CONTENTS

On Supporting Scalable Active Learning-based Interactive Data Exploration with Uncertainty Estimation Index

Xiaoyu Ge
University of Pittsburgh
xiaoyu@cs.pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

ABSTRACT

Driven by the exponential growth in data volume and complexity, and the increasing demand to extract concealed value from it, interactive data exploration (IDE) approaches have recently received a great amount of attention in both industry and academia. To achieve interactiveness, most existing active learning-based IDE systems operate on main-memory databases, which inherently limits the scalability of these IDE systems. In this paper, we propose a novel indexing mechanism, called *Uncertainty Estimation Index* (`UEI`), which supports the interactivity and scalability of the active learning-based IDE systems. `UEI` combines hierarchical in-memory indexing with columnar and inverted-indexing based secondary storage mechanism. It achieves scalability and efficiency through a dynamic estimation of the set of data that are most beneficial to the current exploration. By intelligently manage the in-memory cache, `UEI` enables active learning-based IDE systems to scale beyond the main memory restriction, while maintains the desired accuracy and convergency speed. We experimentally evaluated `UEI` using a state-of-the-art IDE system with two schemes, one incorporating `UEI`, and one utilizing a standard DBMS. We measure the efficiency of the proposed solution using a large real-world dataset placed on the secondary storage. Our experiments show that (1) `UEI` version outperforms the DBMS one by providing more than 50x runtime efficiency when the size of the dataset exceeds the main memory capacity, and (2) is capable of achieving sub-second interactive response time for data that is 100 times larger than the available memory while achieving the desired exploration accuracy and effectiveness.

1 INTRODUCTION

In recent years, as the data has grown rapidly in both complexity and volume, the traditional search methods relying on explicit keywords or queries can quickly lose their effectiveness. As reported in previous studies [15], it is often difficult for users to construct precise articulations that describe their interests. In such cases, traditional search methods usually fail to deliver satisfying results, and the user often needs to deal with results that are too big in size due to loose queries or keywords. Consequently, to obtain a satisfying result, users need to execute numerous ad-hoc queries with tightened conditionals to reduce the search space, which requires a considerable amount of time and human effort. Thus, novel *interactive data exploration* (IDE) techniques that aim to assist users in finding their intended items has generated a significant amount of interest in research communities [2, 7, 9, 10, 12, 13, 16].

One of the core features of these IDE systems is to employ *human-in-the-loop* (HIL) exploration processes to minimize the overall user effort and time in finding the relevant data items. Instead of providing inaccurate results with one generic predictive model (i.e., engine), by leveraging human-in-the-loop, a uniquely learned predictive model can be created rapidly for each individual user and task. To do so, a common solution to support effective human-in-the-loop operations is to leverage active learning techniques [20]—active learning refers to a set of machine learning approaches that aim to learn an accurate predictive model with minimum labeled data for regression and classification tasks. Clearly, the goal of active learning naturally aligns with the needs of many human-in-the-loop techniques as they seek to quickly and accurately deliver the results that the user needs with a personalized, predictive model.

In previous works, numerous active learning techniques [5, 8, 14, 20–22] have been proposed to boost the convergence of training a predictive model. Among these query strategies, *uncertainty sampling* is the most commonly used one because of its simplicity and efficiency, as pointed out in [20]. Uncertainty sampling trains each predictive model in an iterative fashion, wherein each iteration, it identifies the unlabeled items that are closest to the current decision boundary of the predictive model as these items are believed to be most *uncertain*. Uncertainty Sampling then solicits the user's label on the identified sample and utilizes it in the training of the predictive model.

Although uncertainty sampling is more efficient than alternative active learning methods, in order to find the most uncertain object, it still needs to perform an exhaustive search over the entire database. Therefore, in the case where the size of the data is larger than the main memory capacity, those data that resides on the secondary storage must be loaded into memory at each interation. Due to the limitation imposed by physical I/O of the secondary storages, it takes a significant amount of time for active learning techniques to scan datasets that are considerably larger than the main memory. This essentially makes it impossible for any active learning-based IDE system to explore datasets that are larger than the available memory. As pointed in [15], run time efficiency is critical for the IDE systems as any response time excesses 500ms will severely impact the user's engagement, and hence hinders the usability of the system. Moreover, as the exploration could occur on any subset of the attributes of the dataset, it is nearly impossible to apply any typical indexing in advance to support the exploration task over any arbitrary combination of the attributes.

In order to achieve interactiveness, existing uncertainty sampling-based IDE systems rely on main memory to cache the entire dataset. For datasets that are larger than the main memory, a subset of data objects will need to be sampled from the original dataset on secondary storage (e.g., [7]). While simple and intuitive, this approach could easily lead to very inaccurate results and a waste of user effort since the boundaries of the interesting data regions in the sampled space is likely to be different from the original space [9]. Moreover, small sets of relevant data regions may even be ignored in the resulting sample set.

To overcome this problem, we propose a novel indexing mechanism coined *Uncertainty Estimation Index* (UEI), to facilitate the interactivity and scalability of active learning-based IDE systems. To our knowledge, UEI is the first approach in extending the scalability of active learning-based IDE systems beyond the main memory capacity.

Published in Proceedings of the 24th International Conference on Extending Database Technology (EDBT), March 23-26, 2021, ISBN 978-3-89318-084-4 on OpenProceedings.org.

Instead of relying solely on the main memory to cache the whole dataset during the exploration, UEI enables caching in memory only the necessary subsets of the data that are needed by the current stage of the exploration. This is achieved through the combination of an effective estimation of the uncertainty of each data object and an efficient data storage mechanism. The essential observation that UEI is based on is that data objects often have additional information that can be used to infer their relationship with other objects, one of which is the *similarity* among data objects. Since the uncertainty essentially represents its distance to the current decision boundary in the high-dimensional data space, thus the uncertainty of an object x is strongly related to the uncertainty of the surrounding objects [14]. This observation allows UEI to informatively select the set of highly uncertain data objects to be loaded into memory before it is needed by the predictive model. Hence, the amount of memory needed to explore a dataset in real-time is significantly reduced.

To evaluate UEI, we employed a state-of-the-art data exploration system REQUEST [9], and compare the performance of UEI against MySQL, which is used by the existing interactive data exploration systems [7, 12]. Our results using a large real-world dataset, namely the Sloan Digital Sky Survey (SDSS) [1], show that UEI outperforms existing solutions by more than 50X in run-time efficiency when the size of the dataset goes beyond the main memory capacity. Furthermore, UEI is well capable of meeting the sub 500 millisecond interactive response time requirement for data that is at least 100 times larger than the main memory capacity, while achieving minimum impact on the convergence of the predictive model.

It should be noted that even though UEI is designed for the IDE systems, where response time and run time efficiency is critical, it can also be used in combination with any active learning-based human-in-the-loop (HIL) applications. Examples of such human-in-the-loop applications including but not limited to: record matching [3], entity resolution [18, 19], and facts checking [4].

2 BACKGROUND

In this section, we provide the necessary background of our `UEI`.

2.1 Active Learning

Active learning refers to a set of approaches that aim to learn an accurate model with minimum labeled data for regression and classification tasks. One key component of active learning is the *query strategy* that sequentially selects the most informative unlabeled sample (i.e., data object) from the entire database to be labeled by the user.

In previous works, a number of active learning techniques have been proposed to define the "informativenes" of samples [20], including: *Uncertainty Sampling* [14], *Query-By-Committee* [21], *Expected Model Change* [5], *Expected Error Reduction* [22], and *Expected Model Output Change* [8]. These techniques are often interchangeable, providing the applications the flexibility to choose the most appropriate query strategy that fits their needs. Among the query strategies, *Uncertainty Sampling* [14] is the most commonly used because of its simplicity and efficiency [20].

Uncertainty Sampling is a query strategy that can be used with any probability-based predictive model (e.g., Naive Bayes, SVM, etc.). The intuition of *Uncertainty Sampling* is that patterns with high uncertainty are hard to classify, and thus, if the labels of those patterns are obtained, they can boost the accuracy of the classification models. Particularly, in binary classification models (e.g., with class labels 0 and 1), the most uncertain example $\mathbf{x}$ is the one which can be assigned to either class label $z(\mathbf{x})$ with probability 0.5. Inspired by the idea of uncertainty, also known as *least confidence* (lc), [14] proposes a measurement of uncertainty for binary classification models:

$$u^{(lc)}(\mathbf{x}) = 1 - p(\hat{y}|\mathbf{x}) \tag{1}$$

where $u^{(lc)}(\mathbf{x})$ is the uncertainty score with the least confidence measurement of $\mathbf{x}$, and $\hat{y}$ is the predicted class label of the unlabeled $\mathbf{x}$. Accordingly, after measuring the uncertainty of each unlabeled sample, the unlabeled sample with highest uncertainty is selected:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} u(\mathbf{x}) \tag{2}$$

where $u(\mathbf{x})$ can be any other measurement of informativeness over the unlabeled sample $\mathbf{x}$.

