Similarity Query Processing for Probabilistic Sets

Ming Gao [1], Cheqing Jin [1], Wei Wang [2], Xuemin Lin [1,2], Aoying Zhou [1*]

[1]*Shanghai Key Laboratory on trustworthy computing, Software Engineering Institute, East China Normal University, Shanghai, China*

[2]*The University of New South Wales, Sydney, Australia*

omega.mgao@gmail.com, {cqjin, ayzhou}@sei.ecnu.edu.cn
{weiw, lxue}@cse.unsw.edu.au

***Abstract*— Evaluating similarity between sets is a fundamental task in computer science. However, there are many applications in which elements in a set may be uncertain due to various reasons. Existing work on modeling such probabilistic sets and computing their similarities suffers from huge model sizes or significant similarity evaluation cost, and hence is only applicable to small probabilistic sets.**

In this paper, we propose a simple yet expressive model that supports many applications where one probabilistic set may have thousands of elements. We define two types of similarities between two probabilistic sets using the possible world semantics; they complement each other in capturing the similarity distributions in the cross product of possible worlds. We design efficient dynamic programming-based algorithms to calculate both types of similarities. Novel individual and batch pruning techniques based on upper bounding the similarity values are also proposed. To accommodate extremely large probabilistic sets, we also design sampling-based approximate query processing methods with strong probabilistic guarantees. We have conducted extensive experiments using both synthetic and real datasets, and demonstrated the effectiveness and efficiency of our proposed methods.

I. Introduction

Similarity query processing is a fundamental and active research area in databases. There are methods to abstract objects into either a multi-dimensional vector or a set of feature values. Similarity between objects then can be evaluated as similarity or distance between their computerized representations, which is amenable to many optimization techniques, such as indexing and top-k query optimization.

While there exists abundant research work in similarity query processing for certain sets, there are only few studies on probabilistic sets. Probabilistic sets arise naturally out of applications where data may contain noise, data values are inherently imprecise, or data values are summarized, so that the existence of an element in a set is characterized by its existential probability. Below are some example applications:

- Personalization systems can automatically construct a user's profile by logging the web sites or terms in the web page browsed by the user [33]. Personalization can be offered based on similarity between users' profiles [16]. Since the data is noisy (e.g., multiple persons using the same computer) and incomplete (e.g., some web sites were browsed using another web browser) in nature, a natural representation of the profile is all the web sites a user visited, each associated with a certain existential probability.
- Multi-label classification [34] is the task of assigning multiple class labels to an object. E.g., a document can belong to multiple topics [37]; a gene may be associated with multiple functions [5]. As the label assignment is usually performed by classifiers or humans, the set of class labels assigned to an object naturally forms a probabilistic set.

A commonality among these applications is the requirement to model a set where each element in the set is associated with some probability. However, there is little research in the database literature on this topic. An element-level and an instance-level model for probabilistic sets are proposed by Lian and Chen [27], which can also be used to model probablistic q-gram sets from uncertain strings [23]. It studies efficient similarity join on probabilistic sets based on several pruning techniques. Another recent work [39] proposes a simpler model for probabilistic set where the existence of elements are independent. However, to apply these models to the above-mentioned application scenarios, the user will face at least one of the following problems: (1) the probabilistic set needs exponentially large space to define if using the set-level model, (2) the similarity computation between two probabilistic sets requires time exponential in the size of the probabilistic sets, if using the element-level model, or (3) there are insufficient query processing techniques for pruning candidate sets especially when many sets in the database contains large numbers of elements. These problems limit the applicability of probabilistic sets to many modern applications which require typically hundreds of elements (or even more) per probabilistic set. For example, the number of classes for the multi-label datasets listed on the LIBSVM page[1] ranges from 6 to 313; the average number of elements in DBLP author's probabilistic profiles in our experiment is 205; the largest set in Del.icio.us dataset contains 0.3 million elements (Table VI).

In this work, we systematically study the problem of modeling uncertainty in sets with the aim to support the above-mentioned applications that demand support for large probabilistic sets. We model a probabilistic set, or a *p-set*, as a set of elements, each with its existential probability. We

* Corresponding author.

[1]http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html

then define two types of similarities between p-sets according to the possible world semantics. The first one is the expected similarity (*ES*), defined as the average similarity between all possible worlds of two p-sets. The other one is the confidence-based similarity (*CS*) with a user-given confidence of $minconf$. It is defined as the maximum similarity τ such that there is at least $minconf$ chance that two random worlds from two p-sets respectively have similarity no less than τ. We show that both types of similarity measures can be answered in $O(n^3)$ time by dynamic programming-based algorithms (n is the number of elements of the larger p-set).

Given the *ES* and *CS* similarity measures, we consider their corresponding similarity queries (termed *ESQ* and *CSQ* respectively) with a given similarity threshold. One of our major contributions is that we develop efficient pruning methods for both *ESQ* and *CSQ* in both individual and batch manners, based on novel derivations of similarity upper bounds. Two distinct features of our upper bounds are (1) we essentially bound the similarity value of two p-sets by a function rather than a specific value, and (2) the upper bounds are based on simple statistics which can be computed in time $O(n)$ and with a space complexity of $O(1)$. This matches or improves the complexity of alternative upper bounding methods in [27], and achieves better pruning effectiveness as demonstrated in our experiments.

Since the exact computation of similarities between p-sets requires $O(n^3)$ time and $O(n^2)$ space, it means it is still too slow or even infeasible for large p-sets with tens of thousands of elements. Therefore, we also propose approximate algorithms that compute the similarity values in $O(n)$ time with strong probabilistic guarantees.

Finally, we evaluate the proposed algorithms and demonstrate their efficiency in our extensive experiments using both synthetic and real-world datasets.

The contributions of the paper are summarized below.

- We propose an intuitive yet expressive model for probabilistic sets appearing in many real applications. As shown in Table I, our proposed model complements the existing ones by supporting applications that need to handle large p-sets due to the computational efficiency.
- We define two types of similarity measures to capture different characteristics of the similarity distributions between two p-sets under the possible world semantics and give efficient dynamic programming-based algorithms to compute these similarities.
- We design novel individual and batch pruning techniques to speed up the query processing. We also propose sampling-based algorithms to approximate the similarity between two p-sets with probabilistic guarantees.
- We conduct comprehensive experiments upon both synthetic and real datasets and demonstrate the efficiency and the effectiveness of the proposed approaches.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 gives the definition of our probabilistic set model and similarity queries. Section 4 presents our dynamic programming-based algorithms to compute these similarities exactly. Pruning rules to further speed up the query processing are given in Section 5. We show the sampling-based approximate algorithms in Section 6. Experimental results and analyses are given in Section 7. Section 8 concludes the paper.

II. Related Work

Uncertain Data Management. Recently, there has been a surge of research on managing uncertain data, partly because of the wide existence of uncertainty in many applications [1], such as sensor networks, automatic information extraction and integration, multimedia retrieval and *optical character recognition (OCR)*.Several prototype systems have been developed to manage such uncertain data, where data values or the existence of data items are associated with certain probability; they include MayBMS [3], MystiQ [14], and Trio [2], to just name a few. A recent survey can be found in [32].

Similarity Search. A similarity search, which returns objects similar or close to a query object, is a well-researched topic in the literature. Different variants of similarity search have been studied, including top-k [13], [20], [21], [22], [29], k-NN [9], reverse k-NN [12], and range queries [10], [20]. Based on the sliding window model, [24] studies top-k queries on uncertain data streams. Compared with the vast amount of similarity search research on certain data, there is relatively small amount of work on uncertain data. [38], [8] propose the frequent item mining on uncertain data. Pei et al. propose skyline query on uncertain data [30]. Xu et al. study similarity search based on the Earth Mover's Distance (EMD) [36]. It is popular to use probabilistic models to capture many alternatives during the OCR process. [26] proposes an approximation scheme to trade recall for query performance.

Similarity Join. A similarity join can be deemed as batch similarity queries. It has many applications, including data integration and data mining. Similarity joins on certain data have been extensively studied for various similarity and distance measures [35], [19].

There have been a few studies on similarity joins over uncertain data [25], [29], [23], [27]. Kriegel et al. study probabilistic spatial similarity join on uncertain data, where probabilistic distance functions are used to measure the similarity between uncertain objects [25]. [29] studies two kinds of probabilistic spatial join (PSJ) over uncertain data: the threshold PSJ and top-k PSJ.

A related work in multimedia retrieval is the recent Probabilistic signature method [6], where the set of feature points of an object are modeled as a distribution in the entire feature space. The distribution can be concisely described as a set of overlapping Gaussian distributions and efficiently learned via the EM algorithm. Unlike our method, the distances between Gaussian distributions are considered using a modified signature quadratic form distance [7].

We note that our Confidence-based similarity is similar to the median/quantile-based rank in [22], as both are based on the cumulative probability distribution.

TABLE I
COMPARISON OF PROBABILISTIC SET MODELS

Model	Expressive Power	Exact Similarity Computation	Upper Bound Computation
Set-level [27]	Most general	$O(N^2)$	$O(N)$
Element-level [27]	Can model exclusion	$\Omega(2^n)$	$O(n^2)$ (online) or $O(n)$ (offline)
Our p-set model	A special case of Element-level model	$O(n^3)$	$O(n)$

Notations: (i) n is the size of a probabilistic set in both the element-level model and our proposed p-set model. (ii) N is the number of possible instances of a probabilistic set under the set-level model. If we use set-level model to model a probabilistic set in the element-level or our proposed p-set model, $N = \Omega(2^n)$.

