Progressive Evaluation of Queries over Tagged Data

Dhrubajyoti Ghosh  
University of California, Irvine  
dhrubajg@uci.edu

Roberto Yus  
University of California, Irvine  
ryuspeir@uci.edu

Yasser Altowim  
King Abdulaziz City for Science and Technology  
yaltowim@kacst.edu.sa

Sharad Mehrotra  
University of California, Irvine  
sharad@ics.uci.edu

ABSTRACT
Modern information systems often collect raw data in the form of text, images, video, and sensor readings. Such data needs to be further interpreted/enriched prior to being analyzed. Enrichment is often a result of automated machine learning and or signal processing techniques that associate appropriate but uncertain tags with the data. Traditionally, with the notable exception of a few systems, enrichment is considered to be a separate pre-processing step performed independently prior to data analysis. Such an approach is becoming increasingly infeasible since modern data capture technologies enable creation of very large data collections for which it is computationally difficult/impossible and ultimately not beneficial to derive all tags as a preprocessing step. Hence, approaches that perform tagging at query-/analysis time on the data of interest need to be considered. This paper explores the problem of joint tagging and query processing. In particular, the paper considers a scenario where tagging can be performed using several techniques that differ in cost and accuracy and develops a progressive approach to answering Select-Project-Join (SPJ) queries (with a restricted version of the join predicates) that enriches the right data to the right degree so as to maximize the quality of the query results. The experimental results show that the proposed approach performs significantly better compared to baseline approaches.

PVldb Reference Format:  
Dhrubajyoti Ghosh, Roberto Yus, Yasser Altowim, Sharad Mehrotra. Progressive Evaluation of Queries over Tagged Data. *PVLDB*, 11(44): xxxx-yyyy, 2018.  
DOI: https://doi.org/TBD

1 Such approaches can be viewed as motivated by the “schema on query” principle espoused for the data lakes.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 45th International Conference on Very Large Data Bases, August 2019, Los Angeles, California.  
Proceedings of the VLDB Endowment, Vol. 11, No. 44  
Copyright 2018 VLDB Endowment 2150-8097/18/10.. $ 10.00.  
DOI: https://doi.org/TBD

1. INTRODUCTION
Modern information systems often collect raw data in the form of text, images, video, and sensor readings. Queries over such data arise in several domains such as social media analysis, multimedia data analysis, and Internet-of-Things applications. However, such data is often too low-level for the end applications due to the existing semantic gap and hence need to be further enriched/tagged before they can be queried. Given the large amount of data, such tags are generated using automated functions (e.g., pre-trained classifiers) and/or data extraction or analysis mechanisms (e.g., sentiment detection) and hence are often uncertain.

The amount of uncertainty present in the generated tags depends on the cost and quality of these tagging functions. A more expensive tagging function usually produces better quality tags (i.e., tags with less uncertainty). For instance, executing more than one tagging function and combining their results, although computationally expensive, has been shown to reduce the uncertainty of tags [13, 25, 29]. As an example, Table I shows the trade-off between quality and cost of different tagging functions that extract the gender from images using two real datasets used in our experiments.

| Dataset   | Tagging Function     | Quality (AUC) | Cost (Sec.) |
|-----------|----------------------|---------------|-------------|
| MUCT      | Decision Tree        | 0.64          | 0.023       |
| MUCT      | Gaussian Naive-Bayes | 0.67          | 0.114       |
| MUCT      | Support Vector Machine | 0.71   | 0.949       |
| Multi-PIE | Decision Tree        | 0.55          | 0.018       |
| Multi-PIE | Gaussian Naive-Bayes | 0.84          | 0.096       |
| Multi-PIE | Support Vector Machine | 0.89   | 0.886       |

Table 1: Average cost per image and quality of different tagging functions.

Our goal in this paper is to perform query processing on such datasets using such tagging functions. Query processing would be simple if the underlying dataset could be pre-tagged prior to the arrival of queries. However, there are several reasons why such pre-tagging is not possible. First and foremost, pre-tagging a dataset requires that we exactly know a priori all the analysis tasks that we want to implement over the dataset in the future. Let us consider an example of a dataset of surveillance camera images and a query for catching a perpetrator of a theft. The fact that one is interested in a person with a purple shirt with a backpack might be a result of a memory of a person who saw someone...
Figure 1: Comparison of different evaluation approaches.

suspicious that instigated the query. It is virtually impossible to know that such an analysis query would be of interest beforehand. Second, even if we have created the tags for each object, we might not have used the most accurate tagging function for generating those tags which will result in low quality results.

Motivated by the above scenarios, we propose a progressive approach to evaluating queries over untagged data. Figure 1 illustrates the concept of progressive evaluation and how it differs from traditional and incremental evaluation approaches. In the traditional approach, all the required tagging functions are executed on all the objects first before answering any query. In this scenario, the quality of the result will stay low for a long time (i.e., until all the tags have been generated using all the tagging functions and for all the objects) and then reaches a very high value. Answering a query using this approach will take a large amount of time especially if the dataset is large and/or the number of tags and the number of functions for evaluating those tags are large. In the incremental evaluation, the algorithm is configured to constantly improve the quality of the answer by improving the quality of the tags of objects.

Both of these approaches are not suitable for applications that require low latency response (and thus cannot tolerate delays caused by generating accurate tags for each object), and/or interactive applications that need to show preliminary analysis results based on moderately tagged objects and progressively refine their results by executing more tagging functions to improve the tagging accuracy. On the other hand, a progressive approach aims to improve the quality of the tags in such a way so that the rate at which the quality of the answer set improves is maximized. Such an approach can substantially reduce the query evaluation cost since the evaluation process can be prematurely terminated whenever a satisfactory level of quality is achieved. Progressive evaluation has been studied in the literature in various contexts such as in adaptive query processing through sampling, in wavelet-based approximate query processing, and in data cleaning.

The main idea of our progressive approach is to prioritize/rank which tags should be evaluated on which objects first in order to improve the quality of the answer set as quickly as possible. To this end, our approach divides the query execution time into several epochs and analyzes the evaluation progress at the beginning of each epoch to generate an execution plan for the current epoch. An evaluation plan consists of multiple triples that have the highest potential of improving the quality of the answer in that epoch, where a triple consists of an object, a query predicate, and a tagging function for evaluating the predicate on the object. In summary, our contributions in this paper are as follows:

- We propose a progressive approach to evaluating queries over untagged datasets.
- We present an algorithm for the problem of generating an execution plan that has the highest potential of improving the quality of the answer set in the corresponding epoch.
- We present an efficient strategy that estimates the benefits of the possible triples (i.e., object-predicate-function triple) based on different properties (i.e., output of previous functions, cost and quality of remaining functions).
- We experimentally evaluated our approach in different domains (i.e., tweets and images) using real datasets and tagging functions and demonstrate the efficiency of our proposed solution.

| Object | Gender | Expression | Glasses | Timestamp       |
|--------|--------|------------|---------|----------------|
| o1     | Male   | Smile      | False   | 2018-05-15T03:48 |
| o2     | Male   | Smile      | False   | 2018-05-15T03:52 |
| o3     | Male   | Neutral    | False   | 2018-05-15T03:54 |
| o4     | Female | Neutral    | True    | 2018-04-11T11:08 |
| o5     | Female | Smile      | True    | 2018-04-11T11:12 |
| o6     | Female | Neutral    | False   | 2018-04-11T11:14 |

Table 2: Running Example.

2This example is inspired by a real occurrence in the authors’ Computer Science Department building wherein the research infrastructure of cameras was used to create an image profile of a perpetrator who stole several computers, iPads, and phones from several offices.

2. PROBLEM SETUP

In this section we first describe the notations used in this paper and then define our problem formally.

**Dataset.** Let $O = \{o_1, o_2, \ldots, o_{|O|}\}$ be a dataset of objects that have the same object type (i.e., images, tweets, etc.). Attributes in the dataset can be either precise or imprecise. Precise attributes are those whose values are certain whereas imprecise attributes are those whose initial values are empty and thus require running (possibly a sequence of) tagging functions to identify them. As discussed earlier, the tags/values generated from such functions are often uncertain. For instance, Table 2 shows an example with a dataset of six images. In this dataset, there is a precise attribute, Timestamp, which was associated with the images at the time of their capture, and three imprecise attributes, Gender, Expression, and Glasses, which require running tagging functions on the objects in order to determine their values. (Note that the values in brackets in the table for the imprecise attributes correspond to the values in the ground truth.)

Each imprecise attribute is associated with a tag type $T_i$. The dataset in Table 2 is associated with three tag types $T_1 = \text{Gender}$, $T_2 = \text{Expression}$, and $T_3 = \text{Glasses}$ that are associated with the three attributes Gender, Expression, and Glasses respectively. Each object $o_i$ is associated with a tag $t_j$ per each tag type $T_k$. In this case, we say that $t_j \in T_k$, implying that $t_j$ is a possible tag for $T_k$. Object $o_1$ in Table 2, for example, is associated with three tags Male $\in$ Gender, Smile $\in$ Expression, and True $\in$ Glasses.

**Tagging Functions.** Let $F_i = \{f_1, f_2, \ldots, f_k\}$ be a set of tagging functions associated with the tag type $T_i$. Each
function $f_j^i$ takes as input an object $o_k$ and a tag $t_i \in T_i$, and returns the probability of the object containing the tag $t_i$, denoted as $p_{j,k}^i$. For example, given the dataset in Table 2, the tag type Gender can be evaluated using two tagging functions: a tagging function based on a Support Vector Machine classifier and/or a tagging function based on a Decision Tree classifier. Each of these functions takes as input an image and a tag (e.g., a Male) and outputs the probability value of the image containing that tag (e.g., 0.8).

Each function $f_j^i$ is associated with a quality and cost value, which we denote as $q_j^i$ and $c_j^i$ respectively. The quality of a function measures the accuracy of the function in detecting the tags. For example, the quality of a function that is internally implemented using a machine learning classifier is typically measured using the area under the curve (AUC) score of the classifier [6]. The cost of a function represents the average cost of running the function on an object. Our approach is agnostic to the way the quality and cost values are set. For instance, they can be determined by running the function on a sample validation dataset. In practice, the higher the quality of a function is, the higher its cost is.

For clarity, we will present the paper as if none of tagging functions were run on any objects prior to the arrival of the query. In Section 5, we discuss how our approach deals with the case where the outputs of the tagging functions run in previous queries are cached and study the impact of different levels of caching on the performance of our approach.

**Queries.** We consider Select-Project-Join (SPJ) queries as they are the most common types of queries in our settings of interest. The predicates in the where clause can be either over precise or imprecise attributes. However, we assume that the join predicates do not contain any imprecise attributes as they are probabilistic and contain uncertainty. Such a complex join predicate requires non-trivial extension and hence is out of the scope of the paper.

An example of a query $Q$ on the dataset in Table 2 is as follows:

```
SELECT * FROM ImageDataset
WHERE Gender = Male and Expression != Smile
AND Timestamp between "2018-01-01T00:00:00" and "2018-06-00T00:00:00"
```

This query contains one precise predicate on the Timestamp attribute and two imprecise predicates on the Gender and Expression attributes. Since evaluating imprecise predicates requires running tagging functions (and hence is more expensive than evaluating precise predicates), the query execution plan will evaluate precise predicate first in order to reduce the amount of tagging required [17]. Note that we assume that the objects that satisfy the precise predicates, and thus require tagging, can fit in memory.

