or the data [11]. Therefore analysts/users/applications issuing queries that do not have a *satisfiable score*. Moreover, taking into consideration the available computational and networking resources, the cost for processing an exploratory and analytics query should be carefully considered in the total cost of holistic task. It is claimed in this paper that such challenges can be avoided if the scores of queries can be reliably predicted in order to decide in advance whether they are worth executing based on user criteria. We focus this research on the exploration queries that are directed towards datasets involving multidimensional data points. Previous research in [2, 3] has evidenced the potential of predicting the results of aggregate queries, through clustering, where specific clustering methods are adopted to provide estimations of the query answers.

Our methodology here departs from the limitations of the previous works in [2, 3] *by adopting query vectorial space clustering and using the resulting closest and rival centroids to provide more reliable and robust predictions.* We then investigate whether *refining* centroids to increase the *reliability* of a query's closest centroid can improve *predictability*. Based on this refinement, we examine different approaches of refinement and prediction, as well as how the behaviour of these approaches can change based on the number of centroids. The final outcome of this study involves determining *whether a query is worth executing based on user criteria*. As of that reason, we define sets of user criteria, that define what score a query should have in order to be worth executing. We can then examine how successfully our predicted scores can determine whether a query should be executed or not, i.e., our initial research hypothesis. Before elaborating on our motivation, objectives, novelty and related work discussions, we provide the fundamental definitions to establish the narrative of our predictive intelligence approach. We follow the notation used in the previous research approaches [2, 3] to avoid confusion to the reader with different mathematical symbols and notation. Consider a d-dimensional data space $\mathbf{x} = [x_1, \ldots, x_d] \in \mathbb{R}^d$:

Definition 1 (Exploration Query) Let a d-dimensional box be defined by two boundary vectors $[l_1, \ldots, l_d]^\top$ and $[u_1, \ldots, u_d]^\top$ such that $l_i \leq u_i, \forall i$. An exploration query, $\mathbf{q} \in \mathbb{R}^{2d}$, is defined by the $2d$-dimensional vector $[l_1, u_1, \ldots, l_d, u_d]^\top$.

Definition 2 (Query Distance) The Manhattan (L_1 norm) distance between two queries $\mathbf{q}$ and $\mathbf{q}'$ is $\|\mathbf{q} - \mathbf{q}'\|_1 = \sum_{i=1}^{d} |l_i - l'_i| + |u_i - u'_i|$. This distance metric is subject to change to adapt to different types of datasets and/or correlation among dimensions by adopting the Mahalanobis distance.

Definition 3 (Score) Given a query $\mathbf{q}$ and a dataset B that consists of data points $\mathbf{x} \in \mathbb{R}^d$, $y \in \mathbb{N}$ is the score or the cardinality of the subset containing those $\mathbf{x} \in B$ being in the interior of the hyper-rectangle defined by the query $\mathbf{q}$ that matches $l_i \leq x_i \leq u_i, \forall i$.

The structure of the paper is as follows: Section 2 focuses on the motivation behind this research along with the related work and our contribution. Section 3 introduces our methodology, while Section 4 reports on a comprehensive experimental evaluation and comparative assessment with other works found in the literature. Section 5 concludes the paper with suggestions for future work. Refer to Table 1 in the Appendix for a nomenclature of the parameters/notations used in this paper.

2 Related work & contribution

2.1 Motivation

Advances in Approximate Query Processing (AQP) involve the use of sampling, histograms, self-tuning histograms, wavelets, and sketches [1, 8, 12, 13, 22]. AQP structures are based on the fact that the dataset on is always available and accessible in a distributed computing environment, when required by analysts to be explored. Data analysis based on histograms and sampling requires an extensive execution of queries over subsets of datasets to acquire a score prediction. As we mentioned, the execution of such queries in distributed computing environments has certain obstacles due to the lack of knowledge about the underlying data distribution. These obstacles can complicate the task of finding *what query should be executed*; which can lead to executing queries with unsatisfiable scores. This will inevitably lead to the waste of network, storage and financial resources as well as time. Furthermore, there are other scenarios that can make the importance of finding the right query fast, with less query executions, more significant. Certain real life data management systems can allow only up to a certain number of query executions and/or charge any excesses. For instance, web interface queries can involve per-IP limits while API based queries can involve per developer key limits. In either of the mentioned scenarios, consequences will either involve further waste of resources or the risk of being denied to execute further queries. We claim that the aforesaid risks could be avoided if the scores of queries were predicted in advance; as predicted scores would be used to determine if the queries are worth executing based on user criteria. We expect this to reduce the number of queries that are executed, but the degree to which this number is decreased will depend on the accuracy of our prediction methodology.

While taking the above-mentioned arguments into account, we present a list of requirements & desiderata in order to develop a rationale for predicting the scores of queries:

- **R1**: The rationale must avoid constant data-accesses / interactions with the distributed dataset, in order to avoid wasting computational and communication resources as well as making the provision of a prediction fast.
- **R2**: The rationale must provide score predictions that are accurate and reliable enough to determine if a query is worth executing.
- **R3**: The rationale must be applicable to datasets that consist of multidimensional data points.

A possible baseline solution would involve storing locally AQP structures and using them to predict the scores of unseen queries or such structures could be held by the distributed nodes, who would receive queries and respond with a prediction. The authors in [2, 3] presented a query-driven solution to this problem that involves training an ML model with a set of `COUNT` queries and their results, and using it to predict the results of new / unseen queries. Our solution is based on the query-driven initiative and involves the adoption of query vectorial space clustering, where centroids are formed with a training set of exploration queries. The score prediction is associated with the closest cluster-head to an incoming/unseen query $\mathbf{q}$ are going to work as the basis for making a score prediction. Notably, we have noticed via experimentation in our comparative assessment section that through the query-driven methodology in the literature [2, 3], the closest cluster-head to a query (Definition 2) has not always had a better prediction than its *rival*, the second closest centroid! In fact, we propose a methodology to refine and locate the two closest cluster-head to each query $\mathbf{q}$, from a set of queries, and noticed that in approximately 60% of the predictions, the best prediction derived from $\mathbf{q}$'s closest centroid, while in the remaining 40% it was obtained from its second closest cluster-head. This acted as our major motivator on how the existing query-driven approach should be further investigated to provide *robust and reliable predictions*.

2.2 Related work & contribution

Queries executed over distributed data for exploration purposes result to a huge amount of data points to be further investigated. As of that reason, ways of narrowing down query results by presenting only the top-k results based on a scoring function through joining and aggregating multiple inputs, were introduced in [6, 10, 15]. Other research approaches like [5, 16, 17] and [7] face with minimizing the number of relevant data points retrieved. Undeniably, research focuses on the significance of query's answer amount of retrieved data points, which challenges us to whether *we can decide if a query is worth executing based on the number of results it can return, more relevant*.

Histograms is a fundamental tool for a fast estimation of the number of data points retrieved per aggregate and/or exploration query. However, histograms to be able to accurately provided us this information/estimation they should be updated at times; and this does not benefit a viable solution in a distributed computing environment. Moreover, other score prediction methods adopt histograms, but they are still not applicable for our solution due to their requirement for data access; the reader could refer to wavelets [20, 22]. Further approaches that have been studied for solving the score prediction problem include sampling [13] and sketching [8]. However, these methodologies are not suitable in our context since they require all the underlying data to construct predictors and approximation models, which is not a viable solution in the case where the data are distributed in different geo-locations. As we already mentioned, in this paper we depart from our recent research on the query-driven ML approach of the relevant research approaches in [3] and [2] adopt clustering of queries (using variants of [14, 18] clustering methods) in order to predict the score. This is achieved by estimating certain query representative or cluster-heads, used further to predicting the score of an unseen query. However, as it will be shown in our experimental and comparative assessment section, the use of a unique and the closest cluster-head to an unseen query is not a panacea for an accurate and robust score prediction. In this case, we drastically depart from this and define methods for *refining* the clustering setting over the query space to increase reliability ad robustness of the score prediction for any random query in a distributed computing system.

The contribution and novelty of our research is:

- We introduce the idea of *centroid or cluster-head refinement*, which aims to increase the reliability of a query's closest cluster-head, as we believe that the closest cluster-head to a query should have the closest prediction. We introduce different refinement approaches and through the use of experiments we derive which one is the most effective.
- We present and examine three different approaches on how cluster-head can be used to make predictions for a query **q**. The *average prediction error*, as well as the *specificity* and *sensitivity* metrics, act as guides to help us determine which prediction approach is the most effective.
- We provide separate predictions using both refined and unrefined cluster-head to determine whether improved reliability leads to more accurate predictions, based on prediction error.
- We provide a comparative assessment with the previous approaches [3] and [2] found in the literature, and showcase the benefits of our methodology in providing robust and reliable predictions.
- Throughout our experiments, we always take into account how the behaviour of our methodologies can be influenced by the number of cluster-head.

3 Predictive intelligence methodology

3.1 Rationale

We introduce our methodology on how the scores of queries can be predicted with the use of clustering as well as the rationale for involving cluster-head refinement in the process. Recall that our study focuses on queries that are examining multivariate data, therefore a dataset includes scalar values in all dimensions. As in our first step we issue random queries to be executed against the dataset, we scale all values of the dataset in $\mathbf{x} = [x_1, \ldots, x_d] \in [0, 1]^d$. We then generate a random query set that consists of exploration queries of (l_i, u_i) pairs for each dimension $i = 1, \ldots, d$; see Definition 1. In order for a data point $\mathbf{x} = [x_1, \ldots, x_d]$ to match a query $\mathbf{q} = [l_1, u_1, \ldots, l_d, u_d]$, each x_i value that is being referred to by a (l_i, u_i) pair has to be: $l_i \leq x_i \leq u_i$.

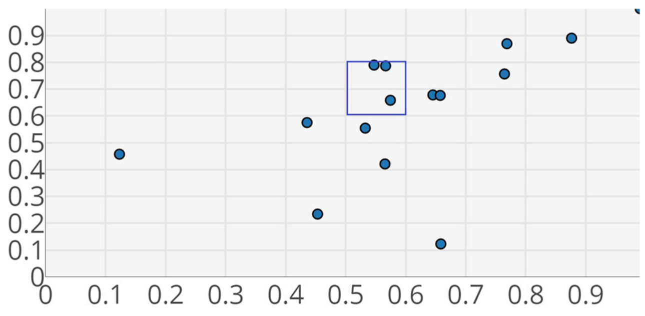


Fig. 2 An example exploration query projected onto the 2-dim. data space that consists of a (*min,max*) pair in each dimension

Each time a point matches a query, it will cause the score of that query **q** to be increased by 1. For example, in Fig. 2 the query [0.5, 0.6, 0.6, 0.8], represented by the square, has a score of 3, i.e., there are three data points *included* in this square.