The closest work to ours is [39], [23], [27]. [39] considers efficient query processing methods for set containment queries for probabilistic sets. Their probabilistic set model is identical to ours. One of the similarity measure considered (expected Jaccard containment) and its dynamic programming-based computation algorithm are similar to our expected Jaccard similarity and its computation algorithm. However, [39] does not consider pruning method for individual sets or batch pruning, nor do they consider approximate query processing methods, as the probabilistic sets considered contain small number of elements (up to 25).

[23] investigates the problem of probabilistic similarity joins over strings regarding the expected edit distance measure (EED), and a set of novel techniques have been developed. Nevertheless, there are several major technical differences between [23] and ours. For example, although character-level uncertain strings are decomposed into probabilistic sets of q-grams, there are non-neglectible correlations among these elements. In addition, the major technical challenge in [23] is to establish lower and upper bounds for EED based on some functions involving aggregated probabilities of constituent q-grams.

A. Comparison with [27]

We compare our work with [27] from several different aspects below. A summary of comparisons is listed in Table I.

Models and Similarity Evaluation. Lian and Chen propose two probabilistic set models, namely the *set-level* model and the *element-level* model [27]. The former model explicitly enumerates all possible set instances with existential probabilities, and all these set instances exist exclusively. The latter model enumerates all possible element instances with their existential probabilities at each of the n positions in a set; all element instances at the same position exist exclusively, while element instances at different positions exist independently.

The set-level model is more expressive than the element-level model, as the latter can always be encoded by the former. The main limitation of the set-level and element-level models is their intractability of model size and computational complexity, respectively. For the element-level model, while it can concisely encode the p-set in space $O(n)$, the only way to evaluate the similarity measure is by the definition, i.e., enumerating all the possible worlds of two p-sets and computing the similarities for the cross product of the possible worlds. For two element-level encoded p-sets of sizes n and m, the similarity computation time is $\Omega(2^{n+m})$. Given the sizes of the probabilistic sets used in our experiments (See Table VI), it is obvious that both models in [27] cannot be used.

As a comparison, our model achieves a good balance between the model complexity and the efficiency of similarity retrieval algorithms. For the preceding example, our p-set instance has a size of $O(n)$, and various similarity values can be computed exactly in time $O(n^3)$, or approximately with strong probabilistic guarantees in time $O(n)$.

Pruning Rules. [27] proposes two pruning rules to evaluate the similarity metric between a pair of uncertain sets. One is Jaccard Distance pruning, which uses the property of triangle inequality of *Jaccard distance* to prune dissimilar set pairs. The other one, probability upper bound pruning (*PUBP* in short), computes an upper bound of the similarity metric in time $O(n^2)$ or time $O(n)$ if certain statistics have been pre-calculated offline.

The following negative results show that it is pointless to adapt and use the Jaccard distance pruning [27] in our probabilistic set model.

Theorem 1: The Jaccard distance pruning rule in [27] cannot prune any candidate p-set for the *CSQ* query if none of the elements' probabilities is 1.0 (i.e., all elements are uncertain).

Hence, we will develope both batch and individual pruning methods for our two similarity measures. Our pruning methods are both efficient and have demonstrated better pruning power than PUBP in our experiments.

III. PROBLEM DEFINITION AND DATA NORMALIZATION

A. Probabilistic Set Model

We model a probabilistic set $\mathcal{A}$ in a domain $\mathcal{D}$ as

$$\mathcal{A} = \{a_i : p_{a_i} | a_i \in \mathcal{D}, \forall i \in [1, n]\}$$

where $\forall i \neq j$, $a_i \neq a_j$. Note that we require $p_{a_i} > 0$, as otherwise it can be trivially removed from the p-set.

The semantics is that each element a_i has an existential probability p_{a_i}, and the existence of the element is independent of each other. Although simple, it can model uncertainty in many applications. For example, [8] performs frequent itemset mining on uncertain transaction databases, where each "tuple" in the transaction database is a collection of items, each with its existential probability. In fact, in one of the popular approaches to model probabilistic relational databases [14], a probabilistic relation can be modeled in our model as a probabilistic set made up of tuples.

Under the *possible world* semantics [15], a probabilistic set $\mathcal{A}$ corresponds to many *possible world*s, where each possible

world, $w(\mathcal{A})$, is a (certain) set which is a subset of the elements in $\mathcal{A}$. The probability of a possible world w, $\mathbf{Pr}[w]$, can be computed as $\mathbf{Pr}[w] = \prod_{t \in w} p_t \prod_{t \notin w}(1 - p_t)$, where t is an element in $\mathcal{A}$. For simplicity, we call a probabilistic set under our model simply as a *p-set*.

Now consider similarity between two p-sets, $\mathcal{A}$ and $\mathcal{B}$. Here we assume independence between the possible worlds of $\mathcal{A}$ and $\mathcal{B}$. This assumption is reasonable in the absence of correlation between two p-sets, and it has been widely employed in other probabilistic database studies [14]. The comparison of two p-sets generates joint possible worlds, $\mathcal{W}(\mathcal{A}, \mathcal{B})$, which are all the combination of possible worlds of individual p-sets, i.e., $\mathcal{W}(\mathcal{A}, \mathcal{B}) = \mathcal{W}(\mathcal{A}) \times \mathcal{W}(\mathcal{B})$. The existential probability of $(w_a, w_b) \in \mathcal{W}(\mathcal{A}, \mathcal{B})$ is $\mathbf{Pr}[w_a] \cdot \mathbf{Pr}[w_b]$.

For each possible world of $\mathcal{W}(\mathcal{A}, \mathcal{B})$, a similarity value can be obtained. These values collectively form a discrete probabilistic distribution of similarity values, which fully characterizes the similarity between two p-sets.

A natural way to concisely capture the similarity by one single value is to use the mean of the similarities. Hence we define *expected similarity* (*ES*) as follows.

Definition 1 (Expected Similarity, ES): The expected similarity between two p-sets, $\mathcal{A}$ and $\mathcal{B}$, is the expected similarity between possible worlds of $\mathcal{A}$ and $\mathcal{B}$, i.e.,

$$\begin{aligned} ES(\mathcal{A}, \mathcal{B}) &= \sum_{(w_a, w_b) \in \mathcal{W}(\mathcal{A}, \mathcal{B})} sim(w_a, w_b) \cdot \mathbf{Pr}[(w_a, w_b)] \\ &= \sum_{w_a \in \mathcal{W}(\mathcal{A}) \wedge w_b \in \mathcal{W}(\mathcal{B})} sim(w_a, w_b) \cdot \mathbf{Pr}[w_a] \cdot \mathbf{Pr}[w_b] \end{aligned}$$

where sim is a similarity function.

The choice of the similarity function, sim, depends on the application scenario. In this paper, we follow previous work [27] and focus on using Jaccard coefficient as the similarity function. The Jaccard coefficient between two sets is defined as their intersection size over their union size, i.e., $jac(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. For simplicity, we define $sim(w_a, w_b) = 0$ when $w_a = \emptyset$ and $w_b = \emptyset$. Note that our methods can be generalized to other measures.

Note that commonly used set similarity functions naturally assume that the similarity between two different elements is zero. We follow the same convention here and will consider more general models that consider inter-element similarity in future work.

ES, being a concise summary, may not capture the overall distribution of similarity values [17]. We complement it with the notion of *confidence-based similarity*, or *CS*. We first define the *conditioned cumulative probability*, $\mathbf{CPr}(x, \mathcal{A}, \mathcal{B})$, as the total probability associated with possible worlds where the similarity value is no less than x, i.e.,

$$\mathbf{CPr}(x, \mathcal{A}, \mathcal{B}) = \sum_{(w_a, w_b) \in \mathcal{W}(\mathcal{A}, \mathcal{B}) \wedge sim(w_a, w_b) \geq x} \mathbf{Pr}[(w_a, w_b)]$$

Then, we define the confidence-based similarity as follows.

Definition 2 (Confidence-based Similarity, CS): Let $minconf \in [0, 1]$ be a user-defined minimum confidence threshold. The confidence-based similarity between two p-sets, $\mathcal{A}$ and $\mathcal{B}$, is the maximum similarity value x such that the corresponding cumulative probability is at least $minconf$, i.e., $CS(\mathcal{A}, \mathcal{B}, minconf) = \max\{\, x \mid \mathbf{CPr}(x, \mathcal{A}, \mathcal{B}) \geq minconf \,\}$.

TABLE II

POSSIBLE WORLDS AND SIMILARITIES

(a) Example P-sets

$\mathcal{A}$	$\mathcal{B}$
$\{1 : 0.7,\ 2 : 1.0\}$	$\{1 : 1.0,\ 2 : 0.5,\ 3 : 0.8\}$

(b) All the Joint Possible Worlds with Non-zero Probabilities ($e^{\mathcal{X}}$ is the element e of p-set $\mathcal{X}$)

w_a	w_b	$\mathbf{Pr}[(w_a, w_b)]$	**Jaccard**
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}\}$	0.03	0
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}\}$	0.03	0.5
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.12	0
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.12	0.333
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}\}$	0.07	0.5
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}\}$	0.07	1
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.28	0.333
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.28	0.666

TABLE III

SIMILARITIES OF TWO P-SETS $\mathcal{A}$ AND $\mathcal{B}$

	Jaccard
$ES(\mathcal{A}, \mathcal{B})$	0.44
$CS(\mathcal{A}, \mathcal{B}, minconf = 0.3)$	0.666
$CS(\mathcal{A}, \mathcal{B}, minconf = 0.5)$	0.333

Example 1: Consider two p-sets $\mathcal{A}$ and $\mathcal{B}$ in Table II(a). They have eight (joint) possible worlds, which are listed together with their similarities and existential probabilities in Table II(b). For example, the 2nd possible world is made up of $\{\, 2^{\mathcal{A}} \,\}$ and $\{\, 1^{\mathcal{B}}, 2^{\mathcal{B}} \,\}$. The former's existential probability is $(1 - 0.7) \cdot 1 = 0.3$, and the latter's existential probability is $1 \cdot 0.5 \cdot (1 - 0.8) = 0.1$. Therefore, the existential probability of the joint possible world is $0.3 \cdot 0.1 = 0.03$.