Algorithm 1 Typical Workflow of Active Learning-based IDE

```
Require: The raw data set $D$; Batch Size $B$
Ensure: A set of results $R$
1: Labeled set $L \leftarrow \emptyset$
2: Unlabeled set $U \leftarrow D$
3: $M \leftarrow$ initialize query strategy
4: while user continues the exploration do
5:     for $i = 1$ to $B$ do
6:         Choose one $x$ from $U$ using $M$
7:         Solicit user's label on $x$
8:         $L \leftarrow L \cup \{x\}$
9:         $U \leftarrow U - \{x\}$
10:    end for
11:    $M \leftarrow$ trained with $L$ to update $M$.
12: end while
13: Return the set of results $R$ classified as positive by $M$.
```

2.2 Interactive Data Exploration

Active learning-based IDE systems can effectively find relevant items that are often undiscoverable using traditional search methods [7, 9, 10, 12]. In particular, active learning-based IDE systems do not require users to formulate any complex queries, nor does it need any form of description of the target items. The entire exploration can be done by answering simple binary (i.e., yes or no) questions. More importantly, active learning-based IDE systems can further be used to enhance traditional search results. For instance, they can be used to address the problem of having overwhelming results due to a broad query or search conditions.

As shown in Algorithm 1, a typical active learning-based IDE system works in the following steps: first it incorporates a query strategy (e.g., uncertainty sampling), which is used for selecting the example objects to be presented to the user for labeling (line 3). As long as the user is willing to label more examples (line 4), active learning-based IDE system will keep invoking the query strategy M to select a new example object x from D, and present it to the user to label it as *relevant* or *irrelevant* (lines 5-9). Once the amount of labeled samples received from the user reaches a sample batch size (denoted as B), which is a tunable parameter of the active learning-based IDE balancing the effectiveness and efficiency, then the classifier model employed by the query strategy will be updated according to the label assigned to x (line 11). In particular, the label assigned to x will be used for retraining the classifier model, which is an essential step towards selecting the object presented to the user in the next iteration. Once the iterative labeling process is completed, the obtained results will be presented to the user (line 13).

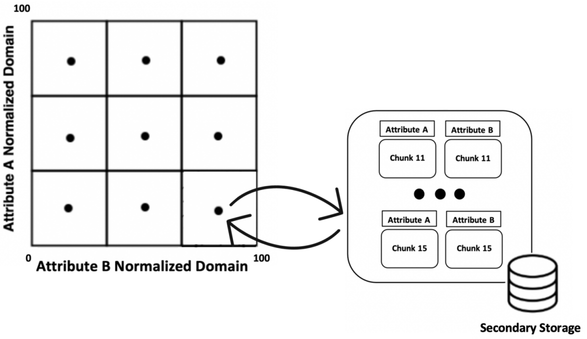


Figure 1: Illustrates UEI with a 2D data space, each grid represents a subspace, the dot in the center of each grid represents a symbolic point p. With chunks stored as separated files on the secondary storage.

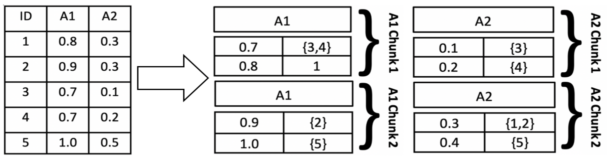
3 UNCERTAINTY ESTIMATION INDEX

Maintaining interactive response time for large datasets that are beyond the main memory capacity has always been one of the major challenges of active learning-based IDE systems. In tackling this problem, we focused our efforts on uncertainty sampling and proposed a novel index approach coined *Uncertainty Estimation Index* (UEI). Our UEI can be easily incorporated in any existing systems that leverage the uncertainty sampling and can be used in conjunction with any probabilistic-based classifiers as discussed in [14]. In the next section (Section 3.1) we provide the details of UEI's main component, and discuss how UEI leverages the inverted index-based data secondary storages to support scalable exploration. Following that, in Section 3.2, we provide a walkthrough of a complete example to illustrate how a typical active learning-based IDE workflow (i.e., Algorithm 1) performs when enhanced with UEI.

3.1 UEI Components

A key observation underlying UEI is that data objects often have additional information that can be used to inference their relationship with other objects, one of which is the similarity (i.e., distance) between data objects [14]. In other words, the uncertainty value of an object x is strongly related to the uncertainty of the surrounding objects. For example, if x is located near to one of the current decision boundaries, then x would have higher uncertainty due to a mixed set of relevant and irrelevant neighbors. Such continuity between data points is also generally assumed in both supervised and semi-supervised learning algorithms where data points that are closer to each other are more likely to share a label [23]. More precisely, we observed that the uncertainty of a data object x can be approximated through its spatial relationships with the labeled data objects. UEI explores this spatial relationship to load into memory the set of highly uncertain data objects before they are needed by the query strategy.

Specifically, as illustrated in Figure 1, the main idea of UEI is to divide the exploration space D into equal-size subspaces (i.e., d-dimensional grids) g_i's of D ($g_i \in D$), and build a set of symbolic (virtual) index points $P = \{p_1, ...p_c\}$, such that each index point p_i represents a subspace g_i. In each iteration, UEI estimates the uncertainty of each subspace based on the uncertainty of its corresponding index point p, then loads only the data in the subspace that is predicted to be most uncertain into memory. Conceptually, UEI is based on the same principle as hash-based multidimensional indexing, such as the traditional grid file structures, which splits the space into a non-periodic grid where one or more cells of the grid refer to a small set of points. As opposed to a grid file, designed to efficiently reference a single value with multiple keys, UEI is designed to scale out the uncertainty sampling by estimating the distribution of the uncertainty of data objects through the set of symbolic index points.



ID	A1	A2
1	0.8	0.3
2	0.9	0.3
3	0.7	0.1
4	0.7	0.2
5	1.0	0.5

A1	
0.7	{3,4}
0.8	1

A1 Chunk1

A1	
0.9	{2}
1.0	{5}

A1 Chunk2

A2	
0.1	{3}
0.2	{4}

A2 Chunk1

A2	
0.3	{1,2}
0.4	{5}

A2 Chunk2

Figure 2: Before storing the data, UEI vertically decompose the data into an inverted index form, and then store them in separate chunks.

To do so, UEI comprises five components: 1) an index set P of symbolic index points p_i; 2) a mapping method $m : p \longmapsto C$ that maps each index point p to a set of data chunk C; 3) a data cache U that caches a subset of uniformly sampled unlabeled data; 4) a set of labeled data L that contains all data that has been labeled by the user, and 5) the exploration dataset D stored in a fully inverted columnar format on a hard drive. The first four components reside in the main memory.

UEI divides the operation of exploration into two phases: an *Index Initialization* phase, and *Interactive Exploration* phase. The first phase only needs to be executed once per each new dataset. The second phase is specific to each exploration and discussed in the next section (Section 3.2).

Index Initialization Phase: As illustrated in Algorithm 2, to work with a new dataset, UEI first vertically (i.e., attribute-wise) decompose the whole data set and sort each dimension (i.e., attribute) based on the values in ascending order (lines 2 - 4). Since each dimension of a typical exploratory dataset (e.g., scientific, business analysis) can contain values of an arbitrary length, and one specific value for a dimension may appear multiple times, we compress the data by organizing it in a key-value fashion ($< key, \{values\} >$), where each value of the dimension would be used as a *key* and the ids of the corresponding objects as *values* (as illustrated in Figure 2). Note that for each exploration task, UEI stores all needed data in one location, thus when exploring data that are distributed in multiple locations (e.g., tables, files), the data needs to be merged before being utilized in the exploration.

During the process of storing the data, UEI splits the distinct values of each dimension d into a set of equal-sized data chunks $C^d = \{c^d{}_i, ..., c^d{}_u\}$, where each chunk will be stored as a separate file on the disk, and the size of each chunk can be adjusted based on the size of the data and the available hardware resources (line 5). UEI also ensures that the values of each dimension are stored in a sequential order, meaning values stored in each subsequence chunk $c^d{}_{i+1}$ will be larger than the values that have been stored in $c^d{}_i$ for efficient lookup.

Once the data are partitioned and stored on the disk, UEI would start construct the set of symbolic index points P by divide the original data space D into a set of equilateral d-dimensional subspaces $G = \{g_i, ..., g_j\}$, where $|G| = |P|$, then for each subspace g_i, UEI constructs a symbolic index point p_i that represents g_i by using the coordinates of the "virtual" center point of g_i (lines 7-11). To ensure UEI can be deployed in resource restricted environments, the number of symbolic index point can be adjusted based on the size of the dataset and the available hardware resources.