In the following, we will focus only on the set of imprecise predicates. This set of predicates is used henceforth to refer to an imprecise predicate and are connected by boolean connective operators: And = Λ or Or = ∨. A predicate, denoted as $R_k^i$, contains a tag type $T_i$, a tag $t_j$, and a boolean connective operator. The operator can be of two types: equal (denoted as “==”) or not equal (denoted as “≠=”) An example of a predicate can be $Value(Gender) == Male$. (For ease of notation, we will omit word Value from the left operand henceforth).

We assume that if a predicate refers to a particular tag type, then the system has at least one tagging function available for that type and, as stated before, join predicates involving imprecise attributes (e.g., Dataset1.Gender == Dataset2.Gender) are outside the scope of this paper.

Given a query $Q$ containing $n$ predicates, we associate a predicate probability with the objects in $O$. It is defined as the probability of the object satisfying a particular predicate mentioned in $Q$. Predicate probability of an object $o_i$ w.r.t. a predicate $R_k^i$ is denoted by $p_{i,k}^j$, and it is a function of the probability outputs of the tagging functions. We calculate predicate probability of an object as follows:

$$p_{j,k}^i = M_j(p_{1,k}^i, p_{2,k}^i, \ldots, p_{n,k}^i)$$

where function $M_j$ is a combine function that takes the output of the tagging functions in $F_j$ and returns the predicate probability of $o_i$ satisfying predicate $R_k^i$. This function is learned offline using a labeled training dataset.

**Quality Metric.** Each query answer has a notion of quality associated with it. A quality metric represents how close the answer set $A$ is to the ground truth set $G$. In a set based answer, $F_o$-measure is the most popular and widely used quality metric. It is the harmonic mean of precision and recall. Precision (Pre) is defined as the fraction of correct objects in $A$ to the total number of objects in $A$, whereas the recall (Rec) is defined as the fraction of correct objects in $A$ to the total number of objects in $G$. More formally, the $F_o$-measure is computed as follows:

$$Pre = \frac{|A \cap G|}{|A|}, \quad Rec = \frac{|A \cap G|}{|G|}, \quad F_o = \frac{(1 + \alpha) \cdot Pre \cdot Rec}{(\alpha \cdot Pre + Rec)}$$

where $\alpha \in [0,1]$ denotes the weight factor assigned to the precision value.

**Problem Statement.** Given a dataset $O$, a query $Q$, and a set of tagging functions for each tag type, the goal is to develop a progressive approach that maximizes the rate at which the quality of the answer set $A$ improves.

The quality of a progressive approach can be measured by the following discrete sampling function [26]:

$$Qty(A) = \sum_{i=1}^{V} W(v_i) \cdot Imp(v_i)$$

where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ is a set of sampled cost values (s.t. $v_i > v_{i-1}$), $W$ is a weighting function that assigns a weight value $W(v_i) \in [0,1]$ to every cost value $v_i$ (s.t. $W(v_i) > W(v_{i-1})$), and $Imp(v_i)$ is the improvement in the $F_o$ measure of $A$ that occurred in the interval $(v_{i-1}, v_i]$. (For convenience, we assume the existence of a cost value $v_0 = 0$ henceforth.) In other words, $Imp(v_i)$ is the $F_o$ measure of $A$ at $v_i$ minus the $F_o$ measure of $A$ at $v_{i-1}$.

3. OUR APPROACH

Our approach divides the execution time into multiple epochs. Each epoch consists of three phases: plan generation, plan execution, and answer set selection. A plan consists of a list of triples where each triple consists of an object
o_j, a predicate R^R_k, and a tagging function f^R_m. In the plan generation phase, our approach determines the list of triples that should be evaluated in the plan execution phase of that epoch. A triple (o_j, R^R_k, f^R_m) is evaluated by simply calling the function f^R_m and passing to it the object o_j and the tag t_l present in the predicate R^R_k. Identifying the list of triples in the plan depends on a trade-off between the cost of evaluating those triples and the expected improvement in the quality of the answer set that will result from evaluating them.

3.1 Plan Generation Phase

The process of generating a plan takes three inputs: the state, joint probability, and uncertainty values of each object in O. We define these terms as follows:

Definition 1. (State). A State of an object o_i w.r.t. a predicate R^R_k present in Q is denoted as s^R_i,k, and defined as the set of tagging functions in f^R_k that have already been executed on o_i to determine if it is associated with tag t_k.

Our approach maintains for each object a state value for each predicate mentioned in the query. The set of all state values of an object o_i w.r.t. a query Q is denoted to as a state vector and denoted as S^i.

Definition 2. (Joint Probability). Joint probability value of an object o_i w.r.t. a query Q is denoted by P_{i}, and is defined as the probability of the object satisfying all the predicates in Q.

To compute the joint probability, we assume that any two predicates containing two different tags of the same tag type (e.g., Expression == Smile and Expression == Neutral) to be mutually exclusive and any two predicates containing tags of different tag types (e.g., Expression == Neutral and Gender == Male) to be independent.

To illustrate, consider the example query provided in the previous section. Let us denote the tag types Gender, Glasses, and Expression as T_1, T_2, and T_3, respectively, and the predicates as R^R_1, R^R_2, and R^R_3, respectively. Suppose the predicate probability values for object o_m are p_{1,m} = 0.8, p_{2,m} = 0.7, and p_{3,m} = 0.9, respectively. Then, the joint probability value P_m is (0.8 · 0.7) + 0.9 · (0.8 · 0.7) · 0.9 = 0.956.

Definition 3. (Uncertainty). We associate a concept of uncertainty with each object o_i in O. The uncertainty of o_i is measured using the entropy of the object. Given a discrete random variable X, the entropy is defined as follows:

$$H(X) = -\sum_x P_r(X=x) \cdot \log(P_r(X=x))$$  

where x is a possible value of X and P_r(X=x) is the probability of X taking the value x. In our setup, we consider a tag type T_l as a random variable and tags of this tag type are considered as the possible values of the random variable.

We denote uncertainty value of an object o_i with respect to a predicate R^R_k as h^R_{i,k}. We calculate this value using the predicate probability value as follows:

$$h^R_{i,k} = \begin{cases} -p^R_{i,k} \cdot \log(p^R_{i,k}) & \text{if } 0 \leq p^R_{i,k} \leq 1 \\ -(1 - p^R_{i,k}) \cdot \log(1 - p^R_{i,k}) & \text{if } p^R_{i,k} = \{0, 1\} \end{cases}$$

Given the three input above, the process of generating a plan can be viewed as consisting of the following four steps:

1. Candidate Set Selection. The objective of this step is to choose a subset of objects from O which will be considered for the selection of a plan. We denote the candidate set of epoch i as Candidate. This step aims at reducing the complexity of plan generation by considering a less number of objects for selection.

2. Generation of Triples. This step generates triples for the objects present in Candidate. For each object o_j ∈ Candidate, it determines a tagging function for o_j for each predicate in the query Q. Assuming that Q consists of n predicates, this step will generate n triples for each object o_j ∈ Candidate.

3. Benefit Estimation of Triples. This step estimates the benefit of each triple present in Triples. As our quality metric is the F_α-measure, we define a metric called the expected F_α-measure, denoted as E(F_α), to estimate the quality of the answer set at each epoch. Based on this metric, we define the Benefit of a triple as the amount of improvement in the E(F_α) measure of the answer set per unit cost that will be caused by processing that triple.

4. Selection of Triples. This step compares between the benefit values of the triples and generates a plan that consists of the triples with the highest benefit values. The triples in the plan will be sorted in a decreasing order according to that value. We denote the plan selected in epoch i as Plan_i.

We will describe these four steps in details in Section 3.2.

3.2 Plan Execution Phase

This phase iterates over the list of triples in the plan generated in the previous phase and, for each triple (o_j, R^R_k, f^R_m), executes it by calling the function f^R_m with these two parameters: the object o_j and the tag t_l present in the predicate R^R_k. Triples are executed until the allotted time for the epoch is consumed.

After the plan is executed, our approach updates the state, predicate probability and uncertainty values for the objects involved in the executed triples of the Plan_i. In Appendix B, we discuss in details how these three values can be efficiently updated.

3.3 Answer Set Selection Phase

At the end of an epoch i, we determine an answer set (Answer_i) from O based on the output of plan execution phase. The strategy to choose Answer_i is based on choosing a subset from O which optimizes the quality of the answer set. As the true ground truth of the objects in O is not known, we use the joint probability values of the objects in O to measure the quality metrics (precision, recall, and F_α-measure) in an expected sense.

Consider the joint probability values of the objects in O at a particular time instant P_{i1}, P_{i2}, ..., P_{i|O|}, P_{i|O|}. Let us consider an answer set A that consists of objects with probability values P_{i1}, P_{i2}, ..., P_{i|A|}. The expected quality of the answer set A is defined as follows:

$$E(Pre) = \frac{\sum_{i=1}^{|A|} P_{i}}{|A|}, E(Rec) = \frac{\sum_{i=1}^{|A|} P_{i}}{\sum_{i=1}^{|O|} P_{i}}, E(F_\alpha) = \frac{(1 + \alpha) \cdot \sum_{i=1}^{|A|} P_{i}}{\alpha \cdot \sum_{i=1}^{|O|} P_{i} + |A|}$$  

(6)
Our strategy of choosing Answer, is based on sorting the objects in O in the descending order of their joint probability values and selecting those objects with joint probability value higher than a particular threshold probability value. We use the notation of \( P_i \) to refer to the threshold of epoch i. The following theorem shows that this will guarantee obtaining the best possible answer set.

**Theorem 1.** Let \( P_1, P_2, P_3, \ldots, P_{|O|} \) be the joint probability values of the objects in O in epoch i such that \( P_1 \geq P_2 \geq P_3 \geq \ldots \geq P_{|O|} \). A subset of objects chosen as Answer, from O, will have maximum expected quality, if \( \text{Answer}, = \{o_i \mid P_i > P^*_i\} \) where \( P^*_i \) is the joint probability value of the object \( o_i \), for which the following condition holds:

- The \( E(F_o) \) measure of the answer set consisting of the first \( \tau + 1 \) objects \( \{o_1, o_2, \ldots, o_{\tau+1}\} \) is less than that of the answer set that consists of only the first \( \tau \) objects \( \{o_1, o_2, \ldots, o_{\tau}\} \).

The proofs of all theorems and lemmas can be found in Appendix A. In the proof of Theorem 1, we show that if we keep including objects in the answer set starting from the object with highest probability \( P_i \), then the \( E(F_o) \) measure of the answer set increases monotonically. Then, we show that if the \( E(F_o) \) measure decreases for the first time due to inclusion of a particular object, then it will decrease monotonically with the inclusion of any further objects.

The actual value of \( P_i \) depends on the joint probability values of the objects in O and it can be computed as follows:

**Lemma 1.** Let \( P_1, P_2, P_3, \ldots, P_{|O|} \) be the joint probability values of the objects in O in epoch i such that \( P_1 \geq P_2 \geq P_3 \geq \ldots \geq P_{|O|} \). The threshold probability \( P^*_i \) is defined as the lowest value \( P_i \) for which the following conditions hold:

- \( P_j > \frac{(P_1 + P_2 + \ldots + P_{j-1})}{(j-1)k} \), and
- \( P_{j+1} < \frac{(P_1 + P_2 + \ldots + P_{j})}{(j+1)k} \), where \( k = \sum_{i=1}^{|O|} P_i \).

