Ranking Mutual Information Dependencies in a Summary-based Approximate Analytics Framework

Dominik Ślęzak*, Janusz Borkowski† and Agnieszka Chądzyńska-Krasowska‡
*Institute of Informatics, University of Warsaw
ul. Banacha 2, 02-097 Warsaw, Poland
Email: slezak@mimuw.edu.pl
†Security On-Demand
12121 Scripps Summit Dr 320, San Diego, CA 92131, USA
Email: janusz.borkowski@securityondemand.com
‡Polish-Japanese Academy of Information Technology
ul. Koszykowa 86, 02-008 Warsaw, Poland
Email: honzik@pjwstk.edu.pl

***Abstract*—We continue our research on utilizing histogram-based data summaries in approximate derivation of mutual information scores in large relational data sets. Our methodology of creating, storing and using summaries has been designed for the purpose of developing an approximate database engine that is currently deployed commercially in the area of cyber-security data analytics. However, a similar idea of approximate data processing operations can be considered also in other fields, including machine learning whereby heuristic calculations are a component of many methods. In this paper, we focus on investigation of one possible source of inaccuracy of our previously proposed approach to approximating mutual information – that is, neglecting a kind of column domain drift during distributed summary-based computations. We illustrate it using an artificially created benchmark data set and we discuss how to cope this particular challenge in the future.**

***Index Terms*—Approximate Data Processing, Granulated Data Summaries, Approximate Mutual Information**

I. Introduction

A growing need for scalable solutions for machine learning and business intelligence exists in the area of applications involving large machine-/service-generated data sets. Most research teams and companies address this challenge by scaling out computational power even though this strategy is increasingly inefficient. At the same time, people begin to realize that some computational tasks could be performed in approximate fashion, which leads toward opportunity of working with tradeoffs between the speed, the resource consumption and the accuracy of outcomes of data operations.

In [1], we presented a new approximate database engine attempting to take advantage of such tradeoffs. Our engine captures information upon data load, in form of one- and two-dimensional summaries. It composes groups of data rows (called *packrows*, containing 2^{16} rows each by default) incoming into a given data table and summaries are built for each group separately. The engine stores only such *granulated* summaries without assuming any access to the original data. SQL queries are executed directly on summaries, i.e., each data operation (filtering, aggregating, etc.) *transforms* summaries of its input directly into summaries of its output.

The considered engine is designed to perform on petabytes of the summarized data. In [1], one can find empirical comparison of the speed of our style of approximate calculations versus state-of-the-art methods of scaling by means of adding computational resources. In [2], we reported the current major commercial deployment of our engine in the field of online cyber-security, whereby ad-hoc analytical queries need to be executed against data sets containing detailed event logs growing with intensity of over 300 billions of new rows per month. In both works, we emphasized that our ultimate goal is to integrate the proposed methodology with aforementioned state-of-the-art solutions, so it is possible to work with granulated summaries representing pairwise disjoint pieces of the data in a fully parallel/distributed environment.

In another thread of research, we have examined whether our approach based on approximate summary transformations could be applied in data mining. In [3], we investigated how to utilize granulated data summaries to boost basic methods of feature selection. We also worked on exposing summaries in form of metadata tables that can be accessed using standard SQL for the purposes of visual data analytics [4]. As the underlying data operations performed during database querying and data mining are quite analogous to each other, we expect observing comparable speed-up and scalability capabilities. On the other hand, there is still a lot to be done with regard to better understanding and measuring (in)accuracy of approximate results of data mining and data analytics processes.

In this article, we continue aforementioned research and focus on a task of approximate derivation of entropy-based mutual information scores for pairs of data columns. Herein, there are two sources of inaccuracy. First, for each packrow – a fragment of the original data set – we store only quantized and partial characteristics of columns and their interdependencies. Thus, any reference to pairwise probability distributions may yield imperfect results. On the other hand, the good news is that slight imperfections may not invalidate decision-making processes based on approximate scores, with an expected advantage of significant acceleration of calculations.

978-1-5386-7879-4/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCS.2018.00137

The second source of inaccuracy refers to the fact that the technique considered in [3] is fully distributed, whereby each packrow (or rather its summary) quickly produces its own set of scores and then, local scores are averaged for a given pair of data columns over all packrows. Packrow-level computations of mutual information are currently implemented in our approximate database engine for the purpose of spanning locally optimal belief propagation trees that serve as the basis for SQL filtering [1]. For a given query, trees maximizing overall mutual information (summed over their edges) can take different forms for different packrows. This diversity provides us with more accurate approximate query execution than it would be a case for belief propagation trees optimized globally for the whole data. However, if the task is to find pairs of data columns that are interdependent globally, then such purely local calculations may suffer from mistakes.

This study serves as a step toward understanding cases when aforementioned mistakes are most visible. Our hypothesis is that the latter above type of inaccuracy corresponds to a "domain drift", which is evolution of single-column summarized characteristics from packrow to packrow. According to the presented experimental results, this is indeed valid observation – for columns whose values are distributed across packrows in a *uniform* way, their approximated mutual information scores (linking them with other columns) are relatively more reliable than for columns whose values are *drifting* along the data. At this stage of our research, we do not present a complete solution to this problem yet. However, we discuss whether it is possible to achieve it by enriching our current model of distributed approximation of mutual information by additional calculations conducted over an aggregated table gathering together simplified summaries of all packrows.