In order to construct and load each subspace g_i into memory, UEI employed a hash-based mapping method m that records for

Algorithm 2 Typical Exploration Flow with UEI

```
Require: The raw data set $D$
Ensure: Result set $T$
 1: $P \leftarrow \emptyset, L \leftarrow \emptyset, U \leftarrow \emptyset$
 2: $DC \leftarrow verticalDecompose(D)$
 3: for $d = 1$ to $|DC|$ do
 4:     $sort(DC_d)$
 5:     $C \leftarrow splitIntoChunks(DC_d)$
 6: end for
 7: $G \leftarrow splitIntoSubspaces(D)$
 8: for each grid $g_i \in G$ do
 9:     $p_i \leftarrow computeCenter(g_i)$
10:     $P \leftarrow P \cup \{p_i\}$
11: end for
12: $U \leftarrow sample(D, \gamma)$
13: $M \leftarrow$ initialize predictive model for uncertainty estimation
14: while user continues the exploration do
15:     drop any previously loaded data regions from $U$
16:     $M \leftarrow$ trained with $L$ to update $M$
17:     $P \leftarrow updateUncertainty(P, M)$
18:     $p_i^* \leftarrow$ choose the most uncertainty index point from $P$
19:     $g_i^* \leftarrow$ load data region with $m(p_i^*)$
20:     $U \leftarrow U \cup g_i$
21:     choose one $x$ from $U$ using $M$
22:     solicit user's label on $x$
23:     $L \leftarrow L \cup \{x\}$
24:     $U \leftarrow U - \{x\}$
25: end while
26: $T \leftarrow resultRetrieval(L)$
27: Return the set of interesting data objects $T$
```

each symbolic index point p_i, the set of chunks that are needed to construct g_i. As chunks are stored separately on the disk, this approach allows UEI to quickly identify the data that needs to be loaded. Since each data subspace g is stored as series of one-dimensional data chunks, to reconstruct each g when needed, UEI utilizes a hash table for efficiently merge of those data chunks. During the merge process, UEI iterates through each dimension and loads the corresponding chunks to the memory one at a time, and each entry in the chunk would be visited in a sequential manner. For each object ID that is recorded in a loaded data chunk c_i, the value associated with the ID will be inserted into the corresponding entry in the hash table. Once a chunk has been examined, UEI will release the memory space used to hold the data chunk and reuse the space for the subsequent chunk.

3.2 UEI in Action

In the previous section, we have discussed the components of UEI, as well as how the data are being stored and indexed, which essentially covers the first half (i.e., lines 1 - 11) of the exploration workflow, shown in Algorithm 2. In this section, we will discuss the interactive exploration phase of UEI, which illustrates how a typical active learning-based IDE task (i.e., Algorithm 1) can be performed when incorporating UEI (lines 12 - 27, Algorithm 2).

Interactive Exploration Phase: After the index set has been constructed, UEI begins the exploration by filling the unlabeled set U. Specifically, for the original data space D, UEI would uniformly sample a set of data from the underlying dataset (line 12), where the size of the samples γ can be adjusted based on the system hardware specs (e.g., available main memory size). As a result, a set of unlabeled objects U would be sampled and cached in the main memory. These unlabeled objects will then be used in the acquisition of the set of initial examples that will be labeled by the user to construct the initial predictive model M_0 for uncertainty estimation. Query strategy will randomly sample examples from U until the set of initial examples contains at least one positive example and one negative example (line 13).

In each iteration, UEI updates the uncertainty of all index points $p_i \in P$ based on the most recently trained predictive model M_{t-1} (line 17), which serves as the uncertainty estimator. Here the uncertainty of a data object is essentially equals to the probability of one object being either positive or negative class, with a value that equal to 50% being the most uncertain. Then, the index point p_i^* for which the current exploration model is most uncertain, will be chosen (line 18), such that:

$$p_i^* = \operatorname*{Argmax}_{p_i \in P} M_{t-1}(Y|p_i) \tag{3}$$

where Y = {0,1} is set of binary labels.

Based on the chosen p_i^*, UEI uses the mapping method m to identify and load (into the memory) all data chunks that correspond to the subspace g_i^*, which was represented by p_i^* (line 19). As mentioned earlier, the mapping method m is simply a hash table that maps a single index point p into a set of data chunks located on the disk. Later, the data of subspace g_i^* together with the unlabeled dataset U will be used by the query strategy (i.e., uncertainty sampling) in the selection of the example to be labeled in the current iteration (lines 20 - 22). To reduce memory usage, by default UEI kept only one uncertain data region g_i^* in the memory at any given time. Once the user is satisfied with the exploration result, the *resultRetrieval* method will be invoked to retrieve the exploration results and present them to the user (lines 26 - 27).

Tuning Interactive Exploration: In addition to the above typical exploration flow, UEI further allows the user to specify a response latency threshold σ that determines the latency between each exploration iteration (i.e., two subsequent examples). Using the user-specified σ, UEI determines whether or not to defer the swap between the current in-memory uncertain region g_i^* and the next uncertain region g_{i+1}^*, when g_i^* is no longer the most uncertain region.

In the case when an extremely low σ is specified that makes it impossible for the system to load the entire subspace g_i^* into main memory, UEI would start fetching the corresponding data chunks that associated with g_{i+1}^* (in the background) θ iterations before g_{i+1}^* is loaded into the memory. Here, θ is a tunable variable that can also be inferred based on the average loading time τ of data regions, and the configurable latency threshold σ, such that $\theta = \left\lceil \frac{\tau}{\sigma} \right\rceil$.

3.3 Time Complexity of UEI

Clearly, the time complexity of UEI is dominated by the interactive exploration phase. As discussed in Section 3.1, the initialization phase is done once for each dataset and the time required for UEI to prepare and store a dataset D on secondary storage is simply *linear* with respect to the number items n stored in D.

As discussed in Section 3.2, each iteration of the interactive exploration phase in UEI is dominated by the time taken to load the data from the chunks stored on the disk into the memory, which is *linear* with respect to the number of dimensions k and the number entries e stored in the loaded chunks. In contrast, each iteration in the current IDE approaches needs to load and examine all data items n ($e <<< n$).

Therefore, the time complexity of the UEI-enhanced data exploration generally is reduced from $O(kn)$ where n is the number of data objects to $O(ke)$ where e is the number entries associated

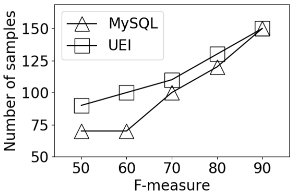


Figure 3: UEI Accuracy (Small Target Region).

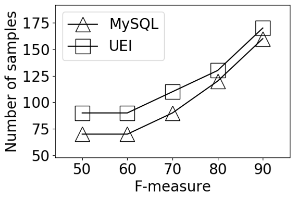


Figure 4: UEI Accuracy (Medium Target Region).

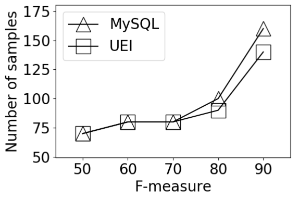


Figure 5: UEI Accuracy (Large Target Region).

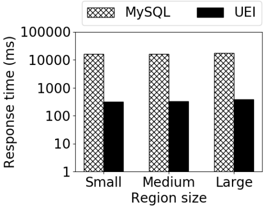


Figure 6: UEI Response Time.

Table 1: PARAMETERS

Number of runs per result	10
Number of dimensions (D)	5
Number of relevant regions	1
Cardinality of relevant regions	0.1% (S), 0.4% (M), 0.8% (L)
Uncertainty Estimator	DWKNN [11]
Label Type	Binary
Data Storage Engine	UEI, MySQL
Size of Individual Data Chunk	470KB
Number of Symbolic Index Points	3125
Latency Threshold	500ms
Performance Measurement	F-Measure (Accuracy)

with the loaded chunks for the current most uncertain subspace g_i^*.

4 EXPERIMENTAL EVALUATION

In our experimental evaluation of our UEI, we use REQUEST [9], a state-of-the-art IDE system, with two schemes, one incorporating UEI, and one utilizing MySQL, used in the existing IDE systems [7, 12]. After describing our experimental setup in Section 4.1, we present the findings of our experimental evaluation in Section 4.2.

4.1 Experiment Setup

Dataset: We used 40 GB of real-world dataset from Sloan Digital Sky Survey (SDSS) [1] that consists of 10×10^6 tuples.

IDE System: In our experiments, we employed our REQUEST [9] with traditional uncertainty sampling and the *dual weighted k-nearest neighbor* (DWKNN) [11] as the uncertainty estimator.

Environment: We implemented both REQUEST and UEI with Java JRE 1.7. All the experiments were run on a machine with Intel Core i7 Processor, 32 GiB RAM and 2 TB of NVMe SSD. The fast NVMe SSD was used to eliminate any potential bottleneck due to physical IO limitation. All experiments reported are averages of 10 complete runs. We have considered five numerical attributes *rowc, colc, ra, dec* and *field* of the PhotoObjAll table.

Target Interest Regions: The exploration task characterizes user interests and eventually predicts the relevant regions by iteratively gathering user labeled tuples. We experimented with 1 region per each exploration task. In addition, we vary the single region complexity based on the data space coverage of the relevant regions. Specifically, we categorize relevant regions to small, medium and large. Small regions have cardinality with an average of 0.1% of the entire experimental dataset, medium regions a cardinality of 0.4%, and large regions a cardinality of 0.8%. Furthermore, the dimensionality of the target interest regions is the same as the dimensionality of the dataset across the entire experiment.

User Simulation For experiment evaluation purpose, we simulate the user behavior using the following method. For each target interest region, we simulate the user by executing the corresponding range query to collect the exact target set of relevant tuples. We rely on this "oracle" set to assign confidence score p to the tuples we extract in each iteration based on their location in the data space against the target region.

More specifically, for each relevant region, there is a region center and a set of region widths, one for each dimension. We define the *maximum relative distance d* of an example against the region center as:

$$d = max_{i=1..l}(|x_i - c_i|/w_i) \tag{4}$$

where l is the dimension number, $|\cdot|$ is the absolute value operator, x_i, c_i and w_i are the attribute value of the example, of the center and region width in each dimension.

Parameters: Table 1 summarizes the important settings in the experiments.