Example 2: We also show the similarity values of two p-sets in Table III. The *ES* value is the average of Jaccard values in Table II(b) weighted by their respective probabilities. The *CS* value for Jaccard similarity in the 2nd row is 0.5 because there are two possible worlds where the similarity is no less than 0.666 in Table II(b); the total probability of these possible worlds is: $0.07 + 0.28 = 0.35$ and hence is greater than the $minconf = 0.3$. In addition, it can be verified that this is the maximum value. E.g., the total probability of possible worlds with a similarity greater than 0.666 is only 0.07.

We can also observe that *CS* can give a more detailed picture of the similarity value distribution between two p-sets by varying the $minconf$ value.

In many applications, we are mostly interested in highly similar results. Therefore, given a query p-set, we can define *expected similarity query* (*ESQ*) as finding all p-sets such that the expected similarity between the query p-set and the result p-set is no less than a user-given threshold. We can define *confidence-based similarity query* (*CSQ*) in a similar fashion.

Although one may informally interpret the probabilities in our p-set model as weights, our similarity measures are fundamentally *different* from the *weight Jaccard*

similarity [18], as the latter essentially only works on certain sets. For a simple example, consider $\mathcal{A} = \{1 : 0.8\}$ and $\mathcal{B} = \{1 : 0.4\}$. Weighted Jaccard will give similarity 0.5, while ES between the two is 0.32.

B. Normalization of Two P-Sets

Given two p-sets $\mathcal{A}$ and $\mathcal{B}$ of sizes n and m, respectively, we call the elements that occur (albeit probabilistically) in both p-sets as *common elements*, denoted as c_i, and other elements *distinct elements*, denoted as d_j. Let the number of common elements be k ($0 \leq k \leq \min(n, m)$), then the two sets can also be represented in the following forms:

$$\mathcal{A} = \{ c_1 : p^{\mathcal{A}}_{c_1}, \ldots, c_k : p^{\mathcal{A}}_{c_k},\ d_1 : p_{d_1}, \ldots, d_{n-k} : p_{d_{n-k}} \}$$
$$\mathcal{B} = \{ c_1 : p^{\mathcal{B}}_{c_1}, \ldots, c_k : p^{\mathcal{B}}_{c_k},\ d_{n-k+1} : p_{d_{n-k+1}}, \ldots d_{n+m-2k} : p_{d_{n+m-2k}} \}$$

We call this process *normalization*, and it can be performed by intersecting two p-sets to find the common elements, which can be generally performed in $O(n + m)$ time.

In the rest of the paper, we assume all pairs of p-sets are normalized.

Example 3: Consider the two p-sets $\mathcal{A}$ and $\mathcal{B}$ in Table II(a). The common elements are $\{1, 2\}$, and the distinct element set is $\{3\}$. The two sets are already in the normalized forms.

C. Notations

We use $|\mathcal{A}|$ to denote the random variable that corresponds to the size of the set in a possible world of $\mathcal{A}$. $\mathbf{E}[|\mathcal{A}|]$ is its expectation.

The *size* of a p-set is defined as the number of elements it contains. This is to be distinguished from its *expected size*, which is the average size of its possible worlds. For example, a p-set with huge size may actually have a small expected size if every element is associated with a small probability.

IV. Exact Similarity Computation

In this section, we propose efficient methods to answer *ESQ* and *CSQ* queries. The key is efficient methods to calculate *ES* and *CS* values for two p-sets.

The naïve method to compute these similarity values is based on their definitions, which needs to explore all possible world pairs from two p-sets, hence resulting in computational cost exponential in the size of the p-sets.

The key observation to more efficient computational methods is the fact that our similarity functions only rely on a few key statistics, and these statistics can be computed efficiently via dynamic programming, and hence avoid the exponential cost of enumerating all possible worlds. In the following, we will introduce the overall method first, and then give further details.

A. Overview

We can partition the joint possible worlds of $\mathcal{A}$ and $\mathcal{B}$ according to *equivalent classes* based on their intersection and union sizes. Specifically, we use $H[i, j]$ to denote the total existential probabilities of possible worlds (w_a, w_b) where w_a and w_b's intersection size is exactly i and their union size is exactly j, i.e.,

$$H[i, j] = \sum_{(w_a, w_b) \in \mathcal{W}(\mathcal{A}, \mathcal{B}) \wedge |w_a \cap w_b| = i \wedge |w_a \cup w_b| = j} \mathbf{Pr}[w_a] \cdot \mathbf{Pr}[w_b]$$

TABLE IV
Equivalent Classes

w_a	w_b	$\mathbf{Pr}[(w_a, w_b)]$	i	j	**Jaccard**
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}\}$	0.03	0	2	0
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.12	0	3	0
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.12	1	3	0.333
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.28	1	3	0.333
$\{2^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}\}$	0.03	1	2	0.5
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}\}$	0.07	1	2	0.5
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}, 3^{\mathcal{B}}\}$	0.28	2	3	0.666
$\{2^{\mathcal{A}}, 1^{\mathcal{A}}\}$	$\{1^{\mathcal{B}}, 2^{\mathcal{B}}\}$	0.07	2	2	1

TABLE V
$H[i, j]$

	$j = 0$	$j = 1$	$j = 2$	$j = 3$
$i = 0$	0	0	0.03	0.12
$i = 1$		0	0.1	0.4
$i = 2$			0.07	0.28

Example 4: Consider again the sixteen possible worlds in Table II(b). We regroup them into six classes, as shown in Table IV. Then it is easy to calculate $H[i, j]$, as shown in Table V. For example, $H[1, 3]$ corresponds to the third and fourth possible worlds in Table IV. So $H[1, 3] = 0.12 + 0.28 = 0.4$. Once $H[i, j]$ is calculated, we can calculate *ES* and *CS* easily as follows. For *ES*, we simply compute the weighted average of all the entries in H, i.e, $ES = \sum_{i=1}^{k} \sum_{j=i}^{m+n-k} H[i, j] \cdot (i/j)$, where k is the number of common elements. For *CS*, a naïve method is to sort all entries of H by decreasing order of similarity values (i.e., $\frac{i}{j}$), and then scan the sorted entry until the prefix sum of the probabilities is at least $minconf$.

When $minconf$ is large, it is expected that only a few entries need to be summed up. We can calculate *CS* more efficiently by accessing entries by the decreasing order of the similarity values and stop whenever the answer is found. The algorithm is described in Algorithm 1. We use a max heap that can return entries with the maximum similarity values. First, we push all the "diagonal" entries into the heap, as they have the maximum similarity value of 1.0 (Line 1). Then we repeatedly pop the entry with the maximum similarity, sum up the total probabilities, and break the loop if the current total probability is no smaller than $minconf$ (Lines 4–6). Finally, we push the "right" neighbor of the current entry into the heap. This is because, this "right" neighbor has the larger similarity than the "above" neighbor (since $\frac{i}{j+1} > \frac{i-1}{j}$).

The time complexity of Algorithm 1 is $O(k + out \cdot \log k)$, where $k \leq \min(n, m)$ is the number of common elements, and $out \leq k(n + m - k) - k(k - 1)/2$ is the number of entries that must be retrieved from the heap to calculate the *CS* value under a given $minconf$.

Algorithm 1: Calculate *CS* from $H[i,j]$

Input: $H[i,j]$, $minconf$
Data: $heap$ is a max-heap on the similarity values.
1 **for** $i = 1$ **to** k **do** $heap.push(1.0, i, i)$;
2 $CPr \leftarrow 0$; $sim \leftarrow 0$;
3 **while** $heap$.empty $=$ **false do**
4 $(sim, i, j) \leftarrow heap$.pop;
5 $CPr \leftarrow CPr + H[i,j]$;
6 **if** $CPr \geq minconf$ **then break**;
7 **if** $j < m + n - k$ **then** $heap$.push$(\frac{i}{j+1}, i, j+1)$;
8 **return** sim

B. Computing H

Now we show that $H[i,j]$ can be computed in polynomial time via dynamic programming. Given normalized p-sets $\mathcal{A}$ and $\mathcal{B}$, we construct an array $\vec{e} = (c_1, c_2, \ldots, c_k, d_1, d_2, \ldots, d_{m+n-2k})$ that collects all the elements from two p-sets. Consider the l-th element (e_l). If it is a distinct element (i.e., $k < l \leq m + n - k$), then we use p_l to denote its associated probability in one of the p-sets. If e_l is a common element (i.e., $1 \leq l \leq k$), then we use $p_l^{\mathcal{X}}$ to denote its probability in p-set $\mathcal{X}$.

Now we consider the sub-problem of calculating H where only the first l elements $(e_1, \ldots, e_l)$ and their associated probabilities are considered. We denote the results as H^l. Obviously, $H^{m+n-k}[i,j]$ is the H we need.

