Query Log Compression for Workload Analytics

Ting Xie
University at Buffalo, SUNY
tingxie@buffalo.edu

Varun Chandola
University at Buffalo, SUNY
chandola@buffalo.edu

Oliver Kennedy
University at Buffalo, SUNY
okennedy@buffalo.edu

ABSTRACT

Analyzing database access logs is a key part of performance tuning, intrusion detection, benchmark development, and many other database administration tasks. Unfortunately, it is common for production databases to deal with millions or even more queries each day, so these logs must be summarized before they can be used. Designing an appropriate summary encoding requires trading off between conciseness and information content. For example: simple workload sampling may miss rare, but high impact queries. In this paper, we present LogR, a lossy log compression scheme suitable use for many automated log analytics tools, as well as for human inspection. We formalize and analyze the space/fidelity trade-off in the context of a broader family of "pattern" and "pattern mixture" log encodings to which LogR belongs. We show through a series of experiments that LogR compressed encodings can be created efficiently, come with provable information-theoretic bounds on their accuracy, and outperform state-of-art log summarization strategies.

1. INTRODUCTION

Automated analysis of database access logs is critical for solving a wide range of problems, from database performance tuning [6], to compliance validation [11] and query recommendation [8]. For example, the Peloton self-tuning database [36] searches for optimal configurations by repeatedly simulating database performance on historical queries drawn from access logs. Unfortunately, query logs for production databases can grow to be extremely large — A recent study of queries at a major US bank for a period of 19 hours found that the bank's systems produced nearly 17 million SQL queries and over 60 million stored procedure execution events [27]. Tracking only a sample of these queries is not sufficient, as rare queries can disproportionately affect database performance, for example, if they are using an otherwise unused index.

We propose LogR, a lossy compression scheme for summarizing database query logs. The summarized log facilitates efficient (both in terms of storage and time) approximation of workload statistics. Hence applications like database tuning optimizers and query recommendation systems that need to repeatedly compute statistics can benefit by compressing the log with LogR during a pre-processing step. As an additional benefit, the summarized log is interpretable and can be used for human inspection.

By adjusting a tunable parameter in LogR, a user can choose to obtain a summary that incurs a low loss of information (high-fidelity) but is verbose, or obtain a more compact summary (high-compression) that incurs a greater loss of information. To manage the loss-rate, we develop a framework for reasoning about the trade-off between space and fidelity. While LogR does not admit closed-form solutions to classical information theoretical fidelity measures like information loss, we propose an efficiently computable fidelity measure called *Reproduction Error*. We show through a combination of analytical and experimental evidence that Reproduction Error is a meaningful measure of the quality in LogR.

LogR-compressed data relies on a codebook based on structural elements like `SELECT` items, `FROM` tables, or conjunctive `WHERE` clauses [1]. This codebook provides a bidirectional mapping from SQL queries to a bit-vector encoding and back again. In effect, we treat the query log as a collection of feature-vectors, with each structural element as one feature. The compression problem thus reduces to one of compactly encoding a collection of feature-vectors.

We further simplify this problem by observing that a common theme in use cases like automated performance tuning or query recommendation is the need for predominantly *aggregate* workload statistics. Because such statistics do not depend on the order of the query log, the compression problem further reduces to compacting a *bag* of feature-vectors.

LogR works by identifying groups of co-occurring structural elements that we call patterns. We define a family of *pattern encodings* of access logs, which map patterns to their frequencies in the log. For pattern encodings, we define two idealized measures of fidelity: (1) Ambiguity, which measures how much room the encoding leaves for interpretation; and (2) Deviation, which measures how reliably the encoding approximates the original log. Neither Ambiguity nor Deviation can be computed efficiently for pattern encodings. Hence we propose a third measure called *Reproduction Error* that is efficiently computable and that closely tracks both Ambiguity and Deviation.

In general, the size of the encoding is inversely related with Reproduction Error: The more detailed the encoding, the more faithfully it represents the original log. Thus, log compression may be defined as a search over the space of pattern based encodings to identify the one that best trades off between these two properties. Unfortunately, searching for such an ideal encoding from the full space can be computationally expensive. To overcome this limitation, we reduce the search space by first clustering entries in the log and then encoding each cluster separately, an approach that we call pattern mixture encoding. Finally we identify a simple approach to encoding individual clusters that we call naive mixture encodings, and show experimentally that it produces results competitive with more powerful techniques for log compression and summarization.

Concretely, in this paper we make the following contributions: (1) We define two families of compression for query logs: pattern and pattern mixture, (2) We define Ambiguity and Deviation, two principled measures of the quality of any pattern or pattern mixture encoding, (3) We define a computationally efficient measure, Reproduction Error, and demonstrate that it is a close approximation of Ambiguity and Deviation, (4) We propose a clustering-based approach to efficiently search for naive mixture encodings, and show how these encodings can be further optimized, and, (5) We experimentally validate our approach and show that it produces more precise encodings faster than several state-of-the-art pattern encoding algorithms.

2. PROBLEM DEFINITION

In this section, we introduce and formally define the log compression problem. We begin by exploring several applications that need to repeatedly analyze query logs.

Index Selection. Selecting an appropriate set of indexes requires trading off between update costs, access costs, and limitations on available storage space. Existing strategies for selecting a (near-)optimal set of indexes typically repeatedly simulate database performance under different combinations of indexes, which in turn requires repeatedly estimating the frequency with which specific predicates appear in the workload. For example, if `status = ?` occurs in 90% of the queries in a workload, a hash index on `status` is beneficial.

Materializing Views Selection. The results of joins or highly selective selection predicates are good candidates for materialization when they appear frequently in the workload. Like index selection, view selection is a non-convex optimization problem, typically requiring exploration by repeated simulation, which in turn requires repeated frequency estimation over the workload.

Online Database Monitoring. In production settings, it is common to monitor databases for atypical usage patterns that could indicate a serious bug or security threat. When query logs are monitored, it is often done retrospectively, some hours after-the-fact [27]. To support real-time monitoring it is necessary to quickly compute the frequency of a particular class of query in the system's typical workload.

In each case, the application's interactions with the log amount to counting queries that have specific features: selection predicates, joins, or similar.

2.1 Notation

Let L be a log, or a finite collection of queries $\mathbf{q} \in L$. We write $f \in \mathbf{q}$ to indicate that $\mathbf{q}$ has some *feature* f, such as a specific predicate or table in its `FROM` clause. We assume (1) that the universe of features in both a log and a query is enumerable and finite, and (2) that a query is isomorphic to its feature set. We outline one approach to extracting features that satisfies both assumptions below. We abuse syntax and write $\mathbf{q}$ to denote both the query itself, as well as the set of its (relevant) features.

Let $\mathbf{b}$ denote some set of features $f \in \mathbf{b}$, which we call a *pattern*. We write these sets using vector notation: $\mathbf{b} = (x_1, \ldots, x_n)$ where n is the number of distinct features appearing in the log and x_i indicates the presence (absence) of ith feature with a 1 (resp., 0). For any two patterns $\mathbf{b}$, $\mathbf{b}'$, we say that $\mathbf{b}'$ *is contained* in $\mathbf{b}$ if $\mathbf{b}' \subseteq \mathbf{b}$. Equivalently, using vector notation $\mathbf{b} = (x_1, \ldots, x_n)$, $\mathbf{b}' = (x'_1, \ldots, x'_n)$:

$$\mathbf{b}' \subseteq \mathbf{b} \quad \equiv \quad \forall i,\, x'_i \leq x_i$$

Our goal then is to be able to query logs for the number of times a pattern $\mathbf{b}$ appears:

$$|\{\ \mathbf{q} \mid \mathbf{q} \in L \wedge \mathbf{b} \subseteq \mathbf{q}\ \}|$$

2.2 Coding Queries

For this paper, we specifically adopt the feature-extraction conventions of a query summarization scheme by Aligon et al. [1]. In this scheme, each feature is one of the following three query elements: (1) a table or sub-query in the `FROM` clause, (2) a column in the `SELECT` clause, and (3) a conjunctive clause of the `WHERE` clause.

EXAMPLE 1. *Consider the following example query.*

```
SELECT _id, sms_type, _time FROM Messages
WHERE status = 1 AND transport_type = 3
```

This query uses 6 features: ⟨ `sms_type`, `SELECT` ⟩, ⟨ `_id`, `SELECT` ⟩, ⟨ `_time`, `SELECT` ⟩, ⟨ `Messages`, `FROM` ⟩, ⟨ `status = 1`, `WHERE` ⟩, *and* ⟨ `transport_type = 3`, `WHERE` ⟩

Although this scheme is simple and limited to conjunctive queries (or queries with a conjunctive equivalent), it fulfills both assumptions we make on feature extraction schemes. The features of a query (and consequently a log) are enumerable and finite, and the feature set of the query is isomorphic (modulo commutativity and column order) to the original query. Furthermore, even if a query is not itself conjunctive, it often has a conjunctive equivalent. We quantify this statement with Table 1, which provides two relevant data points from production query logs; In both cases, *all* logged queries can be rewritten into equivalent queries compatible with the Aligon scheme.

Although we do not explore more advanced feature encoding schemes in detail here, we direct the interested reader to work on query summarization [30, 2, 27]. For example, a scheme by Makiyama et. al. [30] also captures aggregation-related features like group-by columns, while an approach by Kul et. al. [27] encodes partial tree-structures in the query.

2.3 Log Compression

As a lossy form of compression, LogR only approximates the information content of a query log. We next develop a simplified form of LogR that we call pattern-based compression, and develop a framework for reasoning about the fidelity of a LogR-compressed log. As a basis for this framework, we first reframe the information content of a query log

```
SELECT  sms_type, external_ids, _time, _id
  FROM  messages
 WHERE  (sms_type=1) ∧ (sms_type=0) ∧ (status=1)
```

(a) *Correlation-ignorant*: Features are highlighted independently

```
SELECT sms_type FROM messages WHERE sms_type=1
SELECT sms_type FROM messages WHERE status=1
```

(b) *Correlation-aware*: Pattern groups are highlighted together.

Figure 1: **Example Encoding Visualizations**

to allow us to adapt classical information-theoretical measures of information content.

2.3.1 Information Content of Logs

We define the information content of the log as a distribution $p(Q \mid L)$ of queries Q drawn uniformly from the log.