From a general perspective of parallel and distributed computing, our current approach to approximating global (table-level) mutual information is a typical example of decomposing a given task onto pre-arranged data fragments and then, aggregating local outcomes in quite a naïve way. On the other hand, the discussion triggered by our experiments serves as a guideline how to make that second phase of calculations more sensitive with respect to the data. It is worth noting that further development at this level would require us to pay attention also to the first aforementioned source of inaccuracy – operating with quantized column domains. Indeed, calculations on an aggregated table representing all packrows would need to be preceded by assembling global quantization of a domain of each single column. This can be done by merging one-column summaries available for particular packrows into global representations – a mechanism that is already used inside our approximate database engine for other purposes.

The paper is structured as follows. In Section II, we recall our methodology of deriving meaningful summaries of the original data. In Section III, we refer to our previous studies on approximating mutual information. In Section IV, we present new experiments on an artificially generated data set. In Section V, we discuss how to make our current approach more accurate. In Section VI, we conclude the paper.

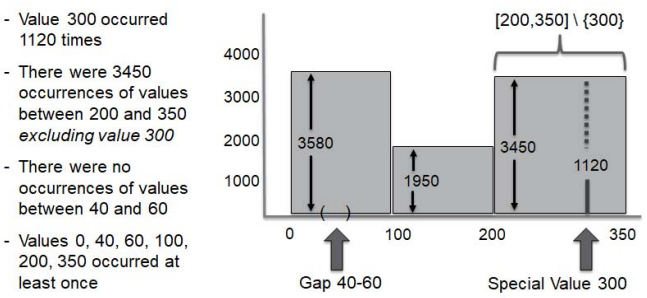

Fig. 1. Summarized content of a single data pack corresponding to a single numeric column, over a single packrow. Parts of quantized data pack's domain take form of complements (such as $[200, 350] \setminus \{300\}$) and special values (such as 300) annotated with frequencies derived from the original data.

II. Background on Granulated Data Summaries

There are many methods to develop approximate databases. Quite often, results are estimated by running queries on data samples [5]. However, for big data sets, good-quality samples need to be big as well, which limits query acceleration capabilities. The second category of methods is based on summaries (sketches, etc.) [6]. Our approach drops into this category, as it forms summaries represented as extended histograms. There is a long tradition of using histograms in the realm of database engines, with a lot of effort devoted to updating histogram structures while loading new data. In our approach, separate summaries are built for subsequently loaded chunks of rows – so-called *packrows*. Hence, newly loaded packrows do not interfere with previously captured summaries.

Figure 1 shows how the domain of a given numeric column (alphanumeric columns are still under investigation), within a given packrow, can be quantized onto ranges and exceptions, called *special values*. There is a lot of work behind designing heuristics that choose ranges and special values for particular *data packs*, i.e., collections of values of a single column within a single packrow [1], [3]. Technically, we operate on *complements*, i.e., ranges with special values excluded. As a result, for each column within a packrow we obtain its local domain partition, with its parts annotated with frequencies of rows having the corresponding values. Besides special values and complements there are also gaps, although they have no influence on experiments conducted in this paper.

Figure 2 and Table I represent complete summary contents. Besides single-pack parts and their frequencies, the key aspect is to derive and store the most meaningful co-occurrences (correlations, interdependencies) involving pairs of columns in particular packrows. For packrow t and columns a and b, let us refer to a's and b's parts (special values or complements) using iterators i and j, respectively. Let $P_t\left(part_t^a[i]\right)$, $P_t\left(part_t^b[j]\right)$ and $P_t\left(part_t^a[i], part_t^b[j]\right)$ denote probabilities of occurrence of a's values within its i-th part, b's values within its j-th part and pairs of a's and b's values within their i-th and j-th parts, respectively. We define co-occurrence ratios as:

$$\tau_t\left(part_t^a[i], part_t^b[j]\right) = \frac{P_t\left(part_t^a[i], part_t^b[j]\right)}{P_t\left(part_t^a[i]\right) P_t\left(part_t^b[j]\right)} \qquad (1)$$

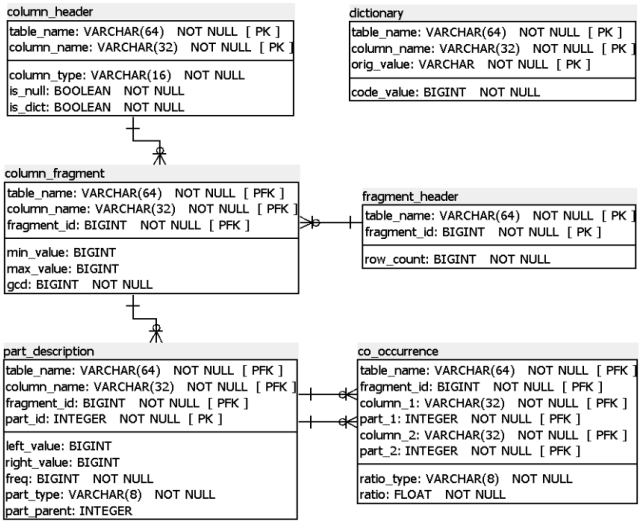

Fig. 2. A revised (comparing to [1], [4]) metadata schema representing our granulated summaries. For data sets stored in the engine, their corresponding metadata can be accessed via PostgreSQL-like interface as virtual tables.