Once the random query set is generated, it executes each query **q** against the dataset to obtain its score, $\mathbf{q}(y)$ or simply $y \geq 0$. The score is associated with each query, thus a *scored* query **q** vector is expanded at this point to accommodate y obtaining the following representation:

$$\mathbf{q} = [l_1, u_1, \ldots, l_d, u_d, y] \in \mathbb{R}^{2d+1} \tag{1}$$

The random query set is usually divided into two sets, 60% of the random query set becomes a training set while the remaining 40% becomes a testing set. The training set is the input to the k-means clustering algorithm which produces the k centroid, coined here as query representative patterns:

$$\mathbf{w}_k = [l_{k1}, u_{k1}, \ldots, l_{kd}, u_{kd}, y_k] \in \mathbb{R}^{2d+1}. \tag{2}$$

Different sets of centroids are created at different values of k, in order to examine changes in the behaviour of our refinement and prediction approaches based on the numbers of centroids. For each query **q** in the testing set, we find its two closest centroids. The distance between **q** and a centroid is measured through the Manhattan distance between the values of the (l_i, u_i) pairs. We refer to these two centroids as the **winner representative** and the **rival representative**, where the former is the closest centroid to **q**. Specifically, given a query **q**, the winner (closest) representative **w** (closest) out of the k representatives $\mathcal{W} = \{\mathbf{w}_i\}_{i=1}^k$ is defined as:

$$\mathbf{w} = \arg \min_{\mathbf{w}_i \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}_i\|_1, \tag{3}$$

where the Manhattan (L1 norm) distance is defined as:

$$\|\mathbf{q} - \mathbf{w}_k\|_1 = \sum_{i=1}^{d} |l_i - l_{ki}| + |u_i - u_{ki}|. \tag{4}$$

The rival representative **r** (2nd closest) is defined as:

$$\mathbf{r} = \arg \min_{\mathbf{w}_i \in \{\mathcal{W} \setminus \mathbf{w}\}} \|\mathbf{q} - \mathbf{w}_i\|_1, \tag{5}$$

i.e., after excluding the winner (closest) representative **w** of the $\mathcal{W}$ set, the rival representative is the second closest representative to the query **q**.

We subsequently measure the score prediction error for each of the two representatives (closest/winner and second closest/rival), where that is the absolute difference between a representative's score and **q**'s score. That is, the score prediction error w.r.t. winner representative is defined as:

$$e_{\mathbf{w}} = |y - \hat{y}_{\mathbf{w}}|, \tag{6}$$

where y is the actual score of the query **q** and $\hat{y}_{\mathbf{w}}$ is the winner representative's score $y_{\mathbf{w}}$ of the winner representative **w**. Similarly, the the score prediction error w.r.t. rival representative is defined as:

$$e_{\mathbf{r}} = |y - \hat{y}_{\mathbf{r}}|, \tag{7}$$

where $\hat{y}_{\mathbf{r}}$ is the rival representative's score $y_{\mathbf{r}}$ of the rival representative **r**.

The winner and rival representative along with their score prediction errors expand the vectorial representation of the scored query vector **q**, thus **q** now has the holistic vectorial representation:

$$\mathbf{q} = [l_1, u_1, \ldots, l_d, u_d,\ y, \mathbf{w}, e_{\mathbf{w}}, \mathbf{r}, e_{\mathbf{r}}], \tag{8}$$

where **w** is the winner representative with its associated score prediction score $e_{\mathbf{w}}$, and **r** is the rival representative with its associated score prediction $e_{\mathbf{r}}$.

Given a query **q** with actual score y, we define the indicator $I_{\mathbf{q}} = 1$ if the rival prediction error is less than the winner prediction error w.r.t. query **q**; 0 otherwise, i.e., $I_{\mathbf{q}} = 1$ if $e_{\mathbf{r}} < e_{\mathbf{w}}$, while $I_{\mathbf{q}} = 0$ if $e_{\mathbf{r}} \geq e_{\mathbf{w}}$. Let f_0 and f_1 be the percentage of cases (out of N predictions) that the rival representative has smaller prediction error than the winner representative, and vice versa, respectively, i.e.,

$$f_1 = \frac{1}{N} \sum_{i=1}^{N} I_{\mathbf{q}_i} \text{ and } f_0 = 1 - f_1 \tag{9}$$

We have noticed that in a huge number of cases $N > 10,000$, the rival representative has had a lower score prediction error than the winner representative about 40%, i.e., $f_1 = 0.4$, while $f_0 = 0.6$. This has lead to our study to introduce our rationale: centroid refinement, i.e., optimally updating the positions of the rival and winner representative such that $f_0 \rightarrow 1$ and $f_1 \rightarrow 0$. This means that, we cater for updating the representatives such that the winner representative will highly likely provide more accurate predictions that its rival. The algorithm for estimating the probabilities f_0 and f_1 of prediction for the winner and rival query representatives, respectively, is shown in Algorithm 1.

Algorithm 1 Winner & rival prediction probabilities.

```
Result: prediction probabilities $f_0$; $f_1$
$f_0 \leftarrow 0; f_1 \leftarrow 0; t \leftarrow 1$;
while $t \leq N$ do
    Receive random query $\mathbf{q}$ with score $y$;
    Winner query representative
    $\mathbf{w} = \arg\min_{\mathbf{w}_i \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Rival query representative
    $\mathbf{r} = \arg\min_{\mathbf{w}_i \in \{\mathcal{W} \setminus \mathbf{w}\}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Winner's predicted score $\hat{y}_{\mathbf{w}}$;
    Rival's predicted score $\hat{y}_{\mathbf{r}}$;
    if $|y - \hat{y}_{\mathbf{w}}| < |y - \hat{y}_{\mathbf{r}}|$ then
        $f_0 \leftarrow f_0 + 1$;
    else
        $f_1 \leftarrow f_1 + 1$;
    end
end
$f_0 \leftarrow \frac{f_0}{N}$; $f_1 \leftarrow \frac{f_1}{N}$;
```

3.2 Centroid refinement mechanism

3.2.1 Refinement methods

The centroid refinement aims to increase the *reliability* of the winner representative in the score prediction. In other words, the centroid refinement aims to decrease the cases where the winner representative has a higher score prediction error than the rival representative. The reason behind this is that, we will be basing our score predictions for new sets of queries on the values of representative centroids. We argue that the closest centroid to a query should have the closest score prediction, therefore by improving reliability we expect also to improve predictability. Although, the use of the appropriate experiments shall determine whether that statement holds as will be shown later.

In order to initiate centroid refinement, a new random query set must be generated called the **refinement set**. This refinement process involves **penalty** and **reward** formulas on the winner and rival representatives. The reward formula, shown below, aims to *shift* the representative vector **w** closer to the query **q**, where **q** is a query from the refinement set, i.e.,

$$\mathbf{w} = \mathbf{w} + a(\mathbf{q}) \cdot (\mathbf{q} - \mathbf{w}). \tag{10}$$

The parameter $a(\mathbf{q}) \in (0, 1)$ is a query-driven parameter that determines the *rewarding* effect; if $a = 1$ then the representative would take the exact same form as **q**, while if $a = 0$ the representative would remain unaltered. The penalty formula behaves in exactly the opposite manner, it aims to *shift* a representative further from the $\mathbf{q}$:

$$\mathbf{w} = \mathbf{w} - a(\mathbf{q}) \cdot (\mathbf{q} - \mathbf{w}). \tag{11}$$

We introduce three different methods for acquiring a rewarding value of a defined below. All of the three rewarding methods will be examined to determine which leads to the *best refinement*. Given a query $\mathbf{q}$:

Rewarding Method A suggests a be a scalar independent of the query $\mathbf{q}$.

Rewarding Method B includes two functions for a. The first function is a rewarding a value depending on the winner representative score prediction error. That is, the greater the winner score error is, the greater the reward effect will be:

$$a(\mathbf{q}) = 1 - e^{\frac{e_{\mathbf{w}}}{e_{\mathbf{w}}+e_{\mathbf{r}}}} \tag{12}$$

where $e_{\mathbf{w}} = |y - \hat{y}_{\mathbf{w}}|$ is the absolute error difference between the $\mathbf{q}$'s score and the winner representative's score and $e_{\mathbf{r}} = |y - \hat{y}_{\mathbf{r}}|$ is the absolute error difference between $\mathbf{q}$'s score and the rival representative's score.

The second function is a penalizing a value depending on the rival score prediction error, such that the greater the rival score error is, the greater the penalty effect will be:

$$a(\mathbf{q}) = 1 - e^{\frac{e_{\mathbf{r}}}{e_{\mathbf{w}}+e_{\mathbf{r}}}} \tag{13}$$

Rewarding Method C suggests that a depends on $n_{\mathbf{w}} > 1$, where $n_{\mathbf{w}}$ is the number of the query members of a representative's cluster. Specifically, let $\mathcal{Q} = \{\mathbf{q}_m\}_{m=1}^{M}$ be the set of all issued queries over the dataset $\mathcal{B}$. The query set $\mathcal{Q}$ is split into $k > 0$ groups $\mathcal{Q}_m$ of queries, where each group is associated with a query representative $\mathbf{w}_m, m = 1, \ldots, k$. A query $\mathbf{q}$ belongs to a query group $\mathcal{Q}_m$ iff the distances of this query $\mathbf{q}$ with the corresponding query representative $\mathbf{w}_m$ is the smallest compared to all the distances of $\mathbf{q}$ with the other query representatives. Thus, we define the cardinality of a query group $n_{\mathbf{w}_m}$ represented by the query representative $\mathbf{w}_m$ as:

$$n_{\mathbf{w}_m} = |\mathcal{Q}_m|, \tag{14}$$

and therefore $M = \sum_{m=1}^{k} n_{\mathbf{w}_m}$. Based on this cardinality, we obtain that, given a query $\mathbf{q}$ whose winner representative is $\mathbf{w}$, the rewarding value $a(\mathbf{q})$ is a reciprocal function of the winner's group cardinality:

$$a(\mathbf{q}) = \frac{1}{n_{\mathbf{w}} + 1}. \tag{15}$$

The refinement mechanism makes use of the penalties and rewards to refine centroids incrementally. In this context, we introduce three refinement approaches that will be examined to determine our final refinement mechanism. We then eventually choose the approach that *maximizes the reliability of the winner representative*. We also test how the different refinement approaches behave with the three methods for acquiring the value of a as well as how the refinement's effect can vary with different numbers of centroids.

During the refinement phase, we visit every query $\mathbf{q}$ from the refinement set and determine whether the score prediction error of the winner representative is higher than the rival representative's. If it is found that the winner representative has the lowest score error then the reward formula is imposed on each winner representative (including its score), while the rival representative remains unaltered. If it is found that the rival representative has a lower score error than the winner representative, then the three candidate refinement approaches are taken into consideration:

– **Refinement Approach 1** suggests imposing the reward formula on each rival representative attribute including its score, while the winner representative remains unaltered.
– **Refinement Approach 2** suggests imposing the reward formula on each rival representative attribute including its score, while also imposing the penalty formula on each winner representative attribute excluding its score.
– **Refinement Approach 3** is similar with Refinement Approach 2, but only in this case, the penalty formula is also imposed on the winner representative's score. It shall be noted though that in this case a winner representative's score might suffer from too many penalties and cause its score to be constantly dropping. If it keeps dropping, it might drop below zero where it is impossible for a query to return a negative number of data points. In these cases, the score will be held at zero value.

As the refinement set is entirely random, it can be assumed that there could be queries within our set that *worsen* the refinement process. That is to say, there could be queries that lead to imposing too many penalties or rewards on a specific representative and as a result, refinement could not be working to its full potential. As of that reason, instead of generating a single refinement set, we generate a series of random refinement sets. We refer to this process as the **refinement cycles** methodology. During the first refinement cycle, a refinement set refines the initial centroids and in every succeeding refinement cycle, a refinement set refines the resulting centroids from the previous refinement set. For each of those refinement cycles, we aim to pick the refinement set that maximizes reliability. This includes trying multiple refinement sets at each cycle and finally picking the one with the best performance. This will help us determine the full potential of refinement. The algorithm for the centroid refinement is shown in Algorithm 2.