Example 2. *Consider the following query log, which consists of four conjunctive queries.*

```
1. SELECT _id FROM Messages WHERE status = 1
2. SELECT _time FROM Messages
          WHERE status = 1 AND sms_type = 1
3. SELECT _id FROM Messages WHERE status = 1
4. SELECT sms_type, _time FROM Messages
          WHERE sms_type = 1
```

Drawing uniformly from the log, each entry will appear with probability $\frac{1}{4} = 0.25$*. The query* q_1 $(= q_3)$ *occurs twice, so the probability of drawing it is double that of the others (i.e.,* $p(\mathbf{q}_1 \mid L) = p(\mathbf{q}_3 \mid L) = \frac{2}{4} = 0.5$*)*

Treating a query as a vector of its component features, we can define a query $\mathbf{q} = (x_1, \ldots, x_n)$ to be an observation of the multivariate distribution over variables $Q = (X_1, \ldots, X_n)$ corresponding to features. The event $X_i = 1$ occurs if feature i appears in a query uniformly drawn from the log.

Example 3. *Continuing, the universe of features for this query log is (1)* $\langle$ `_id`, `SELECT` $\rangle$*, (2)* $\langle$ `_time`, `SELECT` $\rangle$*, (3)* $\langle$ `sms_type`, `SELECT` $\rangle$*, (4)* $\langle$ `status = 1`, `WHERE` $\rangle$*, (5)* $\langle$ `sms_type = 1`, `WHERE` $\rangle$*, and (6)* $\langle$ `Messages`, `FROM` $\rangle$*. Accordingly, the queries can be encoded as feature vectors, with fields counting each feature's occurrences:* $\mathbf{q}_1 = \langle 1, 0, 0, 1, 0, 1 \rangle$, $\mathbf{q}_2 = \langle 0, 1, 0, 1, 1, 1 \rangle$, $\mathbf{q}_3 = \langle 1, 0, 0, 1, 0, 1 \rangle$, $\mathbf{q}_4 = \langle 0, 1, 1, 0, 1, 1 \rangle$

Patterns. Our target applications require us to count the number of times features (co-)occur in a query. For example, materialized view selection requires counting tables used together in queries. Motivated by this observation, we begin by defining a broad class of *pattern based encodings* that directly encode co-occurrence probabilities. A *pattern* is an arbitrary set of features $\mathbf{b} = (x_1, \ldots, x_n)$ that may co-occur together. Each pattern captures a piece of information from the distribution $p(Q \mid L)$. In particular, we are interested in the probability of uniformly drawing a query $\mathbf{q}$ from the log that *contains* the pattern $\mathbf{b}$ (i.e., $\mathbf{q} \supseteq \mathbf{b}$):

$$p(Q \supseteq \mathbf{b} \mid L) = \sum_{\mathbf{q} \in L \wedge \mathbf{q} \supseteq \mathbf{b}} p(\mathbf{q} \mid L)$$

When it is clear from context, we abuse notation and write $p(\cdot)$ instead of $p(\cdot \mid L)$. Recall that $p(Q)$ can be represented as a joint distribution over variables $(X_1, \ldots, X_n)$ and probability $p(Q \supseteq \mathbf{b})$ is thus equivalent to the marginal probability or simply marginal $p(X_1 \geq x_1, \ldots, X_n \geq x_n)$.

Pattern-Based Encodings. Denote by $\mathcal{E}_{max} : \mathbb{N}^n \to [0, 1]$, the mapping from the space of all possible patterns $\mathbf{b} \in \mathbb{N}^n$ to their marginals. A *pattern based encoding* $\mathcal{E}$ is any such partial mapping $\mathcal{E} \subseteq \mathcal{E}_{max}$. We denote the marginal of pattern $\mathbf{b}$ in encoding $\mathcal{E}_L$ by $\mathcal{E}_L[\mathbf{b}]$ $(= p(Q \supseteq \mathbf{b} \mid L))$. When it is clear from context, we abuse syntax and also use $\mathcal{E}$ to denote the set of patterns it maps (i.e., *domain*$(\mathcal{E})$). Hence, $|\mathcal{E}|$ is the number of mapped patterns, which we call the encoding's *Verbosity*. A *pattern based encoder* is any algorithm $\texttt{encode}(L, \epsilon) \mapsto \mathcal{E}$ whose input is a log L and whose output is a set of patterns $\mathcal{E}$, with Verbosity thresholded at some integer ϵ. Many pattern mining algorithms [12, 31] can be used for this purpose.

2.3.2 Communicating Information Content

A side-benefit of pattern based encodings is that they are interpretable: patterns and their marginals can be used for human analysis of the log. Figure 1 shows two examples. The approach illustrated in Figure 1a uses shading to show each feature's frequency in the log, and communicates frequently occurring constraints or attributes. This approach might, for example, help a human to manually select indexes. A second approach illustrated in Figure 1b conveys correlations, showing the frequency of entire patterns.

3. INFORMATION LOSS

Our goal is to encode the distribution $p(Q)$ as a set of patterns: obtaining a less verbose encoding (i.e., with fewer patterns), while also ensuring that the encoding captures $p(Q)$ without information loss. We will start by defining information loss for pattern based encodings and generalize the definition to probabilistic models.

3.1 Lossless Summaries

To establish a baseline for measuring information loss, we begin with the extreme cases. At one extreme, an empty encoding ($|\mathcal{E}| = 0$) conveys no information. At the other extreme, we have the encoding $\mathcal{E}_{max}$ which is the full mapping from all patterns. Having this encoding is a sufficient condition to exactly reconstruct the original distribution $p(Q)$.

Proposition 1. *For any query* $\mathbf{q} = (x_1, \ldots, x_n) \in \mathbb{N}^n$*, the probability of drawing exactly* $\mathbf{q}$ *at random from the log (i.e.,* $p(X_1 = x_1, \ldots, X_n = x_n)$*) is computable, given* $\mathcal{E}_{max}$*.*

3.2 Lossy Summaries

Although lossless, $\mathcal{E}_{max}$ is also verbose. Hence, we will focus on lossy encodings that can be less verbose. A lossy encoding $\mathcal{E} \subset \mathcal{E}_{max}$ may not be able to precisely identify the distribution $p(Q)$, but can still be used to approximate the distribution. We characterize the information content of a lossy encoding $\mathcal{E}$ by defining a *space* (denoted by $\Omega_{\mathcal{E}}$) of distributions $\rho \in \Omega_{\mathcal{E}}$ allowed by an encoding $\mathcal{E}$. This space is defined by constraints as follows: First, we have the general properties of probability distributions:

$$\forall \mathbf{q} \in \mathbb{N}^n : \rho(\mathbf{q}) \geq 0 \qquad \sum_{\mathbf{q} \in \mathbb{N}^n} \rho(\mathbf{q}) = 1$$

Each pattern $\mathbf{b}$ in the encoding $\mathcal{E}$ constraints the marginal probability over its component features:

$$\forall \mathbf{b} \in domain(\mathcal{E}) : \quad \mathcal{E}[\mathbf{b}] = \sum_{\mathbf{q} \supseteq \mathbf{b}} \rho(\mathbf{q})$$

Note that the dual constraints $1 - \mathcal{E}[\mathbf{b}] = \sum_{\mathbf{q} \not\supseteq \mathbf{b}} \rho(\mathbf{q})$ are redundant under constraint $\sum_{\mathbf{q} \in \mathbb{N}^n} \rho(\mathbf{q}) = 1$.

The resulting space $\Omega_{\mathcal{E}}$ is the set of all query logs, or equivalently the set of all possible distributions of queries, that obey these constraints. From the outside observer's perspective, the distribution $\rho \in \Omega_{\mathcal{E}}$ that the encoding conveys is ambiguous: We model this ambiguity with a random variable $\mathcal{P}_{\mathcal{E}}$ with support $\Omega_{\mathcal{E}}$. The true distribution $p(Q)$ derived from the query log must appear in $\Omega_{\mathcal{E}}$, denoted $p(Q) \equiv \rho^* \in \Omega_{\mathcal{E}}$ (i.e., $p(\mathcal{P}_{\mathcal{E}} = \rho^*) > 0$). Of the remaining distributions ρ admitted by $\Omega_{\mathcal{E}}$, it is possible that some are more likely than others. For example, a query containing a column (e.g., `status`) is only valid if it also references a table that contains the column (e.g., `Messages`). This prior knowledge may be modeled as a prior on the distribution of $\mathcal{P}_{\mathcal{E}}$ or by an additional constraint. However, for the purposes of this paper, we take the uninformed prior by assuming that $\mathcal{P}_{\mathcal{E}}$ is uniformly distributed over $\Omega_{\mathcal{E}}$:

$$p(\mathcal{P}_{\mathcal{E}} = \rho) = \begin{cases} \frac{1}{|\Omega_{\mathcal{E}}|} & \text{if } \rho \in \Omega_{\mathcal{E}} \\ 0 & \text{otherwise} \end{cases}$$

Naive Encodings. One specific family of lossy encodings that treats each feature as being independent (e.g., as in Figure 1a) is of particular interest to us. We call this family *naive encodings*, and return to it throughout the rest of the paper. A naive encoding is composed of all patterns that have exactly one feature and a non-zero marginal.

$$\{\ \mathbf{b} = (0, \ldots, 0, x_i, 0, \ldots, 0) \mid i \in [1, n],\ x_i = 1\ \}$$

3.3 Idealized Information Loss Measures

Based on the space of distributions constrained by the encoding, the information loss of a encoding can be considered from two related, but subtly distinct perspectives: (1) *Ambiguity* measures how much room the encoding leaves for interpretation, (2) *Deviation* measures how reliably the encoding approximates the target distribution $p(Q)$.

Ambiguity. We define the Ambiguity $\mathrm{I}(\mathcal{E})$ of an encoding as the entropy of the random variable $\mathcal{P}_{\mathcal{E}}$. The higher the entropy, the less precisely $\mathcal{E}$ identifies any specific distribution.

$$\mathrm{I}(\mathcal{E}) = \sum_{\rho} p(\mathcal{P}_{\mathcal{E}} = \rho) \log\left(p(\mathcal{P}_{\mathcal{E}} = \rho)\right)$$

Deviation. The deviation from any permitted distribution ρ to the true distribution ρ^* can be measured by the Kullback-Leibler (K-L) divergence [28] (denote as $\mathcal{D}_{KL}(\rho^* || \rho)$). We define Deviation $\mathrm{d}(\mathcal{E})$ of a encoding as the expectation of the K-L divergence over all permitted $\rho \in \Omega_{\mathcal{E}}$:

$$\mathrm{d}(\mathcal{E}) = \mathbb{E}_{\mathcal{P}_{\mathcal{E}}}\left[\mathcal{D}_{KL}(\rho^* || \mathcal{P}_{\mathcal{E}})\right] = \sum_{\rho \in \Omega_{\mathcal{E}}} p(\mathcal{P}_{\mathcal{E}} = \rho) \cdot \mathcal{D}_{KL}(\rho^* || \rho)$$

Limitations. There are two limitations to these idealized measures in practice. First, K-L divergence is not defined from any probability measure ρ^* that is not *absolutely continuous* with respect to a second (denoted $\rho^* \ll \rho$). Second, neither Deviation nor Ambiguity has a closed-form formula.