4.2 Experiment Results

In our experiments we aimed to test UEI's ability to provide an interactive response time for datasets that are beyond the size of main memory capacity, and to illustrate the benefit of searching only a small set of cached objects with UEI against performing an exhaustive search over the entire dataset. In our experiments, we stored 10 million data items with both UEI and MySQL, and restricted the memory footprint for both UEI and MySQL to be within 400MB, which is ~ 1% of the entire dataset. Figures 3 to 6 illustrates the effectiveness and efficiency of our proposed UEI.

UEI Accuracy (Figures 3 to 5): Compared to MySQL, we have noticed that our proposed UEI requires more labeled examples in the early stage of the experiment (e.g., below 70% of accuracy). This is due to the fact that in the early stage, the classifier does not have enough training samples to learn an accurate uncertainty estimator (i.e., DWKNN classifier) that captures precisely the user's interesting regions. This causes the predictive model to select less informative examples to be labeled by the user. Since UEI uses uncertainty as the criteria for both the example selection and the loading of data regions (i.e., subspaces), therefore the negative effect of an inaccurate classifier has been magnified.

However, as the accuracy of the uncertainty estimator improves with more labeled examples, we observed a significant boost in performance of UEI in the later stages (e.g., above 80% accuracy) of the exploration. This is expected as the the predictive model gets more and more accurate with respect to the decision boundaries, and thus can estimate more accurate uncertainties. Since UEI rely on the uncertainty estimation for both the query strategy and cache management, therefore, it benefits more noticeably than the MySQL, which only rely on uncertainty estimation for query strategy.

UEI Response time (Figure 6): Finally, we have measured the response time for both UEI and MySQL based schemes. As shown in Figure 6, UEI achieves 50*x* faster response time than MySQL, and ensures the sub-second interactive response time across all data region sizes. Note the response time remains the same across all three target interest regions sizes, which is as expected because the runtime complexity of the uncertainty sampling-based systems only depends on the size of the dataset and not the size of the target interest regions.

From the experiments, it is clear that due to the fact that uncertainty sampling requires an exhaustive search over the entire data spare, thus the physical bandwidth of the secondary storage has become the major bottleneck that severely limits the scalability of active learning-based IDE systems.

Even though in our experiments, we have used NVMe based SSD with I/O throughput of around 3.4GB/s, the uncertainty sampling still takes over 12 seconds to complete the exhaustive search in each iteration. Therefore, it is still impossible to explore datasets that exceed the main memory capacity without UEI.

5 RELATED WORK

Traditionally, indexing has been the core technique for optimizing response time in database systems. Recently, main-memory indexing and specialized access methods have been proposed to support domain-specific query processing and analytics (e.g., [6, 17]). UEI is based on similar principles as these specialized access methods. However, UEI, to the best of our knowledge, is the first domain-specific access method with in-memory and disk components that support interactivity and scalability of active learning-based IDE systems.

The active learning-based IDE systems that can leverage our proposed UEI for better scalability includes *REQUEST* [9], the first active learning-based IDE system, and the two more recently proposed systems, *Dual-Space Model* [12] and *ExNav* [10]. REQUEST utilizes a two stages approach; a data reduction stage aims to selectively reduce the search space while keeping all relevant data regions, and a query selection stage that utilizes an active learning-based predictive model to iteratively improve the accuracy of the constructed exploratory query through interactions with the user. Dual-Space Model uses a new uncertainty sampling-based predictive model and a new dual-space pruning technique that focuses solely on exploration tasks with a single relevant region. It also optimizes these tasks for faster model convergence. ExNav is the first uncertainty sampling-based IDE system that specializes in exploring a variety of unstructured data sets by leveraging the corresponding data embedding methods for each unstructured data type [10].

In addition to the active learning-based IDE systems, UEI can also be utilized in other active learning-based *Human-in-the-loop* (HIL) systems. For example, in [18], the authors have proposed an active learning-based HIL system, called SystemER, for learning Entity Resolution models through user interactions. Another example of an active learning-based HIL application is fact-checking. In [4], the author has proposed an effective active learning-based HIL system for identifying various types of potentially misleading or false information for news contents. Most recently, [19] has proposed to use active learning for learning the implicit structured representations of entity names, which can be useful for many entity-related tasks such as entity normalization and variant generation. To facilitate the process of learning such structured representations, a user-friendly interface called PARTNER has been designed to enhance the user's interaction experience.

6 CONCLUSION

In this paper, we present the *Uncertainty Estimation Index* (UEI), the first indexing mechanism that enables active learning-based IDE systems to explore datasets that exceed the main memory capacity. Instead of requiring all data to be loaded into the main memory as in existing active learning-based IDE systems for sub-second respond times, UEI enables the scalability by dynamically identify and caching the set of objects that are most uncertain to the current stage of the exploration. It achieves this by maintaining a small in-memory index that estimates the aggregated uncertainty value of the data items in the entire data subspaces. UEI also employees columnar-based secondary storage and combines it with an inverted index to support efficient loading of the necessary data items at each iteration.

Our experimental evaluation using real-world data show that a state-of-the-art IDE systems using UEI greatly outperforms a DBMS-based version of the same IDE system by provide more than 50x runtime efficiency when the size of the dataset exceed the main memory capacity, and is capable of achieving sub-second response time for data that is 100 times larger than the available memory while achieving the desired exploration accuracy and effectiveness. In conclusions, UEI can be used not only with existing active learning-based IDE but with other active learning-based Human-in-the-loop systems as well to achieve significantly higher scalability by removing the main memory restriction.

REFERENCES

[1] SDSS Samples Queries - http://cas.sdss.org/dr4/en/help/docs/realquery.asp.

[2] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis. 2010. An Optimization Framework for Query Recommendation. In *ACM WSDM*. 161–170.

[3] A. Arasu, M. Götz, R. Kaushik. 2010. On active learning of record matching packages. In *ACM SIGMOD*. 783–794.

[4] S. Bhattacharjee, A. Talukder, B. Balantrapu. 2017. Active learning based news veracity detection with feature weighting and deep-shallow fusion. In *IEEE BigData*. 556–565.

[5] W. Cai, Y. Zhang, J. Zhou. 2013. Maximizing Expected Model Change for Active Learning in Regression. In *IEEE ICDE*. 51–60.

[6] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, T. Palpanas, S. Athanasiou, S. Skiadopoulos. 2019. Local Similarity Search on Geolocated Time Series Using Hybrid Indexing. In *ACM SIGSPATIAL*. 179–188.

[7] K. Dimitriadou, O. Papaemmanouil, Y. Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. In *TKDE*, 28:2842–2856.

[8] A. Freytag, E. Rodner, J. Denzler. 2014. Selecting Influential Examples: Active Learning with Expected Model Output Changes. In *ECCV*. 562–577.

[9] X. Ge, Y. Xue, Z. Luo, M. Sharaf, P. Chrysanthis. 2016. REQUEST: A Scalable Framework for Interactive Construction of Exploratory Queries. In *IEEE BigData*. 4566–4579.

[10] X. Ge, X. Zhang, P. Chrysanthis. 2020. ExNav: An Interactive Big Data Exploration Framework for Big Unstructured Data. In *IEEE BigData*.

[11] J. Gou, L. Du, Y. Zhang, T. Xiong, et al. 2012. A new distance-weighted k-nearest neighbor classifier. In *J. Inf. Comput. Sci*, 9(6):1429–1436.

[12] E. Huang, L. Peng, L. Palma, A. Abdelkafi, A. Liu, Y. Diao. 2019. Optimization for active learning-based interactive database exploration. In *VLDB*. 71–84.

[13] S. Islam, C. Liu, R. Zhou. 2013. A framework for query refinement with user feedback. In *J. Syst. Softw., 86(6):1580-1595*.

[14] D. Lewis W. Gale. 1994. A sequential algorithm for training text classifiers. In *ACM SIGIR*. 3–12.

[15] Z. Liu , J. Heer. 2014. The Effects of Interactive Latency on Exploratory Visual Analysis. In *IEEE TVCG*, 20(12):2122–2131.

[16] B. McCamish, V. Ghadakchi, A. Termehchy, B. Touri, and L. Huang. 2018. The Data Interaction Game. In *ACM SIGMOD*. 83–98.

[17] B. Peng, P. Fatourou, T. Palpanas. 2020. MESSI: In-Memory Data Series Indexing. In *IEEE ICDE*. 337–348.

[18] K. Qian, L. Popa, P. Sen. 2019. SystemER: A Human-in-the-loop System for Explainable Entity Resolution. *PVLDB* 12, 12, 1794–1797.

[19] K. Qian, P. Raman, Y. Li, L. Popa. 2020. Learning Structured Representations of Entity Names using Active Learning and Weak Supervision. *CoRR* abs/2011.00105.

[20] B. Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison.

[21] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *ACM Workshop on Computational Learning Theory*.

[22] Y. Zhang, Y. Wang, W. Cai, S. Zhou, Y. Zhang. 2017. From Theory to Practice: Efficient Active Cost-sensitive Classification with Expected Error Reduction. In *SIAM*. 153–161.

[23] Zheng Zhao and Huan Liu. 2007. Semi-supervised Feature Selection via Spectral Analysis. In *SDM*. 641–646.