Algorithm 2 Centroid refinement.

```
Result: Refined centroids in $\mathcal{W}$
while TRUE do
    Receive random query $\mathbf{q}$ with score $y$;
    Find winner representative
    $\mathbf{w} = \arg\min_{\mathbf{w}_i \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Update winner cardinality index $n_\mathbf{w} \leftarrow n_\mathbf{w} + 1$;
    Find rival representative
    $\mathbf{r} = \arg\min_{\mathbf{w}_i \in \{\mathcal{W} \setminus \mathbf{w}\}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Set winner's and rival's predicted scores $\hat{y}_\mathbf{w}$ and $\hat{y}_\mathbf{r}$;
    Set winner and rival prediction errors:
    $e_\mathbf{w} \leftarrow |y - \hat{y}_\mathbf{w}|$; $e_\mathbf{r} \leftarrow |y - \hat{y}_\mathbf{r}|$;
    Update reward/penalty factor $a(\mathbf{q})$ based on $e_\mathbf{w}$ and
    $e_\mathbf{r}$ (see Rewarding Methods A, B, C);
    if $e_\mathbf{w} < e_\mathbf{r}$ then
        Reward winner: $\mathbf{w} \leftarrow \mathbf{w} + a(\mathbf{q})(\mathbf{q} - \mathbf{w})$;
        Penalize rival: $\mathbf{r} \leftarrow \mathbf{r} - a(\mathbf{q})(\mathbf{q} - \mathbf{r})$;
    else
        Reward rival: $\mathbf{r} \leftarrow \mathbf{r} + a(\mathbf{q})(\mathbf{q} - \mathbf{r})$;
        Penalize winner: $\mathbf{w} \leftarrow \mathbf{w} - a(\mathbf{q})(\mathbf{q} - \mathbf{w})$;
    end
end
```

3.2.2 Refinement evaluation metrics

We measure the effectiveness of our refinement techniques by creating a new random set of queries, **the test refinement set**, to obtain the evaluation metrics:

- **B.R.** (Before Refinement) percentage is defined as the percentage representing the number of cases in the test refinement set where there is a higher winner representative score error, using the **unrefined** centroids.
- **A.R.** (After Refinement) percentage is defined as the percentage representing the number of cases in the test refinement set where there was a higher winner representative score error, using the **refined** centroids.
- **IMP** (Improvement) due to refinement is defined as:

$$\mathbf{IMP{=}B.R. - A.R.} \tag{16}$$

Once the appropriate experiments have been conducted, we are now able to determine which refinement approach along with which method for acquiring the value of a should be used for centroid refinement. We can then use these refined centroids to make predictions for the scores of queries.

3.3 Query score prediction mechanism

3.3.1 Prediction methods

In order to carry out score prediction, a query set is required where for each query $\mathbf{q}$ the two representatives and the actual score are known. We provide three prediction approaches to be taken into account:

- **Prediction Approach 1**: the predicted score of $\mathbf{q}$ is the score of its winner representative:

$$\hat{y} = \mathbf{w}.y \tag{17}$$

  where $\hat{y}$ is the predicted score of $\mathbf{q}$ and $\mathbf{w}.y$ is the score of the winner representative $\mathbf{w}$.
- **Prediction Approach 2**: the predicted score of $\mathbf{q}$ is the result of the weighted sum of the scores of the two representatives:

$$\hat{y} = f_0 \cdot \mathbf{w}.y + f_1 \cdot \mathbf{r}.y \tag{18}$$

  where $\mathbf{r}.y$ is the score of the rival representative. The empirical probabilities $f_0, f_1 \in (0, 1)$ defined in (9) represent the portion of queries the winner representative had had a lower prediction score than the rival representative, and vice versa, respectively.
- **Prediction Approach 3**: In this stochastic approach, the predicted score is that of the rival representative with probability $f_1 \in [0, 1]$ (defined in Prediction Approach 2), otherwise it is the score of the winner representative, i.e.,

$$\hat{y} = \begin{cases} \mathbf{r}.y & \text{with probability } f_1 \\ \mathbf{w}.y & \text{with probability } f_0 = 1 - f_1 \end{cases} \tag{19}$$

The algorithm for the score prediction given the refined centroids is shown in Algorithm 3.

Algorithm 3 Score prediction w.r.t. refined centroids.

```
Result: Average absolute score prediction error $e$
$t \leftarrow 1$;
while $t \leq n$ do
    Receive random query $\mathbf{q}$ with actual score $y$;
    Find winner representative
    $\mathbf{w} = \arg\min_{\mathbf{w}_i \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Find rival representative
    $\mathbf{r} = \arg\min_{\mathbf{w}_i \in \{\mathcal{W} \setminus \mathbf{w}\}} \|\mathbf{q} - \mathbf{w}_i\|_1$;
    Prediction Approach 1: predicted score $\hat{y} \leftarrow \mathbf{w}.y$;
    Prediction Approach 2: predicted score
    $\hat{y} \leftarrow f_0 \cdot \mathbf{w}.y + f_1 \cdot \mathbf{r}.y$;
    Prediction Approach 3:;
    Get a uniformly distributed random number
    $p \sim U(0, 1)$;
    if $p \leq f_1$ then
        predicted score $\hat{y} \leftarrow \mathbf{r}.y$;
    else
        predicted score $\hat{y} \leftarrow \mathbf{w}.y$;
    end
    Prediction error $e \leftarrow e + |y - \hat{y}|$
end
$e \leftarrow \frac{e}{n}$
```

3.3.2 Predictability & reliability evaluation metrics

The predicted score of $\mathbf{q}$ is kept in pairs with its actual score to allow an easy prediction error calculation. We can then calculate the average absolute prediction error e of a set of n queries:

$$e = \frac{1}{n}\sum_{i=1}^{n}(|\mathbf{q}_i.\hat{y} - \mathbf{q}_i.y|) \tag{20}$$

We explore the average prediction errors of our three prediction approaches at different values of k (the number of centroids). We concluded that predictions get better as k increases until improvement ceased to exist.

Two separate prediction errors are obtained for each prediction approach that use the refined or unrefined centroids as the prediction basis. This allows us to determine whether *improved reliability also benefits predictability*. We can then determine which prediction approach at which k is the most effective, which will be the one that has the lowest average prediction error. We refer to the chosen prediction approach given a chosen k to answer the question:

Can we use the predicted score to determine whether it is worth executing a query?

In order to answer this, user criteria must be defined first to declare what the score of a query should be in order to be worth executing, for instance: $10 < score < 350$.

The sensitivity metric is used to determine how effective our prediction model is in determining whether a query is worth executing based on user criteria, where TP (True Positive) is the number of cases where the predicted scores correctly met user criteria, while P (True Positive + False Negative) is the number of cases where actual scores met user criteria:

$$\textbf{Sensitivity} = \frac{TP}{P}. \tag{21}$$

We determine how effective our prediction model is in determining whether a query is *not* worth executing based on user criteria through a specificity metric where: TN (True Negative) is the number of cases where the predicted scores correctly did not meet user criteria and N (True Negative + False Positive) is the number of cases where the actual scores did not meet user criteria:

$$\textbf{Specificity} = \frac{TN}{N} \tag{22}$$

4 Experimental evaluation & comparative assessment

4.1 System workflow

Before reporting on the experimental evaluation of our approach, we would like to consolidate all the intelligent components of the proposed methodology to demonstrate the system workflow. Upon recording predictive analytics queries and their corresponding scores (cardinality values), we first quantize the query-space into a set of centroids, representing the patterns of the issued queries. Then, the novel cluster-head refinement method is applied to increase the reliability of a query's closest cluster-head, such that the closest cluster-head of a given query should have the most accurate score prediction.

The refinements are achieved by investigating the statistical patterns of the winner and the rival cluster-heads, which are adjusted to minimize the prediction error. The adjustment mechanism adopts a reward-penalty methodology to fine tuning the positions of the winner and the rival cluster-head vectors in the query space in light of increasing the reliability in the score prediction and, at the same time, minimizing the prediction error.

Finally, the system hosting the refined cluster-heads is ready to provide reliable score predictions for any given query.

4.2 Dataset & experiment set-up

We conducted a series of experiments to answer the different questions that have been asked in our methodology section. In order to carry out these experiments, the use of a real life data sets and query work loads were required.

Dataset We adopted the data set of gas contextual sensor data in [21] and [19] publicly available from the UCI Machine Learning Repository.[1] The data set consists of 14,000 measurements from 16 chemical sensors exposed to 6 gases at different concentration levels, where there are 8 features for each chemical sensor. Our exploration queries focus on the measurements of the first two features of the first chemical sensor for ethanol.

Query workload We used the query workload set of size 1000 adopted from UCI ML Repository,[2] where each query

[1] https://archive.ics.uci.edu/ml/datasets/Gas+Sensor+Array+Drift+Dataset+at+Different+Concentrations

[2] http://archive.ics.uci.edu/ml/datasets/Query+Analytics+Workloads+Dataset

has two (min, max) pairs, and run it against the data set to obtain the score y of each query. Therefore the resulting training set was of size 600 while the testing set was of size 400.

Experimental set-up For the query space quantization, we experimented with the k-means algorithm with values $k \in \{7, 15, 22, 40\}$. As there are no known methods for acquiring the optimal k value in our specific query-driven clustering process, we decided to start off with small values of k to be able to monitor each centroid. These k values were also considered in the related work [3], which will be used for our comparative assessment.

In order to compare our model with the predictive models [3] and [2], we adopted n-fold cross-validation with $n = 10$. To check whether there is a statistically significant difference between the means of the score prediction accuracy values, we fixed a significance level $\alpha = 0.95$, where the reasoning is that if difference is significant at the $\alpha\%$ level, there is a $100 - \alpha\%$ chance that there really is a difference. We divided the significance level by two using the *two-tailed test*,[3] i.e., the significance level is set to $(1 - \alpha)/2 = 0.025$. In all the reported experiments and comparison, the derived probability values (p-values) were less that $(1 - \alpha)/2 = 0.025$ (with average $p-$value being 0.0127). This indicates that the comparison assessment of the score prediction accuracy is statistically significant.

4.3 Reliable refinement methodology & comparative assessment

We examine the behaviour of the three refinement methods of reward/penalty w.r.t. value of $a(\mathbf{q})$ given a query $\mathbf{q}$. We firstly report on the results of the Rewarding Methods A and B, as the method C is investigated later. We decided that an appropriate way to determine which the best method was, was by measuring the performance of each possible combination between Method A (or B) and the three Refinement Approaches using the same initial centroids.

For each of these tests, we initially created a new refinement set of 1500 queries and uses a method-refinement approach combination to produce a new set of refined centroids. A test refinement set of 400 queries is used to obtain the average BR, AR and IMP metrics that quantify the refinement's effect. For each refinement set, five testing refinement sets were produced and their BR, AR and IMP values were averaged. This process is repeated three times for each experiment with three different refinement sets. The average BR, AR and IMP values for an experiment are once again averaged.