4. PRACTICAL LOSS MEASURE

Computing either Ambiguity or Deviation requires enumerating the entire space of possible distributions, or an approximation. One approach to estimating either measure is to repeatedly sample from, rather than enumerate the space. However accurate measures require a large number of samples, rendering this approach similarly infeasible. In this section, we propose a faster approach to assessing the fidelity of a pattern encoding. We select a single representative distribution $\overline{\rho}_{\mathcal{E}}$ from the space $\Omega_{\mathcal{E}}$, and use $\overline{\rho}_{\mathcal{E}}$ to approximate both Ambiguity and Deviation.

4.1 Reproduction Error

Maximum Entropy Distribution. Inspired by the principle of maximum entropy [21], we select a single distinguished representative distribution $\overline{\rho}_{\mathcal{E}}$ from the space $\Omega_{\mathcal{E}}$:

$$\overline{\rho}_{\mathcal{E}} = \arg\min_{\rho \in \Omega_{\mathcal{E}}} -\mathcal{H}(\rho) \qquad \text{where } \mathcal{H}(\rho) = \sum_{\mathbf{q} \in \mathbb{N}^n} -\rho(\mathbf{q}) \log \rho(\mathbf{q})$$

Maximizing an objective function belonging to the exponential family (entropy in our case) under a mixture of linear equalities/inequality constraints is a convex optimization problem [5] which guarantees a *unique* solution and can be efficiently solved [10], using the cvx toolkit [17, 33], and/or subdividing the problem by grouping variables of $\mathbf{q} \in \mathbb{N}^n$ into equivalence classes. For naive encodings specifically, we can assume independence across word occurrences X_i. Under this assumption, $\overline{\rho}_{\mathcal{E}}$ has a closed-form solution:

$$\overline{\rho}_{\mathcal{E}}(\mathbf{q}) = \prod_i p(X_i = x_i) \qquad \text{where } \mathbf{q} = (x_1, \ldots, x_n) \tag{1}$$

Reproduction Error. Using the representative distribution $\overline{\rho}_{\mathcal{E}}$, we define *Reproduction Error* $e(\mathcal{E})$ as the entropy difference between the representative and true distributions:

$$e(\mathcal{E}) = \mathcal{H}(\overline{\rho}_{\mathcal{E}}) - \mathcal{H}(\rho^*) \qquad \text{where } \overline{\rho}_{\mathcal{E}} = \arg\min_{\rho \in \Omega_{\mathcal{E}}} -\mathcal{H}(\rho)$$

Relationship to K-L Divergence. Reproduction Error is closely related to the K-L divergence from the representative distribution $\overline{\rho}_{\mathcal{E}}$ to the true distribution ρ^*.

$$\mathcal{D}_{KL}(\rho^* || \overline{\rho}_{\mathcal{E}}) = \mathcal{H}(\rho^*, \overline{\rho}_{\mathcal{E}}) - \mathcal{H}(\rho^*)$$
$$\text{where } \mathcal{H}(\rho^*, \overline{\rho}_{\mathcal{E}}) = \sum_{\mathbf{q}} -\rho^*(\mathbf{q}) \log \overline{\rho}_{\mathcal{E}}(\mathbf{q})$$

$\mathcal{H}(\rho^*, \overline{\rho}_{\mathcal{E}})$ is called the *cross-entropy*. Replacing cross-entropy by entropy $\mathcal{H}(\overline{\rho}_{\mathcal{E}})$, the formula becomes the same as Reproduction Error. Though cross entropy is different from entropy in general, e.g., it is only defined when $\rho^* \ll \overline{\rho}_{\mathcal{E}}$, they are closely correlated. However, for the specific case of naive encodings the two are equivalent.

LEMMA 1. *For any naive encoding $\mathcal{E}$, $\mathcal{H}(\rho^*, \overline{\rho}_{\mathcal{E}}) = \mathcal{H}(\overline{\rho}_{\mathcal{E}})$*

PROOF. With $\mathbf{q} = (x_1, \ldots, x_n)$ and applying Equation 1:

$$\begin{aligned}
\sum_{\mathbf{q} \in L} -\rho^*(\mathbf{q}) \log \overline{\rho}_{\mathcal{E}}(\mathbf{q}) &= -\sum_{\mathbf{q}} p(\mathbf{q}) \cdot \sum_i \log p(X_i = x_i) \\
&= -\sum_{i,k} \log p(X_i = k) \cdot \sum_{\mathbf{q} \mid x_i = k} p(\mathbf{q}) \\
&= -\sum_{i,k} \log p(X_i = k) \cdot p(X_i = k) \\
&= \sum_i \mathcal{H}(X_i)
\end{aligned}$$

The variables in a naive encoding are independent, so: $\sum_i \mathcal{H}(X_i) = \mathcal{H}(\overline{\rho}_{\mathcal{E}})$, and we have the lemma. □

While we do not provide a similar proof for more general encodings, we show it experimentally in Section 7.3.

4.2 Practical vs Idealized Information Loss

In this section we prove that Reproduction Error closely parallels Ambiguity. We define a partial order lattice over encodings and show that for any pair of encodings on which the partial order is defined, a like relationship is implied for both Reproduction Error and Ambiguity. We supplement the proofs given in this section with an empirical analysis relating Reproduction Error to Deviation in Section 7.3.

Containment. We define a partial order over encodings $\leq_\Omega$ based on *containment* of their induced spaces $\Omega_{\mathcal{E}}$:

$$\mathcal{E}_1 \leq_\Omega \mathcal{E}_2 \quad \equiv \quad \Omega_{\mathcal{E}_1} \subseteq \Omega_{\mathcal{E}_2}$$

That is, one encoding (i.e., $\mathcal{E}_1$) precedes another (i.e., $\mathcal{E}_2$) when all distributions admitted by the former encoding are also admitted by the latter.

Containment Captures Reproduction Error. We first prove that the total order given by Reproduction Error is a superset of the partial order $\leq_\Omega$.

LEMMA 2. *For any two encodings $\mathcal{E}_1, \mathcal{E}_2$ with induced spaces $\Omega_{\mathcal{E}_1}, \Omega_{\mathcal{E}_2}$ and maximum entropy distributions $\overline{\rho}_{\mathcal{E}_1}, \overline{\rho}_{\mathcal{E}_2}$ it holds that $\mathcal{E}_1 \leq_\Omega \mathcal{E}_2 \to e(\mathcal{E}_1) \leq e(\mathcal{E}_2)$.*

PROOF. Firstly $\Omega_{\mathcal{E}_2} \supseteq \Omega_{\mathcal{E}_1} \to \overline{\rho}_{\mathcal{E}_1} \in \Omega_{\mathcal{E}_2}$. Since $\overline{\rho}_{\mathcal{E}_2}$ has the maximum entropy among all distributions $\rho \in \Omega_{\mathcal{E}_2}$, we have $\mathcal{H}(\overline{\rho}_{\mathcal{E}_1}) \leq \mathcal{H}(\overline{\rho}_{\mathcal{E}_2}) \equiv e(\mathcal{E}_1) \leq e(\mathcal{E}_2)$. ☐

Containment Captures Ambiguity. Next, we show that the partial order based on containment implies a like relationship between Ambiguities of pairs of encodings.

LEMMA 3. *Given encodings $\mathcal{E}_1, \mathcal{E}_2$ with uninformed prior on $\mathcal{P}_{\mathcal{E}_1}, \mathcal{P}_{\mathcal{E}_2}$, it holds that $\mathcal{E}_1 \leq_\Omega \mathcal{E}_2 \to I(\mathcal{E}_1) \leq I(\mathcal{E}_2)$.*

PROOF. Given an uninformed prior: $\mathrm{I}(\mathcal{E}) = \log |\Omega_{\mathcal{E}}|$. Hence $\mathcal{E}_1 \leq_\Omega \mathcal{E}_2 \to |\Omega_{\mathcal{E}_1}| \leq |\Omega_{\mathcal{E}_2}| \to \mathrm{I}(\mathcal{E}_1) \leq \mathrm{I}(\mathcal{E}_2)$ ☐

5. PATTERN MIXTURE ENCODINGS

Thus far we have defined the problem of log compression, treating the query log as a single joint distribution $p(Q)$ that captures the frequency of feature occurrence and/or co-occurrence. Patterns capture positive information about correlations. However in cases like logs of *mixed* workloads, there are also many cases of anti-correlation between features. For example, consider a log that includes queries drawn from two workloads with disjoint feature sets. Patterns with features from both workloads never actually occur in the log. However, unless explicitly defined otherwise, many such patterns will have non-zero marginals. Consistently excluding anti-correlations, in general, requires adding many patterns to the encoding.

In this section, we explore an alternative to capturing correlation and anti-correlation with a single set of patterns. Instead, we propose a generalization of pattern encodings where the log is modeled not as a single probability distribution, but rather as a mixture of several simpler distributions. The resulting encoding is likewise a mixture: Each component of the mixture of distributions is stored independently. Hence, we refer to it as a *pattern mixture encoding*, and it forms the basis of LOGR compression.

To start, we explore a simplified form of the problem, where we only mix *naive* pattern encodings. We refer to the resulting scheme as *naive mixture encodings*, and give examples of the encoding, as well as potential visualizations in Section 5.1. Using naive mixture encodings as an example, we generalize Reproduction Error and Verbosity for pattern mixture encodings in Section 5.2. With generalized encoding evaluation measures, we then evaluate several encoding strategies based on different clustering methods for creating naive mixture encodings. Finally, in Section 6.4, we discuss strategies for refining naive mixture encodings into more precise, general mixture encodings.

5.1 Example: Naive Mixture Encodings

Consider a toy query log with only 3 conjunctive queries.