If the current element e_l is a common element, then

$$\begin{aligned} H^l[i,j] = {} & H^{l-1}[i,j](1-p_l^{\mathcal{A}})(1-p_l^{\mathcal{B}}) \\ & + H^{l-1}[i,j-1](p_l^{\mathcal{A}}(1-p_l^{\mathcal{B}}) + (1-p_l^{\mathcal{A}})p_l^{\mathcal{B}}) \\ & + H^{l-1}[i-1,j-1]p_l^{\mathcal{A}}p_l^{\mathcal{B}} \end{aligned} \quad (1)$$

The above recurrence equation corresponds to the four possible cases regarding the occurrence of e_l in $\mathcal{A}$ and $\mathcal{B}$, respectively. For example, the first part is for the case where e_l does not appear in either $\mathcal{A}$ or $\mathcal{B}$; therefore, in this case, $H^l[i,j] = H^{l-1}[i,j](1-p_l^{\mathcal{A}})(1-p_l^{\mathcal{B}})$.

If the current element is a distinct element, then

$$H^l[i,j] = H^{l-1}[i,j](1-p_l) + H^{l-1}[i,j-1]p_l \quad (2)$$

The above recurrence equation corresponds to the two possible cases regarding the occurrence of e_l.

Initially, we set $H^0[i,j] = 1$ when $i = 0 \wedge j = 0$, and $H^0[i,j] = 0$ otherwise. We also return $H^{l-1}[i,j] = 0$ if one of the following conditions is violated: (i) $0 \leq i \leq k$; (ii) $i \leq j \leq l \leq n + m - k$.

We omit the details of the dynamic programming-based algorithm based on Equations (1) and (2) in the interest of space. We note that the method is actually a special case of the marginal vector convolution method [31] on a semi ring with $S = \mathbb{Z}_{\max(n,m)}$.

Assuming $n \geq m \geq k$, the time complexity of computing $H^{m+n-k}[i,j]$ ($0 \leq i \leq k$ and $0 \leq j \leq m + n - k$) can be shown to be $\Theta(kn^2)$ or $O(n^3)$, and the space complexity is $\Theta(kn)$, or $O(n^2)$.

V. Pruning Techniques

Given that we can compute the *ES* and *CS* similarities efficiently using dynamic programming-based algorithms, the naïve query processing algorithm to answer *ESQ* or *CSQ* queries is to compute the respective similarity between the query p-set and each p-set in the database and output only those that pass the similarity threshold.

The obvious disadvantage of the basic query processing algorithm is that it needs to run a costly similarity computation algorithm for *every* p-set in the database irrespective of the similarity threshold. In the following, we will develop an efficient pruning method based on estimating upper bounds for both *ES* and *CS* queries. Exact similarity computation is only performed on p-sets that survive our two levels of pruning: individual pruning and batch pruning. This is depicted in Algorithm 2.

In the rest of this section, we first show how these non-trivial upper bounds are derived (Lines 4–9 in Algorithm 2), and then consider their generalization to allow efficient batch pruning (Line 1 in Algorithm 2).

Algorithm 2: Answer Queries with Pruning ($\mathcal{Q}$, $\{\mathcal{O}_i\}$, τ, $minconf$)

1 $C \leftarrow$ candidates that survive the batch pruning (c.f., Sec. V-D);
2 **foreach** *p-set in* C **do**
3 $pruned \leftarrow$ **false**;
4 **if** *the query type is ESQ* **then**
5 $ub \leftarrow$ calcESUpperBound($\mathcal{Q}$, $\mathcal{O}_i$) (c.f., Sec. V-B);
6 **if** $ub < \tau$ **then** $pruned \leftarrow$ **true**
7 **if** *the query type is CSQ* **then**
8 $ub \leftarrow$ calcCSUpperBound($\mathcal{Q}$, $\mathcal{O}_i$, τ) (c.f., Sec. V-C);
9 **if** $ub < minconf$ **then** $pruned \leftarrow$ **true**
10 **if** $pruned =$ **false then**
11 $sim \leftarrow$ the similarity value between $\mathcal{Q}$ and $\mathcal{O}_i$;
12 **if** $sim \geq \tau$ **then**
13 output $\mathcal{O}_i$;

A. Preliminaries

Our upper bound calculation only requires two simple statistics, namely, the *expected set intersection size and union size*, which can be computed in $O(n+m)$ time due to linearity of expectation. First, the *expected size of a p-set* $\mathcal{A}$, $\mathbf{E}[|\mathcal{A}|]$, is $\sum_{w \in \mathcal{W}(\mathcal{A})} |w| \cdot \mathbf{Pr}[w] = \sum_{l=1}^{n} p_l^{\mathcal{A}}$. Second, the *expected intersection size of two p-sets* $\mathcal{A}$ *and* $\mathcal{B}$, $\mathbf{E}[|\mathcal{A} \cap \mathcal{B}|]$, is $\sum_{(w_a, w_b) \in \mathcal{W}(\mathcal{A},\mathcal{B})} |w_a \cap w_b| \cdot \mathbf{Pr}[(w_a, w_b)] = \sum_{l=1}^{k} p_l^{\mathcal{A}} \cdot p_l^{\mathcal{B}}$. Last, the *expected union size of two p-sets* $\mathcal{A}$ *and* $\mathcal{B}$, $\mathbf{E}[|\mathcal{A} \cup \mathcal{B}|]$, is $\mathbf{E}[|\mathcal{A}|] + \mathbf{E}[|\mathcal{B}|] - \mathbf{E}[|\mathcal{A} \cap \mathcal{B}|] = \sum_{l=1}^{k}(p_l^{\mathcal{A}} + p_l^{\mathcal{B}} - p_l^{\mathcal{A}} \cdot p_l^{\mathcal{B}}) + \sum_{l=k+1}^{n+m-k} p_l$.

B. Pruning Rule for ESQ

The *ES* value of two p-sets $\mathcal{A}$ and $\mathcal{B}$ is the expected Jaccard similarity between two random possible worlds of $\mathcal{A}$ and $\mathcal{B}$, respectively. Hence, its value should be concentrated around the expected set intersection size over the expected set union size of $\mathcal{A}$ and $\mathcal{B}$ under certain conditions. Below we show how to derive *non-trivial* upper bounds based on this idea.

We first introduce the following theorem and then apply it to obtain an upper bound of the *ES* value. The core idea is that if a random variable X is the sum of *Poisson trials*, we can use the Chernoff bound to bound the total probabilities that its value deviates much from its expectation. Note that the set intersection size and union size are both sums of Poisson trials. For example, random variable $|\mathcal{A} \cap \mathcal{B}|$ can be deemed as the sum of performing k coin tosses, each with $p_i^{\mathcal{A}} p_i^{\mathcal{B}}$ probability, which is the probability that i-th common element in $\mathcal{A}$ and $\mathcal{B}$ appears at the same time.

Theorem 2: Let X and Y denote two sums of Poisson trials. $Y \geq X \geq 0$. Then $\mathbf{E}[X/Y] < UB_1(\mathbf{E}[X], \mathbf{E}[Y])$ where [2]

$$UB_1(u, v) = \min_{\exp(-u/3) \leq \epsilon \leq 1} \left(2\epsilon + \frac{u + \sqrt{-3u \ln \epsilon}}{v - \sqrt{-2v \ln \epsilon}}\right)$$

Proof: We define two events, H_1 and H_2, for X and Y. $H_1 : X > (1 + \lambda_X)\mathbf{E}[X]$. $H_2 : Y < (1 - \lambda_Y)\mathbf{E}[Y]$. For any ϵ satisfying $\exp(-\frac{\mathbf{E}[X]}{3}) \leq \epsilon \leq 1$, we set $\lambda_X = \sqrt{\frac{-3 \ln \epsilon}{\mathbf{E}[X]}}$ and $\lambda_Y = \sqrt{\frac{-2 \ln \epsilon}{\mathbf{E}[Y]}}$. Applying the Chernoff inequality, we have

$$\mathbf{Pr}[H_1] < \exp\left(-\frac{\lambda_X^2 \mathbf{E}[X]}{3}\right) = \epsilon$$
$$\mathbf{Pr}[H_2] < \exp\left(-\frac{\lambda_Y^2 \mathbf{E}[Y]}{2}\right) = \epsilon$$

Then, $\mathbf{E}[X/Y] = V_1 + V_2 + V_3 < 2\epsilon + \frac{1+\lambda_X}{1-\lambda_Y} \cdot \frac{\mathbf{E}[X]}{\mathbf{E}[Y]}$, where

$$V_1 = \mathbf{E}[X/Y \mid H_2] \cdot \mathbf{Pr}[H_2] < 1 \cdot \epsilon$$
$$V_2 = \mathbf{E}\left[X/Y \mid H_1, \overline{H_2}\right] \cdot \mathbf{Pr}\left[H_1 \wedge \overline{H_2}\right] < 1 \cdot \epsilon$$
$$V_3 = \mathbf{E}\left[X/Y \mid \overline{H_1}, \overline{H_2}\right] \cdot \mathbf{Pr}\left[\overline{H_1} \wedge \overline{H_2}\right] < \frac{(1 + \lambda_X)\mathbf{E}[X]}{(1 - \lambda_Y)\mathbf{E}[Y]} \cdot 1$$

We will have the UB_1 function by expanding the definitions of λ_X and λ_Y and the theorem is proved. The range of ϵ in UB_1 is set because $\lambda_X, \lambda_Y \in [0, 1]$ [3] and $e^{-u/3} > e^{-v/2}$. ■

Unlike many common approaches where a value is obtained as an upper bound, our upper bound is the minimum value of a function. One can either choose an appropriate numerical analysis method to obtain the minimum upper bound, or use an appropriate fixed ϵ value to obtain a loose upper bound.