CONTENTS

On Supporting Scalable Active Learning-based Interactive Data Exploration with Uncertainty Estimation Index

Xiaoyu Ge
University of Pittsburgh
xiaoyu@cs.pitt.edu

Panos K. Chrysanthis
University of Pittsburgh
panos@cs.pitt.edu

ABSTRACT

Driven by the exponential growth in data volume and complexity, and the increasing demand to extract concealed value from it, interactive data exploration (IDE) approaches have recently received a great amount of attention in both industry and academia. To achieve interactiveness, most existing active learning-based IDE systems operate on main-memory databases, which inherently limits the scalability of these IDE systems. In this paper, we propose a novel indexing mechanism, called *Uncertainty Estimation Index* (`UEI`), which supports the interactivity and scalability of the active learning-based IDE systems. `UEI` combines hierarchical in-memory indexing with columnar and inverted-indexing based secondary storage mechanism. It achieves scalability and efficiency through a dynamic estimation of the set of data that are most beneficial to the current exploration. By intelligently manage the in-memory cache, `UEI` enables active learning-based IDE systems to scale beyond the main memory restriction, while maintains the desired accuracy and convergency speed. We experimentally evaluated `UEI` using a state-of-the-art IDE system with two schemes, one incorporating `UEI`, and one utilizing a standard DBMS. We measure the efficiency of the proposed solution using a large real-world dataset placed on the secondary storage. Our experiments show that (1) `UEI` version outperforms the DBMS one by providing more than 50x runtime efficiency when the size of the dataset exceeds the main memory capacity, and (2) is capable of achieving sub-second interactive response time for data that is 100 times larger than the available memory while achieving the desired exploration accuracy and effectiveness.

1 INTRODUCTION

In recent years, as the data has grown rapidly in both complexity and volume, the traditional search methods relying on explicit keywords or queries can quickly lose their effectiveness. As reported in previous studies [15], it is often difficult for users to construct precise articulations that describe their interests. In such cases, traditional search methods usually fail to deliver satisfying results, and the user often needs to deal with results that are too big in size due to loose queries or keywords. Consequently, to obtain a satisfying result, users need to execute numerous ad-hoc queries with tightened conditionals to reduce the search space, which requires a considerable amount of time and human effort. Thus, novel *interactive data exploration* (IDE) techniques that aim to assist users in finding their intended items has generated a significant amount of interest in research communities [2, 7, 9, 10, 12, 13, 16].

One of the core features of these IDE systems is to employ *human-in-the-loop* (HIL) exploration processes to minimize the overall user effort and time in finding the relevant data items. Instead of providing inaccurate results with one generic predictive model (i.e., engine), by leveraging human-in-the-loop, a uniquely learned predictive model can be created rapidly for each individual user and task. To do so, a common solution to support effective human-in-the-loop operations is to leverage active learning techniques [20]—active learning refers to a set of machine learning approaches that aim to learn an accurate predictive model with minimum labeled data for regression and classification tasks. Clearly, the goal of active learning naturally aligns with the needs of many human-in-the-loop techniques as they seek to quickly and accurately deliver the results that the user needs with a personalized, predictive model.

In previous works, numerous active learning techniques [5, 8, 14, 20–22] have been proposed to boost the convergence of training a predictive model. Among these query strategies, *uncertainty sampling* is the most commonly used one because of its simplicity and efficiency, as pointed out in [20]. Uncertainty sampling trains each predictive model in an iterative fashion, wherein each iteration, it identifies the unlabeled items that are closest to the current decision boundary of the predictive model as these items are believed to be most *uncertain*. Uncertainty Sampling then solicits the user's label on the identified sample and utilizes it in the training of the predictive model.

Although uncertainty sampling is more efficient than alternative active learning methods, in order to find the most uncertain object, it still needs to perform an exhaustive search over the entire database. Therefore, in the case where the size of the data is larger than the main memory capacity, those data that resides on the secondary storage must be loaded into memory at each interation. Due to the limitation imposed by physical I/O of the secondary storages, it takes a significant amount of time for active learning techniques to scan datasets that are considerably larger than the main memory. This essentially makes it impossible for any active learning-based IDE system to explore datasets that are larger than the available memory. As pointed in [15], run time efficiency is critical for the IDE systems as any response time excesses 500ms will severely impact the user's engagement, and hence hinders the usability of the system. Moreover, as the exploration could occur on any subset of the attributes of the dataset, it is nearly impossible to apply any typical indexing in advance to support the exploration task over any arbitrary combination of the attributes.

In order to achieve interactiveness, existing uncertainty sampling-based IDE systems rely on main memory to cache the entire dataset. For datasets that are larger than the main memory, a subset of data objects will need to be sampled from the original dataset on secondary storage (e.g., [7]). While simple and intuitive, this approach could easily lead to very inaccurate results and a waste of user effort since the boundaries of the interesting data regions in the sampled space is likely to be different from the original space [9]. Moreover, small sets of relevant data regions may even be ignored in the resulting sample set.

To overcome this problem, we propose a novel indexing mechanism coined *Uncertainty Estimation Index* (UEI), to facilitate the interactivity and scalability of active learning-based IDE systems. To our knowledge, UEI is the first approach in extending the scalability of active learning-based IDE systems beyond the main memory capacity.

Published in Proceedings of the 24th International Conference on Extending Database Technology (EDBT), March 23-26, 2021, ISBN 978-3-89318-084-4 on OpenProceedings.org.

Instead of relying solely on the main memory to cache the whole dataset during the exploration, UEI enables caching in memory only the necessary subsets of the data that are needed by the current stage of the exploration. This is achieved through the combination of an effective estimation of the uncertainty of each data object and an efficient data storage mechanism. The essential observation that UEI is based on is that data objects often have additional information that can be used to infer their relationship with other objects, one of which is the *similarity* among data objects. Since the uncertainty essentially represents its distance to the current decision boundary in the high-dimensional data space, thus the uncertainty of an object x is strongly related to the uncertainty of the surrounding objects [14]. This observation allows UEI to informatively select the set of highly uncertain data objects to be loaded into memory before it is needed by the predictive model. Hence, the amount of memory needed to explore a dataset in real-time is significantly reduced.

To evaluate UEI, we employed a state-of-the-art data exploration system REQUEST [9], and compare the performance of UEI against MySQL, which is used by the existing interactive data exploration systems [7, 12]. Our results using a large real-world dataset, namely the Sloan Digital Sky Survey (SDSS) [1], show that UEI outperforms existing solutions by more than 50X in runtime efficiency when the size of the dataset goes beyond the main memory capacity. Furthermore, UEI is well capable of meeting the sub 500 millisecond interactive response time requirement for data that is at least 100 times larger than the main memory capacity, while achieving minimum impact on the convergence of the predictive model.

It should be noted that even though UEI is designed for the IDE systems, where response time and run time efficiency is critical, it can also be used in combination with any active learning-based human-in-the-loop (HIL) applications. Examples of such human-in-the-loop applications including but not limited to: record matching [3], entity resolution [18, 19], and facts checking [4].

2 BACKGROUND

In this section, we provide the necessary background of our `UEI`.

2.1 Active Learning

Active learning refers to a set of approaches that aim to learn an accurate model with minimum labeled data for regression and classification tasks. One key component of active learning is the *query strategy* that sequentially selects the most informative unlabeled sample (i.e., data object) from the entire database to be labeled by the user.

In previous works, a number of active learning techniques have been proposed to define the "informativenes" of samples [20], including: *Uncertainty Sampling* [14], *Query-By-Committee* [21], *Expected Model Change* [5], *Expected Error Reduction* [22], and *Expected Model Output Change* [8]. These techniques are often interchangeable, providing the applications the flexibility to choose the most appropriate query strategy that fits their needs. Among the query strategies, *Uncertainty Sampling* [14] is the most commonly used because of its simplicity and efficiency [20].

Uncertainty Sampling is a query strategy that can be used with any probability-based predictive model (e.g., Naive Bayes, SVM, etc.). The intuition of *Uncertainty Sampling* is that patterns with high uncertainty are hard to classify, and thus, if the labels of those patterns are obtained, they can boost the accuracy of the classification models. Particularly, in binary classification models (e.g., with class labels 0 and 1), the most uncertain example $\mathbf{x}$ is the one which can be assigned to either class label $z(\mathbf{x})$ with probability 0.5. Inspired by the idea of uncertainty, also known as *least confidence* (lc), [14] proposes a measurement of uncertainty for binary classification models:

$$u^{(lc)}(\mathbf{x}) = 1 - p(\hat{y}|\mathbf{x}) \tag{1}$$

where $u^{(lc)}(\mathbf{x})$ is the uncertainty score with the least confidence measurement of $\mathbf{x}$, and $\hat{y}$ is the predicted class label of the unlabeled $\mathbf{x}$. Accordingly, after measuring the uncertainty of each unlabeled sample, the unlabeled sample with highest uncertainty is selected:

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}} u(\mathbf{x}) \tag{2}$$

where $u(\mathbf{x})$ can be any other measurement of informativeness over the unlabeled sample $\mathbf{x}$.