1. `SELECT id FROM Messages WHERE status = 1`
2. `SELECT id FROM Messages`
3. `SELECT sms_type FROM Messages`

The vocabulary of this log consists of 4 features: ⟨ `id`, `SELECT` ⟩, ⟨ `sms_type`, `SELECT` ⟩, ⟨ `Messages`, `FROM` ⟩, and ⟨ `status = 1`, `WHERE` ⟩. Re-encoding the three queries as vectors, we get:

1. $\langle\, 1,0,1,1 \,\rangle$ 2. $\langle\, 1,0,1,0 \,\rangle$ 3. $\langle\, 0,1,1,0 \,\rangle$

A naive encoding of this log can be expressed as:

$$\left\langle\ \frac{2}{3},\ \frac{1}{3},\ 1,\ \frac{1}{3}\ \right\rangle$$

This encoding captures that all queries in the log pertain to the Messages table, but obscures the relationship between the remaining features. For example, this encoding obscures the anti-correlation between `id` and `sms_type`. Similarly, the encoding hides the association between `status = 1` and `id`. Such relationships are critical for evaluating the effectiveness of views or indexes.

EXAMPLE 4. *The maximal entropy distribution for a naive encoding assumes that features are independent. Assuming independence, the probability of query 1 from the log is:*

$$p(\texttt{id}) \cdot p(\neg\texttt{sms\_type}) \cdot p(\texttt{Messages}) \cdot p(\texttt{status=1}) = \frac{4}{27} \approx 0.148$$

This is a significant difference from true probability of this query (i.e., $\frac{1}{3}$). Conversely queries not in the log, such as the following, would be assumed to have non-zero probability.

`SELECT sms_type FROM Messages WHERE status = 1`

$$p(\neg\texttt{id}) \cdot p(\texttt{sms\_type}) \cdot p(\texttt{Messages}) \cdot p(\texttt{status=1}) = \frac{1}{27} \approx 0.037$$

To achieve a more faithful representation of the original log, we could partition it into two components, with the corresponding encoding parameters:

Partition 1 (L_1)	**Partition 2** (L_2)
$(1,0,1,1)$ $(1,0,1,0)$	$(0,1,1,0)$
$\downarrow$ $\downarrow$	$\downarrow$
$\langle\ 1,\ 0,\ 1,\ \frac{1}{2}\ \rangle$	$\langle\ 0,\ 1,\ 1,\ 0\ \rangle$

The resulting encoding only has one non-integral probability: $p(\texttt{status = 1} \mid L_1) = 0.5$. Although there are now two encodings, the encodings are not ambiguous. The feature `status = 1` appears in exactly half of the log entries, and is indeed independent of the other features. All other attributes in each encoding appear in all queries in their respective partitions. Furthermore, the maximum entropy distribution induced by each encoding is exactly the distribution of queries in the compressed log. Hence, the Reproduction Error is zero for both of the two encodings.

5.2 Generalized Encoding Fidelity

We next generalize our definitions of Reproduction Error and Verbosity from pattern to pattern mixture encodings. Suppose query log L has been partitioned into K clusters with L_i, S_i, $\overline{\rho}_{S_i}$ and ρ_i^* (where $i \in [1, K]$) representing the log of queries, encoding, maximum entropy distribution and true distribution (respectively) for ith cluster. First, observe that the distribution for the whole log (i.e., ρ^*) is the sum of distributions for each partition (i.e., ρ_i^*) weighted by proportion of queries (i.e., $\frac{|L_i|}{|L|}$) in the partition.

$$\rho^*(\mathbf{q}) = \sum_{i=1,\ldots,K} w_i \cdot \rho_i^*(\mathbf{q}) \qquad \text{where } w_i = \frac{|L_i|}{|L|}$$

Generalized Reproduction Error. Similarly, the maximum entropy distribution $\overline{\rho}_S$ for the whole log can be obtained as

$$\overline{\rho}_S(\mathbf{q}) = \sum_{i=1,\ldots,K} w_i * \overline{\rho}_{S_i}(\mathbf{q})$$

We define the *Generalized Reproduction Error* of a pattern mixture encoding similarly, as the weighted sum of the errors for each partition:

$$\mathcal{H}(\overline{\rho}_S) - \mathcal{H}(\rho^*) = \sum_i w_i \cdot \mathcal{H}(\overline{\rho}_{S_i}) - \sum_i w_i \cdot \mathcal{H}(\rho_i^*) = \sum_i w_i \cdot e(S_i)$$

As in the base case, a pattern mixture encoding with low generalized Reproduction Error indicates a high-fidelity representation of the original log. A process can infer the probability of any query $p(Q = \mathbf{q} \mid L)$ drawn from the original distribution, simply by inferring its probability drawn from each cluster i (i.e., $p(Q = \mathbf{q} \mid L_i)$) and taking a weighted average over all inferences. When it is clear from context, we refer to generalized Reproduction Error simply as Error in the rest of this paper.

Generalized Verbosity. Verbosity can be generalized to mixture encodings in two ways. First we could measure the *Total Verbosity* ($\sum_i |S_i|$), or the total size of the encoded representation. This approach is ideal for our target applications, where our aim is to reduce the representational size of the query log. As an alternative, we could express the complexity of the encoding by computing the *Average Verbosity* ($\sum_i w_i \cdot |S_i|$), which measures the expected size of each cluster. The latter measure is specifically useful in cases where LogR encodings are presented for human consumption, as lower average verbosity encodings are simpler.

6. PATTERN MIXTURE COMPRESSION

We are now ready to describe the LogR compression scheme. Broadly, LogR attempts to identify a pattern mixture encoding that optimizes for some target trade-off between Total Verbosity and Error. A naive — though impractical — approach to finding such an encoding would be to search the entire space of possible pattern mixture encodings. Instead, LogR approximates the same outcome by identifying the *naive* pattern mixture encoding that is closest to optimal for the desired trade-off. As we show experimentally, the naive mixture summary produced by the first stage is competitive with more complicated, slower techniques for summarizing query logs. We also explore a hypothetical second stage, where LogR refines the naive mixture encoding to further reduce error. The outcome of this hypothetical stage has a slightly lower Error and Verbosity, but does not admit efficient computation of database statistics.

6.1 Constructing Naive Mixture Encodings

LogR compression searches for a naive mixture summary that best optimizes for a requested tradeoff between Total Verbosity and Error. As a way to make this search efficient, we observe that a log (or log partition) uniquely determines its naive mixture encoding. Thus the problem of searching for a naive mixture encoding reduces to the problem of searching for the corresponding log partitioning.

We further observe that the Error of a naive mixture encoding is proportional to the diversity of the queries in the log being encoded; The more uniform the log (or partition), the lower the corresponding error. Hence, the partitioning problem further reduces to the problem of clustering queries in the log by feature overlap.

To identify a suitable clustering scheme, we next evaluate four commonly used partitioning/clustering methods: (1) KMeans [20] with Euclidean distance (i.e., l_2-norm) and Spectral Clustering [24] with (2) Manhattan (i.e., l_1-norm), (3) Minkowski (i.e., l_p-norm) with $p = 4$, and (4) Hamming ($\frac{Count(\mathbf{x} \neq \mathbf{y})}{Count(\mathbf{x} \neq \mathbf{y}) + Count(\mathbf{x} = \mathbf{y})}$) distances[1]. Specifically, we evaluate these four strategies with respect to their ability to create naive mixture encodings with low Error and low Verbosity.

Experiment Setup. Spectral and KMeans clustering algorithms are implemented by *sklearn* [37] in Python. We gradually increase K (i.e., the number of clusters) configured for selected clustering algorithm that simulates the process of continuously sub-clustering the log, tolerating higher Total Verbosity for lower Error. To compare clustering methods fairly, we reduce randomness in clustering (e.g., random initialization in KMeans) by running each of them 10 times for each K and averaging Error of resulting encodings. We used two datasets: "US Bank" and "PocketData." We describe both datsets in detail in Section 7.1 and the data preparation process in Section 7.2. All results for our clustering experiments are shown in Figure 2.

6.1.1 Clustering

In this section, we show that (1) Clustering is an effective way to consistently reduce Error, and that (2) Classical clustering methods differ in at least one of Error, Verbosity, and/or runtime.

Adding more clusters reduces Error. Figure 2a compares the relationship between the number of clusters (x-axis) and Error (y-axis), showing the varying rates of convergence to zero Error for each clustering method. We observe that adding more clusters consistently reduces Error for both data sets, regardless of clustering method and distance measures. We note that the US Bank dataset is significantly more diverse than the PocketData dataset, with respect to the total number of features (See Table 1) and that more than 30 clusters may be required for reaching near-zero Error. In general, Manhattan distance is not competitive as others for both datasets and Hamming distance converges faster than other methods on PocketData. Minkowski distance shows faster convergence rate than Hamming within 14 clusters on US bank dataset. However, it is also the only one that shows consistently *increasing* trend on Error when the number of clusters exceeds 22 for both datasets. This

[1]We also evaluated Spectral Clustering with Euclidean, Chebyshev and Canberra distances; These did not perform better and we omit them in the interest of conciseness.

is because l_p-norm is more aggressive in distinguishing between vectors as p grows (i.e., l_4 vs l_2) and since US bank dataset is more diverse, Minkowski helps by separating the highly mixed workloads aggressively. But it begin to break down when the data is already well-organized (e.g., being partitioned into more than 22 clusters).

Adding more clusters increases Total Verbosity. Figure 2b compares the relationship between the number of clusters (x-axis) and Total Verbosity (y-axis). We observe that Total Verbosity increases with the number of clusters. This is because when a partition is split, features common to both partitions each increase the Total Verbosity by 1.

Hierarchical Clustering. Classical clustering methods produce non-monotonic cluster assignments. That is, Error/Total Verbosity can actually grow/shrink with more clusters, as seen in Figure 2a and 2b. Notably, a class of clustering approaches called hierarchical clustering [22] does create monotonic assignments, and can offer more flexible control on deciding which cluster to further explore, by subclustering and constructing naive encodings with lower Reproduction Error on sub-clusters.

Error v. Average Verbosity. We observe from Figure 2c that Average Verbosity (x-axis) *positively* correlates with Error (y-axis). By comparing Figure 2a with 2c, we also observe that the clustering method that achieves lower Error also has lower Average Verbosity. Recall that Average Verbosity is the weighted sum of the number of distinct features in each cluster for naive mixture encodings. Misplacing queries with less overlap in features (i.e., less similar) into the same cluster increases its number of distinct features, and Average Verbosity measures the weighted average degree of such misplacement over all clusters.

Running Time Comparison. The total running time (y-axis) in Figure 2d is measured in seconds and includes both distance matrix computation time (if any) and clustering running time. Since the total running time of KMeans is significantly lower than that of Spectral Clustering, we logarithmically scaled Y-axis. In addition, the figure shows that Hamming distance is also the winner among other distances with respect to running time.

Take-Aways. Under Spectral Clustering, Hamming distance helps to construct naive mixture encodings faster and of lower Error than other distance measures. Additionally, in cases where it may be necessary to dynamically vary the accuracy (e.g., log visualizations), KMeans or its variants (e.g., KMedoids [35]) with customized distance measures (e.g., Hamming) are more effective, since KMeans is orders of magnitude faster than Spectral Clustering.