Based on this lemma, we can conclude that \( P^*_i \) is dependent on the joint probability value of the objects inside of the answer set (numerator and denominator) as well as those outside of the answer set (denominator). Hence, a change in the joint probability value of any object as a result of executing a plan can change the threshold probability. Hence, the threshold will be computed after each epoch.

## 4. PLAN GENERATION

In this section, we explain in details the four steps of generating an execution plan.

### 4.1 Candidate Set Selection

The objective of this step is to choose a subset of objects Candidate, from O which will be considered for the generation of Plan. Ideally, Candidate, should contain all the objects in O as potential candidates for generating Plan. However, this strategy will result in a large number of triples generated in the next step and comparison between such triples will result in a very high overhead.

Hence, our strategy restricts Candidate, to only objects from outside Answer, i.e., we determine Candidate, to be the set \( \{o_j \mid o_j \notin O-\text{Answer}_{i-1}\} \). In general, choosing an object from the Answer, has a higher chance of improving the quality of the answer set than choosing an object from outside Answer,._1. To illustrate, let us study how the \( E(F_o) \) measure and threshold value \( P^*_i \) will be affected in epoch i as a result of a change in the joint probability value of an object \( o_i \) involved in a triple executed in epoch \( i - 1 \). We will first consider the case where \( o_i \in \text{Answer},_{i-1} \) and then consider the case where \( o_i \notin \text{Answer},_{i-1} \).

**Case 1: Object \( o_i \in \text{Answer},_{i-1} \).** In epoch i, the joint probability value of \( o_i \) can (i) increase, (ii) decrease but still remain higher than the threshold \( P^*_i \), or (iii) decrease and become lower than \( P^*_i \). The effects of these possibilities on the \( E(F_o) \) measure and the new threshold probability (i.e., \( P^*_i \)) are explained in the following lemmas.

**Lemma 2.** If the joint probability value of \( o_i \) increases in epoch i, then the threshold \( P^*_i \) can remain the same as \( P^*_i \) or increase (some objects which were part of \( \text{Answer},_{i-1} \) can move out of \( \text{Answer},_i \)). In both cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will be higher than that of \( \text{Answer},_{i-1} \).

**Lemma 3.** If the joint probability value of \( o_i \) decreases in epoch i but still remains higher than \( P^*_i \), then the threshold \( P^*_i \) can remain the same as \( P^*_i \) or decrease. This implies that the objects which were already part of \( \text{Answer},_{i-1} \) will still remain in \( \text{Answer},_i \), and some new objects might be added to \( \text{Answer},_i \). In both the cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will be lower than that of \( \text{Answer},_{i-1} \).

**Case 2: Object \( o_i \notin \text{Answer},_{i-1} \).** In epoch i, the joint probability value of \( o_i \) can (i) increase and become higher than the threshold \( P^*_i \), (ii) increase but still remain lower than the threshold \( P^*_i \), (iii) decrease. The effects of these possibilities on the \( E(F_o) \) measure and the new threshold probability (i.e., \( P^*_i \)) are explained in the following lemmas.

**Lemma 4.** If the joint probability value of \( o_i \) decreases in epoch i and becomes less than \( P^*_i \), then the threshold \( P^*_i \) can increase or decrease. In both the cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will be lower than that of \( \text{Answer},_{i-1} \).

**Lemma 5.** If the joint probability value of \( o_i \) increases in epoch i and becomes higher than \( P^*_i \), then the threshold \( P^*_i \) can remain the same as \( P^*_i \) or increase (i.e., some objects which were part of \( \text{Answer},_{i-1} \), might move out of \( \text{Answer},_i \)). In both the cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will be higher than that of \( \text{Answer},_{i-1} \).

**Lemma 6.** If the joint probability value of \( o_i \) increases in epoch i but does not become higher than \( P^*_i \), then the threshold \( P^*_i \) will remain the same as \( P^*_i \) or decrease. In both the cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will remain the same as that of \( \text{Answer},_{i-1} \).

**Lemma 7.** If the joint probability value of \( o_i \) decreases in epoch i, then the threshold \( P^*_i \) will remain the same as \( P^*_i \) or increase. In these cases, the \( E(F_o) \) measure of \( \text{Answer},_i \) will be higher than or equal to that of \( \text{Answer},_{i-1} \).

Based on Lemmas 2, 3, and 4, we can conclude that, given an object \( o_i \in \text{Answer},_{i-1} \), the \( E(F_o) \) measure of \( \text{Answer},_i \) will increase w.r.t. \( \text{Answer},_{i-1} \) only if the joint probability of \( o_i \) increases in epoch i. However, according to Lemmas 5, 6, and 4, given an object \( o_i \notin \text{Answer},_{i-1} \), the \( E(F_o) \) measure of \( \text{Answer},_i \) increases or remains the same w.r.t. \( \text{Answer},_{i-1} \) but it never decreases. Hence, our strategy considers Candidate, to be the set \( \{o_j \mid o_j \notin O-\text{Answer},_{i-1}\} \).
4.2 Generation of Triples

The objective of this step is to generate a set of (object, predicate, function) triples, denoted as Triples, from the objects present in Candidate. For each object \(o_k \in \text{Candidate}\), this step identifies which function should be executed next on the object w.r.t. each predicate present in a query \(Q\). Assuming that \(Q\) contains \(n\) predicates, this step will generate \(n\) triples for \(o_k\).

Function Selection. The tagging functions that should be included in the triples are determined using a decision table. The structure of the decision table is shown in Table 3. For each pair of a tag type \(T_i\) and tag \(t_m \in T_i\), this table stores, for each possible state value, a list of \(w\) uncertainty ranges, a list of \(w\) tagging functions and a list of \(w\) \(\Delta\) uncertainty values.

Given an object \(o_k \in \text{Candidate}\), and a predicate \(R_{l,m}^k\), the function that should be executed on \(o_k\) w.r.t. \(R_{l,m}^k\) is determined using the state value \(s_{l,m}^k\) and the uncertainty value \(h_{l,m}^k\) as follows. If \(h_{l,m}^k\) lies in the \(x^k\) uncertainty range (where \(x < w\) of state \(s_{l,m}^k\), then the next function that should be executed on \(o_k\) is the \(x^k\) function in the list of tagging functions. For example, if \(s_{l,m}^k = [0, 0, 0, 1]\) (the first row in Table 3) and \(h_{l,m}^k = 0.92\), then our technique returns \(f_o^l\) (the last function in the list). Such a function is expected to provide the highest reduction in \(h_{l,m}^k\) among the remaining functions that have not been executed on \(o_k\).

The estimated amount of reduction in \(h_{l,m}^k\) is shown in the last column of the table. In our example, applying \(f_o^l\) on \(o_k\) is expected to reduce the uncertainty of \(o_k\) by 0.22 (the last value of the last column). (The \(\Delta\) uncertainty value will be used in estimating the benefit of triples as we will see in the next section.)

Learning the Decision Table. This table is learned off-line using a validation dataset. Given a range of uncertainty values \([a, b]\), we use the state, tag type, tag and uncertainty values of the objects in the validation dataset as meta-features and learn the function that is expected to provide the highest uncertainty reduction among the other remaining functions. (This is similar to the technique used in the META-DES framework [III].) Such a function is stored as the next function in the decision table. Then, for the chosen function, we learn the average amount of uncertainty reduction when this function is executed on those objects. This value is stored as the \(\Delta\) uncertainty value in the decision table.

4.3 Benefit Estimation of Triples

In this step, we derive the benefit of triples present in Triples, using a benefit metric.

Definition 4. Benefit of a triple \((o_k, R_{l,m}^k, f_{i,n}^k)\) denoted as Benefit\((o_k, R_{l,m}^k, f_{i,n}^k)\), is defined as the increase in the \(E(F_a)\) measure of the answer set (that is caused by the evaluation of the triple) per unit cost. Formally, it is calculated as follows:

\[
\text{Benefit}(o_k, R_{l,m}^k, f_{i,n}^k) = \frac{\hat{E}(F_a)(o_k, R_{l,m}^k, f_{i,n}^k) - E(F_a)}{c_{i,n}}
\]

where \(\hat{E}(F_a)(o_k, R_{l,m}^k, f_{i,n}^k)\) is the estimated quality of Answer, if only triple \((o_k, R_{l,m}^k, f_{i,n}^k)\) is executed in epoch \(i\), \(E(F_a)\) is the \(E(F_a)\) measure of Answer\(_{-1}\), and \(c_{i,n}\) is the cost of function \(f_{i,n}^k\). For ease of notation, we will refer to the estimated quality as \(\hat{E}(F_a)^i\).
This implies that if triple \((o_k, R'_m, f'_n)\) is executed in epoch \(i\), then the joint probability of \(o_k\) will either increase from 0.66 to 0.76 or decrease from 0.66 to 0.12.

Based on Lemma \[8\] we only consider the estimated joint probability value that is higher than the current one \(P_k\) (i.e., 0.76 in our example above). This is because this higher joint probability value always increases the \(E(F_n)\) measure whereas the lower one may increase \(E(F_n)\) measure or keep it the same.

### 4.3.2 Efficient Benefit Estimation

Our approach for estimating the benefit values is based on the relationship between the benefit metric of a triple, as defined in Equation \[7\] and the local properties of such a triple (i.e., joint probability of the object, cost of the function, etc.). The objective is to eliminate the step of computing the probability threshold that is needed to estimate the increase in the quality of the answer set.

Let \((o_k, R'_m, f'_n)\) and \((o_l, R'_l, f'_l)\) be two triples present in Triuples, \(i\). Suppose the estimated joint probability value of \(o_k\) (i.e., \(P_k\)) if triple \((o_k, R'_m, f'_n)\) is executed in epoch \(i\) is as follows: \(P_k = (P_k + \Delta P_k)\) and the estimated joint probability value of \(o_l\) if triple \((o_l, R'_l, f'_l)\) is executed in epoch \(i\) is as follows: \(P_l = (P_l + \Delta P_l)\).

Let \(m_1\) be the number of objects that are part of Answer, \(-1\) and will move out of Answer as a result of changing the joint probability of object \(o_k\) from \(P_k\) to \(P_l\). According to Lemma \[8\] since \(o_k \notin \text{Answer}_{-1}\) and \(P_l > P_k\), the threshold probability \(P'_l\) (after changing the joint probability of \(o_k\) from \(P_k\) to \(P_l\)) can remain the same as \(P'_l-1\) or increase. Hence, \(m_1 \geq 0\). Furthermore, let \(m_2\) be the number of objects that are part of Answer, \(-1\) and will move out of Answer as a result of changing the joint probability of object \(o_l\) from \(P_l\) to \(P_l\). Similarly, \(m_2 \geq 0\).

The possible values of \(m_1\) and \(m_2\) are as follows:

\[m_1 = m_2 = 0\]
\[m_1 > m_2, m_1 > 0, \text{and } m_2 > 0\]
\[m_2 > m_1, m_1 > 0, \text{and } m_2 > 0\]
\[m_1 = m_2, m_1 > 0, \text{and } m_2 > 0\]

Given these four scenarios, we can state the following theorem about the benefit values of the triples in Triuples.

**Theorem 2.** The triple \((o_k, R'_m, f'_n)\) will have a higher benefit value than the triple \((o_l, R'_l, f'_l)\) in epoch \(i\) irrespective of the values of \(m_1\) and \(m_2\) if the following condition holds:

\[
\frac{P_l(P_l + \Delta P_l)}{c_l^2} > \frac{P_k(P_k + \Delta P_k)}{c_k^2}
\]  
(9)

We prove the theorem as follows. Given each of the four scenarios of \(m_1\) and \(m_2\), we consider all the possible combinations of the \(P_k, P_q, \Delta P_k, \text{and } \Delta P_q\) values and show that if the condition in Equation \(8\) holds, then the benefit of triple \((o_k, R'_m, f'_n)\) will be higher than that of triple \((o_l, R'_l, f'_l)\).