Algorithm 1 Typical Workflow of Active Learning-based IDE

```
Require: The raw data set $D$; Batch Size $B$
Ensure: A set of results $R$
 1: Labeled set $L \leftarrow \emptyset$
 2: Unlabeled set $U \leftarrow D$
 3: $M \leftarrow$ initialize query strategy
 4: while user continues the exploration do
 5:     for $i = 1$ to $B$ do
 6:         Choose one $x$ from $U$ using $M$
 7:         Solicit user's label on $x$
 8:         $L \leftarrow L \cup \{x\}$
 9:         $U \leftarrow U - \{x\}$
10:     end for
11:     $M \leftarrow$ trained with $L$ to update $M$.
12: end while
13: Return the set of results $R$ classified as positive by $M$.
```

2.2 Interactive Data Exploration

Active learning-based IDE systems can effectively find relevant items that are often undiscoverable using traditional search methods [7, 9, 10, 12]. In particular, active learning-based IDE systems do not require users to formulate any complex queries, nor does it need any form of description of the target items. The entire exploration can be done by answering simple binary (i.e., yes or no) questions. More importantly, active learning-based IDE systems can further be used to enhance traditional search results. For instance, they can be used to address the problem of having overwhelming results due to a broad query or search conditions.

As shown in Algorithm 1, a typical active learning-based IDE system works in the following steps: first it incorporates a query strategy (e.g., uncertainty sampling), which is used for selecting the example objects to be presented to the user for labeling (line 3). As long as the user is willing to label more examples (line 4), active learning-based IDE system will keep invoking the query strategy M to select a new example object x from D, and present it to the user to label it as *relevant* or *irrelevant* (lines 5-9). Once the amount of labeled samples received from the user reaches a sample batch size (denoted as B), which is a tunable parameter of the active learning-based IDE balancing the effectiveness and efficiency, then the classifier model employed by the query strategy will be updated according to the label assigned to x (line 11). In particular, the label assigned to x will be used for retraining the classifier model, which is an essential step towards selecting the object presented to the user in the next iteration. Once the iterative labeling process is completed, the obtained results will be presented to the user (line 13).

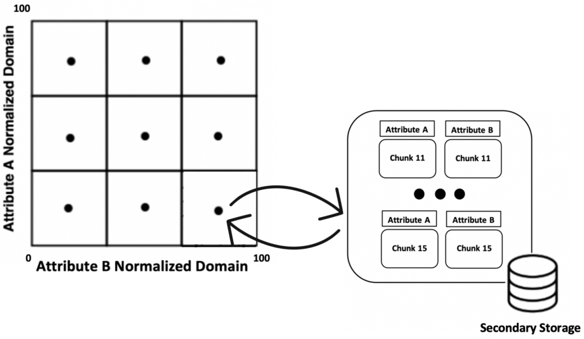


Figure 1: Illustrates UEI with a 2D data space, each grid represents a subspace, the dot in the center of each grid represents a symbolic point p. With chunks stored as separated files on the secondary storage.

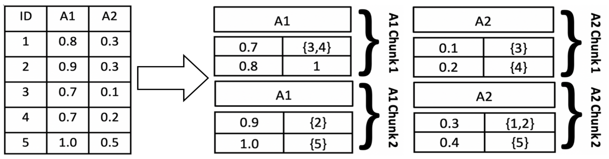
3 UNCERTAINTY ESTIMATION INDEX

Maintaining interactive response time for large datasets that are beyond the main memory capacity has always been one of the major challenges of active learning-based IDE systems. In tackling this problem, we focused our efforts on uncertainty sampling and proposed a novel index approach coined *Uncertainty Estimation Index* (UEI). Our UEI can be easily incorporated in any existing systems that leverage the uncertainty sampling and can be used in conjunction with any probabilistic-based classifiers as discussed in [14]. In the next section (Section 3.1) we provide the details of UEI's main component, and discuss how UEI leverages the inverted index-based data secondary storages to support scalable exploration. Following that, in Section 3.2, we provide a walkthrough of a complete example to illustrate how a typical active learning-based IDE workflow (i.e., Algorithm 1) performs when enhanced with UEI.

3.1 UEI Components

A key observation underlying UEI is that data objects often have additional information that can be used to inference their relationship with other objects, one of which is the similarity (i.e., distance) between data objects [14]. In other words, the uncertainty value of an object x is strongly related to the uncertainty of the surrounding objects. For example, if x is located near to one of the current decision boundaries, then x would have higher uncertainty due to a mixed set of relevant and irrelevant neighbors. Such continuity between data points is also generally assumed in both supervised and semi-supervised learning algorithms where data points that are closer to each other are more likely to share a label [23]. More precisely, we observed that the uncertainty of a data object x can be approximated through its spatial relationships with the labeled data objects. UEI explores this spatial relationship to load into memory the set of highly uncertain data objects before they are needed by the query strategy.

Specifically, as illustrated in Figure 1, the main idea of UEI is to divide the exploration space D into equal-size subspaces (i.e., d-dimensional grids) g_i's of D ($g_i \in D$), and build a set of symbolic (virtual) index points $P = \{p_1, ...p_c\}$, such that each index point p_i represents a subspace g_i. In each iteration, UEI estimates the uncertainty of each subspace based on the uncertainty of its corresponding index point p, then loads only the data in the subspace that is predicted to be most uncertain into memory. Conceptually, UEI is based on the same principle as hash-based multidimensional indexing, such as the traditional grid file structures, which splits the space into a non-periodic grid where one or more cells of the grid refer to a small set of points. As opposed to a grid file, designed to efficiently reference a single value with multiple keys, UEI is designed to scale out the uncertainty sampling by estimating the distribution of the uncertainty of data objects through the set of symbolic index points.



ID	A1	A2
1	0.8	0.3
2	0.9	0.3
3	0.7	0.1
4	0.7	0.2
5	1.0	0.5

Figure 2: Before storing the data, UEI vertically decompose the data into an inverted index form, and then store them in separate chunks.

To do so, UEI comprises five components: 1) an index set P of symbolic index points p_i; 2) a mapping method $m : p \longmapsto C$ that maps each index point p to a set of data chunk C; 3) a data cache U that caches a subset of uniformly sampled unlabeled data; 4) a set of labeled data L that contains all data that has been labeled by the user, and 5) the exploration dataset D stored in a fully inverted columnar format on a hard drive. The first four components reside in the main memory.

UEI divides the operation of exploration into two phases: an *Index Initialization* phase, and *Interactive Exploration* phase. The first phase only needs to be executed once per each new dataset. The second phase is specific to each exploration and discussed in the next section (Section 3.2).

Index Initialization Phase: As illustrated in Algorithm 2, to work with a new dataset, UEI first vertically (i.e., attribute-wise) decompose the whole data set and sort each dimension (i.e., attribute) based on the values in ascending order (lines 2 - 4). Since each dimension of a typical exploratory dataset (e.g., scientific, business analysis) can contain values of an arbitrary length, and one specific value for a dimension may appear multiple times, we compress the data by organizing it in a key-value fashion ($< key, \{values\} >$), where each value of the dimension would be used as a *key* and the ids of the corresponding objects as *values* (as illustrated in Figure 2). Note that for each exploration task, UEI stores all needed data in one location, thus when exploring data that are distributed in multiple locations (e.g., tables, files), the data needs to be merged before being utilized in the exploration.

During the process of storing the data, UEI splits the distinct values of each dimension d into a set of equal-sized data chunks $C^d = \{c^d{}_i, ..., c^d{}_u\}$, where each chunk will be stored as a separate file on the disk, and the size of each chunk can be adjusted based on the size of the data and the available hardware resources (line 5). UEI also ensures that the values of each dimension are stored in a sequential order, meaning values stored in each subsequence chunk $c^d{}_{i+1}$ will be larger than the values that have been stored in $c^d{}_i$ for efficient lookup.

Once the data are partitioned and stored on the disk, UEI would start construct the set of symbolic index points P by divide the original data space D into a set of equilateral d-dimensional subspaces $G = \{g_i, ..., g_j\}$, where $|G| = |P|$, then for each subspace g_i, UEI constructs a symbolic index point p_i that represents g_i by using the coordinates of the "virtual" center point of g_i (lines 7-11). To ensure UEI can be deployed in resource restricted environments, the number of symbolic index point can be adjusted based on the size of the dataset and the available hardware resources.

In order to construct and load each subspace g_i into memory, UEI employed a hash-based mapping method m that records for

Algorithm 2 Typical Exploration Flow with UEI

```
Require: The raw data set $D$
Ensure: Result set $T$
 1: $P \leftarrow \emptyset, L \leftarrow \emptyset, U \leftarrow \emptyset$
 2: $DC \leftarrow verticalDecompose(D)$
 3: for $d = 1$ to $|DC|$ do
 4:     $sort(DC_d)$
 5:     $C \leftarrow splitIntoChunks(DC_d)$
 6: end for
 7: $G \leftarrow splitIntoSubspaces(D)$
 8: for each grid $g_i \in G$ do
 9:     $p_i \leftarrow computeCenter(g_i)$
10:     $P \leftarrow P \cup \{p_i\}$
11: end for
12: $U \leftarrow sample(D, \gamma)$
13: $M \leftarrow$ initialize predictive model for uncertainty estimation
14: while user continues the exploration do
15:     drop any previously loaded data regions from $U$
16:     $M \leftarrow$ trained with $L$ to update $M$
17:     $P \leftarrow updateUncertainty(P, M)$
18:     $p_i^* \leftarrow$ choose the most uncertainty index point from $P$
19:     $g_i^* \leftarrow$ load data region with $m(p_i^*)$
20:     $U \leftarrow U \cup g_i$
21:     choose one $x$ from $U$ using $M$
22:     solicit user's label on $x$
23:     $L \leftarrow L \cup \{x\}$
24:     $U \leftarrow U - \{x\}$
25: end while
26: $T \leftarrow resultRetrieval(L)$
27: Return the set of interesting data objects $T$
```

each symbolic index point p_i, the set of chunks that are needed to construct g_i. As chunks are stored separately on the disk, this approach allows UEI to quickly identify the data that needs to be loaded. Since each data subspace g is stored as series of one-dimensional data chunks, to reconstruct each g when needed, UEI utilizes a hash table for efficiently merge of those data chunks. During the merge process, UEI iterates through each dimension and loads the corresponding chunks to the memory one at a time, and each entry in the chunk would be visited in a sequential manner. For each object ID that is recorded in a loaded data chunk c_i, the value associated with the ID will be inserted into the corresponding entry in the hash table. Once a chunk has been examined, UEI will release the memory space used to hold the data chunk and reuse the space for the subsequent chunk.