6.2 Approximating Log Statistics

Recall that our primary goal is estimating statistics about the log. In particular, we are interested in counting the number of occurrences (i.e., marginal) of a particular pattern $\mathbf{b}$ in the log:

$$\Gamma_{\mathbf{b}}(L) = |\{\ \mathbf{q}\ |\ \mathbf{q} \in L \wedge \mathbf{b} \subseteq \mathbf{q}\ \}|$$

Recall that a naive encoding $\mathcal{E}_L$ of the log carries only patterns with only single feature present. In the absence of a prior over the space of encodings allowed by $\mathcal{E}_L$ — that is, assuming a uniform distribution over log distributions allowed by this encoding — the expected log distribution is the maximal entropy distribution $\overline{\rho}_{\mathcal{E}_L}$ introduced in Section 4.1. Hence, the expectation of $\Gamma_{\mathbf{b}}(L)$ is computed by multiplying the probability of the feature occurring by the size of the log: $|L| \cdot \overline{\rho}_{\mathcal{E}_L}$, or:

$$E[\ \Gamma_{\mathbf{b}}(L) \mid \mathcal{E}_L\ ] = |L| \cdot \left(\prod_{f \in \mathcal{E} \textbf{ where } f \subseteq \mathbf{b}} \mathcal{E}[f] \right)$$

This process trivially generalizes to naive pattern mixture encodings by mixing distributions. Specifically, given a set of partitions $L_1 \cup \ldots \cup L_K = L$, the expected counts for each individual partition L_i may be computed based on the partition's encoding $\mathcal{E}_i$

$$E[\ \Gamma_{\mathbf{b}}(L_i) \mid \mathcal{E}_1, \ldots, \mathcal{E}_K\ ] = \sum_{i \in [1,K]} E[\ \Gamma_{\mathbf{b}}(L_i) \mid \mathcal{E}_i\ ]$$

6.3 Pattern Synthesis & Marginal Estimation

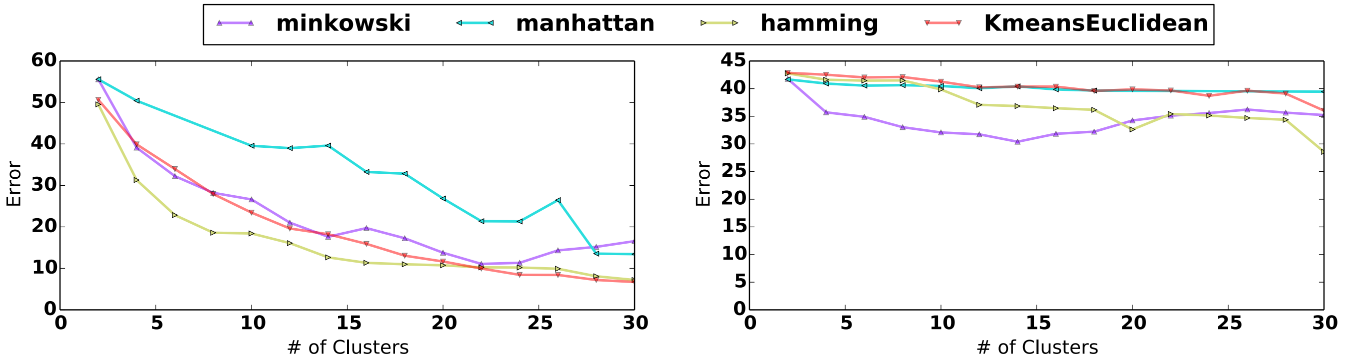
In this section, we empirically verify the effectiveness of naive mixture encoding in approximating log statistics from two related perspectives. The first perspective focuses on *synthesis error*. It measures whether patterns synthesized by naive mixture encoding actually exist in the log. From the second perspective, we would like to further investigate *marginal deviation*. It measures whether naive mixture encoding gives correct marginal values to patterns that one may query for. Specifically, *synthesis error* is measured as $1 - \frac{M}{N}$ where N is the total number of randomly synthesized patterns and M is the number of synthesized patterns with positive marginals in the log. *Marginal deviation* is measured as $\frac{|ESTM - TM|}{TM}$ where TM is true marginal of a pattern and $ESTM$ is the one estimated by naive mixture encoding.

We then set up experiments and empirically show that both synthesis error and marginal deviation consistently decreases with lower Reproduction Error. Unless explicitly specified, we adopt the same experiment settings as Section 6.1.1 and experiment results are shown in Figure 3.

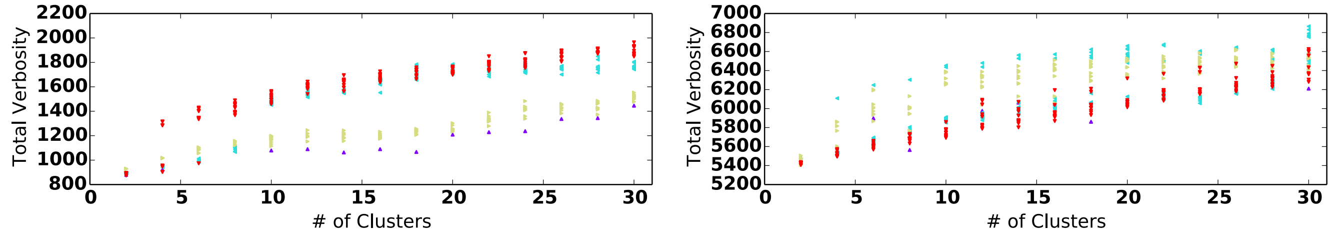
Figure 3a shows synthesis error (y-axis) versus Reproduction Error (x-axis). The figure is generated by randomly synthesizing $N = 10000$ patterns from each partition of the log. Different values of N are tested and they show similar result. Similar to Section 6.1.1, we gradually increase the number of clusters to reduce Reproduction Error. Synthesis error is then measured on each partition and the final synthesis error is the average over all partitions weighted by proportion of queries in the partition.

Figure 3b shows marginal deviation (y-axis) versus Reproduction Error (x-axis). It is not possible to enumerate all patterns that exist in the data. As an alternative, we treat each distinct query in the log as a pattern and measure marginal deviation only on distinct queries. This is because we observe that marginal deviation tends to be smaller if it is measured on a pattern that is contained in the other one. Hence marginal deviation measured on an query is the upper bound of those measured on all possible patterns that this query contains. To measure marginal deviation for each partition, we enumerate all of its distinct queries and take an equally weighted average[2]. The final marginal deviation for the whole log is an weighted average similar to that of synthesis error.

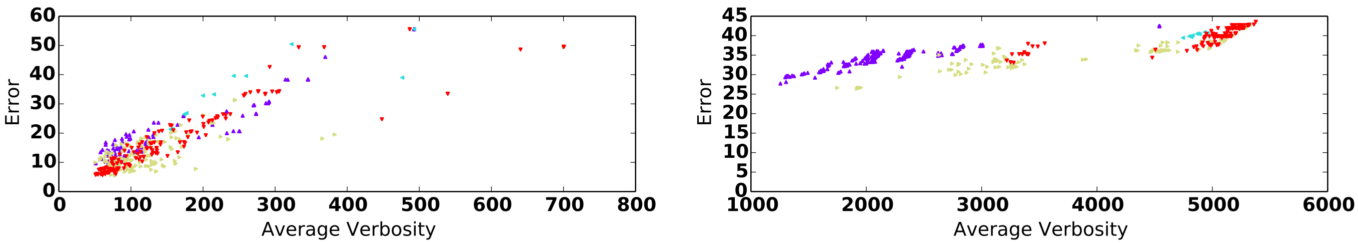
[2]Both frequent and infrequent patterns are treated equally.



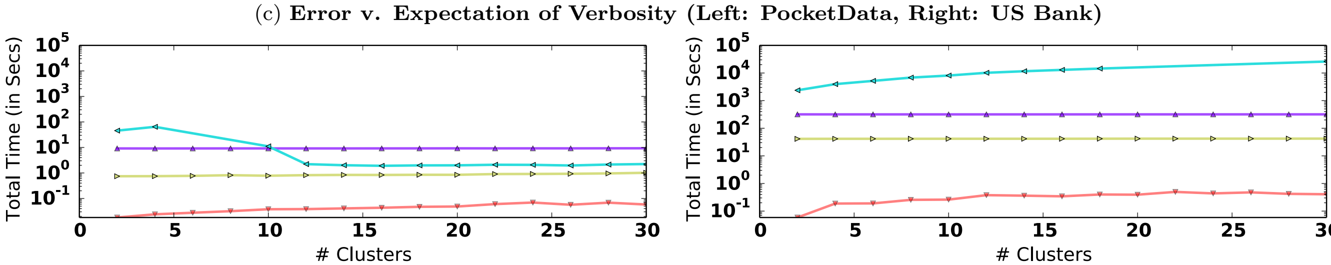
(a) **Error v. Number of Clusters (Left: PocketData, Right: US Bank)**



(b) **Total Verbosity v. Number of Clusters (Left: PocketData, Right: US Bank); We offset Y-axis by Verbosity of the naive encoding without clustering (i.e., the total number of distinct features in the log) to show the change.**

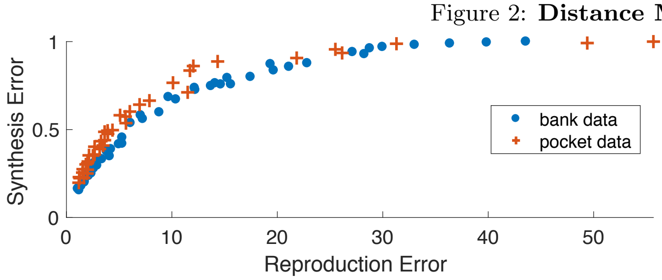


(c) **Error v. Expectation of Verbosity (Left: PocketData, Right: US Bank)**

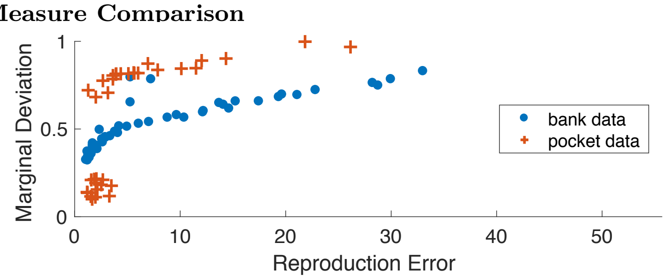


(d) **Running Time v. Number of Clusters (Left: PocketData, Right: US Bank)**

Figure 2: **Distance Measure Comparison**



(a) **Synthesis Error v. Reproduction Error**



(b) **Marginal Deviation v. Reproduction Error**

Figure 3: **Effectiveness of Naive Mixture Encoding**

6.4 Naive Encoding Refinement

Naive mixture encodings can already achieve close to near-zero Error (Figure 2a), have low Verbosity, and admit efficiently computable log statistics $\Gamma_{\mathbf{b}}(L)$. Although doing so makes estimating statistics more computationally expensive, as a thought experiment, we next consider how much of an improvement we could achieve in the Error/Verbosity trade-off by exploring a hypothetical second stage that enriches naive mixture encodings by adding non-naive patterns.