Theorem 3: Given an *ES* query p-set $\mathcal{Q}$, a data p-set $\mathcal{O}$, the similarity threshold τ, $\mathcal{O}$ can be pruned if $UB_1(\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|], \mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|]) \leq \tau$.

Proof: We define two random variables X and Y as: $X = |\mathcal{Q} \cap \mathcal{O}|$, $Y = |\mathcal{Q} \cup \mathcal{O}|$. $Y \geq X \geq 0$. Applying Theorem 2, we have:

$$ES(\mathcal{Q}, \mathcal{O}) = \mathbf{E}[X/Y] < UB_1(\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|], \mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|])$$

Therefore, if the upper bound obtained by the UB_1 function is no larger than τ, we can safely prune $\mathcal{O}$. ■

The calcESUpperBound function in Algorithm 2 implements the pruning due to Theorem 3.

[2]We assume $X/Y = 0$ when $Y = 0$.

[3]The upper bound of the Chernoff bound can be simplified as $\mathbf{Pr}[X > (1 + \lambda)\mathbf{E}[X]] < \exp(-\frac{\lambda^2 \mathbf{E}[X]}{3})$ when $0 \leq \lambda \leq 1$.

C. Pruning Rule for CSQ

We first introduce Theorem 4 and then apply it to our *CSQ* problem to obtain the pruning rule.

Theorem 4: Let X and Y be two sums of Poisson trials. For a given $\alpha \in (0, 1]$, if $\mathbf{E}[X] \leq \alpha \mathbf{E}[Y]$, we have:

$$\mathbf{Pr}[X \geq \alpha Y] < UB_2(\mathbf{E}[X], \mathbf{E}[Y], \alpha)$$

where

$$UB_2(u, v, \alpha) = \min_{u \leq \xi \leq \min(\alpha v, 2u)} \left(e^{\frac{-(\alpha v - \xi)^2}{2\alpha^2 v}} + e^{\frac{-(\xi - u)^2}{3u}}\right)$$

Proof: Consider the following two events, H_1 and H_2, where ξ is between $\mathbf{E}[X]$ and $\alpha \mathbf{E}[Y]$. $H_1 : X > \xi$. $H_2 : Y \leq \xi/\alpha$. Applying the Chernoff inequality, we have

$$\mathbf{Pr}[H_1] < \exp\left(-\frac{(\xi - \mathbf{E}[X])^2}{3\mathbf{E}[X]}\right) = \epsilon_1$$
$$\mathbf{Pr}[H_2] \leq \exp\left(-\frac{(\alpha \mathbf{E}[Y] - \xi)^2}{2\alpha^2 \mathbf{E}[Y]}\right) = \epsilon_2$$

$$\begin{aligned}\mathbf{Pr}[X \geq \alpha Y] &= \mathbf{Pr}[(X \geq \alpha Y) \wedge (X \leq \xi)] \\ &\quad + \mathbf{Pr}[(X \geq \alpha Y) \wedge (X > \xi)] \\ &\leq \mathbf{Pr}[\xi \geq \alpha Y] + \mathbf{Pr}[X > \xi] \qquad < \epsilon_2 + \epsilon_1\end{aligned}$$

Hence, we have the UB_2 function by expanding the definitions of ϵ_1 and ϵ_2, and the theorem is proved. The range of ξ is because $\xi \in [\mathbf{E}[X], \alpha \mathbf{E}[Y]]$ and $\xi \leq 2\mathbf{E}[X]$. ■

Theorem 5: Given a *CS* query p-set $\mathcal{Q}$, a data p-set $\mathcal{O}$, parameters $minconf$ and τ. Then $\mathcal{O}$ can be pruned if (i) $\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|] \leq \tau \cdot \mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|]$, and (ii) $UB_2(\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|], \mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|], \tau) \leq minconf$.

Proof: We define two random variables X and Y as: $X = |\mathcal{Q} \cap \mathcal{O}|$, $Y = |\mathcal{Q} \cup \mathcal{O}|$. Applying Theorem 4, we have:

$$\mathbf{Pr}\left[\frac{|\mathcal{Q} \cap \mathcal{O}|}{|\mathcal{Q} \cup \mathcal{O}|} \geq \tau\right] < UB_2\left(\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|], \mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|], \tau\right)$$

Note that the *CS* value decreases monotonically with $minconf$. Hence if the confidence upper bound obtained by the UB_2 function is smaller than $minconf$, we can safely conclude that *CS* of $\mathcal{Q}$ and $\mathcal{O}$ is strictly smaller than τ. Therefore, $\mathcal{O}$ can be pruned. ■

The calcCSUpperBound function in Algorithm 2 implements the pruning due to Theorem 5.

D. Batch Pruning

The above pruning procedure needs to evaluate the upper bound between the query p-set and *every* p-set in the database, hence the query processing cost scales at least linearly with the size of the database. In this subsection, we introduce *batch pruning* techniques aiming at discarding many p-sets in the database without even evaluating their similarity upper bounds.

The framework is as follows.

1) We index all the p-sets in the database by their expected sizes, such that we can efficiently retrieve p-sets whose expected sizes fall within a given range $[a, b]$.

2) Given a query with a p-set $\mathcal{Q}$, a threshold bound τ, and confidence parameter $minconf$ (only for *CSQ*), we compute a lower bound S_L and an upper bound S_U of the expected size for the appropriate query type.
3) We only consider p-sets in the database whose expected sizes fall within $[S_L, S_U]$.

The main challenge is to decide the two bounds S_U and S_L given only the query p-set $\mathcal{Q}$. We achieve this based on two key observations: (i) Since the upper bound for *ESQ* (resp. *CSQ*) is the minimum value of a function of a free variable (ϵ (resp. ξ)), we still have a valid (albeit a bit loose) upper bound if we fix the free parameter. (ii) Both upper bounds only depend on the expected intersection and union sizes, both can be bounded as $\mathbf{E}[|\mathcal{Q} \cap \mathcal{O}|] \leq \min(\mathbf{E}[|\mathcal{Q}|], \mathbf{E}[|\mathcal{O}|])$ and $\mathbf{E}[|\mathcal{Q} \cup \mathcal{O}|] \geq \max(\mathbf{E}[|\mathcal{Q}|], \mathbf{E}[|\mathcal{O}|])$.

1) Batch Pruning for ESQ: We consider the UB_1 function with a fixed $\epsilon = \epsilon^*$.

Lemma 1: Let $\epsilon^* \in (\exp(-\mathbf{E}[|\mathcal{Q}|]/3), \tau/2)$. Let S_L be the smaller root of the following equation, provided that $\epsilon^* > \exp(-S_L/3)$:[4]

$$x + \sqrt{-3x \ln \epsilon^*} = (\tau - 2\epsilon^*)(\mathbf{E}[|\mathcal{Q}|] - \sqrt{-2\mathbf{E}[|\mathcal{Q}|] \ln \epsilon^*})$$

S_U is the larger root of the following equation:

$$x - \sqrt{-2x \ln \epsilon^*} = \left(\mathbf{E}[|\mathcal{Q}|] + \sqrt{-3\mathbf{E}[|\mathcal{Q}|] \ln \epsilon^*}\right)/(\tau - 2\epsilon^*)$$

We can prune all p-sets whose expected sizes are not within $[S_L, S_U]$ for *ESQ*.

We omit the proof in the interest of space. The key to the establishment of these bounds is that UB_1 is monotonically increasing with u and monotonically decreasing with v. By considering two possible cases regarding the relative size of $\mathbf{E}[|\mathcal{Q}|]$ and $\mathbf{E}[|\mathcal{O}|]$, we can establish if $\mathbf{E}[|\mathcal{O}|]$ is below S_L or larger than S_U, the UB_1 function value with $\epsilon = \epsilon^*$ will be below τ, and hence can be pruned safely.

2) Batch Pruning for CSQ: We select ξ_1^* such that $\exp\left(\frac{-(\tau \cdot \mathbf{E}[|\mathcal{Q}|] - \xi_1^*)^2}{2\tau^2 \cdot \mathbf{E}[|\mathcal{Q}|]}\right) = minconf/2$. Let S_L be the smaller root of $\exp\left(\frac{-(\xi_1^* - x)^2}{3x}\right) = minconf/2$ with the constraints (i) $\tau\mathbf{E}[|\mathcal{Q}|] \geq \xi_1^*$, and (ii) $\xi_1^* < 2x$.

Similarly, we select ξ_2^* such that $\exp\left(\frac{-(\xi_2^* - \mathbf{E}[|\mathcal{Q}|])^2}{3\mathbf{E}[|\mathcal{Q}|]}\right) = minconf/2$. Here, $\mathbf{E}[|\mathcal{Q}|] \leq \xi_2^* \leq 2\mathbf{E}[|\mathcal{Q}|]$. Let S_U be the larger root of $\exp\left(\frac{-(\tau \cdot x - \xi_2^*)^2}{2\tau^2 \cdot x}\right) = minconf/2$.

Lemma 2: With the S_L and S_U computed as above, we can prune all p-sets whose expected sizes are not within $[S_L, S_U]$ for *CSQ*.

Line 1 of Algorithm 2 implements the batch pruning according to Lemma 1 or 2.

Astute readers may find that there are corner cases where there may be no appropriate value for the fixed parameter (ϵ^*, ξ_1^*, ξ_2^*). We will abandon batch pruning should these cases occur.

[4]The two roots for $x \pm \sqrt{ax} = b$ are $\frac{1}{2}(\pm\sqrt{a(a+4b)} + a + 2b)$.