3.2 UEI in Action

In the previous section, we have discussed the components of UEI, as well as how the data are being stored and indexed, which essentially covers the first half (i.e., lines 1 - 11) of the exploration workflow, shown in Algorithm 2. In this section, we will discuss the interactive exploration phase of UEI, which illustrates how a typical active learning-based IDE task (i.e., Algorithm 1) can be performed when incorporating UEI (lines 12 - 27, Algorithm 2).

Interactive Exploration Phase: After the index set has been constructed, UEI begins the exploration by filling the unlabeled set U. Specifically, for the original data space D, UEI would uniformly sample a set of data from the underlying dataset (line 12), where the size of the samples γ can be adjusted based on the system hardware specs (e.g., available main memory size). As a result, a set of unlabeled objects U would be sampled and cached in the main memory. These unlabeled objects will then be used in the acquisition of the set of initial examples that will be labeled by the user to construct the initial predictive model M_0 for uncertainty estimation. Query strategy will randomly sample examples from U until the set of initial examples contains at least one positive example and one negative example (line 13).

In each iteration, UEI updates the uncertainty of all index points $p_i \in P$ based on the most recently trained predictive model M_{t-1} (line 17), which serves as the uncertainty estimator. Here the uncertainty of a data object is essentially equals to the probability of one object being either positive or negative class, with a value that equal to 50% being the most uncertain. Then, the index point p_i^* for which the current exploration model is most uncertain, will be chosen (line 18), such that:

$$p_i^* = \operatorname*{Argmax}_{p_i \in P} M_{t-1}(Y|p_i) \tag{3}$$

where Y = {0,1} is set of binary labels.

Based on the chosen p_i^*, UEI uses the mapping method m to identify and load (into the memory) all data chunks that correspond to the subspace g_i^*, which was represented by p_i^* (line 19). As mentioned earlier, the mapping method m is simply a hash table that maps a single index point p into a set of data chunks located on the disk. Later, the data of subspace g_i^* together with the unlabeled dataset U will be used by the query strategy (i.e., uncertainty sampling) in the selection of the example to be labeled in the current iteration (lines 20 - 22). To reduce memory usage, by default UEI kept only one uncertain data region g_i^* in the memory at any given time. Once the user is satisfied with the exploration result, the *resultRetrieval* method will be invoked to retrieve the exploration results and present them to the user (lines 26 - 27).

Tuning Interactive Exploration: In addition to the above typical exploration flow, UEI further allows the user to specify a response latency threshold σ that determines the latency between each exploration iteration (i.e., two subsequent examples). Using the user-specified σ, UEI determines whether or not to defer the swap between the current in-memory uncertain region g_i^* and the next uncertain region g_{i+1}^*, when g_i^* is no longer the most uncertain region.

In the case when an extremely low σ is specified that makes it impossible for the system to load the entire subspace g_i^* into main memory, UEI would start fetching the corresponding data chunks that associated with g_{i+1}^* (in the background) θ iterations before g_{i+1}^* is loaded into the memory. Here, θ is a tunable variable that can also be inferred based on the average loading time τ of data regions, and the configurable latency threshold σ, such that $\theta = \left\lceil \frac{\tau}{\sigma} \right\rceil$.

3.3 Time Complexity of UEI

Clearly, the time complexity of UEI is dominated by the interactive exploration phase. As discussed in Section 3.1, the initialization phase is done once for each dataset and the time required for UEI to prepare and store a dataset D on secondary storage is simply *linear* with respect to the number items n stored in D.

As discussed in Section 3.2, each iteration of the interactive exploration phase in UEI is dominated by the time taken to load the data from the chunks stored on the disk into the memory, which is *linear* with respect to the number of dimensions k and the number entries e stored in the loaded chunks. In contrast, each iteration in the current IDE approaches needs to load and examine all data items n ($e <<< n$).

Therefore, the time complexity of the UEI-enhanced data exploration generally is reduced from $O(kn)$ where n is the number of data objects to $O(ke)$ where e is the number entries associated

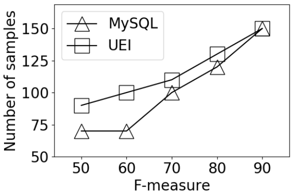


Figure 3: UEI Accuracy (Small Target Region).

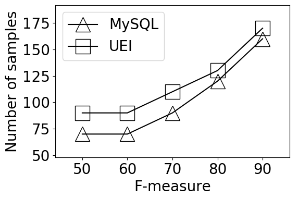


Figure 4: UEI Accuracy (Medium Target Region).

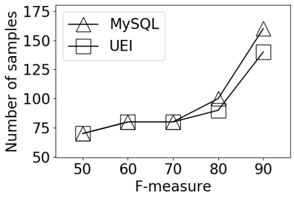


Figure 5: UEI Accuracy (Large Target Region).

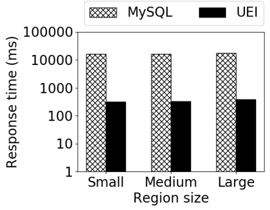


Figure 6: UEI Response Time.

Table 1: PARAMETERS

Number of runs per result	10
Number of dimensions (D)	5
Number of relevant regions	1
Cardinality of relevant regions	0.1% (S), 0.4% (M), 0.8% (L)
Uncertainty Estimator	DWKNN [11]
Label Type	Binary
Data Storage Engine	UEI, MySQL
Size of Individual Data Chunk	470KB
Number of Symbolic Index Points	3125
Latency Threshold	500ms
Performance Measurement	F-Measure (Accuracy)

with the loaded chunks for the current most uncertain subspace g_i^*.

4 EXPERIMENTAL EVALUATION

In our experimental evaluation of our UEI, we use REQUEST [9], a state-of-the-art IDE system, with two schemes, one incorporating UEI, and one utilizing MySQL, used in the existing IDE systems [7, 12]. After describing our experimental setup in Section 4.1, we present the findings of our experimental evaluation in Section 4.2.

4.1 Experiment Setup

Dataset: We used 40 GB of real-world dataset from Sloan Digital Sky Survey (SDSS) [1] that consists of 10×10^6 tuples.

IDE System: In our experiments, we employed our REQUEST [9] with traditional uncertainty sampling and the *dual weighted k-nearest neighbor* (DWKNN) [11] as the uncertainty estimator.

Environment: We implemented both REQUEST and UEI with Java JRE 1.7. All the experiments were run on a machine with Intel Core i7 Processor, 32 GiB RAM and 2 TB of NVMe SSD. The fast NVMe SSD was used to eliminate any potential bottleneck due to physical IO limitation. All experiments reported are averages of 10 complete runs. We have considered five numerical attributes *rowc, colc, ra, dec* and *field* of the PhotoObjAll table.

Target Interest Regions: The exploration task characterizes user interests and eventually predicts the relevant regions by iteratively gathering user labeled tuples. We experimented with 1 region per each exploration task. In addition, we vary the single region complexity based on the data space coverage of the relevant regions. Specifically, we categorize relevant regions to small, medium and large. Small regions have cardinality with an average of 0.1% of the entire experimental dataset, medium regions a cardinality of 0.4%, and large regions a cardinality of 0.8%. Furthermore, the dimensionality of the target interest regions is the same as the dimensionality of the dataset across the entire experiment.

User Simulation For experiment evaluation purpose, we simulate the user behavior using the following method. For each target interest region, we simulate the user by executing the corresponding range query to collect the exact target set of relevant tuples. We rely on this "oracle" set to assign confidence score p to the tuples we extract in each iteration based on their location in the data space against the target region.

More specifically, for each relevant region, there is a region center and a set of region widths, one for each dimension. We define the *maximum relative distance d* of an example against the region center as:

$$d = max_{i=1..l}(|x_i - c_i|/w_i) \tag{4}$$

where l is the dimension number, $|\cdot|$ is the absolute value operator, x_i, c_i and w_i are the attribute value of the example, of the center and region width in each dimension.

Parameters: Table 1 summarizes the important settings in the experiments.