Feature-Correlation Refinement. The first challenge we need to address is that our closed-form formula for Reproduction Error only works for naive summaries. Hence, we first consider the simpler problem of identifying the individual pattern that most reduces the Reproduction Error of a naive summary.

Recall that for the Reproduction Error of a naive encoding has the closed-form representation by assuming independence among features (i.e., $\overline{\rho}_S(Q = \mathbf{q}) = \prod_i p(X_i = x_i)$). Similarly, under naive encodings we have a closed-form estimation of marginals $p(Q \supseteq \mathbf{b})$ (i.e., $\overline{\rho}_S(Q \supseteq \mathbf{b}) = \prod_i p(X_i \geq x_i)$). We define the *feature-correlation* of pattern $\mathbf{b}$ as the log-difference from its actual marginal to the estimation, according to naive encoding.

$$WC(\mathbf{b}, S) = \log\left(p(Q \supseteq \mathbf{b})\right) - \log\left(\overline{\rho}_S(Q \supseteq \mathbf{b})\right)$$

Intuitively, patterns with higher feature correlations create higher Errors, which in turn makes them ideal candidates for addition to the compressed log encoding. For two patterns with the same feature-correlation, the one that occurs more frequently will have the greatest impact on Error [19]. As a result, we compute an overall score for ranking patterns involving feature-correlation:

$$corr\_rank(\mathbf{b}) = p(Q \supseteq \mathbf{b}) \cdot WC(\mathbf{b}, S)$$

We show in Section 7.3 that *corr_rank* closely correlates with Reproduction Error. That is, a higher *corr_rank* value indicates that a pattern produces a greater Reproduction Error reduction if introduced into the naive encoding.

Pattern Diversification. This greedy approach only allows us to add a single pattern to each cluster. In general, we would like to identify a *set* of patterns to add to each cluster. We cannot sum up *corr_rank* of each pattern in the set to estimate its Reproduction Error, because partial information content carried by patterns may overlap. To counter such overlap, or equivalently to *diversify* patterns, inevitably we will need to search through the space of combinatorially large number of candidate pattern sets. The search process can be time-consuming even using heuristics. In other words, the potential Reproduction Error reduction by plugging-in state-of-the-art pattern mining algorithms may come with burden on computation efficiency. We experimentally analyze cases where naive mixture encodings are kept intact, refined or replaced by patterns mined from them in Section 7.4.

7. EXPERIMENTS

In this section, we design experiments to empirically (1) validate that Reproduction Error correlates with Deviation and (2) evaluate the effectiveness of LogR compression.

7.1 Data Sets

We use two specific datasets in the experiment: (1) SQL query logs of the Google+ Android app extracted from the PocketData public dataset [25] and (2) SQL query logs that capture all query activity on the majority of databases at a major US bank over a period of approximately 19 hours. A summary of these two datasets is given in Table 1.

Table 1: **Summary of Data sets**

Statistics	PocketData	US bank
# Queries	629582	1244243
# Distinct queries	605	188184
# Distinct queries (w/o const)	605	1712
# Distinct conjunctive queries	135	1494
# Distinct re-writable queries	605	1712
Max query multiplicity	48651	208742
# Distinct features	863	144708
# Distinct features (w/o const)	863	5290
Average features per query	14.78	16.56

The PocketData-Google+ query log. The dataset consists of SQL logs that capture all database activities of 11 Android phones. We selected Google+ application for our study since it is one of the few applications where all users created a workload. This dataset can be characterized as a stable workload of exclusively machine-generated queries.

The US bank query log. These logs are anonymized by replacing all constants with hash values generated by SHA-256, and manually vetted for safety. Of the nearly 73 million database operations captured, 58 million are not directly queries, but rather invocations of stored procedures and 13 million not able to be parsed by standard SQL parser. Among the rest of the 2.3 million parsed SQL queries, since we are focusing on conjunctive queries, we base our analysis on the 1.25 million valid `SELECT` queries. This dataset can be characterized as a diverse workload of both machine- and human-generated queries.

7.2 Common Experiment Settings

Experiments were performed on operating system macOS Sierra with 2.8 GHz Intel Core i7 CPU, 16 GB 1600 MHz DDR3 memory and a SSD.

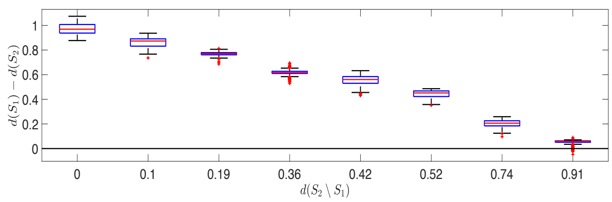
Constant Removal. A number of queries in US Bank differ only in hard-coded constant values. Table 1 shows the total number of queries, as well as the number of distinct queries if we ignore constants. By comparison, queries in PocketData all use JDBC parameters. For these experiments, we ignore constant values in queries.

Query Regularization. We apply query rewrite rules (similar to [7]) to regularize queries into equivalent conjunctive forms, where possible. Table 1 shows that $\frac{135}{605}$ and $\frac{1494}{1712}$ of distinct queries are in conjunctive form for PocketData and US bank respectively. After regularization, all queries in both data sets can be either simplified into conjunctive queries or re-written into a `UNION` of conjunctive queries compatible with the Aligon et. al.'s feature scheme [1].

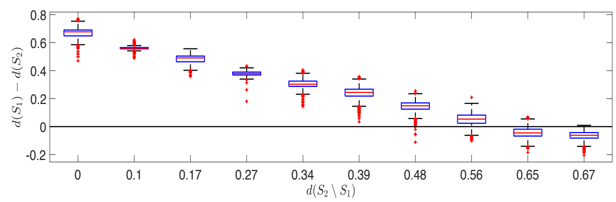
Convex Optimization Solving. All convex optimization problems involved in measuring Reproduction Error and Deviation are solved by the *successive approximation heuristic* implemented by Matlab CVX package [17] with Sedumi solver.

7.3 Validating Reproduction Error

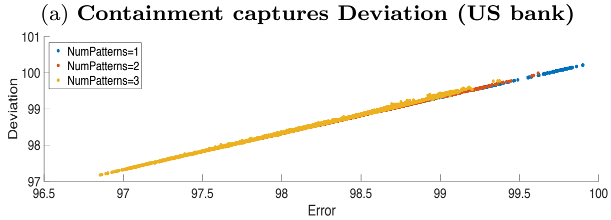
In this section, we validate that Reproduction Error is the practical alternative for Deviation. In addition, we also offer measurements on its correlation with Deviation, as well as two other related measures. As it is impractical to enumerate all possible encodings, we choose a subset of encodings for both PocketData and US bank datases. Specifically, we first select a subset of features having a marginal within



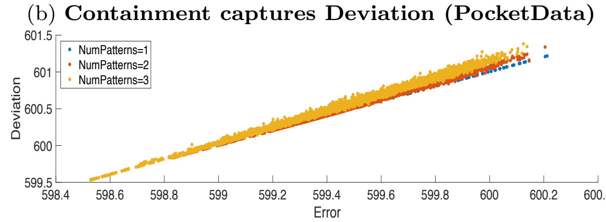
(a) **Containment captures Deviation (US bank)**



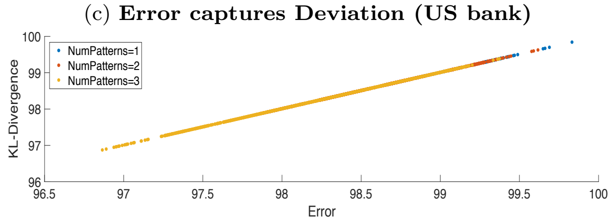
(b) **Containment captures Deviation (PocketData)**



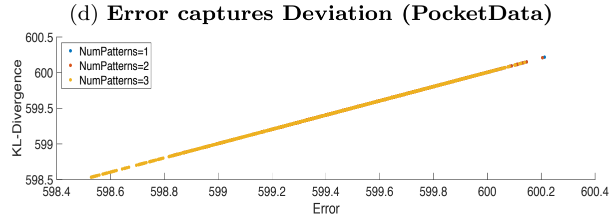
(c) **Error captures Deviation (US bank)**



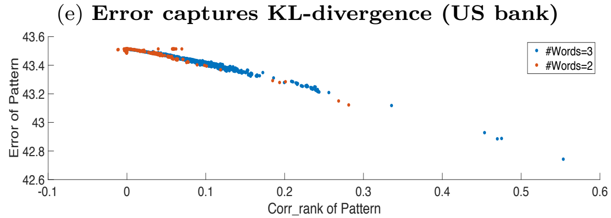
(d) **Error captures Deviation (PocketData)**



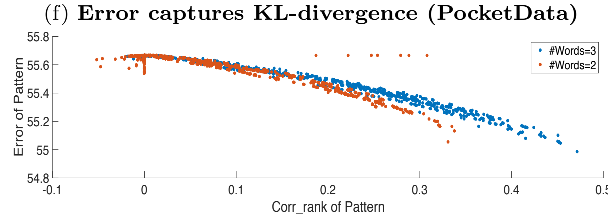
(e) **Error captures KL-divergence (US bank)**



(f) **Error captures KL-divergence (PocketData)**



(g) **Error captures Correlation (US bank)**



(h) **Error captures Correlation (PocketData)**

Figure 4: **Validating The Reproduction Error Metric**

$[0.01, 0.99]$ and use these features to construct patterns. We then enumerate combinations of K (up to 3) patterns as our chosen encodings.