Based on Theorem \[2\] our approach calculates a relative benefit value for each triple \((o_k, R'_m, f'_n)\) in epoch \(i\) as follows:

\[
\text{Benefit}(o_k, R'_m, f'_n) = \frac{P_k(P_k + \Delta P_k)}{c_k^2}
\]  
(10)

After calculating the relative benefit values for the triples in Triuples, we pass them to the triple selection step.

### 4.4 Selection of Triples

The objective of this step is to choose the list of triples that should be included in Plan, \(s\). We compare between the benefit values of triples in Triuples, and generate a plan that consists of the triples with the highest benefit values. To this end, our approach maintains a priority queue (denoted as \(PQ\)) of the triples in Triuples, that orders them based on their benefit values. Our approach maintains the same priority queue over the different epochs; it is created from scratch in the first epoch and then updated efficiently in the subsequent epochs.

In the plan execution phase, our approach retrieves one triple at a time from \(PQ\) and then executes it. This process continues until the allotted time for the epoch is consumed.

### 5. EXPERIMENTAL EVALUATION

In this section, we empirically evaluate our proposed approach on real datasets. We compare our approach with two baseline algorithms using two different quality metrics.

#### 5.1 Experimental Setup

**Datasets.** We consider two image and one twitter datasets for our experiments.

- **MUCT Dataset** \[23\] consists of 3,755 face images of people from various age groups and ethnicities. This smaller dataset has been selected so that all approaches can be executed to completion. From this dataset, we have used 850 images for each of the training and validation phase for the tagging functions. The remaining 2,055 images were used in our experiments.
- **CMU Multi-PIE Dataset** \[31\] contains more than 750k face images (305 GB) of 337 people recorded over the span...
of five months. These images were captured using different camera angles and illumination conditions while the subjects displayed a range of facial expressions. From this dataset, we have obtained a subset of 500,000 images. We have chosen 5,000 images for each of the training and validation phases. The remaining 490,000 images were used in our experiments.

- Stanford Twitter Sentiment (STS) Corpus [16] consists of 1.6M tweets collected using Twitter API during a period of about 3 months in 2009. This is one of the largest publicly available dataset for sentiment analysis. From this dataset, we have used 50,000 tweets for each of the training and validation of the tagging functions. The remaining objects were used in our experiments.

**Queries.** The general structure of a query in our experiments follows the format described in Section 2. We consider the following tag types in the set of untagged/imprecise predicates: Gender(G) for the MUCT dataset, Gender(G), Age(A), and Expression(Ex) for the Multi-PIE dataset, and Sentiment(S) for the Twitter dataset. For the precise predicates, we consider the following attributes: Timestamp, Person Id, Session Id, Camera Id, Camera Angle, and Illumination. Condition for the Multi-PIE and MUCT datasets and Timestamp, User, and Location of the tweet for the Twitter dataset. Given these predicates, we define the following query template for our experiments:

Q1: < G == Male ∧ Precise; == P1 ∧ · · · ∧ Precise; == Pn >
Q2: < S == Positive ∧ Precise; == P1 ∧ · · · ∧ Precise; == Pn >
Q3: < G == Male ∧ Ex == Smile ∧ Precise; == P1 ∧ · · · ∧ Precise; == Pn >
Q4: < G == Male ∧ A == 30 ∧ Precise; == P1 ∧ · · · ∧ Precise; == Pn >
Q5: < G == Male ∧ Ex == Smile ∧ A == 30 ∧ Precise; == P1 ∧ · · · ∧ Precise; == Pn >

For each query template, we generate 40 queries with different conditions for the precise predicates. Since we evaluate such predicates before the imprecise predicates, this variation on the conditions allows us to control the number of objects that require tagging. We refer to this as selectivity of the specific query. In the MUCT dataset, the chosen selectivity values are: 2.5%, 5%, 7.5%, and 10%. In the Multi-PIE dataset, the selectivity values were varied from 0.05% to 0.2% with an increment of 0.05%. Similarly, in the Twitter dataset, the chosen selectivity values are 0.1%, 0.2%, 0.4%, and 1%.

**Tagging Functions.** For each of the tag types in the previous datasets, we consider the following classifiers as tagging functions: Decision Tree, Gaussian Naive-Bayes, Random Forest, and Support Vector Machine. We have used the python implementation of the algorithms available in the scikit-learn library. We calibrate the probability output of each tagging function using appropriate calibration mechanisms so that they can generate real probability values for the tags. We have used the isotonic regression model to calibrate the Gaussian Naive-Bayes classifier based tagging function and the Platt’s sigmoid model to calibrate the remaining tagging functions.

In the validation phase, we measure the cost and quality of each tagging function by running it on a sample validation dataset. The quality of a tagging function \( f_i^j \) (i.e., \( q_i^j \)) is measured by calculating the area under the ROC curve [5]. The cost of \( f_i^j \) is measured by the average execution time of the tagging function per object in the validation dataset. Furthermore, in this phase, we generate the decision table as explained in Section 4.

**Preprocessing Step.** We execute a feature extraction code (Histogram of Oriented Gradients features from images and related keywords extraction from tweets) on all the objects in the dataset prior to the arrival of the queries. Note that this preprocessing step mimics the work that would be done on streaming data on its arrival in an online setting.

**Initialization Step.** At the beginning of the query execution time, each of the three approaches (i.e., our approach and the two baseline approaches) executes the cheapest tagging function of each of the available tag types on the objects. In addition, the following data structures of the objects are created as part of this step: a hash map for the state predicate, a hash map for the predicate probability values, and a list for the uncertainty values. Note that in the plots presented in the following sections, we have omitted the time of executing such functions and the time of creating such data structures as they are the same for all approaches.

**Quality Metrics.** We have considered two metrics for comparing the various approaches: the \( F_m \) measure of the answer set where \( a \) is set to 1 and the gain of the answer set. We measure gain at a time instant \( t \), \( gain(t) \), as the relative \( F_1 \) measure achieved by the answer set at that time instant compared to the maximum \( F_1 \) measure achieved by the answer set during the entire course of query execution:

\[
gain(t) = \frac{F_1(t) - F_{1_{\min}}}{F_{1_{\max}} - F_{1_{\min}}} \tag{11}\]

where \( F_{1_{\min}} \) and \( F_{1_{\max}} \) represent respectively the minimum and maximum \( F_1 \) measures achieved by the answer set during the entire query execution time. We have measured the \( F_1 \) measure of the answer set using the real ground truth tags available in these datasets.

**Approaches.** We compare our approach with two baseline approaches:

- **Function-Based Approach,** where a tagging function is chosen first and then executed on all the objects. The tagging functions are chosen in the decreasing order of their \( q_i^j/c_i^j \) values. The objects were chosen based on their joint probability values at the beginning of the execution, starting with the object with the highest joint probability value. We will refer to this approach as Baseline1.

- **Object-Based Approach,** where we choose an object \( o_i \) first and then execute all the tagging functions for evaluating a particular predicate. In case of queries with multiple predicates involving different tag types, we execute all the tagging functions of the required tag types (based on the predicates in \( Q \)) on an object first and then choose the next object. The objects were chosen based on their joint probability values at the beginning of the query execution, starting with the object with the highest joint probability value. We will refer to this approach as Baseline2.

Note that Baseline1 and Baseline 2 are static approaches that decide upon the order in which triples are executed at the very beginning of the execution time. In contrast, our approach adaptively changes the execution plan over time based on the progress of the algorithm.
5.2 Trade-off between Quality and Cost

In this experiment, we compare our approach with the two baseline approaches. For our approach, we have set the optimal epoch time for each selectivity value according to the experiments that will be presented in Section 5.3.1.

The results for the MUCT dataset w.r.t. the gain and $F_1$-measure of the answer set are shown in Figure 2 and Figure 3 respectively. The results for the Multi-Pie and Twitter datasets w.r.t. to the gain metric are shown in Figure 4 and Figure respectively.

Our approach outperforms the baseline approaches significantly for the different selectivity values in all three datasets. In our approach, the answer set achieves a very high quality within a very small number of epochs of the start of the query execution. (Note that each epoch is noted by a point in each plot.) In Figure 2 we can see that our progressive approach achieves a very high quality gain (e.g., 0.98) within the first 8, 16, 28, and 36 seconds of the query execution respectively for the different selectivity values 2% , 5%, 7.5%, and 10%. In each epoch, our approach achieves a very high rate of quality improvement. For example, let us consider the third epoch of the query execution in Figure 2(c). In this epoch, our approach improves the quality of the answer set from 0.4 to 0.7 whereas, in Baseline1, it improves from 0.22 to 0.24 and, in Baseline2, it improves from 0.28 to 0.31.

The reason behind the performance gap of our approach and the baseline approaches is that unlike the latter, that make candidate ordering decision at the beginning, our approach is adaptive. In our approach, the candidate selection step performed at the beginning of each epoch directs the execution to candidates that are expected to yield the highest improvement in quality.

5.3 Configuration Parameters of our Approach

In this section, we study the effect of different parameters used in our approach.

5.3.1 Optimal Plan Generation Time

The objective of this experiment is to determine the optimal amount of time that should be allocated to the plan generation phase of our approach, for a given selectivity value. Given a selectivity value, we execute our approach by setting 10 different time values for the plan generation phase. For each of the time values, we measure the progressiveness of different runs using the progressive metric defined in Equation 5. In this equation, we choose one of the most commonly used weight function $W(t)$ for comparing the different runs of our approach.

$$W(t) = \max(1 - \frac{t}{\text{budget}}, 0)$$ (12)

where $\text{budget}$ is the amount of time required for the approach to complete.

The results are shown in Figure 6. For the lower selectivity value (Figure 6(a)), the optimal amount of time for the plan generation phase is 4% of the budget whereas for the higher selectivity value (Figure 6(b)) it is 9% of the budget. For the lower selectivity value, the significance of the overhead of the plan generation phase becomes more prominent compared to the case of the large selectivity value. Ideally spending more time in the plan generation phase should be better as performing plan generation phase more frequently will let us choose the best set of triples based on the actual output of the tagging functions without depending on any
kind of estimations. However, spending more time in generating the plan increases the overhead. Due to the lower selectivity value of the objects, the total amount of time required to execute more tagging functions on the objects is not as high, compared to the time spent in the plan generation phase, which is the reason why the graphs shows that a lower plan generation time performs better for the lower selectivity value.

5.3.2 Candidate Selection Strategy

In this experiment, we evaluate our optimization strategy for candidate selection, explained in Section 4.1. By default, all the objects in $O$ should be considered as potential candidates for the generation of a plan but this will be computationally expensive. Our optimized approach considered only objects that are outside of the answer set at the end of previous epoch.

In the experiment, as expected, our optimized strategy uses less amount of time than the default strategy in the plan generation phase (on an average 30% less). The reason is that in our approach, the number of objects for which we need to determine benefit values is lower as well as the comparison time between the generated triples from them is lower, due to less number of generated triples. Furthermore, if we compare these two approaches based on their quality variation with respect to time (as shown in Figure 6), we can see that our approach outperforms the default strategy too. This result empirically proves the validity of our candidate selection strategy (as explained in Section 4.1) which states that an object outside of answer set have higher chance of improving $F_0$ measure of the answer set compared to an object present inside of the answer set.