4.2 Experiment Results

In our experiments we aimed to test UEI's ability to provide an interactive response time for datasets that are beyond the size of main memory capacity, and to illustrate the benefit of searching only a small set of cached objects with UEI against performing an exhaustive search over the entire dataset. In our experiments, we stored 10 million data items with both UEI and MySQL, and restricted the memory footprint for both UEI and MySQL to be within 400MB, which is ~ 1% of the entire dataset. Figures 3 to 6 illustrates the effectiveness and efficiency of our proposed UEI.

UEI Accuracy (Figures 3 to 5): Compared to MySQL, we have noticed that our proposed UEI requires more labeled examples in the early stage of the experiment (e.g., below 70% of accuracy). This is due to the fact that in the early stage, the classifier does not have enough training samples to learn an accurate uncertainty estimator (i.e., DWKNN classifier) that captures precisely the user's interesting regions. This causes the predictive model to select less informative examples to be labeled by the user. Since UEI uses uncertainty as the criteria for both the example selection and the loading of data regions (i.e., subspaces), therefore the negative effect of an inaccurate classifier has been magnified.

However, as the accuracy of the uncertainty estimator improves with more labeled examples, we observed a significant boost in performance of UEI in the later stages (e.g., above 80% accuracy) of the exploration. This is expected as the the predictive model gets more and more accurate with respect to the decision boundaries, and thus can estimate more accurate uncertainties. Since UEI rely on the uncertainty estimation for both the query strategy and cache management, therefore, it benefits more noticeably than the MySQL, which only rely on uncertainty estimation for query strategy.

UEI Response time (Figure 6): Finally, we have measured the response time for both UEI and MySQL based schemes. As shown in Figure 6, UEI achieves 50*x* faster response time than MySQL, and ensures the sub-second interactive response time across all data region sizes. Note the response time remains the same across all three target interest regions sizes, which is as expected because the runtime complexity of the uncertainty sampling-based systems only depends on the size of the dataset and not the size of the target interest regions.

From the experiments, it is clear that due to the fact that uncertainty sampling requires an exhaustive search over the entire data spare, thus the physical bandwidth of the secondary storage has become the major bottleneck that severely limits the scalability of active learning-based IDE systems.

Even though in our experiments, we have used NVMe based SSD with I/O throughput of around 3.4GB/s, the uncertainty sampling still takes over 12 seconds to complete the exhaustive search in each iteration. Therefore, it is still impossible to explore datasets that exceed the main memory capacity without UEI.

5 RELATED WORK

Traditionally, indexing has been the core technique for optimizing response time in database systems. Recently, main-memory indexing and specialized access methods have been proposed to support domain-specific query processing and analytics (e.g., [6, 17]). UEI is based on similar principles as these specialized access methods. However, UEI, to the best of our knowledge, is the first domain-specific access method with in-memory and disk components that support interactivity and scalability of active learning-based IDE systems.

The active learning-based IDE systems that can leverage our proposed UEI for better scalability includes *REQUEST* [9], the first active learning-based IDE system, and the two more recently proposed systems, *Dual-Space Model* [12] and *ExNav* [10]. REQUEST utilizes a two stages approach; a data reduction stage aims to selectively reduce the search space while keeping all relevant data regions, and a query selection stage that utilizes an active learning-based predictive model to iteratively improve the accuracy of the constructed exploratory query through interactions with the user. Dual-Space Model uses a new uncertainty sampling-based predictive model and a new dual-space pruning technique that focuses solely on exploration tasks with a single relevant region. It also optimizes these tasks for faster model convergence. ExNav is the first uncertainty sampling-based IDE system that specializes in exploring a variety of unstructured data sets by leveraging the corresponding data embedding methods for each unstructured data type [10].

In addition to the active learning-based IDE systems, UEI can also be utilized in other active learning-based *Human-in-the-loop* (HIL) systems. For example, in [18], the authors have proposed an active learning-based HIL system, called SystemER, for learning Entity Resolution models through user interactions. Another example of an active learning-based HIL application is fact-checking. In [4], the author has proposed an effective active learning-based HIL system for identifying various types of potentially misleading or false information for news contents. Most recently, [19] has proposed to use active learning for learning the implicit structured representations of entity names, which can be useful for many entity-related tasks such as entity normalization and variant generation. To facilitate the process of learning such structured representations, a user-friendly interface called PARTNER has been designed to enhance the user's interaction experience.

6 CONCLUSION

In this paper, we present the *Uncertainty Estimation Index* (UEI), the first indexing mechanism that enables active learning-based IDE systems to explore datasets that exceed the main memory capacity. Instead of requiring all data to be loaded into the main memory as in existing active learning-based IDE systems for sub-second respond times, UEI enables the scalability by dynamically identify and caching the set of objects that are most uncertain to the current stage of the exploration. It achieves this by maintaining a small in-memory index that estimates the aggregated uncertainty value of the data items in the entire data subspaces. UEI also employees columnar-based secondary storage and combines it with an inverted index to support efficient loading of the necessary data items at each iteration.

Our experimental evaluation using real-world data show that a state-of-the-art IDE systems using UEI greatly outperforms a DBMS-based version of the same IDE system by provide more than 50x runtime efficiency when the size of the dataset exceed the main memory capacity, and is capable of achieving sub-second response time for data that is 100 times larger than the available memory while achieving the desired exploration accuracy and effectiveness. In conclusions, UEI can be used not only with existing active learning-based IDE but with other active learning-based Human-in-the-loop systems as well to achieve significantly higher scalability by removing the main memory restriction.

REFERENCES

[1] SDSS Samples Queries - http://cas.sdss.org/dr4/en/help/docs/realquery.asp.

[2] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis. 2010. An Optimization Framework for Query Recommendation. In *ACM WSDM*. 161–170.

[3] A. Arasu, M. Götz, R. Kaushik. 2010. On active learning of record matching packages. In *ACM SIGMOD*. 783–794.

[4] S. Bhattacharjee, A. Talukder, B. Balantrapu. 2017. Active learning based news veracity detection with feature weighting and deep-shallow fusion. In *IEEE BigData*. 556–565.

[5] W. Cai, Y. Zhang, J. Zhou. 2013. Maximizing Expected Model Change for Active Learning in Regression. In *IEEE ICDE*. 51–60.

[6] G. Chatzigeorgakidis, D. Skoutas, K. Patroumpas, T. Palpanas, S. Athanasiou, S. Skiadopoulos. 2019. Local Similarity Search on Geolocated Time Series Using Hybrid Indexing. In *ACM SIGSPATIAL*. 179–188.

[7] K. Dimitriadou, O. Papaemmanouil, Y. Diao. 2016. AIDE: An Active Learning-Based Approach for Interactive Data Exploration. In *TKDE*, 28:2842–2856.

[8] A. Freytag, E. Rodner, J. Denzler. 2014. Selecting Influential Examples: Active Learning with Expected Model Output Changes. In *ECCV*. 562–577.

[9] X. Ge, Y. Xue, Z. Luo, M. Sharaf, P. Chrysanthis. 2016. REQUEST: A Scalable Framework for Interactive Construction of Exploratory Queries. In *IEEE BigData*. 4566–4579.

[10] X. Ge, X. Zhang, P. Chrysanthis. 2020. ExNav: An Interactive Big Data Exploration Framework for Big Unstructured Data. In *IEEE BigData*.

[11] J. Gou, L. Du, Y. Zhang, T. Xiong, et al. 2012. A new distance-weighted k-nearest neighbor classifier. In *J. Inf. Comput. Sci*, 9(6):1429–1436.

[12] E. Huang, L. Peng, L. Palma, A. Abdelkafi, A. Liu, Y. Diao. 2019. Optimization for active learning-based interactive database exploration. In *VLDB*. 71–84.

[13] S. Islam, C. Liu, R. Zhou. 2013. A framework for query refinement with user feedback. In *J. Syst. Softw., 86(6):1580-1595*.

[14] D. Lewis W. Gale. 1994. A sequential algorithm for training text classifiers. In *ACM SIGIR*. 3–12.

[15] Z. Liu , J. Heer. 2014. The Effects of Interactive Latency on Exploratory Visual Analysis. In *IEEE TVCG*, 20(12):2122–2131.

[16] B. McCamish, V. Ghadakchi, A. Termehchy, B. Touri, and L. Huang. 2018. The Data Interaction Game. In *ACM SIGMOD*. 83–98.

[17] B. Peng, P. Fatourou, T. Palpanas. 2020. MESSI: In-Memory Data Series Indexing. In *IEEE ICDE*. 337–348.

[18] K. Qian, L. Popa, P. Sen. 2019. SystemER: A Human-in-the-loop System for Explainable Entity Resolution. *PVLDB* 12, 12, 1794–1797.

[19] K. Qian, P. Raman, Y. Li, L. Popa. 2020. Learning Structured Representations of Entity Names using Active Learning and Weak Supervision. *CoRR* abs/2011.00105.

[20] B. Settles. 2009. *Active learning literature survey*. Technical Report. University of Wisconsin-Madison.

[21] H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by Committee. In *ACM Workshop on Computational Learning Theory*.

[22] Y. Zhang, Y. Wang, W. Cai, S. Zhou, Y. Zhang. 2017. From Theory to Practice: Efficient Active Cost-sensitive Classification with Expected Error Reduction. In *SIAM*. 153–161.

[23] Zheng Zhao and Huan Liu. 2007. Semi-supervised Feature Selection via Spectral Analysis. In *SDM*. 641–646.