Containment Captures Deviation. Here we empirically verify that containment (Section 4.2) captures Deviation (i.e., $\mathcal{E}_1 \leq_\Omega \mathcal{E}_2 \rightarrow d(\mathcal{E}_1) \leq d(\mathcal{E}_2)$) to complete the chain of reasoning that Reproduction Error captures Deviation. Figures 4a and 4b show all pairs of encodings where $\mathcal{E}_2 \supset \mathcal{E}_1$. The y-axis shows the difference in Deviation values (i.e., $d(\mathcal{E}_2) - d(\mathcal{E}_1)$). Deviation $d(S)$ is approximated by drawing 1,000,000 samples from the space of possible patterns. For clarity, we bin pairs of encodings by the degree of overlap between the encodings, measured by the Deviation of the set-difference between the two encodings $d(\mathcal{E}_2 \setminus \mathcal{E}_1)$; Higher $d(\mathcal{E}_2 \setminus \mathcal{E}_1)$ implies less overlap. Values of y-axis fall within each bin are visualized by Matlab boxplot (i.e., the blue box indicates the range within standard deviation and red/black crosses are outliers). Intuitively, all points above zero on the y-axis (i.e., $d(\mathcal{E}_2) - d(\mathcal{E}_1) > 0$) are pairs of encodings where Deviation order agrees with containment order. This is the case for virtually all encoding pairs.

Additive Separability of Deviation. We also observe from Figures 4a and 4b that agreement between Deviation and containment order is correlated with overlap; More similar encodings are more likely to have agreement. Combined with Proposition 2, this first shows that, for similar encodings Reproduction Error is likely to be a reliable indicator of Deviation. This also suggests that Deviation is additively separable: The information loss (measured in $d(\mathcal{E}_2) - d(\mathcal{E}_1)$) by excluding encoding $\mathcal{E}_2 \setminus \mathcal{E}_1$ from $\mathcal{E}_2$ closely correlates with the quality (i.e., $d(\mathcal{E}_2 \setminus \mathcal{E}_1)$) of encoding $\mathcal{E}_2 \setminus \mathcal{E}_1$ itself:

$$\mathcal{E}_2 \supset \mathcal{E}_1 \rightarrow d(\mathcal{E}_2) - d(\mathcal{E}_1) < 0 \quad \text{and} \quad d(\mathcal{E}_2 \setminus \mathcal{E}_1) \approx d(\mathcal{E}_2) - d(\mathcal{E}_1)$$

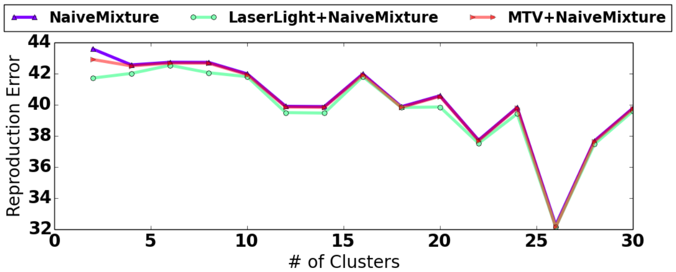
Error Correlates With Deviation. As a supplement, Figures 4c and 4d empirically confirm that that Reproduction Error (x-axis) indeed closely correlates with Deviation (y-axis). Mirroring our findings above, correlation between them is tighter at lower Reproduction Error.

Error Correlates With KL-Divergence. Figures 4e and 4f show the relationship between Reproduction Error (x-axis) and KL-Divergence between the true distribution ρ^* and the space representative distribution $\overline{\rho}_S$ (y-axis), as discussed in Section 4.1. The two are tightly correlated.

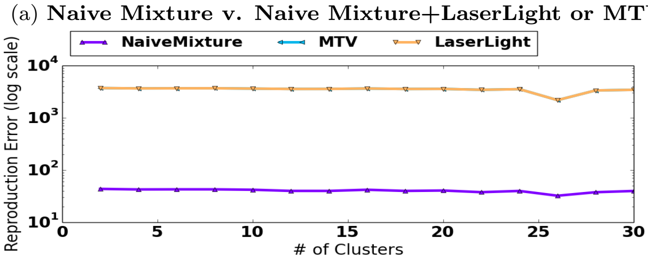
Error and Feature-Correlation. Figure 4g and 4h show the relationship between Reproduction Error (y-axis) and the feature-correlation score *corr_rank* (x-axis), as defined in Section 6.4. Values of y-axis are computed from the naive encoding extended by a single pattern $\mathbf{b}$ containing multiples features (up to 3). One can observe that Reproduction Error of extended naive encodings almost linearly correlates with *corr_rank*($\mathbf{b}$). In addition, one can also observe that *corr_rank* becomes higher when the pattern $\mathbf{b}$ encodes more correlated features.

7.4 Feature-Correlation Refinement

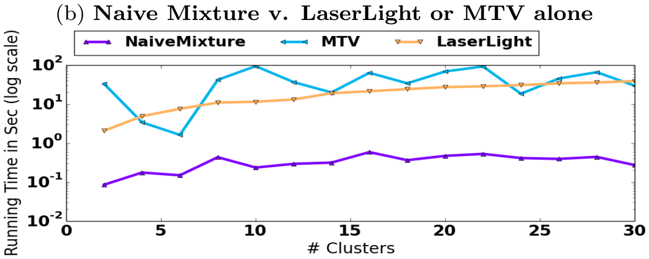
In this section, we design experiments serving two purposes: (1) Evaluating the potential reduction in Error from



(a) **Naive Mixture v. Naive Mixture+LaserLight or MTV**



(b) **Naive Mixture v. LaserLight or MTV alone**



(c) **Running Time Comparison**

Figure 5: **Naive Mixture Encodings (US bank)**

refining naive mixture encodings through state-of-the-art pattern based summarizers, and (2) Evaluating whether we can replace naive mixture encodings by the encodings created from summarizers that we have plugged-in.

Experiment Setup. To serve both purposes, we construct pattern mixture encodings under three different configurations: (1) Naive mixture encodings; (2) Naive mixture encodings refined/plugged-in by pattern based encodings and (3) The pattern mixture encoding that summarizes each cluster using only pattern based summarizers (i.e., replacing the naive mixture encoding). We first choose KMeans with Euclidean distance from which naive mixture encodings are constructed. We then choose two state-of-the-art pattern based summarizers to generate pattern based encodings: (1) *Laserlight* [12] algorithm, which aims at summarizing multi-dimensional data $D = (X_1, \ldots, X_n)$ augmented with an additional binary attribute A; (2) *MTV* [31] algorithm, which aims at mining maximally informative patterns as encodings from multi-dimensional data of binary attributes.

The experiment results are shown in Figure 5 which contains 3 sub-figures. All sub-figures share the same x-axis, i.e., the number of clusters. Figure 5a evaluates the possible change in Error (y-axis) by plugging-in *MTV* and *Laserlight*. Figure 5b compares the Error (y-axis) between the naive mixture encoding and the pattern mixture encoding obtained from only using patterns from *MTV* and *Laserlight*. Figure 5c compares the running time (y-axis) between constructing naive mixture encodings and applying pattern based summarizers. We only show the results for US bank data set as results for PocketData give similar observations.

7.4.1 *Replacing Naive Mixture Encodings*

Figure 5b and 5c show that replacing naive mixture encodings by pattern based encodings is not recommended from two perspectives.

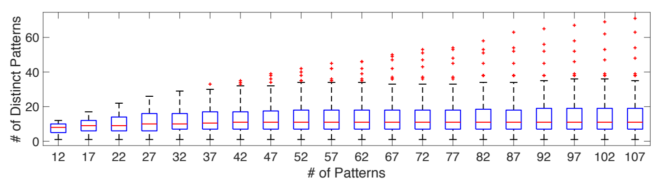
Error Perspective. We observe from Figure 5b that Errors of naive mixture encodings are orders of magnitude lower than Errors of summarizing each cluster using only patterns generated from *Laserlight* or *MTV*. Note that the curve of MTV overlaps with that of Laserlight. In other words, a naive mixture encoding is the basic building block of a more general pattern mixture encoding.

Computation Efficiency Perspective. Furthermore, as one can observe from Figure 5c, that the running time of constructing naive mixture encodings is significantly lower than that of *Laserlight* and *MTV*.

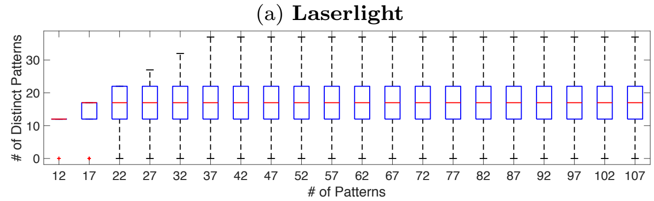
7.4.2 *Refining Naive Mixture Encodings*

The experiment result is shown in Figure 5a. Note that we offset y-axis to show the change in Error. We observe from the figure that reduction on Error contributed by plugging-in pattern based summarizers is small for both algorithms. This is due to (1) restriction on feature vector dimension and (2) pattern diversification failure.

Vector Dimension Restriction. Unrestrictedly high dimensional feature vectors can cause problems in storing and analyzing them. Thus in real life implementations, there is restriction on vector dimensions, e.g. *Laserlight* is implemented in PostgresSQL 9.1 which has a threshold of 1000 features/columns for a table. Dimension reduction methods based on matrix decomposition, e.g. PCA, transform integer-valued feature vectors into a real-valued space where patterns cannot be defined. Instead, practically we simply prune and keep top 1000 features according to variation of their values. This pruning process inevitably reduces the effectiveness of *Laserlight*.



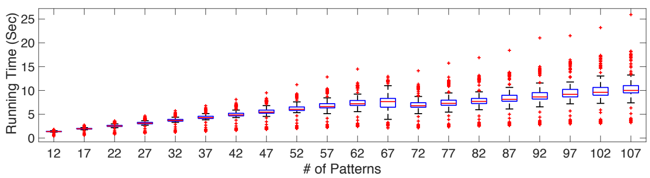
(a) **Laserlight**



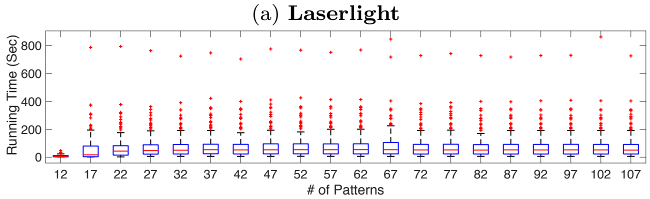
(b) **MTV**

Figure 6: **Distinct Patterns v. Number of Patterns**

Pattern Diversification Failure. Recall in Section 6.4 that pattern diversification can be time-consuming. As a result, both algorithms implement heuristics that sacrifice the capability of pattern diversification to some extent. In other words, increase in the total number of patterns mined from *Laserlight* and *MTV* may not lead to commensurate increase in the number of *distinct* patterns (i.e., pattern duplication implies failure in pattern diversification). Specifically, we run *MTV* and *Laserlight* on each cluster (30 clusters in total) and vary the number of patterns configured for mining from 12 to 107. We collect the number of *distinct* patterns that they have mined as well as the running time under each configuration. The experiment result is shown in Figure 6 where y-axis is the number of distinct patterns and x-axis is the total number of patterns mined through *Laserlight* and *MTV*. We observe that *Laserlight* and *MTV* fail to extensively explore hidden feature-correlation in our chosen datasets.