Method A	B.R.	A.R	IMP
Refinement Approach 1	37.65%	30.15%	7.50%
Refinement Approach 2	37.42%	25.98%	11.44%
Refinement Approach 3	37.20%	30.67%	6.53%

Method B	B.R.	A.R.	IMP
Refinement Approach 1	37.15%	30.55%	6.60%
Refinement Approach 2	37.42%	32.90%	4.52%
Refinement Approach 3	37.30%	30.86%	6.44%

Fig. 3 Evaluation of the Refinement Approaches along with the rewarding/penalizing methods based on the refinement metrics BR, AR, and IMP

The results for Methods A and B for each refinement approach are provided in Fig. 3. Method A with $a(\mathbf{q}) = \mathbf{0.1}$ achieves the best performance in improving the reliability of the winner representative (according to IMP). Using Method A along with Refinement Approach 2 leads to the best refinement in all experiments. At that point, we concluded that Method A was the best choice for acquiring a value of a, but, we were still not convinced that Refinement Approach 2 was the best choice. We expect that the behavior of the refinement approaches will change when the number of centroids increased.

Based on this reasoning, we conducted a new series of experiments to determine which refinement approach is the best using Method A. In these experiments, four sets of centroids were explored corresponding to $k = 7$, $k = 15$, $k = 22$ and $k = 40$. It should be noted that, in all of these experiments, we ensure that the same initial sets of centroids were used for each k. The same methodology as previously was used for producing average BR, AR and IMP values. The results of these experiments are provided in Fig. 4. In most cases, the refinement's effect decreases as the size of k increases. The relationship between k and IMP (the refinement's effect) is further examined in Fig. 5. This inverse-like relationship applies for all refinement approaches at all sizes of k, except when $k = 15$ for the Refinement Approach 3, where the refinement seems to be most effective.

It may also be noticed that the lowest AR may belong to a different Refinement Approach at different k values, although at most cases the difference can be too small to be deemed as important. We consequently concluded that the Refinement Approach 2 would be our choice for refining

[3]Anderson, Dallas W., et al. "On Stratification, Grouping and Matching." Scandinavian Journal of Statistics, vol. 7, no. 2, 1980, pp. 61–66.

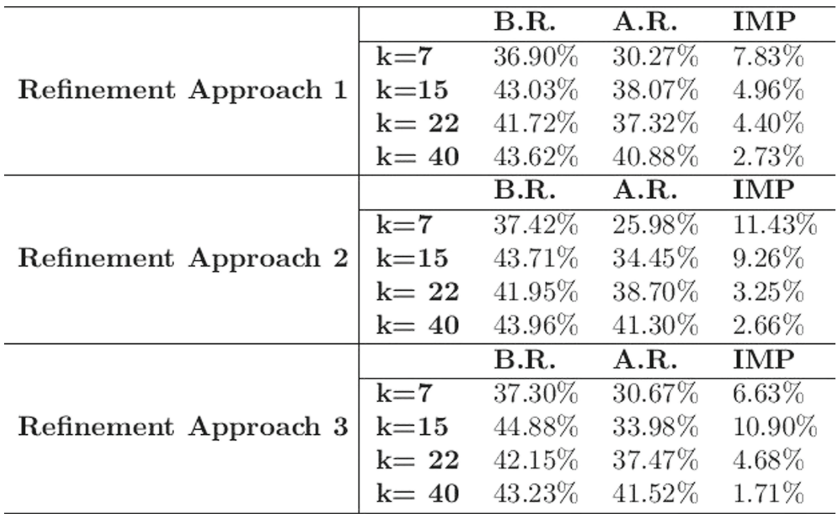


Fig. 4 Evaluation of the Refinement Approaches with different values of k using Rewarding Method A

		B.R.	A.R.	IMP
Refinement Approach 1	k=7	36.90%	30.27%	7.83%
	k=15	43.03%	38.07%	4.96%
	k= 22	41.72%	37.32%	4.40%
	k= 40	43.62%	40.88%	2.73%
		B.R.	A.R.	IMP
Refinement Approach 2	k=7	37.42%	25.98%	11.43%
	k=15	43.71%	34.45%	9.26%
	k= 22	41.95%	38.70%	3.25%
	k= 40	43.96%	41.30%	2.66%
		B.R.	A.R.	IMP
Refinement Approach 3	k=7	37.30%	30.67%	6.63%
	k=15	44.88%	33.98%	10.90%
	k= 22	42.15%	37.47%	4.68%
	k= 40	43.23%	41.52%	1.71%

centroids. We then considered that the method for acquiring a value of a can depend on the number of occupants in a representative's cluster [4], i.e., Method C. As a result of our decision making, experiments were conducted in the same manner as previously, but this time we used Refinement Approach 2 in combination with Method C.

We can see from Fig. 6 that Method C produces the lowest AR results in all cases. Additionally, when $k = 22$, despite the fact that Refinement Approach 2 with Method C produced the lowest AR, the IMP turns negative. This shows that the refinement function slightly decreased the reliability of the winner representative. This also indicates that there is a *limit* to how much we refine a set of centroids until reliability stops improving; this assumption will be further examined later. Similar results are obtained for Method C combined with refinement approaches 1 and 3.

In order to examine whether the centroid refinement method has improved the reliability of the winner representative, we provided our comparison results in the Figs. 7, 8, 9, and 10 with the previous approaches [3] and [2] using the query winner representative for prediction decision. Each of those comparison results represents the number of cases where the rival representative had a *better* prediction than the winner representative using the prediction approach in [3] and [2] (Red Bar) in comparison with one of our centroid refinement methods (Blue Bar) for four sets of centroids. We present the performance of our methodology using the Refinement Approaches 1,2 & 3 with Method A as well as Refinement Approach 2 with Method C.

We conclude by observing these results that our proposed centroid refinement offers *higher reliability* of the winner representative than the existing approaches [3] and [2].

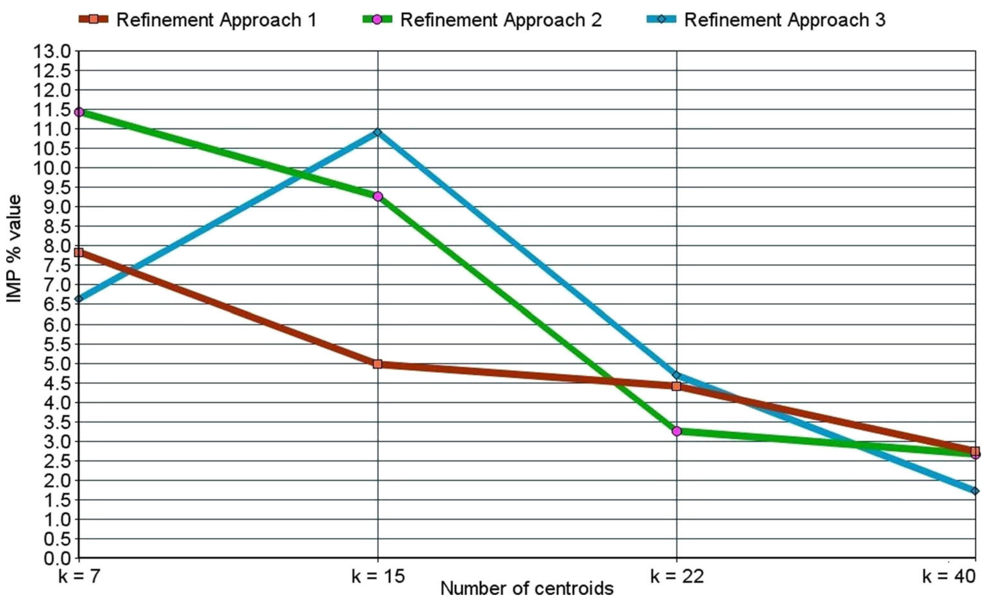
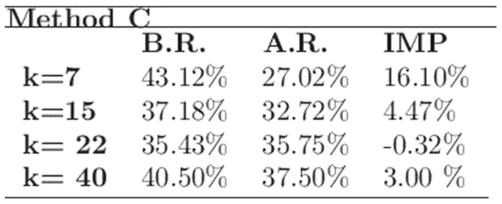


Fig. 5 Impact of number of centroids k on the IMP refinement metric for the three refinement approaches



Method C

	B.R.	A.R.	IMP
k=7	43.12%	27.02%	16.10%
k=15	37.18%	32.72%	4.47%
k= 22	35.43%	35.75%	-0.32%
k= 40	40.50%	37.50%	3.00 %

Fig. 6 Evaluation of the Refinement Approach 2 vs different values of k adopting Method C

Note: we obtain similar comparative assessment results with refinement approaches combined with rewarding methods, illustrating the capability of our methodology to provide accurate and reliable predictions after appropriate centroids refinement.

4.4 Refinement cycles & prediction reliability

We have discussed previously that the contents of a refinement set can be crucial to the refinement's effect on centroids. We suspected that as refinement sets are random, there can be queries within the set that can either improve or worsen the centroid refinement. Hence, we decided to present the idea of the *refinement cycles*.

During a refinement cycle, a new refinement set is generated and imposes the refinement function on the resulting centroids of the previous refinement cycle. This holds unless it is the first refinement cycle, where refinement will occur on the centroids that resulted from the k-means clustering. We keep introducing new refinement cycles as long as there was a positive IMP value; in other words, as long as there is improvement.

We initially created four different sets of centroids, where in all cases $k = 7$. During each refinement cycle, we create a refinement set of 500 queries to be used by the refinement function and a test refinement set of 400 queries to produce the values for BR, AR and IMP. The refinement function made use of the Refinement Approach 2 with Method A. At the end of each refinement cycle, we calculate the average values for BR, AR and IMP for the four sets of centroids. We then attempted ten different refinement sets during each cycle and eventually used the one that had the best performance in improving reliability. This way we ensured that at each cycle, we had the best refinement set we could find.

The results of these experiments are presented in Figs. 11 and 12. It can be seen in all four cases, that no further improvements in reliability could be made during the third refinement cycle as all of our IMP values became negative.

While also taking into account the IMP value when $k = 22$, we conclude that we can refine the centroids up to a certain point before the reliability of the winner representative starts getting worse. We can also see that the average AR of the second refinement cycle is close to the value we obtained 25.98%. Therefore, we deduce that refinement cycles won't lead to a drastic improvement in reliability.