VI. Approximate Solution

The computational cost for *ESQ* and *CSQ* can still be prohibitively high if each p-set has a large number of elements or when the pruning is not so effective. In this section, we devise approximate solutions to compute these queries approximately yet with strong probabilistic guarantees.

A. Approximate Algorithm for *ES*

We devise a sampling-based method for *ES*. The basic idea is to randomly sample $\lceil(\ln \frac{2}{\delta})/(2\epsilon^2)\rceil$ joint possible worlds (where ϵ and δ refer to the error threshold and confidence parameter respectively) and use the average of the similarity in the sampled possible worlds to approximate the expected similarity *ES*. We have the following guarantee of the approximate similarity computed by this method.

Theorem 6: For any $\epsilon > 0$ and $\delta \in (0, 1]$, let the similarity computed by the method mentioned above be $\widehat{ES}$, and let *ES* be the exact *ES* value. We have: $\mathbf{Pr}\left[\left|\widehat{ES} - ES\right| \leq \epsilon\right] \geq 1 - \delta$.

The time complexity of this method is $O(n \cdot \epsilon^{-2} \ln \delta^{-1})$ and space complexity is $O(n)$, assuming $n \geq m$. This is a dramatic improvement in both space and time complexities as compared with the exact algorithm.

B. Approximate Algorithm for *CS*

We also design a sampling-based algorithm for *CS*. We randomly generate G groups of possible worlds, and each group contains M pairs of possible worlds from $\mathcal{A}$ and $\mathcal{B}$. For any ϵ, δ, we set $G = 24 \cdot \lceil \ln \frac{1}{\delta} \rceil$, $M = \lceil 2\epsilon^{-2} \rceil$. In each group, we select the $(minconf \cdot M)$-th largest similarity value into an array sa. Finally, we select the median value from G entries in the sa array.

We have the following guarantee on the approximation quality of the algorithm output.

Theorem 7: For any ϵ, δ and $minconf$, let the similarity computed by the method mentioned above be $\widehat{CS}$, and let CS^- and CS^+ be the exact *CS* values with the minimum confidence $(minconf + \epsilon)$ and $(minconf - \epsilon)$ respectively, i.e, $CS^+ = CS(\mathcal{A}, \mathcal{B}, minconf - \epsilon)$, $CS^- = CS(\mathcal{A}, \mathcal{B}, minconf + \epsilon)$. We have: $\mathbf{Pr}\left[CS^- \leq \widehat{CS} \leq CS^+\right] \geq 1 - \delta$.

The space consumption is $O(\epsilon^{-2} + \ln \frac{1}{\delta})$, since we compute the similarity value group by group and each group contains $O(\epsilon^{-2})$ samples. Selecting of the k-largest value from an unsorted array of size n can be performed in $O(n)$ in a QuickSort-style algorithm. Therefore, the time complexity is $O(n \cdot \epsilon^{-2} \ln \delta^{-1})$, assuming $n \geq m$.

VII. Experiments

In this section, we report the results and analysis of the proposed probabilistic set model and similarity query processing algorithms. All programs were implemented in Java, and were conducted on a Window PC with an Intel Pentium IV 2.8GHz CPU and 4GB memory.

We use both synthetic and real-world datasets.

- **SYNα-U** and **SYNα-G**: These are the synthetic p-sets generated as follows: we form a p-set by randomly sampling

TABLE VI
DESCRIPTIVE STATISTICS

Dataset	DB Size	p-set Min/Max/Avg Size
pDBLP	5,000	27 / 708 / 204.9
pDeli	44,876	50 / 293,214 / 453.2

α elements (without replacement) from a large universe of elements; each element is associated with a probability value generated from one of the following two distributions:

- a uniform distribution within the range of $[\upsilon, 0.9]$ with a default υ value of 0.2.
- a Gaussian distribution $N(\mu, \sigma)$ capped to the range of $(0, 1]$. By default, $\mu = 0.8$ and $\sigma = 0.2$.

The resulting data sets are denoted as SYNα-U and SYNα-G, respectively.

- **pDBLP**: Automatic research profile construction is an essential step in applications such as expert finding, and is a nice feature to have for bibliography search engines. A basic profile is presented in a probabilistic way as $\langle p_1, p_2, \ldots, p_n \rangle$ where p_i quantifies the probability that a person is an expert in one of the n knowledge areas [4].
  While different methods to model and estimate the probabilities have been proposed [4], we use a fairly simple yet effective method based on topical terms used in authors' DBLP entries[5]. Specifically, we extract, tokenize, and preprocess the publication titles of an author to get a *raw profile*, which is a *multiset* of terms. Globally, we select 1,000 general terms as possible descriptions of research interests. We construct each author's probabilistic profile as follows: (i) We remove terms not in the globally selected 1,000 terms; (ii) For the remaining term e, we convert the frequency of the term $c(e)$ into its probability $p(e)$ by adapting the sigmoid function as $p(e) = \frac{2}{1+\exp(-c(e))} - 1$. Similar transformations have been used to convert the numerical values (such as SVM classifier's output) to probability values [28]. Finally, we select the 5,000 most prolific authors and their probabilistic profiles as our pDBLP dataset.
- **pDeli**: We also use the social bookmarking dataset which was crawled from the Del.icio.us web site during 2006 and 2007.[6] We obtain more than 23 million annotations from the original dataset by removing users and urls that occurred less than ten times. We then map each tag to a set of urls to which the tag was used as part of a user's annotation. We associate with each url a probability value, which reflects the possibility or plausibility that the tag is appropriate for the url, by adapting the sigmoid function as $p(u) = \frac{2}{1+\exp(-c(u))} - 1$, where $c(u)$ represents frequency of url u. The final dataset contains 44,876 tags.

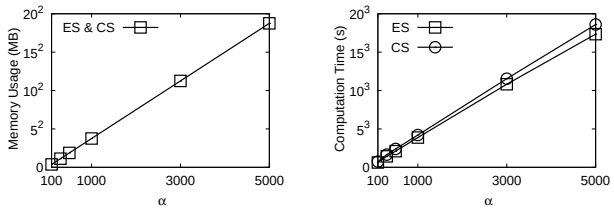
Some statistics of the real datasets are given in Table VI. We construct the queries as follows:

- For a p-set in SYNα-U (resp. SYNα-G), we generate a *controlled* query p-set by adding "noise" to the original p-set. We randomly remove γ percent of elements from the set, and then add the same amount of random elements from the universe. The generated noisy query p-set is denoted as SYNα-U-Nγ (resp. SYNα-G-Nγ). All experiment results on the SYN dataset are averaged over ten runs.

Unless otherwise specified, the default parameter settings are: $minconf = 0.5$ (for *CS*), $\tau = 0.5$, $\alpha = 1000$, $\gamma = 10\%$, $\epsilon = 0.06$, $\delta = 0.06$, $\upsilon = 0.2$, $\sigma = 0.2$, and $\mu = 0.8$.

We take the following measures: (i) **Memory Usage** is the total amount of memory used by an algorithm. (ii) **Computation Time** is the elapsed time to compute an *ES* or *CS* value using either exact or approximate method. (iii) **Query Time** is the query execution time per query for a collection of *ESQ* or *CSQ* queries. **Pruning time** is the time used by the pruning methods in Section V. (iv) **Candidate size** is the number of p-sets in the database that survive the pruning, and **result size** is the number of p-sets in the database that satisfy the query condition, i.e., the number of true positives. (v) **Pruning rate**. Given a query and a database containing N p-sets, let the candidate size be C, and let T be the number of p-sets in the database that does *not* satisfy the query condition. Pruning rate is defined as $\frac{N-C}{T}$. Intuitively, pruning rate reaches 100% when all the true negative results are pruned. (vi) **Average precision** is a common metric in IR to evaluate retrieval effectiveness [11]. It is the precision (i.e., the percentage of retrieved results that are relevant) at a fixed position k average over all the queries.

A. Computing Similarities Exactly



(a) Space consumption (b) Computation time

Fig. 1. Exact Similarity Computation

We first evaluate our exact algorithms to compute the *ES* or *CS* values between two p-sets. Figures 1(a)–1(b) show the computation time between SYN1000-U and SYN1000-U-N10%. Note that the time and space complexity is the same for other distributions. The exact solutions for *ES* and *CS* have similar space costs since they both need to maintain an H table in memory. We can see that the space cost increases quadratically with the increase of p-set size; time cost increases cubically with the increase of the p-set size; both fit well with the theoretical prediction.

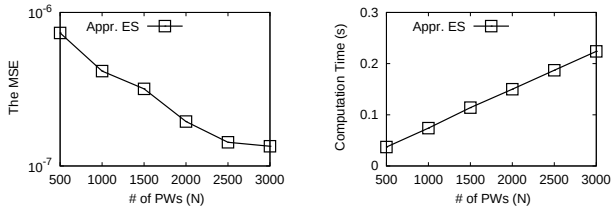
B. Computing Similarities Approximately

We evaluate our approximate similarity evaluation algorithms using SYN1000-G and SYN1000-G-N10%.