5.3.3 Benefit Estimation Strategy

In this experiment we evaluate our optimization strategy of estimating benefit of the triples present in $\text{triples}$, as explained in Section 4.3. We derive the benefit of the triples according to Equation $10$ using the joint probability value of the objects and cost of the tagging functions present in the triples. This way, we avoid performing the threshold selection algorithm for estimating benefit of each triples, and reduce the cost of overall plan generation phase.

By default, for each triple present in $\text{triples}$, benefit should be calculated based on threshold selection algorithm. Specifically, for each triple, we first determine the estimated joint probability value of the object in epoch $i$ and use it to determine the answer set (i.e., perform threshold selection algorithm). Then, we calculate new $E(F_i)$ measure of the chosen answer set and determine the improvement of $E(F_i)$ measure of the answer set in epoch $i$. Figure 8 shows the variation of quality and gain of the answer set with respect to time which shows that our optimized approach outperforms the default strategy significantly.

The reason behind this performance gap is that the default strategy spends a higher amount of time in plan generation phase. For example, for selectivity value of 10% in MUCT dataset, the amount of time spent in plan generation phase of the default strategy is 14.87 seconds whereas as in our optimized approach it is 2.5 seconds. This is due to the execution of threshold selection algorithm for each triples present in $\text{triples}$, which makes this strategy much slower as compared to our approach. This experiment shows the efficiency of our benefit estimation strategy in terms of cost of plan generation phase. Furthermore, it empirically validates our strategy of ordering the triples based on the
In this section, we study the impact of different levels of caching on the benefit metric as shown in Theorem 2.

5.4 Scalability of our Approach

Our approach outperforms both the baseline approaches significantly for all three queries and selectivity values since our approach chooses the best triples at the beginning of each epoch. This implies that our approach can scale well with increasing number of predicates containing different tag types. The result also shows that, as expected, the cost of plan generation phase increases with the increase in number of predicates in these scenarios. However, even after paying high overhead in the plan generation phase, the benefit of choosing right set of triples for execution, overshadows the cost of plan generation phase.

5.5 Caching Previous Query Results

In this experiment, we evaluate the scalability of our approach with the help of multiple predicates containing different tag types. For each of the tag types, we have trained four tagging functions. The results of query Q3 containing two predicates of two tag types with different selectivity values are shown in Figure 9. The results of queries Q4 and Q5, containing two and three predicates respectively, are shown in Figure 11.

Our approach outperforms both the baseline approaches significantly for all three queries and selectivity values since our approach chooses the best triples at the beginning of each epoch. This implies that our approach can scale well with increasing number of predicates containing different tag types. The result also shows that, as expected, the cost of plan generation phase increases with the increase in number of predicates in these scenarios. However, even after paying high overhead in the plan generation phase, the benefit of choosing right set of triples for execution, overshadows the cost of plan generation phase.

6. RELATED WORKS

This paper develops a progressive approach for answering queries that require enrichment of the right amount of data and at the right level of enrichment. We are not aware of any work in the literature that directly addresses this problem. However, there exists related works for some of the components of the problem. In our setup, we assume that a set of expensive tagging functions are available for determining a particular tag and they have to be executed appropriately on the objects for answering a query containing such tags. In this regard, the most relevant works in the databases domain are in the context of query optimization with expensive predicates. Furthermore, enrichment of data efficiently using complex signal processing operators have been studied previously in the context of signal oriented data management systems. In machine learning domain, the problem of determining the optimal set of tagging functions for an object has been addressed in the context of dynamic ensemble selection and ensemble pruning.

Expensive Predicate Optimization. In the expensive predicate optimization problem the goal is to optimize a query containing expensive predicates by appropriately ordering the evaluation of such predicates [8, 9, 17]. A subset of this problem, more relevant to our context, deals with the optimization of multi-version predicates [19]. In these Works, the goal is to optimize the predicate itself with the help of a less computationally costly version of the predicate. Such version of the predicate is used to filter out some objects in the beginning, so that the costly version of the predicate is evaluated only on a small number of objects. In contrast with our context, in these works the predicates are considered to be deterministic.

Adaptive Query Processing. In adaptive query processing the goal is to optimize a query at the execution time with the help of interleaving query execution with exploration of the plan or scheduling space [15, 21, 22]. Broadly, adaptive query processing addresses the problems of how to perform query optimization with missing statistics, unexpected correlations in data, unpredictable cost of operators, etc. Re-optimization of a query during run time is similar in spirit to our approach, where we decide the best set of triples to execute at the beginning of each epoch and execute them in the execution phase. However, these works do not address the issue of evaluating predicates using different non-deterministic functions which vary in cost and quality values.

Signal oriented Data Management Systems. The problem of joint enrichment of data using complex signal processing functions have been studied in the context of signal oriented data stream management systems [4, 13, 24]. The objective is to provide users/developers with a common data processing platform which can be used to specify complex signal processing functions and arbitrary event streams. These approaches are based on the processing of data efficiently at arrival time using complex functions. However, our approach is based on enrichment of data performed in the context of a query by using different data processing techniques.
We have shown empirically that our approach executes the maximization of the quality of the answer at the end of the epoch.

The goal is to select the best set of classifiers for an object to maximize the cardinality of the functional dependencies of that object. The objective of dynamic ensemble selection is to select the most competent classifiers for a given object from a pool of classifiers. Some of the most popular algorithms are META-DES, META-DES oracle, and K-Nearest Output Profiles.

### Dynamic Ensemble Selection

The objective of dynamic ensemble selection approaches is to select the most competent classifiers for a given object from a pool of classifiers. Some of the most popular algorithms are META-DES, META-DES oracle, and K-Nearest Output Profiles.

Another set of works that look into the problem of dynamic selection of classifiers are in the domain of ensemble pruning. The objective of ensemble pruning approaches is to select a subset of classifiers from a large set of classifiers to generate an ensemble which is similar to, or better, than the original ensemble of all the classifiers. Hence, in both dynamic ensemble selection and ensemble pruning, the objective is to select the best set of classifiers for an object to maximize the quality of the answer set.

### 7. CONCLUSIONS AND FUTURE WORK

In this paper, we have developed a progressive approach that enriches the right amount of data to the right degree so as to maximize the quality of the answer set. The goal is to use different tagging techniques, which vary in cost and quality, in such a way that improves the quality of the answer progressively. We have proposed an efficient approach that generates a plan in every epoch with the objective of maximizing the quality of the answer at the end of the epoch.

We have shown empirically that our approach executes the right set of (object, predicate, function) triples in each epoch so that the quality of the answer set improves progressively with respect to time.

This research opens several interesting research challenges for future investigations.

### Complex Queries

Although SPJ queries (with a restricted version of join predicates, as studied in this paper) are an important class of queries on their own, developing a progressive technique in the context of more complex types of queries (e.g., nested sub-queries involving imprecise attributes, top-k queries, aggregation queries) is an interesting direction for future work. As an example, consider a join query where the join condition involves an imprecise attribute in each operand. If we assume that the predicate probability values of two different objects containing a particular tag are independent of each other, then the join technique used in probabilistic databases could be leveraged to address this issue.

### Disk-based Solution

Our approach dealt with scenarios where the queries involve precise predicates that can be used to limit the number of objects on which the expensive tagging functions need to be executed. In some situations (e.g., if the query contains only imprecise predicates), the number of objects for which the functions need to be run can be quite large, requiring a disk-based solution for evaluating the queries. Such a solution needs to tackle several challenges related to how the objects can be grouped together in blocks at the beginning of the query execution, how to decide which blocks should be loaded into memory and which triples should be executed in each epoch, and so on. We plan to leverage our experience in 2 to develop such a disk-based solution.

### ACKNOWLEDGMENT

This material is based on research sponsored by DARPA under agreement number FA8750-16-2-0021. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

### 8. REFERENCES

[1] Dynamic classifier selection: Recent advances and perspectives. Information Fusion, 41:195 – 216, 2018.

[2] Y. Altowim, D. V. Kalashnikov, and S. Mehrotra. Progressive approach to relational entity resolution. Proc. VLDB Endow., 7(11):999–1010, 2014.
[3] H. Altwaijry, S. Mehrotra, and D. V. Kalashnikov. Query: A framework for integrating entity resolution with query processing. *Proc. VLDB Endow.*, 9(3):120–131, Nov. 2015.

[4] R. Bordawekar, B. Blainey, and C. Apte. Analyzing analytics. *SIGMOD Rec.*, 42(4):17–28, 2014.

[5] A. P. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145 – 1159, 1997.

[6] P. R. Cavalin, R. Sabourin, and C. Y. Suen. Dynamic selection approaches for multiple classifier systems. *Neural Computing and Applications*, 22(3):673–688, 2013.

[7] K. Chakrabarti, M. Garofalakis, R. Rastogi, and P. R. Cavalin, R. Sabourin, and C. Y. Suen. Dynamic migration: Optimizing queries with expensive predicates. In *Proc. VLDB Endow.*, 4(2):177–188, 2011.

[8] R. M. Cruz, R. Sabourin, and G. D. Cavalcanti. Meta-des.oracle: Meta-learning and feature selection for dynamic ensemble selection. *Information Fusion*, 38:84 – 103, 2017.

[9] R. M. Cruz, R. Sabourin, G. D. Cavalcanti, and T. I. Ren. Meta-des: A dynamic ensemble selection framework using meta-learning. *Pattern Recognition*, 48(5):1925 – 1935, 2015.

[10] N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *The VLDB Journal*, 16(4):523–544, Oct. 2007.

[11] P. Domingos. Using partitioning to speed up specific-to-general rule induction. In *AAAI Workshop on Integrating Multiple Learned Models*, pages 29–34, 1996.

[12] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. Optimal and approximate computation of summary statistics for range aggregates. In *20th Symposium on Principles of Database Systems, PODS ’01*, pages 227–236, 2001.

[13] L. Girod, Y. Mei, R. Newton, S. Rost, A. Thiagarajan, H. Balakrishnan, and S. Madden. The case for a signal-oriented data stream management system. In *CIDR*, 2007.

[14] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision, 2009.

[15] J. M. Hellerstein and M. Stonebraker. Predicate migration: Optimizing queries with expensive predicates. *SIGMOD Rec.*, 22(2):267–276, 1993.

[16] K. Karanasos, A. Balmin, M. Kutsch, F. Ozcan, V. Ercegovac, C. Xia, and J. Jackson. Dynamically optimizing queries over large scale data platforms. In *Int. Conf. on Management of Data, SIGMOD ’14*, pages 943–954, 2014.

[17] I. Lazaridis and S. Mehrotra. Optimization of multi-version expensive predicates. In *Int. Conf. on Management of Data, SIGMOD ’07*, pages 797–808, 2007.

[18] Z. Lu, X. Wu, X. Zhu, and J. Bongard. Ensemble pruning via individual contribution ordering. In *16th Int. Conf. on Knowledge Discovery and Data Mining, KDD ’10*, pages 871–880, 2010.

[19] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic. Robust query processing through progressive optimization. In *Int. Conf. on Management of Data, SIGMOD ’04*, pages 659–670, 2004.

[20] D. Marmaros, S. E. Whang, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge & Data Engineering*, 25:1111–1124, 2013.

[21] S. Milborrow, J. Morkel, and F. Nicolls. The muct landmarked face database. *Pattern Recognition Association of South Africa*, 2010.

[22] M. Nikolic, B. Chandramouli, and J. Goldstein. Enabling signal processing over data streams. In *Int. Conf. on Management of Data, SIGMOD ’17*, pages 95–108, 2017.