In addition, we used 500 queries less to reach a similar AR value but also used another 500 queries to determine that the best AR has been reached in the previous cycle. As we ensured what the limits of refinement were, we proceeded to examining our predictions approaches while using the Refinement Approach 2 with Method C for centroid refinement. We have discussed three

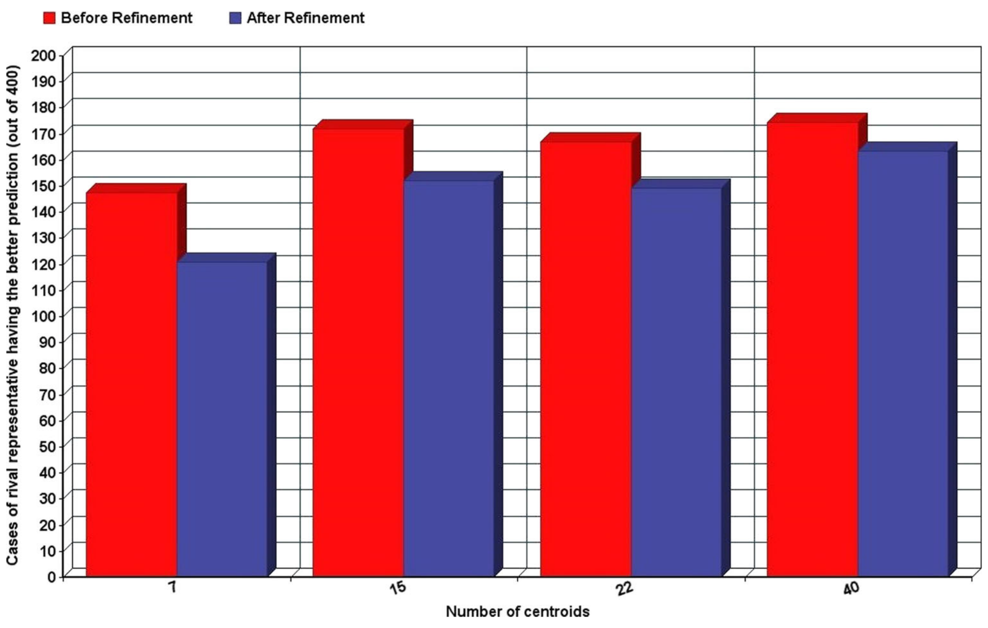


Fig. 7 Existing approaches [2, 3] (red bar) vs. our Refinement Approach 1 with Method A

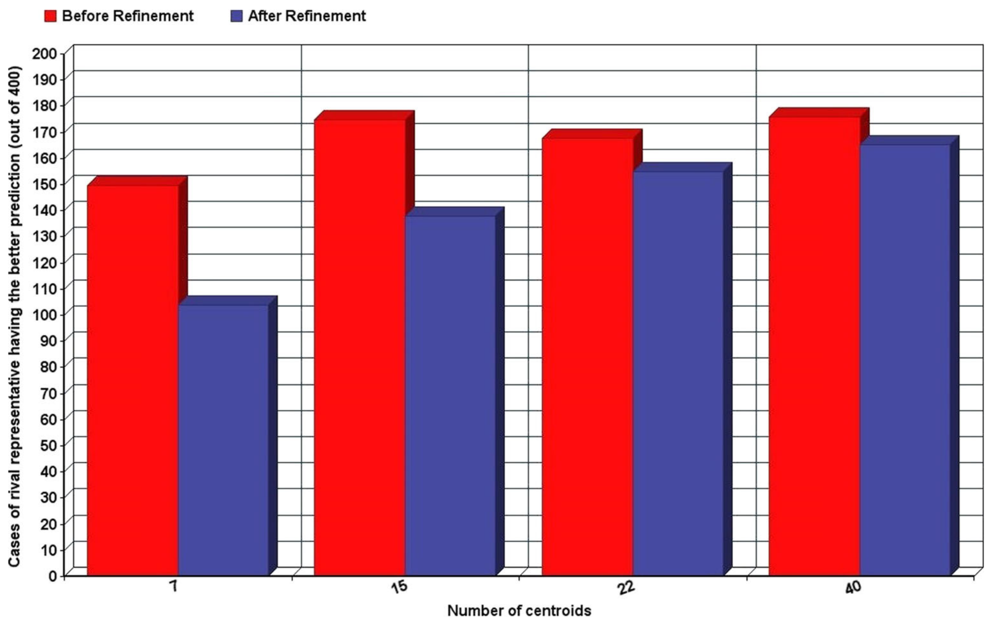


Fig. 8 Existing approaches [2, 3] (red bar) vs. our Refinement Approach 2 with Method A

different prediction approaches in our methodology that will all be examined in this subsection. The conducted experiments here did not only aim to find which is the best prediction approach, *but also* to determine whether improved reliability of the winner representative leads to better predictability. These results are shown in Fig. 13.

We further examined each prediction approach by making predictions for the scores of queries. We made separate predictions for queries that were derived from the three refinement sets, used during centroid refinement and their corresponding test refinement sets. The actual scores of these queries were in all cases already known. This allowed for the prediction errors to be calculated.

All the prediction errors were then used to produce the average absolute prediction error. We made separate predictions using both the Refined (R) and Unrefined (UR) centroids in order to compare their average prediction errors and derive whether centroid refinement has benefited our predictions. We ensured that for all refinement sets and test refinement sets, predictions were made with their corresponding centroids from centroid refinement.

The average prediction errors for unrefined centroids can be found in the columns labelled as UR, while the average prediction errors for refined centroids can be found in the columns labelled as R in Fig. 13. Therefore, for a given k, there are four average prediction error values for each

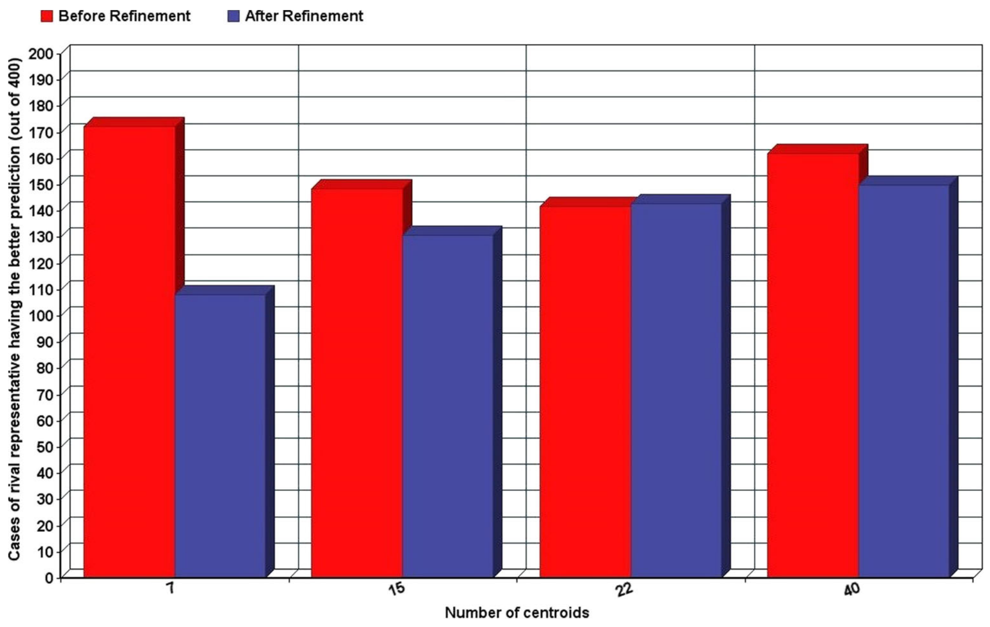


Fig. 9 Existing approaches [2, 3] (red bar) vs. our Refinement Approach 2 with Method C

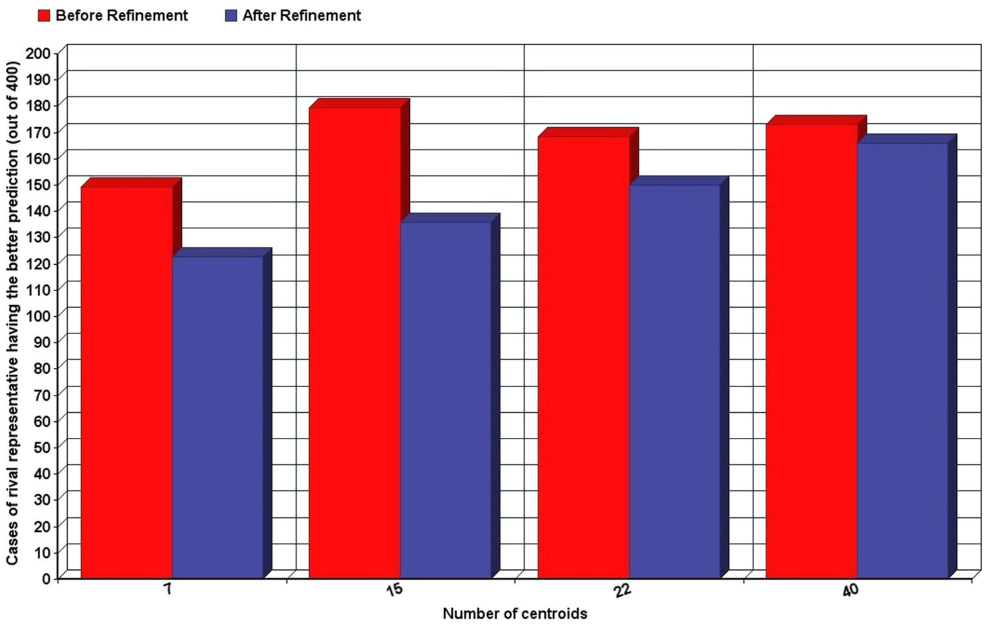
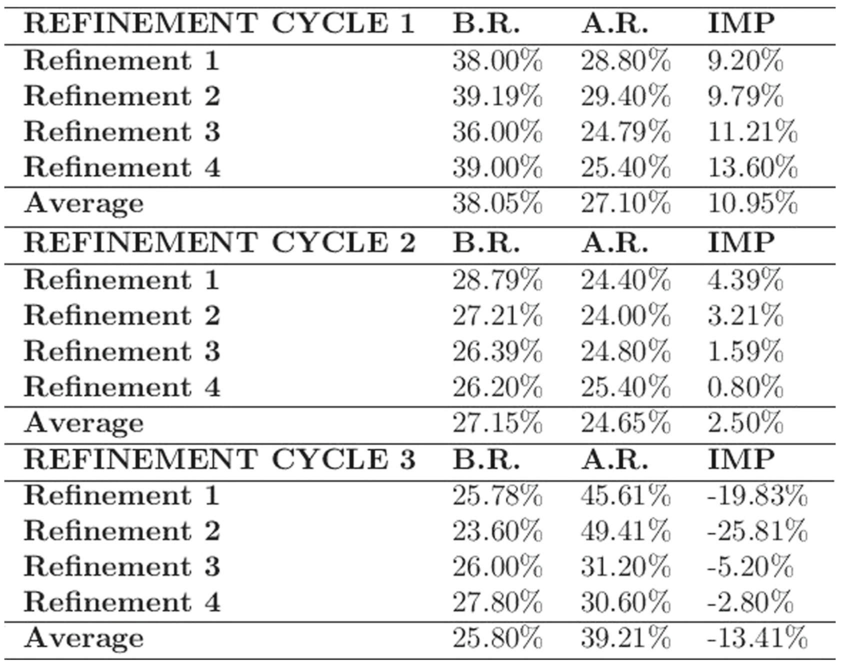


Fig. 10 Existing approaches [2, 3] (red bar) vs. our Refinement Approach 3 with Method A

prediction approach. Our initial tests involved three sets of centroids where $k = 7$, $k = 22$ and $k = 40$. We observed that for all prediction approaches, our average prediction errors tend to drop as the number of centroids was increasing. This lead us to conducting more experiments with higher k numbers, where $k = 85$, $k = 107$ and $k = 130$. There are a couple of key points that we can conclude from our experimental results.