In [3], we investigated several methods of on-load evaluation, which co-occurrences are most worth storing. In [2], we tested how different "budgets" for the number of stored ratios (and other parameters) can influence the speed and accuracy of query execution. Generally, we assume that a complete map of co-occurrences does not need to be maintained. Instead, pairs that were not evaluated as meaningful during data load can be estimated using default ratio $\widetilde{\tau}_t\left(a,b\right) =$

$$= \frac{1-\sum_{\text{stored ratios}} P_t(part_t^a[i])P_t\left(part_t^b[j]\right)\tau_t\left(part_t^a[i],part_t^b[j]\right)}{1-\sum_{\text{stored ratios}} P_t(part_t^a[i])P_t\left(part_t^b[j]\right)} \tag{2}$$

where "$\sum_{\text{stored ratios}}$" means the sum over all combinations of parts indexed by i and j such that $\tau_t\left(part_t^a[i], part_t^b[j]\right)$ has been chosen to be stored by the engine.

In the next sections, we will show several examples how to use above-outlined structures in approximate analytics (both in our already-existing approximate database engine and in new approaches to approximate data mining). However, let us emphasize that our overall methodology makes practical sense only if granulated summaries of the original data are produced fast enough. From software architecture perspective, our solution comprises two fully separate layers, responsible for: 1) distributed and asynchronous acquisition of summaries and 2) utilization of already-stored summaries to run approximate operations. The first layer is supposed to look efficiently through potentially distributed and heterogeneous data sources, leaving the actual data in-place. Given that summaries of particular packrows can be computed independently from each other, this phase can be highly optimized [7]. Still, even from perspective of a single packrow, aforementioned quantization and ranking-based calculations require significant effort and, therefore, it will be always important to investigate new computational techniques at this level [8].

TABLE I
HIGH-LEVEL DESCRIPTION OF METADATA TABLES AND THEIR COLUMNS CURRENTLY SUPPORTED BY OUR APPROXIMATE DATABASE ENGINE.

Table / Column	**Description of Contents**
column_header	Basic information about columns
table_name	Table name
column_name	Column name
column_type	Column type
is_null	Can column have null values
is_dict	Are column's values replaced by dictionary codes
dictionary	Information about dictionary codes
table_name	Table name
column_name	Column name
orig_value	Original column's value
code_value	Its corresponding code used in data summaries
fragment_header	Basic information about packrows
table_name	Table name
fragment_id	Packrow's ordinal number in table
row_count	Amount of original rows represented by packrow
column_fragment	Basic information about data packs
table_name	Table name
column_name	Column name
fragment_id	Packrow's ordinal number in table
min_value	Minimum value occurring in data pack
max_value	Maximum value occurring in data pack
gcd	Greatest common divisor for values in data pack
part_description	Detailed information about data pack summaries
table_name	Table name
column_name	Column name
fragment_id	Packrow's ordinal number in table
part_id	Part's ordinal number (unique in data pack)
left_value	Minimum value occurring in part
right_value	Maximum value occurring in part
freq	Amount of rows with values inside part's domain
part_type	Special value / complement / gap
part_parent	For a value or gap, *id* of a range it belongs to
co_occurrence	Information about co-occurrence ratios
table_name	Table name
fragment_id	Packrow's ordinal number in table
column_1	First column in co-occurrence ratio
part_1	*id* of part corresponding to column_1
column_2	Second column in co-occurrence ratio
part_2	*id* of part corresponding to column_2
ratio_type	Type of ratio (equations (1) or (2))
ratio	Ratio for pair (part_1,part_2) in given packrow

Finally, let us emphasize that summaries described in this section differ slightly from those introduced in [1]. In our original approach, co-occurrence ratios were evaluated at two different levels of hierarchy: pairs of special values and pairs of ranges – but not "range minus special values" complements. (When looking at Figure 1, it would mean counting rows with values in $[200, 350]$ instead of $[200, 350] \setminus \{300\}$.) Then, the engine had to synchronize calculations at both levels of most meaningful co-occurrences. On the other hand, representations referred in Figure 2 and Table I are flattened, forming single-layer partitions of local column domains. Such partitions are much easier to handle, both during data load and any later computations. As a side effect, the engine can now store (if it decides to do so) a co-occurrence ratio linking a special value with a complement defined over another column.

III. Approximating Mutual Information

There are many aspects in which our approximate database engine development corresponds to machine learning research. For instance, our approach to deriving quantizations of local data column domains within subsequent packrows could be compared to start-of-the-art discretization methods [9]. We will go back to this aspect later in this section.