[23] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study.

[24] T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. *IEEE Trans. on Knowledge and Data Engineering, 27(5):1316–1329, 2015.

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

[26] J. R. Quinlan. Bagging, boosting, and c4.5. In *13th Nat. Conf. on Artificial Intelligence - Volume 1*, AAAI’96, pages 725–730, 1996.

[27] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[28] J. R. Quinlan. Bagging, boosting, and c4.5. In *13th Nat. Conf. on Artificial Intelligence - Volume 1*, AAAI’96, pages 725–730, 1996.

[29] V. Raman and J. M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In *27th Int. Conf. on Very Large Data Bases, VLDB ’01*, pages 381–390, 2001.

[30] Z. Lu, X. Wu, X. Zhu, and J. Bongard. Ensemble pruning via individual contribution ordering. In *16th Int. Conf. on Knowledge Discovery and Data Mining, KDD ’10*, pages 871–880, 2010.

[31] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic. Robust query processing through progressive optimization. In *Int. Conf. on Management of Data, SIGMOD ’04*, pages 659–670, 2004.

[32] D. Marmaros, S. E. Whang, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge & Data Engineering*, 25:1111–1124, 2013.

[33] S. Milborrow, J. Morkel, and F. Nicolls. The muct landmarked face database. *Pattern Recognition Association of South Africa*, 2010.

[34] M. Nikolic, B. Chandramouli, and J. Goldstein. Enabling signal processing over data streams. In *Int. Conf. on Management of Data, SIGMOD ’17*, pages 95–108, 2017.

[35] D. Opitz and R. Maclin. Popular ensemble methods: An empirical study.

[36] T. Papenbrock, A. Heise, and F. Naumann. Progressive duplicate detection. *IEEE Trans. on Knowledge and Data Engineering, 27(5):1316–1329, 2015.

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

[38] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

[39] J. R. Quinlan. Bagging, boosting, and c4.5. In *13th Nat. Conf. on Artificial Intelligence - Volume 1*, AAAI’96, pages 725–730, 1996.

[40] V. Raman and J. M. Hellerstein. Potter’s wheel: An interactive data cleaning system. In *27th Int. Conf. on Very Large Data Bases, VLDB ’01*, pages 381–390, 2001.

[41] Z. Lu, X. Wu, X. Zhu, and J. Bongard. Ensemble pruning via individual contribution ordering. In *16th Int. Conf. on Knowledge Discovery and Data Mining, KDD ’10*, pages 871–880, 2010.

[42] V. Markl, V. Raman, D. Simmen, G. Lohman, H. Pirahesh, and M. Cilimdzic. Robust query processing through progressive optimization. In *Int. Conf. on Management of Data, SIGMOD ’04*, pages 659–670, 2004.

[43] D. Marmaros, S. E. Whang, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge & Data Engineering*, 25:1111–1124, 2013.

[44] S. Milborrow, J. Morkel, and F. Nicolls. The muct landmarked face database. *Pattern Recognition Association of South Africa*, 2010.
APPENDIX

A. PROOFS

A.1 Proof of Theorem 1

Proof. If we keep including objects starting with the highest joint probability value of \( P, \) then \( E(F_a) \) measure of the answer set increases monotonically up to a certain point and then it starts to decrease monotonically. We prove this theorem by showing that, if \( E(F_a) \) measure of the answer set decreases for the first time, due to the inclusion of a particular object, then it will keep decreasing monotonically with the inclusion of any further objects. Let \( \alpha_r \) be the object beyond which the \( E(F_a) \) measure of the answer set decreases.

Let us denote the \( E(F_a) \) measure of the answer set, if we include \( \alpha_r \) in the answer set as \( F_r. \) Similarly, the \( E(F_a) \) measures corresponding to the objects \( \alpha_{r+1} \) and \( \alpha_{r+2} \) are denoted as \( F_{r+1} \) and \( F_{r+2} \) respectively. Now we show that, if \( E(F_a) \) of the answer set decreases due to the object \( \alpha_{r+1} \), then it will monotonically decrease for any further objects (i.e., \( \alpha_{r+2}, \alpha_{r+3}, \ldots, \alpha_{r+n} \)). Specifically, we show that if \( F_{r+1} < F_r \), then it implies that \( F_{r+2} < F_{r+1} \).

\[
F_r = \frac{(1 + \alpha) \cdot k_1}{\alpha \cdot \frac{k_1 + k_2}{k_2}} = \frac{(1 + \alpha) \cdot k_1}{\alpha \cdot k_2 + \tau} \tag{13}
\]

where \( k_1 = \sum_{i=1}^{P} P_i, k_2 = \sum_{i=1}^{(\alpha)} P_i \)

\[
F_{r+1} = \frac{(1 + \alpha) \cdot \frac{k_1 + P_{r+1} + \tau}{\tau + 1 + k} \cdot \frac{k_1 + P_{r+1}}{2}}{\alpha \cdot \frac{k_1 + P_{r+1} + \tau}{\tau + 1 + k} \cdot \frac{k_1 + P_{r+1}}{2}} \tag{14}
\]

\[
F_{r+2} = \frac{(1 + \alpha) \cdot (k_1 + P_{r+1} + P_{r+2})}{(\alpha \cdot k_2 + \tau + 2)} \tag{15}
\]

\[
F_{r+1} < F_r \Rightarrow \frac{(1 + \alpha) \cdot (k_1 + P_{r+1})}{(\alpha \cdot k_2 + \tau + 1)} < \frac{(1 + \alpha) \cdot k_1}{\alpha \cdot k_2 + \tau} \Rightarrow (k_1 + P_{r+1} + \tau) < k_1(\alpha k_2 + \tau + 1) \Rightarrow ak_1k_2 + k_1\tau + \alpha k_2 P_{r+1} + \tau P_{r+1} < ak_1k_2 + k_1\tau + k_1 \Rightarrow P_{r+1}(ak_2 + \tau) < k_1 \Rightarrow P_{r+1}(\tau + ak_2 + 1) < k_1 + P_{r+1} \Rightarrow P_{r+2}(\tau + ak_2 + 1) < P_{r+1}(\tau + k_2 + 1) < k_1 + P_{r+1} \Rightarrow P_{r+2}(\tau + ak_2 + 1) < k_1 + P_{r+1} \Rightarrow (k_1 + P_{r+1})(\tau + ak_2 + 1) + P_{r+2}(\tau + k_2 + 1) < (k_1 + P_{r+1})(\tau + k_2 + 1) + (k_1 + P_{r+1}) \Rightarrow (\tau + ak_2 + 1)(k_1 + P_{r+1} + P_{r+2}) < (k_1 + P_{r+1})(\tau + ak_2 + 2) \Rightarrow \frac{(k_1 + P_{r+1} + P_{r+2})}{(ak_2 + \tau + 2)} < \frac{(k_1 + P_{r+1})}{ak_2 + \tau + 1} \Rightarrow F_{r+2} < F_{r+1} \tag{16}
\]

A.2 Proof of Lemma 1

Proof. As shown in theorem 1, \( E(F_a) \) measure of the answer set will monotonically increase up to a certain object \( \alpha_r \) with joint probability value of \( P \) and then it will keep decreasing monotonically. Let us denote, \( F_r \) as the \( E(F_a) \) measure of the answer set if we include \( \tau - th \) object and \( F_{r+1} \) be the \( E(F_a) \) measure, if we include \( (\tau + 1) - th \) object in the answer set.

\[
\Delta(E(F_a))^{r+1} = \frac{(1 + \alpha)(P_1 + P_2 + \ldots P_{r+1})}{\tau + 1 + k} - \frac{(1 + \alpha)(P_1 + P_2 + \ldots P_r)}{\tau + k}
\]

where \( k = \sum_{i=1}^{P} P_i \)

\[
= \frac{(\tau + k)(P_1 + P_2 + \ldots P_{r+1})}{\tau + 1 + k} - \frac{(\tau + k)(P_1 + P_2 + \ldots P_r)}{\tau + k}
\]

\[
= (1 + \alpha) \cdot \left( \frac{P_{r+1}(\tau + k) - (P_1 + P_2 + \ldots P_r)}{(\tau + 1 + k)(\tau + k)} \right) \tag{17}
\]

\( E(F_a) \) measure will keep increasing as long as the value of \( \Delta(E(F_a))^{r+1} \) remains positive. Thus the threshold value will be the lowest value of \( \tau \) for which \( \Delta(E(F_a))^{r+1} \) value becomes negative.

\[
\Delta(E(F_a))^{r+1} < 0 \Rightarrow P_{r+1}(\tau + k) < (P_1 + P_2 + \ldots P_r) < 0 \Rightarrow P_{r+1}(\tau + k) < (P_1 + P_2 + \ldots P_r) \Rightarrow P_{r+1} < \frac{(P_1 + P_2 + \ldots P_r)}{(\tau + k)} \tag{18}
\]

The object \( \alpha_r \) with the joint probability value of \( P \) will be the threshold probability if the following two inequalities hold:

\[
P_r > \frac{(P_1 + P_2 + \ldots P_{r-1})}{(\tau - 1 + k)} \tag{19}
\]

\[
P_{r+1} < \frac{(P_1 + P_2 + \ldots P_r)}{(\tau + k)} \tag{20}
\]

A.3 Proof of Lemma 2

Proof. Let us consider that \( P_r \) (the joint probability value of the object \( \alpha_r \)) is the threshold probability of epoch \( i - 1 \). Then the value of \( P_r \) satisfies the following condition according to Equation 19:

\[
P_r > \frac{(P_1 + P_2 + \ldots P_{r-1})}{(\tau - 1 + k)} \Rightarrow P_r(\tau - 1 + k) > (P_1 + P_2 + \ldots P_{r-1}) \tag{21}
\]
Let us consider that $P_r$ is also the threshold probability in epoch $i$, after considering the estimated joint probability value of the object $o_j$ in epoch $i$. Then the following condition must hold:

$$P_r > \frac{(P_1 + P_2 + \ldots + P_{r-1})}{(\tau - 1 + k + \Delta)}$$

where $k = \sum_{i=1}^{O} P_i$

From equation (21) we can derive that $P_r(\tau - 1 + k + \Delta) > (P_1 + P_2 + \ldots + P_{r-1})$

This implies that the threshold probability value of epoch $i - 1$ will remain the same as epoch $i$ if the $\Delta$ value follows the Equation 23 otherwise it will increase. In both the cases, when threshold remains the same and when the threshold gets increased, the $E(F_a)$ measure of the answer set will increase. The reason behind this result is that in Equation 14 the numerator will be increased in both the cases resulting in increment of the $E(F_a)$ measure of the answer set. □

### A.4 Proof of Lemma 4

**Proof.** Let us consider that the joint probability of object $o_j \in Answer_{i-1}$ gets reduced from $P_j$ to $P_j - \Delta$ and the threshold probability of epoch $i - 1$ is the joint probability value of object $o_j$. According to Equation 14 the threshold probability of epoch $i - 1$ must satisfy the following condition:

$$P_r > \frac{(P_1 + P_2 + \ldots P_{r-1} - \Delta)}{\tau - 1 + k - \Delta}$$

Let us consider the right hand side of the above Equation (i.e., $\frac{(P_1 + P_2 + \ldots P_{r-1} - \Delta)}{\tau - 1 + k - \Delta}$) in epoch $i$. In epoch $i$, considering the new joint probability value of $o_j$, the value in the right hand side will be reduced from the previous epoch $i - 1$ (i.e., $\frac{(P_1 + P_2 + \ldots P_{r-1})}{\tau - 1 + k}$). Hence the above equation will always hold for object $o_j$ and with any values of $\Delta$. Furthermore if Equation 24 also holds for object $o_r$ then the threshold $P_r$ will remain the same as $P_{r-1}$. In both the cases, the $E(F_a)$ measure of the answer set will be reduced in epoch $i$ as the numerator in Equation 14 will decrease in both the cases. □

### A.5 Proof of Lemma 5

**Proof.** Let us consider that the joint probability of the object $o_j$ is reduced from $P_j$ to $P_j - \Delta$. The new estimated joint probability value $(P_j - \Delta)$ is lower than the threshold probability $P_{r-1}$. The value of $P_{r-1}$ was greater than $(P_1 + P_2 + \ldots + P_{r-1})$ according to Equation 14. Let us denote the numerator by $X$ and the denominator by $Y$.