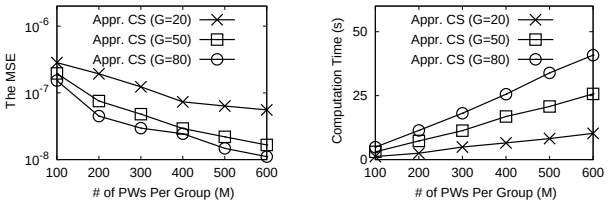
We first consider *ES* computation. We measure the Mean Squared Error (MSE) between the exact similarity values and values returned by our approximate methods. We plot

[5] http://dblp.uni-trier.de/xml/

[6] http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Research/DataSets/PINTSExperimentsDataSets

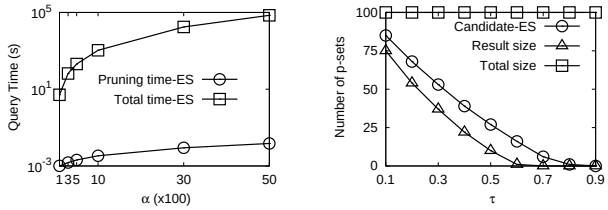


(a) *ES*, MSE (b) *ES*, Computation time

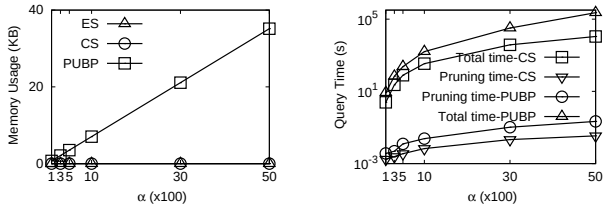


(c) *CS*, MSE (d) *CS*, Computation time

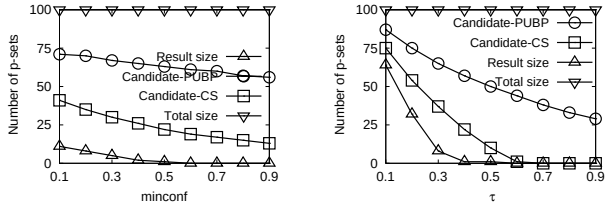
Fig. 2. Approximate Similarity Computation



(a) *ESQ*, Query Time (b) *ESQ*, Candidate Size



(c) *CSQ*, Memory Usage (d) *CSQ*, Query Time



(e) *CSQ*, Candidate Size (f) *CSQ*, Candidate Size

Fig. 3. Pruning Effectiveness and Efficiency on Synthetic Datasets

MSE with respect to the number of samples (i.e, possible worlds) in Figure 2(a). As expected, the error reduces exponentially with the number of samples. On the other hand, the computational cost grows linearly with the sample size, as shown in Figure 2(b).

Next we show the MSE and computation time of the approximate algorithm for *CS* in Figure 2(c) and 2(d). We fix the number of groups to be 20, 50, and 80, and then vary the number of possible worlds within each group. As we can see, the error reduces exponentially with more possible worlds sampled; the error also reduces with more groups, although with a diminishing return. The computation time grows linearly with the number of groups and the number of samples within each group.

C. Evaluating Pruning Efficiency on SYN

We report the performance of our pruning methods on synthetic datasets. We use SYN1000-G as the query and the database consists of 100 p-sets as SYN1000-G-Nγ ($\gamma = 1, 2, \ldots, 100$).

Figures 3(a)–3(b) show the results for *ESQ*. Figure 3(a) shows the total query time and pruning time. The pruning time increases linearly with α, i.e., the p-set size, as the pruning cost is linear in the p-set size. Even when the p-set size is 5,000, the pruning time is only 54.4 milliseconds. The query time includes both the pruning and verification time. It first increases rapidly when the p-set size increases from 100 to 1,000; When the p-set size is above 1,000, the query time starts to increase slowly. This is mainly because (i) The verification time using exact similarity evaluation is much larger than the time for pruning, as can be expected from their respective time complexities. It dominates the total time. (ii) The pruning rate becomes higher with the growth of the expected size of p-set, because both of our upper bounds become tighter with larger sets. When the p-set size is large enough, most of p-sets are pruned and hence the total time increases slowly.

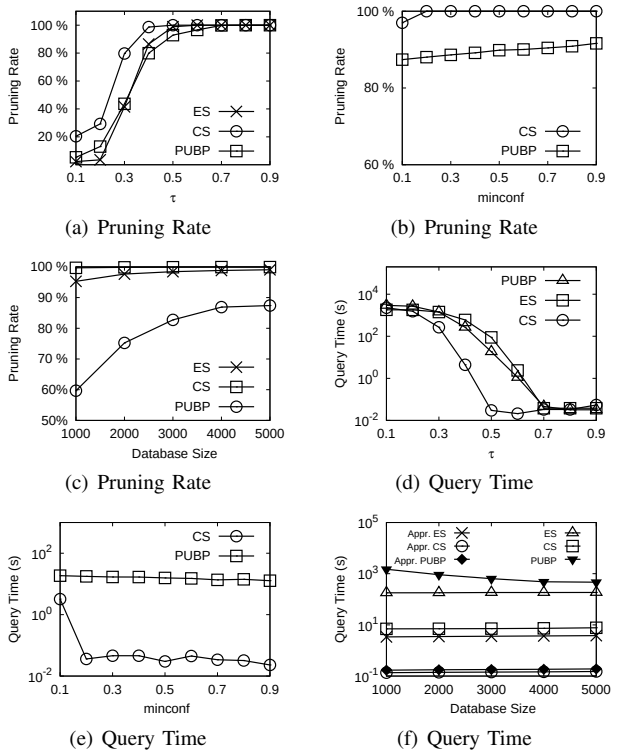
Figure 3(b) shows the candidate size with varying similarity threshold τ. When τ increases, the result size decreases accordingly. The candidate size follows closely with the result size in all the cases. This is because our pruning is based on upper bound estimation and hence will always over-estimate the similarity by definition. The difference becomes negligible for $\tau > 0.7$.

Figures 3(c)–3(f) show the pruning efficiency for *CSQ*. In particular, we adapted the PUBP-based pruning proposed in [27] to our *CSQ*. Note that we incorporated the precomputation optimization into PUBP, which reduces its time from $O(n^2)$ to $O(n)$. As analyzed in Section II-A, our method has a similar time complexity with PUBP. Here we will compare their computational costs and pruning rates in practice. Figures 3(c) and 3(d) show the space and time costs. Our method is immune to the growth of p-set, while PUBP needs to consume more. This is because our memory requirement is $O(1)$ while PUBP needs to load the precomputed cumulative probability vector into the memory, whose size is linear in α. In terms of query time, our method is much faster than PUBP-based method, although the pruning times of both methods are approximately the same. This is mainly because the number of candidate p-sets of our pruning is much less than that of PUBP-based pruning. We also show the candidate size of both pruning methods with respect to other parameters ($minconf$ and τ) in Figures 3(e)– 3(f). It is obvious that our pruning is far more effective than PUBP in almost all cases. In particular, our candidate size follows the result size closely, where PUBP's candidate size is largely insensitive to the change of result size. E.g., when $\tau \geq 0.4$, PUBP still returns about 60% of the data as candidates even if the result set is empty.

TABLE VII
SAMPLE QUERY RESULTS ON PDBLP

Query Author	Top-3 Similar Authors
Hanan Samet	Thomas Seidl, Walid G. Aref, Pavel Zezula
Jeffrey D. Ullman	Leonid Libkin, Yehoshua Sagiv, Richard J. Lipton
Michael I. Jordan	Zoubin Ghahramani, Eric P. Xing, John Shawe-Taylor

D. Performance on the pDBLP Dataset



(a) Pruning Rate (b) Pruning Rate

(c) Pruning Rate (d) Query Time

(e) Query Time (f) Query Time

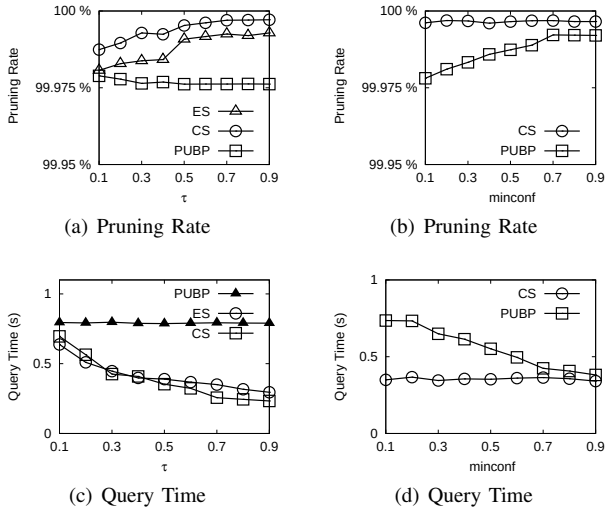
Fig. 4. Performance on the pDBLP dataset

We test our algorithms' performance on the pDBLP dataset. A sample of the queries and their top-3 most similar authors based on *ES* similarity are listed in Table VII. As we can see, the top-3 results are indeed researchers with closely matching research interest with the query author. For example, the first three rows contains well-known authors in the field of spatial database, database theory, and machine learning.

Figures 4(a)–4(c) show the pruning rates with respect to various parameter settings for both types of queries if applicable. Note that PUBP pruning is only applicable to *CSQ* and hence should be compared with the *CSQ* curves. We can see that our pruning method outperforms PUBP substantially under all settings. For example, our pruning rate is close to 100% for $\tau \geq 0.4$, while PUBP's rate is only close to 100% when $\tau \geq 0.7$. The pruning rate of our method remains above 95% irrespective of the database size, while PUBP's rate increases with the database size. For both types of queries, our pruning rates increases with τ and DB sizes, and remain stable with $minconf$.