As another example, the approximate query mechanism introduced in [1] refers strongly to so-called probabilistic graphical models [10]. Namely – as already mentioned in Section I – for any SQL statement with WHERE conditions, the engine constructs an internal tree-based scheme allowing it to propagate influence of those conditions on one-column representations of all data columns involved in the statement. Such trees are spanned over nodes symbolizing columns and they can be optimized for each packrow separately.

Now, in order to span a tree for packrow t, one can rely on well-known idea of maximizing its joint mutual information. If we had full access to packrow's contents, then such local mutual information for columns a and b would take a form of $I_t(a,b) = \sum_{v_a,v_b} P_t(v_a,v_b) \log \frac{P_t(v_a,v_b)}{P_t(v_a)P_t(v_b)}$, where v_a and v_b denote original values of a and b, respectively. In our quantized version, $I_t(a,b)$ could be rewritten as $\sum_{i,j} P_t\left(part_t^a[i], part_t^b[j]\right) \log \tau_t\left(part_t^a[i], part_t^b[j]\right)$, whereby $P_t\left(part_t^a[i], part_t^b[j]\right)$ could be further replaced with $P_t\left(part_t^a[i]\right) P_t\left(part_t^b[j]\right) \tau_t\left(part_t^a[i], part_t^b[j]\right)$. However, given limited information about pairwise probability distributions stored in our framework, we can only approximate it using coefficients in equations (1) and (2):

$$\begin{aligned}\widetilde{I}_t(a,b) = &\sum_{\text{stored ratios}} P_t\left(part_t^a[i]\right) P_t\left(part_t^b[j]\right) \\ &\tau_t\left(part_t^a[i], part_t^b[j]\right) \log \tau_t\left(part_t^a[i], part_t^b[j]\right) \\ &+ \log \widetilde{\tau}_t(a,b) \times \Big(1 - \sum_{\text{stored ratios}} \\ &P_t\left(part_t^a[i]\right) P_t\left(part_t^b[j]\right) \tau_t\left(part_t^a[i], part_t^b[j]\right)\Big)\end{aligned}$$

Given the observed efficiency of the above approach in approximate querying, we started to consider introducing the same style of calculations in other areas. Inspired by an overview in [11], we decided to develop summary-based techniques that could be useful for basic data exploration and machine learning. The first step was to adapt the above way of approximating mutual information for the purpose of accelerating classical minimum redundancy maximum relevance (mRMR) feature selection [12]. However, in this case we needed to operate with global mutual information referring to the whole data table – not its particular fragments represented by separate packrows. In [3], we used the following naïve technique to estimate global mutual information for columns a, b in table T (where N denotes the amount of packrows in T):

$$\widetilde{I}(a,b) = \tfrac{1}{N} \sum_{t=1}^{N} \widetilde{I}_t(a,b) \tag{3}$$

To assess reliability of formula (3), we conducted comparative analysis of mRMR outcomes produced using approximate and exact modes of calculating mutual information. We investigated a data set containing several millions of network transmissions, obtained from a company developing tools for early detection of viruses and worms. Our goal was to identify features characterizing suspiciously large transfers.

In order to calculate exact variant of mutual information measure I, we first discretized the data set using the same procedure as the one applied in our approximate engine to identify meaningful ranges and special values for particular packrows. Thus, for each column, we quantized a single "big data pack" representing the whole column's content. Then, we calculated I for each pair of discretized columns.

We analyzed orderings of additions of columns to feature sets constructed by mRMR algorithm. Although final outcomes differed from each other, one could see that by operating with approximate $\widetilde{I}$ instead of exact I, mRMR could still produce useful feature sets. On the other hand, a disadvantage of mRMR is that its outputs can be sensitive with respect to heuristic choices of features at early stages of selection process. Hence, in Section IV we conduct experiments focusing simply on information scores, without applying them as inputs to any more sophisticated algorithms.

It is also worth noting that we thoroughly examined our quantization procedure from the perspective of its expressive power, including its comparison with other data discretization techniques [9]. In our approach, domain of a given column (within a given packrow) is first split onto eight equal-length intervals in order to assure that all its areas are described in sufficiently detailed way. Then, each of such intervals is partitioned onto eight smaller buckets supported by roughly uniform (within the given interval) number of original rows. In the meantime, a certain amount of special values is identified. This way, we obtain one-column domain characteristics that can be useful for our internal engine mechanisms and – on top of that – for external data representation and visualization purposes [4]. Therefore, aforementioned experiment based on utilizing our quantization algorithm also at the level of full data makes sense from practical perspective.

IV. Experiments with "Car_Sales" Data Set

Experiments reported in [2] and [3] – referring to tuning parameters of our data summaries from the perspectives of, respectively, approximate query and feature selection accuracies – were conducted on proprietary real-world data sets. Although such empirical studies let us validate and improve our approach in various scenarios, it is not fully possible to discuss freely their outcomes. Thus, in this paper, we refer to our own artificially created data set called "car_sales".

The "car_sales" data set was used for the first time while testing our previous database engine, now known as *Infobright DB*, which was based on utilizing (a different version of) granulated data summaries to accelerate classical data analytics [13]. In particular, we used "car_sales" in [14] to investigate influence of intelligent stream-based data reorganization processes on the *quality* of the resulting data summaries.