Firstly, Prediction Approach 2 outperforms all other prediction approaches in all cases. The only case where this does not hold is when $k = 7$, for predictions made using the refined centroids for the test refinement set, where the average prediction error is the same with Prediction Approach 1. Hence, we can determine that using the scores of more than one representatives improves our predictions. We can assume at this point that our predictions can be further improved if the calculations included the scores of even more centroids that are close to a query and weighting them appropriately.

Secondly, improving the winner representative's reliability does not make predictions better at all cases. We can see from our results that the cases where predictions, made



REFINEMENT CYCLE 1	B.R.	A.R.	IMP
Refinement 1	38.00%	28.80%	9.20%
Refinement 2	39.19%	29.40%	9.79%
Refinement 3	36.00%	24.79%	11.21%
Refinement 4	39.00%	25.40%	13.60%
Average	38.05%	27.10%	10.95%
REFINEMENT CYCLE 2	**B.R.**	**A.R.**	**IMP**
Refinement 1	28.79%	24.40%	4.39%
Refinement 2	27.21%	24.00%	3.21%
Refinement 3	26.39%	24.80%	1.59%
Refinement 4	26.20%	25.40%	0.80%
Average	27.15%	24.65%	2.50%
REFINEMENT CYCLE 3	**B.R.**	**A.R.**	**IMP**
Refinement 1	25.78%	45.61%	-19.83%
Refinement 2	23.60%	49.41%	-25.81%
Refinement 3	26.00%	31.20%	-5.20%
Refinement 4	27.80%	30.60%	-2.80%
Average	25.80%	39.21%	-13.41%

Fig. 11 Evaluation of the Refinement Approach 2 at $k = 7$ using different refinement cycles

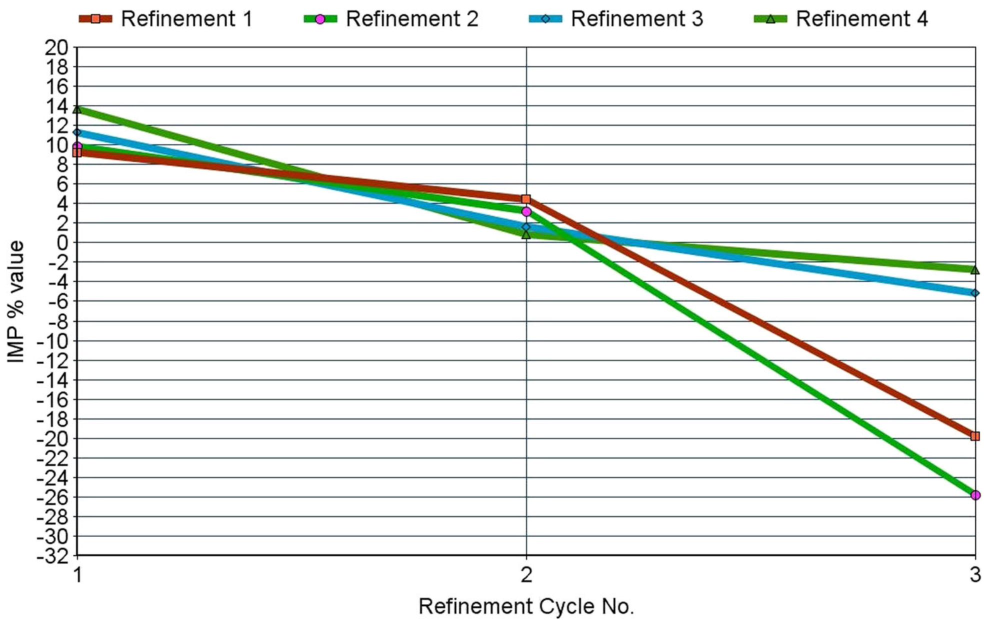


Fig. 12 Impact of the refinement cycles on IMP for $k = 7$

from refined centroids, are consistently better than the unrefined centroids (or at least as good), for all three prediction approaches, for all six sizes of k, are the predictions made for queries of the refinement sets. This comes as no big

Fig. 13 Average prediction errors for the three prediction approaches at different k over Refined (R) and Unrefined (UR) query centroids

k=7	UR	R	k=22	UR	R
Prediction Approach 1: Refinement Set	161.0	158.0	Prediction Approach 1: Refinement Set	134.0	134.0
Prediction Approach 1: Test Refinement Set	151.0	165.0	Prediction Approach 1: Test Refinement Set	148.5	150.0
Prediction Approach 2: Refinement Set	154.0	151.0	Prediction Approach 2: Refinement Set	128.0	125.5
Prediction Approach 2: Test Refinement Set	141.0	165.0	Prediction Approach 2: Test Refinement Set	131.5	132.5
Prediction Approach 3: Refinement Set	205.8	195.4	Prediction Approach 3: Refinement Set	181.3	175.8
Prediction Approach 3: Test Refinement Set	183.8	197.4	Prediction Approach 3: Test Refinement Set	184.7	177.4
k=40	**UR**	**R**	**k=85**	**UR**	**R**
Prediction Approach 1: Refinement Set	116.0	114.0	Prediction Approach 1: Refinement Set	112.0	99.0
Prediction Approach 1: Test Refinement Set	131.0	128.0	Prediction Approach 1: Test Refinement Set	112.0	111.5
Prediction Approach 2: Refinement Set	108.0	105.0	Prediction Approach 2: Refinement Set	110.0	96.0
Prediction Approach 2: Test Refinement Set	115.0	114.0	Prediction Approach 2: Test Refinement Set	102.5	97.0
Prediction Approach 3: Refinement Set	136.5	133.3	Prediction Approach 3: Refinement Set	130.2	125.6
Prediction Approach 3: Test Refinement Set	149.4	140.9	Prediction Approach 3: Test Refinement Set	129.2	127.0
k=107	**UR**	**R**	**k=130**	**UR**	**R**
Prediction Approach 1: Refinement Set	116.3	103.0	Prediction Approach 1: Refinement Set	121.0	107.0
Prediction Approach 1: Text Refinement Set	113.3	108.3	Prediction Approach 1: Test Refinement Set	118.0	131.0
Prediction Approach 2: Refinement Set	101.0	93.6	Prediction Approach 2: Refinement Set	98.0	94.6
Prediction Approach 2: Test Refinement Set	97.0	96.6	Prediction Approach 2: Test Refinement Set	96.0	105.6
Prediction Approach 3: Refinement Set	133.1	126.1	Prediction Approach 3: Refinement Set	131.6	125.0
Prediction Approach 3: Test Refinement Set	128.9	130.8	Prediction Approach 3: Test Refinement Set	131.7	126.9

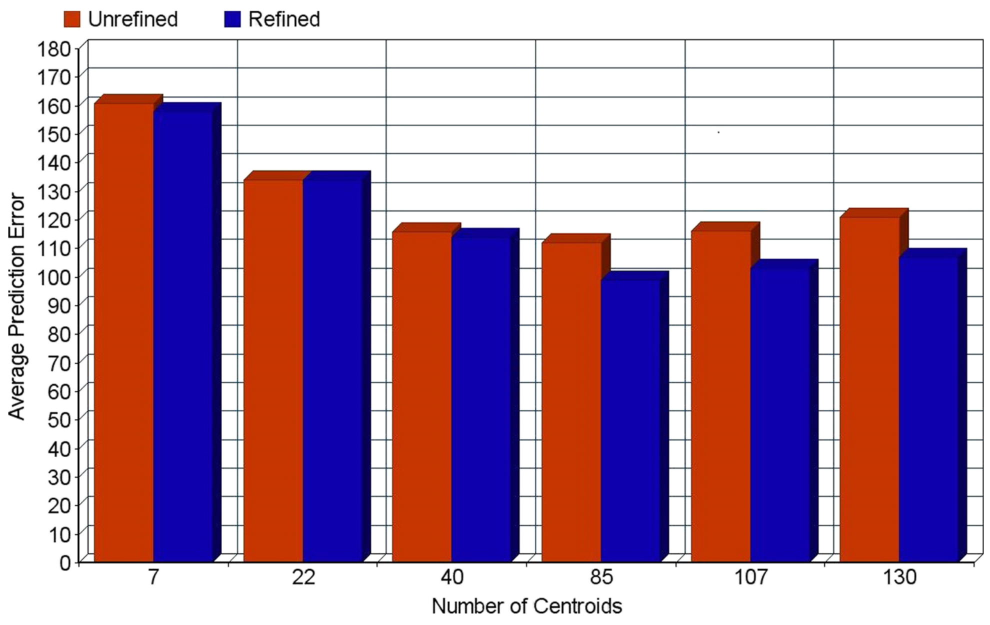


Fig. 14 Prediction Approach 1: Average prediction errors for the predicted scores of the refinement sets using the unrefined centroids vs the refined centroids at different *k*

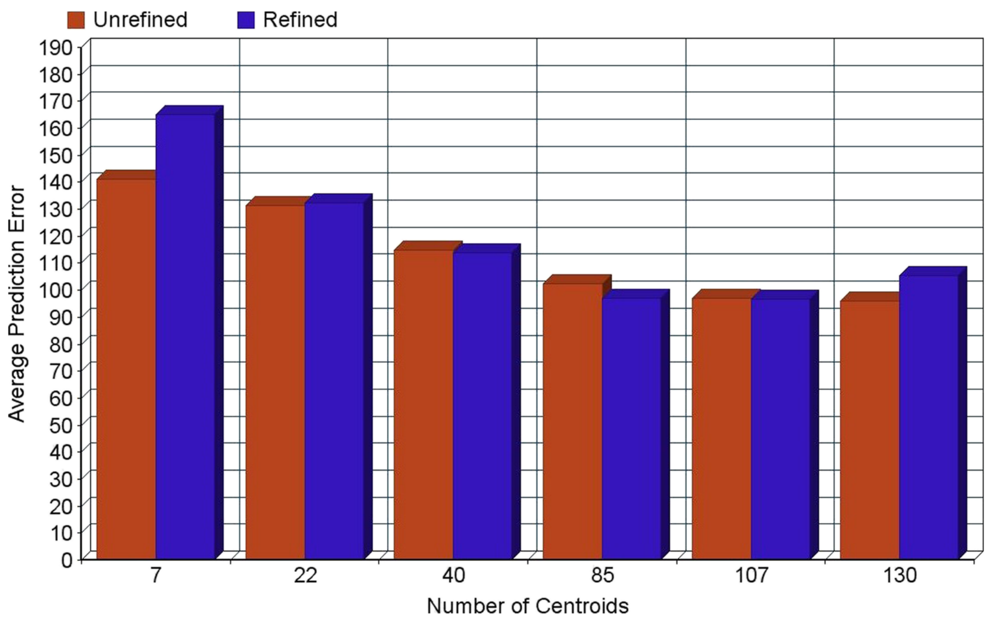


Fig. 15 Prediction Approach 2: Average prediction errors for the predicted scores of the test refinement sets using the unrefined centroids vs the refined centroids at different *k*

Fig. 16 The sensitivity for the predicted scores of the test refinement sets by the best performing prediction models for eight sets of user criteria

	(0,15)	**(0,50)**	**(0,100)**	**(50,100)**	**(200,300)**	**(100,300)**	**(100<)**	**(300<)**
k=85 R	68.19%	83.17%	85.10%	21.89%	34.27%	68.16%	86.70%	52.67%
k=107 UR	62.80%	80.98%	86.02%	40.29%	24.73%	58.04%	84.03%	70.14%
k=107 R	72.50%	80.59%	85.60%	20.88%	28.91%	72.62%	89.26%	64.27%
k=130 UR	71.36%	86.48%	88.26%	20.82%	20.03%	63.41%	86.32%	58.48%