Figures 4(d)–4(f) show the query times with respect to various parameter settings. In Figure 4(d), we can see that for *CSQ*, the query times between our method and PUBP method are close when τ is small; the query time of our method drops quickly with the increase of τ, while PUBP's time decreases moderately; both query times reach their minimum and remain stable after τ is large enough. The time difference can be explained by the difference in pruning rates of the two methods (and hence the candidate size). Since verification cost is relatively more expensive than pruning cost, PUBP results in larger candidate set and this increases their time substantially. Figure 4(e) shows that the query time decreases when $minconf$ increases, with our query time much faster than PUBP's. Figure 4(f) illustrates the scalability and trade-offs among algorithms using either exact or approximate similarity calculation for verification. It can be seen that the overall query time hardly changes with the increase of the database size, thanks to relatively stable candidate size due to our pruning. The time cost using exact algorithm for verification is typically several orders of magnitudes higher than their counterparts using approximate verification.

E. Performance on the pDeli Dataset



(a) Pruning Rate (b) Pruning Rate

(c) Query Time (d) Query Time

Fig. 5. Performance on the pDeli dataset

We also test our algorithms' performance on the pDeli dataset that has very different characteristics from the pDBLP dataset: it is larger, more skewed (See Table VI), and more sparse, since the number of unique elements (i.e., urls) is very large. Due to the large size of the p-sets, we use approximate methods for verification for all algorithms.

We select the url p-sets of twenty tags as queries to evaluate the effectiveness of *ESQ* and *CSQ*. We generate the ground-truth by asking five markers to label returned tags as relevant or not and tags that are labeled relevant by at least three markers are deemed as ground truth. Table VIII gives AP@k on pDeli.

We can see that both *ESQ* and *CSQ* achieve similarly good results in terms of AP@k; as the top-10 query results of *ESQ* and *CSQ* only contain few unrelevant tags, and even at 30-th position, the average precisions of both queries are above

TABLE VIII
AVERAGE PRECISION@k ON PDELI

AP@k	5	10	15	20	25	30
ES	0.7000	0.6675	0.6250	0.5875	0.5825	0.5500
CS	0.7000	0.6785	0.6280	0.5825	0.575	0.5500

0.55. Note that due to the anonymization of the urls and lack of a complete context of tags, we expect our ground truth to be quite conservative and may miss some relevant tags. We can also see that the AP values between the two queries vary, with CS achieving a higher AP value than ES at position 10, but achieving a lower AP value than ES at position 20.

We plot pruning rates and query time for the algorithms with respect to different parameter settings in Figures 5(a)–5(d). Similar trends as those in the pDBLP dataset can be found too. Note that pruning rates for all algorithms are high mainly due to the sparseness of the dataset – only few tag pairs have common URLs. Nevertheless, our pruning methods outperforms PUBP, and improves nicely with τ and remains stable with $minconf$. This also translates to the superiority of our query time over PUBP's under all parameter settings.

VIII. CONCLUSION

The importance of probabilistic similarity queries has been recognized in many application areas, such as spatial databases and data integration. In this paper, we present a comprehensive study of similarity queries for probabilistic sets. We present a simple and computationally tractable model for uncertain sets where each element is associated with its existential probability. We then study two kinds of similarity queries based on the expected and confidence-based similarity measures. Both exact and approximate algorithms are developed to compute these values. In addition, we develop novel pruning techniques based on upper bound estimation. Both the theoretical analysis and our empirical evaluation demonstrate the effectiveness and efficiency of the proposed methods.

Acknowledgement. Ming Gao is supported by the 973 program of China (No. 2012CB316203). Cheqing Jin is supported by NSFC (No. 60933001). Wei Wang is supported by ARC Discovery Projects DP130103401 and DP130103405. Xuemin Lin is supported by NSFC 61021004, ARC Discovery Projects DP120104168, DP110102937 and DP0987557. Aoying Zhou is supported by NSFC (No. 60925008). We also thank anonymous reviewers for their insightful comments and Xiangnan He and Jie Chen for their efforts on the experiment.

REFERENCES

[1] C. C. Aggarwal. Managing and mining uncertain data. *Springer*, 2009.
[2] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. U. Nabar, T. Sugihara, and J. Widom. Trio: A system for data, uncertainty, and lineage. In *VLDB*, pages 1151–1154, 2006.
[3] L. Antova, C. Koch, and D. Olteanu. From complete to incomplete information and back. In *ACM SIGMOD*, pages 713–724, 2007.
[4] K. Balog and M. de Rijke. Determining expert profiles (with an application to expert finding). In *IJCAI*, pages 2657–2662, 2007.
[5] Z. Barutçuoglu, R. E. Schapire, and O. G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7), 2006.
[6] C. Beecks, A. M. Ivanescu, S. Kirchhoff, and T. Seidl. Modeling multimedia contents through probabilistic feature signatures. In *ACM Multimedia*, pages 1433–1436, 2011.
[7] C. Beecks, M. S. Uysal, and T. Seidl. A comparative study of similarity measures for content-based multimedia retrieval. In *ICME*, 2010.
[8] T. Bernecker, H.-P. Kriegel, M. Renz, F. Verhein, and A. Züfle. Probabilistic frequent itemset mining in uncertain databases. In *ACM SIGKDD*, pages 119–128, 2009.
[9] G. Beskales, M. A. Soliman, and I. F. Ilyas. Efficient search for the topk probable nearest neighbors in uncertain databases. *Proceedings of VLDB*, 1(1):326–339, 2008.
[10] A. R. Bharambe, M. Agrawal, and S. Seshan. Mercury: supporting scalable multi-attribute range queries. In *ACM SIGCOMM*, 2004.
[11] C. Buckley and E. M. Voorhees. Evaluating evaluation measure stability. In *SIGIR*, pages 33–40, 2000.
[12] M. A. Cheema, X. Lin, W. Wang, W. Zhang, and J. Pei. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Trans. Knowl. Data Eng.*, 22(4):550–564, 2010.
[13] G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data and expected ranks. In *ICDE*, 2009.
[14] N. N. Dalvi and D. Suciu. Efficient query evaluation on probabilistic databases. *VLDB J.*, 16(4):523–544, 2007.
[15] N. N. Dalvi and D. Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12, 2007.
[16] M. Eirinaki and M. Vazirgiannis. Web mining for web personalization. *ACM Trans. Internet Techn.*, 3(1):1–27, 2003.
[17] T. Ge and Z. Li. Approximate substring matching over uncertain strings. *Proceedings of VLDB*, 4(11):772–782, 2011.
[18] M. Hadjieleftheriou and D. Srivastava. Weighted set-based string similarity. *IEEE Data Eng. Bull.*, 33(1):25–36, 2010.
[19] M. Hadjieleftheriou and D. Srivastava. Approximate string processing. *Foundations and Trends in Databases*, 2(4):267–402, 2011.
[20] M. Hua, J. Pei, W. Zhang, and X. Lin. Ranking queries on uncertain data: a probabilistic threshold approach. In *ACM SIGMOD*, 2008.
[21] I. F. Ilyas, W. G. Aref, and A. K. Elmagarmid. Supporting top-k join queries in relational databases. *VLDB J.*, 13(3):207–221, 2004.
[22] J. Jestes, G. Cormode, F. Li, and K. Yi. Semantics of ranking queries for probabilistic data. *IEEE Trans. Knowl. Data Eng.*, 23(12), 2011.
[23] J. Jestes, F. Li, Z. Yan, and K. Yi. Probabilistic string similarity joins. In *ACM SIGMOD*, pages 327–338, 2010.
[24] C. Jin, K. Yi, L. Chen, J. X. Yu, and X. Lin. Sliding-window top-k queries on uncertain streams. *Proceedings of VLDB*, 1(1), 2008.
[25] H.-P. Kriegel, P. Kunath, M. Pfeifle, and M. Renz. Probabilistic similarity join on uncertain data. In *DASFAA*, pages 295–309, 2006.
[26] A. Kumar and C. Ré. Probabilistic management of ocr data using an rdbms. *Proceedings of VLDB*, 5(4):322–333, 2012.
[27] X. Lian and L. Chen. Set similarity join on probabilistic data. *Proceedings of VLDB*, 3(1):650–659, 2010.
[28] H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3), 2007.
[29] V. Ljosa and A. K. Singh. Top-k spatial joins of probabilistic objects. In *ICDE*, pages 566–575, 2008.
[30] J. Pei, B. Jiang, X. Lin, and Y. Yuan. Probabilistic skylines on uncertain data. In *VLDB*, pages 15–26, 2007.
[31] C. Ré and D. Suciu. The trichotomy of having queries on a probabilistic database. *VLDB J.*, 18(5):1091–1116, 2009.
[32] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Morgan & Claypool Publishers, 2011.
[33] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *WWW*, pages 675–684, 2004.
[34] G. Tsoumakas, I. Katakis, and I. P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*. 2010.
[35] W. Wang. Similarity joins as stronger metric operations. *SIGSPATIAL Special*, 2(2):24–27, 2010.
[36] J. Xu, Z. Zhang, A. K. H. Tung, and G. Yu. Efficient and effective similarity search over probabilistic data based on earth mover's distance. *Proceedings of VLDB*, 3(1):758–769, 2010.
[37] B. Yang, J.-T. Sun, T. Wang, and Z. Chen. Effective multi-label active learning for text classification. In *ACM SIGKDD*, 2009.
[38] Q. Zhang, F. Li, and K. Yi. Finding frequent items in probabilistic data. In *SIGMOD Conference*, pages 819–832, 2008.
[39] X. Zhang, K. Chen, L. Shou, G. Chen, Y. Gao, and K.-L. Tan. Efficient processing of probabilistic set-containment queries on uncertain set-valued data. *Inf. Sci.*, 196:97–117, 2012.