Figure 3 displays the "car_sales" schema. In this paper, we used a relatively small subset containing 100×2^{16} rows (therefore represented by 100 summarized packrows) obtained by joining all tables into a fully denormalized form.

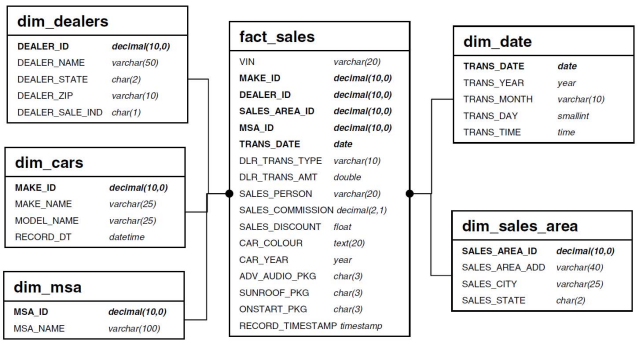

Fig. 3. The "car_sales" database – an artificial data set (with 1,000,000,000 rows in the fact table) used in [14]. In this paper, we consider its denormalized version and test the accuracy of approximate calculation of mutual information measure against the subtable consisting of its first 100×2^{16} rows.

Table II shows mutual information rankings obtained for pairs of "car_sales" columns using exact and approximate modes of calculations. In both cases, we display 50 most strongly interdependent column pairs a-b, ordered descending by $I(a, b)$ and $\widetilde{I}(a, b)$, respectively. As in Section III, exact computation of I was preceded by *global* discretization of the original data set using the same method as the one applied in our approximate database engine *per-packrow*, in order to derive special values and range complements [1], [4].

We focus on rankings – not specific quantities – as this is the key aspect of comparing different pairs of columns while, e.g., conducting feature selection processes. When looking at Table II, for the set of columns with identifiers and time-specific attributes excluded, we can see that general tendencies in score orderings obtained using both variants of calculations are quite comparable to each other. In particular, top four interdependencies in both scenarios are the same.

On the other hand, there are also significant differences. One of them refers to columns whose value ranges vary most often when looking at different packrows, i.e., so-called "time-dependent" columns (where "time" is interpreted as natural flow of rows as they are loaded into a database). We identified such columns by measuring their exact mutual information I with respect to an artificial column indicating ordinal numbers of packrows that particular rows belong to (values from 1 to 100). Outcomes of such scoring are reported in Table III. We denote the column with packrow indicators as *pack*.

One can see that – as we hypothesized in Section I with regard to the second source of inaccuracy in our approach – mutual information scores involving "time-dependent" columns tend to be relatively weaker in approximate variant of computations than in its exact counterpart. For better visualization, we bolded in both tables 10 columns most strongly correlated with packrow ordinal numbers. Clearly, mutual information scores for such columns are underestimated comparing to the others during considered $\widetilde{I}$-approximate calculations.

TABLE II
RANKINGS OF THE MOST STRONGLY INTERDEPENDENT COLUMN PAIRS COMPUTED IN THE EXACT AND APPROXIMATE MODES FOR THE 100×2^{16}-ROW DENORMALIZED SUBSET OF THE "CAR_SALES" DATABASE. MOST "TIME-DEPENDENT" COLUMNS ARE BOLDED. TIME-SPECIFIC COLUMNS AND IDENTIFIERS WERE EXCLUDED FROM THE STUDY.

rank based on exact I-scores	*rank based on approx. $\widetilde{I}$-scores*
model_name-record_dt	**dealer_name-dealer_zip**
make_name-**record_dt**	make_name-**model_name**
make_name-**model_name**	**model_name-record_dt**
dealer_name-dealer_zip	make_name-**record_dt**
dealer_name-dealer_state	dlr_trans_type-sales_commission
dealer_state-**dealer_zip**	dlr_trans_type-**sales_discount**
sales_discount-sales_person	dealer_state-**dealer_zip**
sales_discount-sales_commission	**dealer_name**-dealer_state
sales_person-sales_commission	sales_commission-**sales_discount**
sales_discount-dlr_trans_type	**sales_person**-sales_commission
dlr_trans_type-sales_commission	**sales_person-sales_discount**
dlr_trans_type-**sales_person**	dlr_trans_type-**sales_person**
sales_city-sales_state	**sales_city**-sales_state
sales_area_add-sales_city	dealer_state-dealer_sale_ind
dealer_name-dealer_sale_ind	**dealer_zip**-dealer_sale_ind
dealer_zip-dealer_sale_ind	**dealer_name**-dealer_sale_ind
sales_area_add-sales_state	dlr_trans_type-car_year
dealer_state-dealer_sale_ind	dealer_sale_ind-car_year
record_dt-sales_area_add	car_year-onstart_pkg
dealer_name-sales_city	dlr_trans_type-car_colour
record_dt-dealer_name	car_year-sunroof_pkg
sales_city-sales_person	car_year-adv_audio_pkg
model_name-sales_area_add	car_colour-onstart_pkg
dealer_zip-sales_city	car_colour-sunroof_pkg
dealer_name-sales_person	dealer_state-dlr_trans_type
record_dt-sales_city	make_name-sunroof_pkg
model_name-dealer_name	make_name-dlr_trans_type
record_dt-dealer_zip	dealer_sale_ind-car_colour
sales_area_add-msa_name	dealer_state-sunroof_pkg
sales_discount-record_dt	sales_state-adv_audio_pkg
sales_discount-sales_city	car_colour-adv_audio_pkg
record_dt-sales_person	dealer_state-adv_audio_pkg
sales_discount-dealer_name	make_name-dealer_sale_ind
dealer_zip-sales_person	sales_commission-car_year
model_name-sales_city	**sales_person**-car_year
sales_discount-sales_area_add	**sales_discount**-car_year
dlr_trans_amt-record_dt	sales_state-dlr_trans_type
model_name-dealer_zip	sales_state-onstart_pkg
record_dt-msa_name	make_name-onstart_pkg
model_name-sales_person	**sales_person**-car_colour
sales_city-msa_name	sales_commission-car_colour
sales_discount-model_name	**sales_discount**-car_colour
sales_discount-dealer_zip	dealer_state-onstart_pkg
dealer_name-msa_name	make_name-adv_audio_pkg
msa_name-sales_person	dealer_sale_ind-sales_state
dealer_name-sales_area_add	sales_state-sunroof_pkg
dlr_trans_amt-model_name	dealer_state-**sales_discount**
dealer_zip-msa_name	dealer_state-sales_commission
dlr_trans_amt-msa_name	dealer_state-**sales_person**
dealer_zip-sales_area_add	make_name-sales_commission