In epoch $i$, the joint probability value of object $o_j$ became lower than $P_{r-1}$. This implies that the new numerator of Equation 19 for object $o_r$ will be $X - P_j$ and the denominator will be $Y - 1 - \Delta$. The new value of the right hand side of Equation 19 becomes $\frac{X - P_j}{Y - 1 - \Delta} - \Delta$. If the value $\frac{X - P_j}{Y - 1 - \Delta}$ remains lower than $\frac{X}{Y}$ then the threshold $P_r$ will remain the same as $P_{r-1}$, otherwise it will decrease. The $E(F_a)$ measure of the answer set will reduce in both the cases, as the numerator of Equation 14 will be decreased in both the cases.

□

### A.6 Proof of Lemma 5

**Proof.** Let $P_{r-1}$ be the threshold probability of epoch $i - 1$. This implies that this value is greater than the value of $\frac{(P_1 + P_2 + \ldots + P_{r-1})}{\tau - 1 + k}$, according to Equation 19. Let us denote the fraction by $\frac{X}{Y}$. The right hand side of the Equation 19 in epoch $i$ will be updated to $\frac{X - P_j}{Y - 1 - \Delta}$. If the new value is lower than the previous value of $\frac{X}{Y}$, then the threshold probability $P_r$ will remain the same as $P_{r-1}$, otherwise it will increase.

If the threshold probability value in epoch $i$ remained the same as the threshold probability of epoch $i - 1$, then from Equation 14 we can derive that the $E(F_a)$ measure will be increased in epoch $i$. In the following we consider the case, where the threshold in epoch $i$ is increased compared to epoch $i - 1$.

Let $o_j$ be the object which was part of $Answer_{i-1}$ but moved out of the answer set in epoch $i$ due to the inclusion of object $o_j$. This implies that in epoch $i - 1$, the answer set consisted of the objects with probability values of $(P_1, P_2, \ldots, P_{r-1}, P_r)$ and in epoch $i$ the answer set consists of objects with joint probability values of $(P_1, P_2, \ldots, (P_j + \Delta), \ldots, P_{r-1})$ respectively. The $E(F_a)$ measure of epoch $i - 1$ is as follows:

$$\frac{(1 + \alpha)(P_1 + P_2 + \ldots + P_{r-1})}{\alpha(P_1 + P_2 + \ldots + P_{r-1}) + 1}$$

(23)

Using the previous notations, we can write it as follows:

$$\frac{X}{Y} = \frac{P_1 + P_2 + \ldots + P_{r-1}}{\alpha(P_1 + P_2 + \ldots + P_{r-1}) + 1}$$

As the threshold probability of epoch $i - 1$ was $P_r$, then according to Equation 19 we can derive the following conditions:

$$P_r > \frac{(P_1 + P_2 + \ldots P_{r-1})}{\tau - 1 + (P_1 + P_2 + \ldots P_{r-1})}$$

$$\Rightarrow P_r > \frac{(X - P_j)}{Y - 1}, \Rightarrow P_r > \frac{X}{Y}$$

Furthermore from Equation 20 we can derive the following condition:

$$P_r < \frac{(P_1 + P_2 + \ldots P_{r-1} + P_j + \Delta)}{\tau - 1 + (P_1 + P_2 + \ldots P_{r-1} + P_j + \Delta)}$$

$$\Rightarrow P_r < \frac{(X - P_r + (P_j + \Delta))}{Y - 1 + \Delta}, \Rightarrow P_r < \frac{X + P_j + \Delta}{Y + \Delta}$$

(26)

Using Equations 23 and 26 and simplifying the expressions of $E(F_a)$ measures of epoch $i$ and $i - 1$, we can show that $E(F_a)$ measure of the answer set in epoch $i$ is higher than the $E(F_a)$ measure of the answer set in epoch $i - 1$. □
A.7 Proof of Lemma 6

Proof. Let us consider that $P_{r+1}$ is the threshold probability of the answer set in epoch $i - 1$. This implies that $P_{r+1}$ is higher than $(p_1 + p_2 + ... + p_{r-1})/(r+1)$ (according to Equation 19). We denote this fraction by $\frac{X}{Y}$. If we consider an object $o_j \notin Answer_{i-1}$, the estimated joint probability value of $o_j$ in epoch $i$ is increased from $P_j$ to $P_j + \Delta$. In the new right hand side of Equation 19, the numerator stays same as $X$ (as no extra object was added in the answer set) whereas denominator gets increased from $Y$ to $Y + \Delta$. So the new right hand side $\frac{X}{Y}$ becomes smaller than the previous right hand side of $\frac{X}{Y}$. This implies that object $o_j$ will remain in Answer.

Let us consider Equation 20. In epoch $i - 1$ the following condition is true for object $o_{r+1}$: $P_{r+1} = \frac{(P_1 + P_2 + ... + P_r)}{(r + k)}$, where $k = \sum_{i=1}^{O} P_i$. Let us consider the numerator and denominator as $X$ and $Y$ respectively. In epoch $i$, the denominator $Y$ will be increased as the joint probability of object $P_k$ is increased from $P_k$ to $P_k + \Delta$. This implies that the right hand side of $\frac{X}{Y}$ becomes lower than the previous value of $\frac{X}{Y}$. We can conclude that $o_{r+1}$ can become part of the answer set if the value of $P_{r+1}$ becomes higher than the value of $\frac{X}{Y}$.

Using Equation 21, we can show that $E(F_a)$ measure of epoch $i$ will remain same as the $E(F_a)$ measure of epoch $i - 1$, in both the above scenarios.

A.8 Proof of Lemma 7

Proof. Let $P_r$ be the threshold probability of the answer set in epoch $i - 1$. According to Equation 19, the joint probability value $P_r$ (of object $o_r$) is higher than the value of $(P_1 + P_2 + ... + P_{r-1})/((r-1)+k)$. We denote the value of $\frac{P_1 + P_2 + ... + P_{r-1}}{((r-1)+k)}$ as $\frac{X}{Y}$ in the remaining part of the proof.

Let us consider an object $o_j \notin Answer_{i-1}$. The estimated joint probability value of $o_j$ is decreased from $P_j$ to $P_j - \Delta$ in epoch $i$. In the new right hand side of Equation 19, the numerator stays same as epoch $i - 1$ (as no additional object was added to the answer set). The denominator gets reduced from $Y$ to $Y - \Delta$. So the new right hand side $\frac{X}{Y - \Delta}$ increases from the previous right hand side of $\frac{X}{Y}$. This implies that object $o_j$ will still remain the threshold, if $P_r > \frac{X}{Y - \Delta}$, otherwise the threshold value will be increased.

In the above scenario, where the threshold probability remains the same as epoch $i - 1$, the $E(F_a)$ measure of the answer set will remain same as epoch $i - 1$ (as the numerator of Equation 13 will remain the same as epoch $i - 1$). In the other scenario, where the threshold is increased from epoch $i - 1$, the $E(F_a)$ measure of the answer set will increase (using the Equations 22-23) and simplifying the expression of $E(F_a)$ measure as shown in Equation 14.

A.9 Proof of Theorem 2

Proof. Let us consider two triples $(o_k, R_k^l, f_k^l)$ and $(o_l, R_l^l, f_l^l)$ which were not present in Answer_{i-1}. Let us denote the threshold probability of epoch $i - 1$ as $P_r$. The $E(F_a)$ measure of Answer_{i-1} can be expressed as follows: $\frac{\sum_{O \notin Answer_{i-1}} E(o_j)}{(1 + \alpha) \cdot (P_1 + P_2 + ... + P_{O})}$. We denote the numerator by $X$ and the value of $\alpha(P_1 + P_2 + ... + P_{O})$ in the denominator as $Y$. Let

us denote the value of $\frac{X}{Y}$ as $\nu_k$ and the value of $\frac{Y}{X}$ as $\nu_l$. As $m_1$ is the number of objects that are part of Answer_{i-1} and will move out of Answer, as a result of changing the joint probability value of $o_k$ from $P_k$ to $P_k + \Delta$, the $E(F_a)$ measure of the answer set will be as follows:

$$F_{ak} = \frac{X - (1 + \alpha) \cdot (P_r + P_{r-1} + ... + P_{r-(m_1-1)}) + (1 + \alpha) \cdot (P_k + \Delta P_k)}{Y + (1 + \alpha) \cdot (P_{k-1} + \Delta P_k)}$$

Similarly for object $o_l$, $m_2$ is the number of objects that are part of Answer_{i-1} and will move out of Answer, as a result of changing the joint probability value of $o_l$ from $P_l$ to $P_l + \Delta$. The $E(F_a)$ measure of the answer set in this scenario will be as follows:

$$F_{al} = \frac{X - (1 + \alpha) \cdot (P_r + P_{r-1} + ... + P_{r-(m_1-1)}) + (1 + \alpha) \cdot (P_k + \Delta P_k)}{Y + (1 + \alpha) \cdot (P_{k-1} + \Delta P_k)}$$

For comparing the benefit values of the two triples $(o_k, R_k^l, f_k^l)$ and $(o_l, R_l^l, f_l^l)$, we have to solve the following Equation: $\nu_k F_{ak} > \nu_l F_{al}$.

$$X - (1 + \alpha) \cdot (P_r + P_{r-1} + ... + P_{r-(m_1-1)}) + (1 + \alpha) \cdot (P_k + \Delta P_k) > \nu_k \frac{X}{Y + (1 + \alpha) \cdot (P_{k-1} + \Delta P_k)}$$

We consider the different cases of $\Delta P_k$, $\Delta P_q$, $(P_k + \Delta P_k)$, and $(P_q + \Delta P_q)$ values and establish the conditions in which Equation 24 will be satisfied.

Case 1: $\Delta P_k < \Delta P_q$, $P_k + \Delta P_k > P_q + \Delta P_q$ and $m_1 = m_2 = 0$.

Let us consider the denominators and numerators of the Equation 20 in both the sides. The modified version of the equation considering $m_1 = m_2 = 0$ will be as follows:

$$\frac{X}{Y + (1 + \alpha) \cdot (P_k + \Delta P_k)} > (1 + \alpha) \cdot (P_k + \Delta P_k)$$

In the above equation, we can see that the denominator on the left hand side is smaller than the one on the right hand side in this case (as $\Delta P_k < \Delta P_q$ according to the assumption). Furthermore, if the following condition is also satisfied: $\frac{(P_k + \Delta P_k)}{\nu_k} > \nu_q \frac{P_q + \Delta P_q}{\nu_q}$, then the numerator in the left hand side will also be higher than the right hand side. This implies that the benefit of triple $(o_k, R_k^l, f_k^l)$ will be higher than the triple $(o_l, R_l^l, f_l^l)$, if the condition $\nu_k (P_k + \Delta P_k) > \nu_q (P_q + \Delta P_q)$ is satisfied.