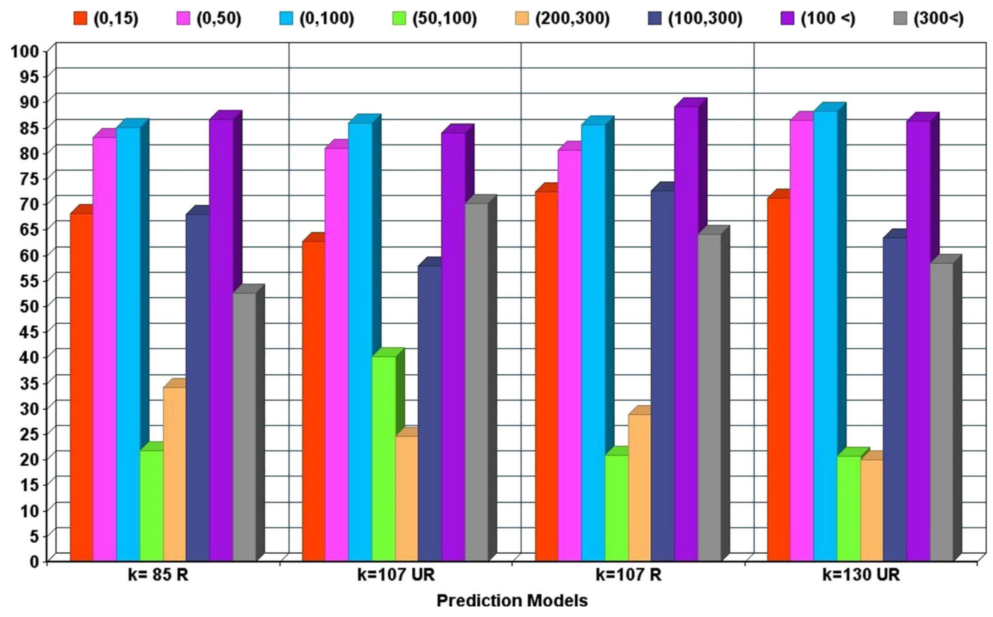


Fig. 17 Bar chart visualizing the sensitivity in Fig. 16

surprise of course, as they are the set that *re-adjusted* the centroids. Centroid refinement seemed to be most effective in Prediction Approaches 1 and 3, where the former has a better performance for $k \in \{85, 107, 130\}$, while the latter has a better performance for $k \in \{7, 22, 40\}$.

Figure 14 displays how score predictions from the refined centroids outperformed all score predictions from the unrefined centroids (using Prediction Approach 1); except in $k = 22$, where predictions from the unrefined and refined centroids had the exact same average prediction error. As that is the only case where there was no drop in the average prediction error, we can take a look at Fig. 6 and observe that $k = 22$ was the test where we actually had a small negative IMP value. In the rest of our k values, IMP was always positive and predictions were improved; we now provide the IMP values for $k = 80$, $k = 107$ and $k = 130$ respectively: 0.40%, 0.20% and 0.57%.

Thirdly, the test refinement set experiments are where we ideally like to see consistent improvement due to refinement, as these can act as indicators of how effective our rationale is in terms of generalization.In the cases of $k = 40$ and $k = 85$, lower average prediction errors were achieved due to centroid refinement in all prediction approaches, even if at some cases the improvement is not that big to be deemed significant. The comparison of the average prediction errors, between predictions made from unrefined and refined centroids using Prediction Approach 2 for the test refinement sets, is shown in Fig. 15.

In these cases, there are no obvious indicators on when refinement seems to help or at which prediction approaches. We can further support this claim by taking into account our BR, AR and IMP values. The greatest IMP value was noticed during $k = 7$ and the lowest at $k = 22$ (where IMP is actually negative). When we look at the corresponding case of Prediction Approach 2 at $k = 7$ in Figs. 13 and 15, the unrefined centroids offered the better predictions by a significant margin. While in the corresponding case for Prediction Approach 2 at $k = 22$, the refined centroids had an average prediction error of 132.5 while the unrefined centroids had an average prediction error of only 131.5.

Therefore our conclusions from the above analysis are:

- Firstly, the centroid refinement has the potential to always improve score predictions for queries in the refinement set. We expect to see the average prediction error decrease as long as there is a positive IMP value (see Fig. 14).
- Secondly, there does not seem to be a direct relationship between IMP and the average prediction error when predicting scores for the test refinement queries, as there are both cases where predictions got worse or better due to centroid refinement. Increasing the number of centroids to improve predictions works up to a certain extend. We can see from Fig. 13, that in most cases when we increased k, the average prediction error for the same test at a lower k would be higher.

Fig. 18 The specificity for the predicted scores of the test refinement sets by the best performing prediction models for eight sets of user criteria

	(0,15)	(0,50)	(0,100)	(50,100)	(200,300)	(100,300)	(100<)	(300<)
k=85 R	95.66%	91.45%	87.12%	92.89%	89.40%	79.53%	82.68%	95.84%
k=107 UR	93.04%	89.76%	85.99%	92.55%	93.23%	84.99%	86.98%	96.78%
k=107 R	97.15%	91.35%	88.28%	91.28%	93.52%	86.96%	88.01%	96.47%
k=130 UR	95.66%	91.45%	87.12%	92.84%	89.40%	79.54%	82.68%	95.84%

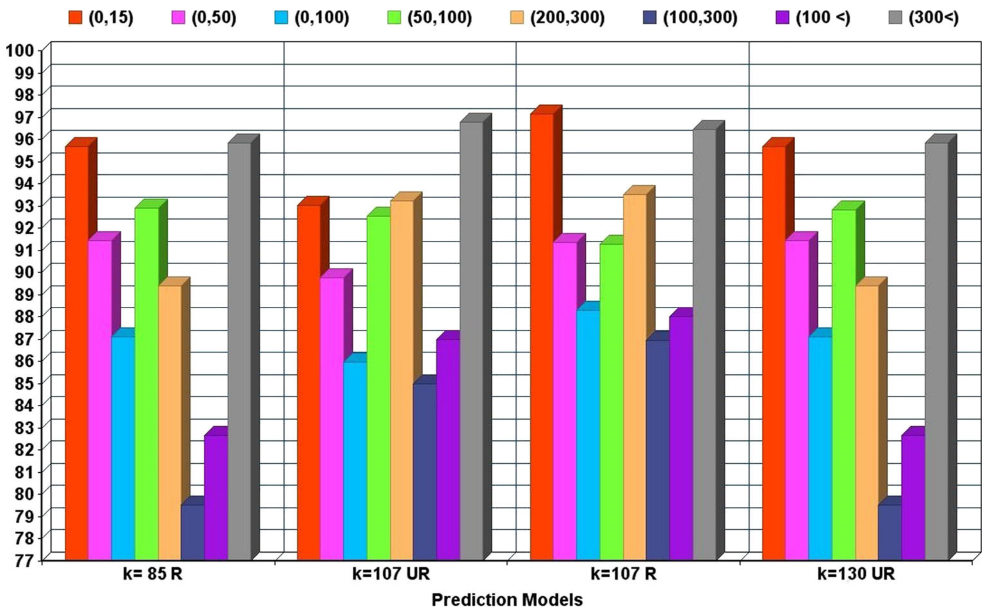


Fig. 19 Bar chart visualizing the table presented in Fig. 18

Despite that, the drop in the average prediction error, gets smaller every time we increase k. In fact, at certain cases the average prediction error can get worse as k increases, e.g., almost all results of Refinement Approach 1 from $k = 85$ to $k = 130$.

– We can also observe from our bar charts in Figs. 14 and 15 that the effect that k has on the average predictions errors becomes more subtle as k increases. We can safely conclude that predictions can get better at a specific range of high k values, but this range can be acquired only through experimentation, as k should differ between datasets and queries with different numbers of dimensions.

At the end of these experiments, we picked our best performing prediction models to investigate whether we could answer the question: *Can we use the predicted score of a query to determine if it is worth executing based on user criteria?* All of the selected prediction models made use of Prediction Approach 2 and were using the following sets of centroids as their prediction basis: $k = 85$ with UR, $k = 107$ with UR, $k = 107$ with R and $k = 130$ with UR. We decided to pick more than one model, as the average prediction errors of these four prediction models were too close one another to single one out.

4.5 Discussion on query execution

We handle the question: *is a query worth executing or not?* as a simple binary classification problem. We classify queries, based on user criteria, into queries that are worth executing (class label: 1) and queries that are not worth executing (class label: 0). In order to do that, we firstly had to define sets of user criteria, where these can be seen in the headings of columns of the tables displayed in Figs. 16 and 19. A set of user criteria in the format (c_1, c_2) declares that:

$$c_1 \leq y \leq c_2, \tag{23}$$

while a set of user criteria in the format $(c_1 <)$ declares that:

$$c_1 < y \tag{24}$$

For each prediction model, represented by a row in Figs. 16 and 18, we used the predicted scores for the queries of the test refinement sets (the same test refinement sets for which we can see their average prediction errors in Fig. 13) along with their actual scores, to measure sensitivity and specificity. Each sensitivity/specificity value displayed is an average from measuring the sensitivity/specificity of those test refinement sets. The sensitivity metric indicates the percentage of queries that we predicted are worth executing and indeed were, according to their actual score.

The sensitivity results can be seen in Fig. 16 while a visual representation of these tests is available through the bar chart in Fig. 19. The specificity metric indicates the percentage of queries that we predicted are not worth executing and indeed weren't according to their actual score. The specificity results can be seen in Fig. 18; and as previously, a visual representation of these results is provided in Fig. 19.

As we can recall from our previous tests in Fig. 13 the average prediction errors of the four prediction models were very similar, too similar to safely determine which one was better. As one may notice, there are plenty of similarities between the sensitivity and specificity measurements of prediction models for certain user criteria as well. We have drawn a couple of conclusions from these experiments that we are presenting below.

Sensitivity We notice the refinement's effect in the cases of $k = 107; UR$ and $k = 107; R$ for the tests $(50, 100)$ and $(100, 300)$. We can see that the UR model had a performance almost twice as good as the R model in the $(50, 100)$ test, while the R model outperformed the UR model in the $(100, 300)$ test by a significant amount. In order to determine the best performing prediction model, we have calculated the average sensitivity for each prediction model for the declared user criteria, these are (respectively as they appear in Fig. 16 from top to bottom): 62.52%, 63.38%, 64.33% and 61.90%. Surely, such sensitivity values are only relevant for the specified user criteria, but we can see that there is indeed no prediction model that particularly stands out. There are tests where all models scored well, such as $(0, 100)$, and tests were they all scored poorly, such as $(200, 300)$. This could be due to the fact that we are only relying on the scores of only two of the closest centroids, which might not always offer the closest prediction. This could again be overcome if we further managed to reduce our average prediction errors, by involving more than two centroids in our prediction process.

Specificity As one may notice from observing Figs. 16, 17, 18, and 19 when we changed our question from *Is a query worth executing?* to *Is a query not worth executing?*, the performance of our prediction models got significantly better. This is reasonable as the sensitivity tests have more boundaries on what the predicted score should be, therefore the specificity metric is more forgiving when it comes to the error of our predicted scores.