(a) **Laserlight**



(b) **MTV**

Figure 7: **Running Time v. Number of Patterns**

Time Restriction. Computation efficiency will also influence effectiveness of pattern based summarizers when it takes too long to mine a potentially large set of informative patterns under from the data. The experiment result for running time analysis is shown in Figure 7. We observe that *Laserlight* shows a close-to-linear growth rate. For *MTV* the running time spikes at 17 patterns and remains oscillating between high levels afterwards.

8. RELATED WORK

An extensively studied problem *Text Summarization* is closely-related. Its objective is condensing the source text, which is a collection/sequence of sentences into a shorter version preserving its information content and overall meaning. Methods achieving such objective can be classified into two categories [18]: (1) abstractive and (2) extractive.

An extractive summarization is simply ranking and extracting *representative* or *relevant* records from the source log. Different extractive methods only differ in the measure of relevance. The simplest measure would be *Term frequency-inverse document frequency (TF-IDF)* [15]. More sophisticated measures include but not limited to *Lexical Chain* based [3], *graph-based lexical centrality* [13], *latent semantic analysis* based [16], *machine learning* based [32], *hidden markov model* based [9], *neural network* based [23], *query* based [38], *mathematical regression* based [14] and *fuzzy-logic* based [29].

A survey on extractive methods [18] pointed out two of their problems: (1) Not all text in extracted records are relevant, resulting in unnecessary verbosity and (2) Relevant information tends to either overlap or spread among records, resulting in unexpectedly large amount of records extracted to achieve reasonable coverage on information content. These problems can be solved by extracting only *patterns* or fragments commonly shared by records, e.g *frequent pattern mining* [19]. Frequency is not the only measure of relevance for patterns. Mampaey *et al.* [31] study the problem of summarizing multi-dimensional vectors where all attributes are binary. The summary is represented as a set of *most informative* patterns which maximize the *Bayesian Information Criterion* (BIC) score. Gebaly *et al.* [12] aim at summarizing multi-dimensional categorical-valued vectors augmented by a binary attribute. The relevance of pattern depends on whether it can *explains* the valuation of the augmented attribute.

However, there is still controversy on summary evaluation for extractive methods. Evaluation measures such as ROGUE [40] and BLEU [34] are human-involved. Since human can only participate as reference summarizers in data sets limited in size and number, it is controversial that the performance of a summarizer evaluated on existing limited data sets will generalize to unseen real life data sets.

An abstractive summarization attempts to go beyond extraction and builds up probabilistic generative models. The goal is to search for relevant *topics* [4] or *abstracts* [26] represented by latent variables in the model inferred from the data in the log. A probabilistic model is usually called generative as it is able produce a world of possible logs where the log representing the actual data has the maximum likelihood to be produced (i.e. Maximum Likelihood Estimate). Typical models used for summarization are *probabilistic topic models* [4, 39] and *noisy-channel* model [26].

Abstractive summarization using probabilistic models has its own problems. Firstly, goodness of fit on data (i.e., likelihood) is model-dependent and cannot be used to compare different models. As a result, the general practice of evaluating probabilistic models is also using human as reference summarizers (see [26, 39]). Hence similar controversy in summary evaluation still exists. In addition, as [18] pointed out, the world of logs that any model can produce (i.e., model capacity) is limited. There may exist logs that exceeds model capacity. Finally, unlike understanding extracted records or shared patterns, it may require a steep learning curve for an human observer to consume and utilize information conveyed by probabilistic models.

9. CONCLUSION

In this paper, we introduced the problem of log compression and defined a family of pattern based log encodings. We precisely characterized the information content of logs and offered three principled and one practical measures of encoding quality: Verbosity, Ambiguity, Deviation and Reproduction Error. To reduce the search space of pattern based encodings, we introduced the idea of partitioning logs into separate components, which induces the family of pattern mixture as well as its simplified form: naive mixture encodings. Finally, we experimentally showed that naive mixture encodings are more informative and can be constructed more efficiently than encodings constructed from state-of-the-art pattern based summarization techniques.

10. REFERENCES

[1] Aligon, J., Golfarelli, M., Marcel, P., Rizzi, S., and Turricchia, E. Similarity measures for olap sessions. *Knowledge and Information Systems 39*, 2 (May 2014), 463–489.

[2] Aouiche, K., Jouve, P.-E., and Darmont, J. Clustering-based materialized view selection in data warehouses. In *ADBIS* (2006).

[3] Barzilay, R., and Elhadad, M. Using lexical chains for text summarization. In *In Proceedings of the ACL Workshop on Intelligent Scalable Text Summarization* (1997), pp. 10–17.

[4] Blei, D. M. Probabilistic topic models. *Commun. ACM 55*, 4 (Apr. 2012), 77–84.

[5] Boyd, S., and Vandenberghe, L. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[6] Bruno, N., and Chaudhuri, S. Automatic physical database tuning: A relaxation-based approach. In *ACM SIGMOD* (2005).

[7] Chandra, B., Joseph, M., Radhakrishnan, B., Acharya, S., and Sudarshan, S. Partial marking for automated grading of sql queries. *Proc. VLDB Endow. 9*, 13 (Sept. 2016), 1541–1544.

[8] Chatzopoulou, G., Eirinaki, M., Koshy, S., Mittal, S., Polyzotis, N., and Varman, J. S. V. The QueRIE system for personalized query recommendations. *IEEE Data Eng. Bull.* (2011).

[9] Conroy, J. M., and O'leary, D. P. Text summarization via hidden markov models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2001), SIGIR '01, ACM, pp. 406–407.

[10] Darroch, J. N., and Ratcliff, D. Generalized iterative scaling for log-linear models. *Ann. Math. Statist. 43*, 5 (10 1972), 1470–1480.

[11] Dwork, C. *ICALP 2006, Proceedings, Part II*. Springer, 2006, ch. Differential Privacy.

[12] El Gebaly, K., Agrawal, P., Golab, L., Korn, F., and Srivastava, D. Interpretable and informative explanations of outcomes. *Proc. VLDB Endow. 8*, 1 (Sept. 2014), 61–72.

[13] Erkan, G., and Radev, D. R. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res. 22*, 1 (Dec. 2004), 457–479.

[14] Fattah, M. A., and Ren, F. Automatic text summarization. *World Academy of Science, Engineering and Technology 37* (2008), 2008.

[15] Garca-Hernndez, R. A., and Ledeneva, Y. Word sequence models for single text summarization. In *2009 Second International Conferences on Advances in Computer-Human Interactions* (Feb 2009), pp. 44–48.

[16] Gong, Y., and Liu, X. Generic text summarization using relevance measure and latent semantic analysis. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2001), SIGIR '01, ACM, pp. 19–25.

[17] Grant, M., and Boyd, S. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, Mar. 2014.

[18] Gupta, V., and Lehal, G. A survey of text summarization extractive techniques.

[19] Han, J., Cheng, H., Xin, D., and Yan, X. Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov. 15*, 1 (Aug. 2007), 55–86.

[20] Jain, A. K. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett. 31*, 8 (June 2010), 651–666.

[21] Jaynes, E., and Bretthorst, G. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[22] Johnson, S. C. Hierarchical clustering schemes. *Psychometrika 32*, 3 (Sep 1967), 241–254.

[23] Kaikhah, K. Automatic text summarization with neural networks. In *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference* (June 2004), vol. 1, pp. 40–44 Vol.1.

[24] Kannan, R., Vempala, S., and Vetta, A. On clusterings: Good, bad and spectral. *J. ACM 51*, 3 (May 2004), 497–515.

[25] Kennedy, O., Ajay, J. A., Challen, G., and Ziarek, L. Pocket data: The need for tpc-mobile. In *TPC-TC* (2015).

[26] Knight, K., and Marcu, D. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artif. Intell. 139*, 1 (July 2002), 91–107.

[27] Kul, G., Luong, D., Xie, T., Coonan, P., Chandola, V., Kennedy, O., and Upadhyaya, S. Ettu: Analyzing query intents in corporate databases. In *Proceedings of the 25th International Conference Companion on World Wide Web* (2016), pp. 463–466.

[28] Kullback, S., and Leibler, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86.

[29] Kyoomarsi, F., Khosravi, H., Eslami, E., Dehkordy, P. K., and Tajoddin, A. Optimizing text summarization based on fuzzy logic. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)* (May 2008), pp. 347–352.

[30] Makiyama, V. H., Raddick, M. J., and Santos, R. D. Text mining applied to SQL queries: A case study for the SDSS SkyServer. In *SIMBig* (2015).

[31] Mampaey, M., Vreeken, J., and Tatti, N. Summarizing data succinctly with the most informative itemsets. *ACM Trans. Knowl. Discov. Data 6*, 4 (Dec. 2012), 16:1–16:42.

[32] Neto, J. L., Freitas, A. A., and Kaestner, C. A. A. Automatic text summarization using a machine learning approach. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence* (London, UK, UK, 2002), SBIA '02, Springer-Verlag, pp. 205–215.

[33] O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications 169*, 3 (Jun 2016), 1042–1068.

[34] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Stroudsburg, PA, USA, 2002), ACL '02, Association for Computational Linguistics, pp. 311–318.

[35] Park, H.-S., and Jun, C.-H. A simple and fast algorithm for k-medoids clustering. *Expert Syst. Appl. 36*, 2 (Mar. 2009), 3336–3341.

[36] Pavlo, A., Angulo, G., Arulraj, J., Lin, H., Lin, J., Ma, L., Menon, P., Mowry, T. C., Perron, M., Quah, I., Santurkar, S., Tomasic, A., Toor, S., Aken, D. V., Wang, Z., Wu, Y., Xian, R., and Zhang, T. Self-driving database management systems. In *CIDR* (2017), www.cidrdb.org.

[37] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[38] Pembe, F. C. Automated query-biased and structure-preserving document summarization for web search tasks.

[39] Wang, D., Zhu, S., Li, T., and Gong, Y. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers* (Stroudsburg, PA, USA, 2009), ACLShort '09, Association for Computational Linguistics, pp. 297–300.

[40] yew Lin, C. Rouge: a package for automatic evaluation of summaries. pp. 25–26.