Case 2: $\Delta P_k > \Delta P_q$, $P_k + \Delta P_k > P_q + \Delta P_q$ and $m_1 = m_2 = 0$.

$$\frac{X}{Y + (1 + \alpha) \cdot (P_k + \Delta P_k)} > (1 + \alpha) \cdot (P_k + \Delta P_k)$$
From Equation \(30\), we can derive the following condition:

\[
\nu_k(X + (1 + \alpha) \cdot (P_k + \Delta P_k)) \cdot (Y + \tau + \alpha \cdot \Delta P_k) > \\
\nu_q(X + (1 + \alpha) \cdot (P_q + \Delta P_q)) \cdot (Y + \tau + \alpha \cdot \Delta P_k) \\
\Rightarrow (\nu_k \cdot X + \nu_k \cdot (1 + \alpha) \cdot (P_k + \Delta P_k)) \cdot (Y + \tau + \alpha \cdot \Delta P_k) > \\
(\nu_q \cdot X + \nu_q \cdot (1 + \alpha) \cdot (P_q + \Delta P_q)) \cdot (Y + \tau + \alpha \cdot \Delta P_k)
\]

(31)

In the above equation, the dominant term on the left hand side is the value of \(\nu_k(P_k + \Delta P_k)\) and the dominant term on the right hand side is the value of \(\nu_q(P_q + \Delta P_q)\). This implies that if the value of \(\nu_k(P_k + \Delta P_k)\) is higher than \(\nu_q(P_q + \Delta P_q)\), then the value in the left hand side will become higher than the value on the right hand side.

**Case 3:** \(\Delta P_k > \Delta P_q, P_k + \Delta P_k < P_q + \Delta P_q, \text{ and } m_1 = m_2 = 0.\)

Let us compare the left hand side of the Equation (31) with the right hand side of the equation. In this case, if the value of \(\nu_k(P_k + \Delta P_k)\) is higher than the value of \(\nu_q(P_q + \Delta P_q)\), then the value of the left hand side will become higher than the value on the right hand side.

**Case 4:** \(\Delta P_k < \Delta P_q, P_k + \Delta P_k < P_q + \Delta P_q, \text{ and } m_1 = m_2 = 0.\)

Let us consider Equation (31) and compare the left hand side of the equation with the right hand side of the equation. After calculating few more steps in the equation, we can derive that the condition in which the left hand side will be higher than the right hand side, is as follows:

\[
\nu_k(P_k + \Delta P_k)\Delta P_k \text{ must be higher than the value of } \nu_q(P_q + \Delta P_q)\Delta P_k.
\]

According to the assumption of this case, \(\Delta P_k\) value is higher than the value of \(\Delta P_k\). This implies that, if the condition \(\nu_k(P_k + \Delta P_k) > \nu_q(P_q + \Delta P_q)\) holds, then the left hand side will be higher than the right hand side.

The above proofs will also hold for the scenarios where \(m_1 = m_2 \text{ and } m_1 > 0\). Only difference in the proofs of that scenario from the proofs of Cases 1-4 will be as follows: an additional constant term (i.e., the term \((P_r + P_{r-1} + \cdots + P_{r-(m_1-1)})\)) will be added to the numerators of both the sides of Equation (29). The remaining steps will remain the same as the proofs in Cases 1-4.

**Case 5:** \(\Delta P_k < \Delta P_q, P_k + \Delta P_k > P_q + \Delta P_q, \text{ and } m_1 < m_2.\)

If we compare the denominators of Equation (29) in both the sides, in the left hand side, \((\tau - m_1) < (\tau - m_2)\) and \(\Delta P_k < \Delta P_q\). This implies that, the denominator in the left hand side is smaller than the denominator in the right hand side. If we compare the numerators on both the sides, the value of \((P_k + P_{r-1} + \cdots + P_{r-(m_1-1)})\) is smaller than \((P_q + P_{r-1} + \cdots + P_{r-(m_2-1)})\) as \(m_1\) is smaller than \(m_2\). Furthermore, if \(\nu_k(P_k + \Delta P_k) > \nu_q(P_q + \Delta P_q)\) then the numerator in the left hand side will be higher than the numerator in the right hand side. Thus we can conclude that, Equation (29) will be satisfied when the following condition: \(\nu_k(P_k + \Delta P_k) > \nu_q(P_q + \Delta P_q)\) is satisfied.

**Case 6:** \(\Delta P_k > \Delta P_q, P_k + \Delta P_k > P_q + \Delta P_q, \text{ and } m_1 < m_2.\)

From Equation (29) we can derive the following condition:

\[
\nu_k(X - (1 + \alpha) \cdot (P_r + P_{r-1} + \cdots + P_{r-(m_1-1)}) + (1 + \alpha) \cdot (P_k + \Delta P_k)) \cdot (Y + (\tau - m_2) + \alpha \cdot \Delta P_k) > \\
\nu_q(X - (1 + \alpha) \cdot (P_r + P_{r-1} + \cdots + P_{r-(m_2-1)}) + (1 + \alpha) \cdot (P_q + \Delta P_q)) \cdot (Y + (\tau - m_1) + \alpha \cdot \Delta P_k)
\]

(32)

In the above equation, the value of \((P_r + P_{r-1} + \cdots + P_{r-(m_1-1)})\) is smaller than \((P_q + P_{r-1} + \cdots + P_{r-(m_2-1)})\) as \(m_1\) is smaller than \(m_2\). This favors the value in the left hand side of the equation. Furthermore, the value of \(\nu_k(P_k + \Delta P_k)\) in the left hand side is a more dominant term as compared to the value of \(\Delta P_k\). This implies that if the value of \((P_k + \Delta P_k)\) is higher than the value of \((P_q + \Delta P_q)\), then Equation (29) will hold.

**Case 7:** \(\Delta P_k > \Delta P_q, P_k + \Delta P_k < P_q + \Delta P_q, \text{ and } m_1 < m_2.\)

Let us compare the left hand side of the Equation (32) with the right hand side of the equation. As explained in the previous case, the value of \(\nu_k(P_k + \Delta P_k)\) in the left hand side is a more dominant term as compared to the value of \(\Delta P_k\). This implies that if the value of \((P_k + \Delta P_k)\) is higher than the value of \((P_q + \Delta P_q)\), then Equation (29) will hold and benefit of the triple \((o_k, \hat{R}_k, f_{k}^*)\) will be higher than the benefit of triple \((o_q, \hat{R}_q, f_{q}^*)\).

**Case 8:** \(\Delta P_k < \Delta P_q, P_k + \Delta P_k < P_q + \Delta P_q, \text{ and } m_1 < m_2.\)

Let us consider the Equation (32) and compare the left hand side of the equation with the right hand side. After deriving few more steps in this equation, we can derive that the condition in which the left hand side will be higher than the right hand side is as follows: \(\nu_k(P_k + \Delta P_k)\Delta P_k > \nu_q(P_q + \Delta P_q)\Delta P_k\).

According to the assumption of this case, \(\Delta P_k\) value is higher than the value of \(\Delta P_k\). This implies that, if the condition \(\nu_k(P_k + \Delta P_k) > \nu_q(P_q + \Delta P_q)\) holds, then the left hand side will be higher than the right hand side.

The above proofs (i.e., the proofs of Case 5-8) will also hold for the scenarios where \(m_1 > m_2\), due to the symmetric nature of the assumptions.

Based on the proofs of Cases 1-8, we can conclude that given two triples \((o_k, \hat{R}_k, f_{k}^*)\) and \((o_q, \hat{R}_q, f_{q}^*)\), if the condition of \(\nu_k(P_k + \Delta P_k) > \nu_q(P_q + \Delta P_q)\) is satisfied then the first triple will have higher benefit value than the second triple.

\[\square\]

### A.10 Disk-based Solutions

In the following paragraph, we have considered a simplified scenario of disk-resident objects and explained how our approach can be extended to this scenario with the help of an example.

Let us consider that, there is no particular order in which the objects are stored in the disk at the beginning of query execution. Furthermore, let us consider the strategy of estimating benefit of a block \(b_r\) in epoch \(l\) as follows: the summation of individual benefit values of the tuples present in Plan\(_r\), which contains objects of block \(b_r\).

Let us denote the cost of loading the \(r\)-th block \(b_r\) of \(O\), in memory as \(c_{load}^{b_r}\). Given a tuple \((o_m, R^k_{i}, f_{j}^*)\), if object \(o_m\)
∈ bₗ, then total cost of the tuple is estimated to be the cost of loading the object and executing the tagging function on object i.e., with the following metric: ($c_{\text{load}} + c_i$). If object $o_n$ is already present in memory, then the cost of loading $o_n$ will be zero and the total cost of such a tuple will be only $c_i$.

In our approach, in the plan generation phase there will be an additional step of block selection step, where we determine which block of objects from disk should be brought in memory. We would associate a benefit value with each block $bₗ$, denoted as $\text{BlockBenefit}$. This benefit metric of a block $bₗ$ in epoch $l$ would be calculated by the sum of benefit values of all the tuples present in Plan which contains objects from block $bₗ$. Based on this benefit metric, the block with maximum $\text{BlockBenefit}$ value would be chosen in this step and it is denoted by $\text{Block}_l$. In this scenario, plan execution phase would need to load $\text{Block}_l$ in memory first and then follow the remaining steps of plan execution as mentioned in our approach.

B. EFFICIENT UPDATE OF DATA STRUCTURES

Given a tuple $(o_k, R_{l,m}^l, f_{l,m}^l)$, we update the state hash map of object $o_k$, with respect to predicate $R_{l,m}^l$, by setting up the corresponding bit to $f_{l,m}^l$. Similarly, we update the predicate probability hash map of object $o_k$, w.r.t. predicate $R_{l,m}^l$ by adding the probability output of function $f_{l,m}^l$ in the hash map. Using the new predicate probability $p_{l,m}^l$, we calculate the new uncertainty value of object $o_k$ by Equation 6. We update the uncertainty list of object $o_k$ w.r.t. predicate $R_{l,m}^l$ with this new uncertainty value.

Along with updating object level data structures, we need to update $PQ$, as the benefit values of some tuples will be changed due to the execution of Plan. Let us consider the tuples in $PQ$ which were part of Plan first and then we will consider the tuples which were not part of Plan.

Given a tuple $(o_k, R_{l,m}^l, f_{l,m}^l) \in \text{Plan}_i$, we update $PQ$ as follows: based on the new joint probability value of object $o_k$, it can become part of the answer set in epoch $i$ or it can still be out of the answer set in epoch $i$. If $o_k$ becomes part of the Answer, then we remove all the tuples containing $o_k$ from $PQ$. If $o_k$ is not part of Answer, then we update the benefit values of the tuples containing $o_k$ in $PQ$.

Given a tuple $(o_k, R_{l,m}^l, f_{l,m}^l) \not\in \text{Plan}_i$, we update $PQ$ as follows: if joint probability of object $o_k$ is not changed from previous epoch $i - 1$, then we do not have to update anything for this tuple, as the benefit metric $\nu_k(P_k + \Delta P_k)$ will remain unchanged. If joint probability of object $o_k$ is changed from previous epoch $i - 1$, then we recompute the benefit of the tuples containing object $o_k$ and add it to $PQ$. 

18