V. DISCUSSION

Experimental results reported in Section IV are quite intuitive as our approach to approximating mutual information was originally designed for the purpose of local computations. By averaging local approximations, one cannot fully express dependencies between columns in the entire data. Still, it may be relatively easy to introduce an additional coefficient reflecting "inter-packrow" correlations that would be complementary to "intra-packrow" level that is already in place.

TABLE III
RANKING OF MOST "TIME-DEPENDENT" COLUMNS IN THE "CAR_SALES" DATA SET. $I(*, pack)$ DENOTES MUTUAL INFORMATION (COMPUTED ON THE ORIGINAL DATA) MEASURED BETWEEN PARTICULAR COLUMNS AND COLUMN *pack* LABELING ROWS WITH THEIR PACKROW NUMBERS.

column	$I(*, pack)$	sales_commission	0.000295
sales_area_add	0.000576	sales_state	0.000192
dlr_trans_amt	0.000573	dealer_state	0.000172
dealer_name	0.000548	make_name	0.000142
record_dt	0.000548	car_colour	0.000096
sales_city	0.000532	car_year	0.000053
model_name	0.000530	adv_audio_pkg	0.000004
dealer_zip	0.000527	dlr_trans_type	0.000004
sales_person	0.000521	sunroof_pkg	0.000003
sales_discount	0.000515	onstart_pkg	0.000003
msa_name	0.000488	dealer_sale_ind	0.000002

One possible approach is to *merge* summaries of single packrows into overall data representation and conduct extra calculations at such unified level. Analogously to our studies in [4], the first step may be to derive global quantization of domains of particular columns based on characteristics of their corresponding data packs. Then, by projecting approximations of each of local (per-packrow) pairwise probability distributions onto a "grid" of globally quantized column domains, we could estimate pairwise distributions corresponding to the whole data table. Such estimation is likely to be less accurate with respect to local relationships between columns within particular data fragments. However, this is what we need – scores calculated based on such high-level distributions can be sufficient counterparts for scores $\widetilde{I}_t(a,b)$, $t = 1, ..., N$, that are designed to reflect those local relationships.

Figure 4 illustrates how such global quantization can be obtained in agglomerative way. At each step, we merge summaries of two packrows, composing their unified compacted representation that can serve as input for next steps. Merging is performed by summing up (in a weighted way, if particular packrows consisted originally different amounts of rows) two histograms (as well as special values, etc.) and running quantization algorithm that is analogous to the one used during data load [3]. As the sets of histogram ranges representing two packrows can differ from each other, the algorithm is executed on joint histogram with potentially higher resolution. Hence, its main task is to choose range borders (and special values) providing most reasonable merged representation using limited footprint (that is more comparable to footprint of each of single packrows rather than a sum of their footprints).

Analogous idea of merging packrow summaries was outlined in [1] in context of approximate execution of multi-table queries. In our engine, we follow this kind of strategy to deal with one-to-many join operations. Namely, whenever needed, our algorithms produce a unified "big-packrow" summary of a dimension table and then, in a loop, such summary is *amalgamated* with particular fact table packrows (which lets us compose extended denormalized representations of those packrows). Thus, one can see that mechanism of assembling global representations based on per-packrow summaries may