We can notice patterns in our specificity values as well, there are tests where all models performed consistently well such as $(300<)$ and tests where performance was poorer such as $(100<)$. We have calculated average specificity values for the four prediction models, these are (respectively as they appear in Fig 18 from top to bottom): 89.32%, 90.42%, 91.63% and 89.32%.

Although these average sensitivity and specificity values are not absolute, as it is impossible to declare and test every possible user criteria, we can still make certain conclusions about which the best prediction model is. Prediction model $k = 107; R$ has the highest average sensitivity and specificity values, even if it did not significantly outperform the other models, as well as the lowest average prediction errors for both the refinement and test refinement sets. Surely, this does not mean that this is the definitive prediction model for determining whether a query is worth executing or not, as different data-sets and queries with more/ less dimensions would most likely require a different k number.

5 Conclusions & future work

In this work, we hypothesize whether we can determine if a query is worth executing based on score prediction and user criteria in distributed computing environments based on query-driven mechanisms. We implemented a query-driven machine learning approach, where we took a set of exploration queries, obtained their scores. We obtained the two closest centroids (winner and rival representatives) for each query in a testing set of real query workloads over real data and calculated their score errors. The representatives' score errors acted as a motivator for our decision to perform centroid refinement to improve the reliability of the closest centroid and determine whether our actions would improve the accuracy of our predictions.

We have experimented and compared with other approaches in the literature with different refinement approaches and through our experiments determined which one maximized reliability of the closest centroid. While doing so, we looked into the factors that could influence our refinement: the value of k query representatives and how many queries are needed to reach the best refinement possible. We observed that as k increases the effect of refinement decreases, while other tests have shown that we can refine centroids up to a certain degree before the reliability of the closest centroid starts worsening.

We then used both our refined centroids and unrefined centroids to make separate predictions for different sets of queries using three different predictions approaches. We observed that predictions can get better in terms of accuracy in predictions as the number of centroids increases, but as previously, after a certain point predictions will stop improving.

We observed that increased reliability does improve predictability when predicting scores for the refinement query sets but is not as effective with new query sets. Despite that, it has to be noted that although centroid refinement does not always improve predictability for new query sets, it can often offer an average prediction error similar to what the unrefined centroids would offer.

We also evidenced that the best predictions are acquired when we use the scores of both representatives. This has lead us to the conclusion that we can further improve predictions in the future if we involved more than two representatives in the score prediction process. We would calculate the weighted sum of the two representatives' scores to obtain a score prediction for a query, where these weights would depend on our BR or AR metrics. Future work could therefore involve an appropriate methodology of how the $n > 2$ closest centroids should be weighted according to their score errors.

We finally introduced and measured the sensitivity and specificity for the score predictions of our four best performing prediction models, to see whether a query is worth executing or not worth executing according to user criteria. As much higher percentages were acquired while measuring specificity, we determined that it is easier to answer the question *is a query not worth executing?* rather than *is a query worth executing?*. During the sensitivity tests we also observed that prediction models $k = 107$; UR and $k = 107$; R would outperform each other on different user criteria tests. For that reason, other possible future work could involve combining a set of refined and unrefined centroids to produce a new set of centroids which could lead to more accurate predictions overall.

Our results have acted as proof that our methodology can be used to avoid the execution of bad/non-informative queries that will waste resources and has the potential for further improvement.

Acknowledgments This research received funding from the European's Union Horizon 2020 research and innovation programme EU/INNOVATE under the grant agreement No. 745829 and UK EPSRC grant EP/R018634/1.

Appendix

Table 1 Table of notations

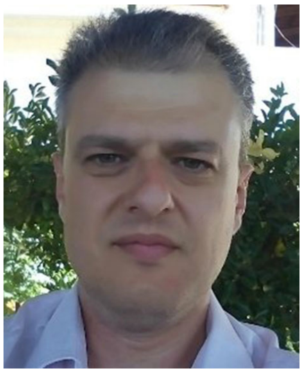
Parameter	Notation
d	data dimensionality
$\mathbf{x}$	data point in d-dimensional space
$\mathbf{q}$	query in $2d$-dimensional space
y	query score in $\mathbb{N}_+$
$\hat{y}$	predicted query score in $\mathbb{N}_+$
$\mathcal{B}$	dataset of d-dimensional points
$\mathcal{W}$	set of k query representatives
$\mathbf{w}$	winner query representative in $2d$-dimensional space
$\mathbf{r}$	rival query representative in $2d$-dimensional space
$e_\mathbf{w}$	score prediction error of the winner query representative
$e_\mathbf{r}$	score prediction error of the rival query representative
f_1, f_0	rival and winner prediction probabilities
$a(\mathbf{q})$	rewarding/penalizing function given a query $\mathbf{q}$
$n_\mathbf{w}$	cardinality of a query group represented by winner $\mathbf{w}$

References

1. Aboulnaga A, Chaudhuri S (1999) Self-tuning histograms: building histograms without looking at data, pp 181–192
2. Anagnostopoulos C, Savva F, Triantafillou P (2018) Scalable aggregation predictive analytics. Appl Intell 48(9):2546–2567
3. Anagnostopoulos C, Triantafillou P (2015) Learning to accurately count with query-driven predictive analytics. In: 2015 IEEE international conference on big data (big data), pp 14–23
4. Bottou L (1998) On-line learning in neural networks. Chapter on-line learning and stochastic approximations, pp 9–42. New York, NY, USA
5. Chaudhuri S (1990) Generalization and a framework for query modification. In: 1990 Proceedings. Sixth international conference on data engineering, pp 138–145
6. Chaudhuri S, Das G, Hristidis V, Weikum G (2004) Probabilistic ranking of database query results. In: Proceedings of the thirtieth international conference on very large data bases - vol 30, VLDB '04, pp 888–899
7. Chu WW, Qiming C (1994) A structured approach for cooperative query answering. IEEE Trans Knowl Data Eng 6(5):738–749
8. Cormode G, Garofalakis M, Haas PJ, Jermaine C (2012) Synopses for massive data: samples, histograms, wavelets, sketches. Found Trends Databases 4:1–294
9. Dean J, Ghemawat S (2010) Mapreduce: a flexible data processing tool. Commun ACM 53:72–77
10. Fagin R, Lotem A, Naor M (2001) Optimal aggregation algorithms for middleware. In: Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on principles of database systems, pp 102–113
11. Google (2019) BigQuery documentation pricing

12. Gunopulos D, Kollios G, Tsotras VJ, Domeniconi C (2005) Selectivity estimators for multidimensional range queries over real attributes. The VLDB Journal 14(2):137–154
13. Haas PJ, Swami AN (1992) Sequential sampling procedures for query size estimation. SIGMOD Rec 21(2):341–350
14. Hu Q, Wu J, Bai L, Zhang Y, Cheng J (2017) Fast k-means for large scale clustering. In: Proceedings of the 2017 ACM on conference on information and knowledge management, CIKM'17. ACM, New York, pp 2099–2102
15. Ilyas IF, Beskales G, Soliman MA (2008) A survey of top-k query processing techniques in relational database systems. ACM Comput Surv 40(4):11:1–11:58
16. Islam MS, Liu C, Zhou R (2012) On modeling query refinement by capturing user intent through feedback. In: Proceedings of the twenty-third australasian database conference, vol 124, pp 11–20
17. Islam MS, Liu C, Zhou R (2013) A framework for query refinement with user feedback. Journal of Systems and Software 86(6):1580–1595
18. Mishra BK, Rath A, Nayak NR, Swain S (2012) Far efficient k-means clustering algorithm. In: Proceedings of the international conference on advances in computing, communications and informatics, ICACCI'12. ACM, New York, pp 106–110
19. Rodriguez-Lujan F, Vergara H (2014) Huerta on the calibration of sensor arrays for pattern recognition using the minimal number of experiments. Chemometrics and Intelligent Laboratory Systems 130:123–134
20. To H, Chiang K, Shahabi C (2013) Entropy-based histograms for selectivity estimation. In: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, CIKM '13. ACM, New York, pp 1939–1948
21. Vergara A, Vembu S, Ayhan T, Ryan MA, Homer ML, Huerta R (2012) Chemical gas sensor drift compensation using classifier ensembles. Sensors and Actuators B: Chemical 166:320–329
22. Vitter JS, Wang M, Iyer B (1998) Data cube approximation and histograms via wavelets. In: Proceedings of the seventh international conference on information and knowledge management, CIKM'98. ACM, New York, pp 96–104

Yiannis Kathidjiotis is a researcher of the Essence: Pervasive & Distributed Intelligence Lab, Information Data and Analysis (IDA) Section at the School of Computing Science, University of Glasgow. His expertise lies in the domains of large-scale data analytics systems and how machine learning can be applied to expedite complex query processing and data analytic tasks. His research focuses on how Machine Learning models can explain data subspaces to guide exploratory analysis, and they can be applied for Approximate Query Processing with error guarantees. He holds a MSc in Data Science at University of Glasgow.



Dr Kostas Kolomvatsos is an Assistant Professor, Department of Informatics & Telecommunications, University of Thessaly and Marie Sklodowska-Curie Research Fellow, University of Glasgow. He received his B.Sc. in Informatics from the Department of Informatics at the Athens University of Economics and Business in 1995, his M.Sc. in Computer Science - New Technologies in Informatics and Telecommunications and his Ph.D. from the Department of Informatics and Telecommunications at the National and Kapodistrian University of Athens (UoA) in 2005 and in the beginning of 2013 respectively. He has participated in several European and national research projects. He is an honorary academic member of the Essence: Pervasive & Distributed Intelligence Lab within the Information, University of Glasgow. His research interests are in the definition of Intelligent Systems and techniques adopting Machine Learning, Computational Intelligence and Soft Computing for Pervasive Computing, Distributed Systems, Internet of Things, Edge Computing and the management of Large Scale Data.

Dr Christos (Chris) Anagnostopoulos is an Assistant Professor of Distributed and Pervasive Computing, School of Computing Science, University of Glasgow. His expertise is in the areas of network-centric pervasive & adaptive systems and in-network information processing in large-scale distributed sensor/UxV/ Edge networks. He has received funding for his research by the EU, UK EPSRC, and the industry. Dr Anagnostopoulos is coordinating the projects: EU GNFUV and EU Marie Sklodowska-Curie (MSCA)/INNOVATE, and is a co-PI of the EU PRIMES and UK EPSRC CLDS. Dr Anagnostopoulos is an author of over 140 refereed scientific journals and conferences. His papers have recieved several awards including IEEE Big Data 2018, IEEE WD 2019. He is leading the Essence: Pervasive & Distributed Intelligence Lab within the Information, Data, and Analysis (IDA), University of Glasgow. He serves as an Editorial Member in the Applied Intelligence, Distributed Sensor Networks, and Open Computer Science journals. Dr Anagnostopoulos before joining Glasgow was an Assistant Professor at Ionian University and Adjunct Assistant Professor at the University of Athens and University of Thessaly. He haheld postdoctoral positions at University of Glasgow and University of Athens in the areas of in-network context-aware distributed & mobile computing systems. He holds a BSc, MSc, and Ph.D. in Computing Science, University of Athens. He is a member of ACM and IEEE.