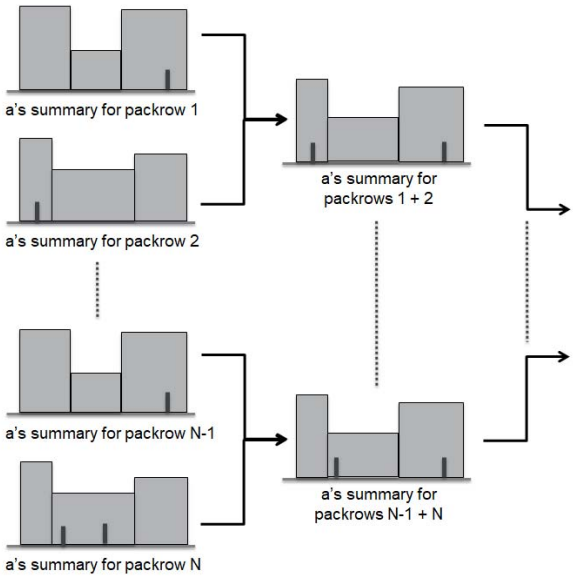

Fig. 4. Construction of global representation of column a based on (multi-threaded) agglomerative merging of its local per-packrow summaries.

be helpful in many different scenarios. Moreover, our current engine-specific multi-threaded implementation shows that such merging operations can be highly optimized.

Let us denote by a^* the obtained global quantization of column a. One way of looking at a^* is by means of CASE WHEN expression that labels values of a with identifiers of domain parts (special values or range complements) of a's global histogram representation that they drop into. Then, one could approximate mutual information $I(a,b)$ by runnning SQL statement "SELECT a^*, b^*, count(*) FROM T GROUP BY a^*, b^*;" and aggregating its outcome as $I(a^*,b^*)$ (or rather $\widetilde{I}(a^*,b^*)$ given the fact that query results produced by our database engine are approximate). This kind of approach could be regarded as an example of SQL-based data exploration that gained significant interest in last decades [9]. Moreover, from our perspective $\widetilde{I}(a^*,b^*)$ is a promising candidate to serve as aforementioned coefficient reflecting "inter-packrow" dependencies between columns a and b.

To summarize, we discussed how to improve approximation of mutual information scores that are insufficiently modeled by equation (3). One might claim that coefficient $\widetilde{I}(a^*,b^*)$ derived above is a good approximation of $I(a,b)$ by itself. However, the key point is to learn how to combine it with local scores $\widetilde{I}_t(a,b)$. This ultimate idea can be expressed by the following equation, where $\otimes$ denotes combination operator that we need to adjust in further research:

$$I(a,b) \approx \widetilde{I}_1(a,b) \otimes ... \otimes \widetilde{I}_N(a,b) \otimes \widetilde{I}\,(a^*,b^*) \qquad (4)$$

Surely, equation (4) represents just one of possible options. In particular, although our engine runs fast on large data sets [2], derivation of $\widetilde{I}\,(a^*,b^*)$ can be still a bottleneck comparing to distributed computations of $\widetilde{I}_t(a,b)$, $t = 1, ..., N$. Thus, let us extend this part with a few more observations.

First, let us refer to a realm of *granular computing* [15], where any calculations are supposed to run over so-called information granules gathering together various forms of entities that are similar or adjacent to each other. If we interpret packrows as groups of adjacent entities and treat their summaries as information granules, then our overall approximate analytics framework, as well as our previous developments in the area of analytical databases [13], can be envisioned as industry-ready deployment of granular computing paradigms.

Second, let us refer to somewhat relevant approach to data clustering outlined in [16], where data rows are dynamically grouped into *micro-clusters* (analogous to our packrows) and then, the final clustering process is conducted on vectors of their averaged summaries. Therein, the contents of particular micro-clusters are assumed to be sufficiently homogeneous to neglect operations at "intra-packrow" level. Actually, this technique inspired our aforementioned research on stream-based reorganization of packrow groupings [14].

In case of both [15] and [16], the fundamental idea is to handle highly aggregated objects whose footprint is even smaller than in case of our one- and two-column summaries. Although such degree of aggregation would be too drastic to assure good accuracy of data analytics in general, those approaches may serve as a guideline to design faster (and still meaningful enough as complementary computation) method to obtain the last component in equation (4).

VI. Conclusions

We continued our investigation on utilization of granulated data summary structures – whose ingestion from original data sources was originally designed for the purpose of our approximate database framework [1] – in some basic data exploration operations. In particular, we extended our work reported in [3] to better understand to what extent the proposed approximate calculations of entropy-based mutual information measure can reflect dependencies in large data.

Our research is motivated by a growth of interest in approximate computing in both, database and machine learning communities. Although there is still a lot to be done to set up proper expectations and fully integrate approximate analytics engines with the mainstream of information technology applications [5], it is tempting to provide the users with faster query answers even at the cost of loosening their accuracy. It is particularly important in scenarios where the speed of reaction really matters and where slight inexactness of analytical results is not likely to damage the quality of final decision-making. Analogous balance between the speed and accuracy of computations should be introduced also in the realm of machine learning solutions, especially if one wants to make them working more interactively [11].

Certainly, there are still several aspects to be addressed. In order to operate with aforementioned balance, one first needs to understand how to express accuracy of approximate calculations. In the field of databases, we can think about it by means of measuring appropriately specified similarities between exact and approximate query outcomes [17]. Analogously, in the area of data mining, one could adapt for this purpose some already-existing approaches to structural comparisons of models learnt from the data [10].

From the perspective of results reported in this paper, we will continue our research on more accurate approximations of mutual information taking into account both, local and global relationships in the summarized data. At global level, we need to better reflect "domain drift" or, in other words, "inter-packrow" correlations that can affect particular columns in various ways. Although we attempted to address this issue in Section V, focusing particularly on deriving additional global coefficients that could work well together with our previously-designed "intra-packrow" calculations, there are still methods of operating with compacted data representations (by means of, e.g., granulation [16] or sampling [18]) that we should investigate and potentially adapt for our purposes.

The idea of mixing together local and global approximations requires further study also from computational perspective. First, our methodology of running data operations on packrow summaries is perfectly combinable with modern parallel processing principles [19]. The only difference is that now calculations take place at different level of granularity, i.e., summaries of larger data collections rather than single data rows. One of our future directions with this respect is to refactor current approximate processes – reflecting both database and data mining operations – to let them perform smoothly within Spark environment [20]. However, some tasks – such as merging packrow summaries (as illustrated by Figure 4) or calculating coefficients $\widetilde{I}\left(a^*, b^*\right)$ (using, e.g., our approximate engine) – can be hard to fully parallelize. Thus, there is a need to design them elegantly for truly big data sets.

References

[1] D. Ślęzak, R. Glick, P. Betliński, and P. Synak, "A New Approximate Query Engine Based on Intelligent Capture and Fast Transformations of Granulated Data Summaries," *Journal of Intelligent Information Systems*, vol. 50, no. 2, pp. 385–414, 2018.

[2] D. Ślęzak, A. Chądzyńska-Krasowska, J. Holland, P. Synak, R. Glick, and M. Perkowski, "Scalable Cyber-security Analytics with a New Summary-based Approximate Query Engine," in *Proceedings of Big-Data 2017*, pp. 1840–1849.

[3] A. Chądzyńska-Krasowska, P. Betliński, and D. Ślęzak, "Scalable Machine Learning with Granulated Data Summaries: A Case of Feature Selection," in *Proceedings of ISMIS 2017*, pp. 519–529.

[4] A. Chądzyńska-Krasowska, S. Stawicki, and D. Ślęzak, "A Metadata Diagnostic Framework for a New Approximate Query Engine Working with Granulated Data Summaries," in *Proceedings of IJCRS 2017 Part I*, pp. 623–643.

[5] S. Chaudhuri, B. Ding, and S. Kandula, "Approximate Query Processing: No Silver Bullet," in *Proceedings of SIGMOD 2017*, pp. 511–519.

[6] G. Cormode, M. N. Garofalakis, P. J. Haas, and C. Jermaine, "Synopses for Massive Data: Samples, Histograms, Wavelets, Sketches," *Foundations and Trends in Databases*, vol. 4, no. 1-3, pp. 1–294, 2012.

[7] J. Borkowski, "Performance Debugging of Parallel Compression on Multicore Machines," in *Proceedings of PPAM 2009 Part II*, pp. 82–91.

[8] B. Cyganek and M. Woźniak, "A Tensor Framework for Data Stream Clustering and Compression," in *Proceedings of ICIAP 2017 Part I*, pp. 163–173.

[9] H. S. Nguyen, "Approximate Boolean Reasoning: Foundations and Applications in Data Mining," in *Transactions on Rough Sets V*. Springer, 2006, pp. 334–506.

[10] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2003.
[11] A. Agrawal, J. Choi, K. Gopalakrishnan, S. Gupta, R. Nair, J. Oh, D. A. Prener, S. Shukla, V. Srinivasan, and Z. Sura, "Approximate Computing: Challenges and Opportunities," in *Proceedings of ICRC 2016*, pp. 1–8.
[12] H. Peng, F. Long, and C. Ding, "Feature Selection based on Mutual Information Criteria of Max-dependency, Max-relevance, and Min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.
[13] D. Ślęzak and V. Eastwood, "Data Warehouse Technology by Infobright," in *Proceedings of SIGMOD 2009*, pp. 841–846.
[14] D. Ślęzak and M. Kowalski, "Intelligent Data Granulation on Load: Improving Infobright's Knowledge Grid," in *Proceedings of FGIT 2009*, pp. 12–25.
[15] W. Pedrycz, *Granular Computing: Analysis and Design of Intelligent Systems*. CRC Press, 2013.
[16] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "On Clustering Massive Data Streams: A Summarization Paradigm," in *Data Streams – Models and Algorithms*, C. C. Aggarwal, Ed. Springer, 2007, pp. 9–38.
[17] A. Chądzyńska-Krasowska, "Similarity-based Accuracy Measures for Approximate Query Results," in *Proceedings of IPMU 2018*.
[18] A. Molina, A. Munteanu, and K. Kersting, "Core Dependency Networks," in *Proceedings of AAAI 2018*, 2018, pp. 3820–3827.
[19] S. Babu and H. Herodotou, "Massively Parallel Databases and MapReduce Systems," *Foundations and Trends in Databases*, vol. 5, no. 1, pp. 1–104, 2013.
[20] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, "Apache Spark: A Unified Engine for Big Data Processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016.