A semi-supervised framework of clustering selection for de-duplication

Shrinu Kushagra, Hemant Saxena, Ihab F. Ilyas, Shai Ben-David
University of Waterloo
{skushagr,h2saxena,ilyas,shai}@uwaterloo.ca

***Abstract*—We view data de-duplication as a clustering problem. Recently, [1] introduced a framework called restricted correlation clustering (RCC) to model de-duplication problems. Given a set X, an unknown target clustering C^* of X and a class $\mathcal{F}$ of clusterings of X, the goal is to find a clustering C from the set $\mathcal{F}$ which minimizes the correlation loss. The clustering algorithm is allowed to interact with a domain expert by asking whether a pair of records correspond to the same entity or not. Main drawback of the algorithm developed by [1] is that the pre-processing step had a time complexity of $\Theta(|X|^2)$ (where X is the input set). In this paper, we make the following contributions. We develop a sampling procedure (based on locality sensitive hashing) which requires a linear pre-processing time $O(|X|)$. We prove that our sampling procedure can estimate the correlation loss of all clusterings in $\mathcal{F}$ using only a small number of labelled examples. In fact, the number of labelled examples is independent of $|X|$ and depends only on the complexity of the class $\mathcal{F}$. Further we show that to sample one pair, with high probability our procedure makes a constant number of queries to the domain expert. We then perform an extensive empirical evaluation of our approach which shows the efficiency of our method.**

***Index Terms*—De-duplication, clustering, semi-supervised, locality sensitive hashing, correlation clustering**

I. Introduction

Modern businesses generate a tremendous amount of datasets. These datasets are then used to make many critical decisions. Therefore, ensuring the quality of these datasets becomes extremely important. As the data accumulates from multiple sources over time, many errors creep into the data. For example, many records end up having duplicate entries. Record de-duplication is a central task in managing large scale databases. The goal is to detect records in a database that correspond to the same real word entity.

The problem of data de-duplication can be viewed as a clustering task. Here, the goal is to put records corresponding to the same physical entity in the same cluster while separating the records corresponding to different entities into different clusters. Clustering for de-duplication has many characteristics which are different from standard clustering problems. Many popular clustering algorithms like k-means or k-median receive as input the value k, that is the number of clusters to output. This information is unknown in de-duplication applications. In any dataset, the number of different-cluster pairs (i.e. different entity pairs) is order of magnitude greater than the number of positive or same-cluster pairs (i.e. same entity pairs). Hence, common machine learning tools of classification prediction (learning a binary classifier over the set of pairs of instances) do not automatically transfer to this domain as the dataset is heavily skewed towards the negative pairs.

The framework of correlation clustering is very natural for modelling the problem of data de-duplication [2]. Here, de-duplication is viewed as an optimization problem over graphs. More formally, given a dataset X and a complete graph G over the set. The edges of the graph are labelled 0 or 1. An edge label of zero indicates that the corresponding vertices have been deemed to be in different cluster while an edge label of one indicates that the corresponding vertices should be in the same cluster. The motivation for edge labelling is the following. Often the practitioner can design a pairwise similarity function over the pairs of points. The pairs whose similarity is above a certain threshold are deemed as positive (or same-cluster) and the remaining pairs are deemed to be negative (or different-cluster). Sometimes, the similarity metric is also learned from training data.

Given the graph, the goal of correlation clustering is to find a clustering of the dataset which correlates 'as much as possible' with the given edges. In other words, find a clustering which minimizes the correlation loss w.r.t the given edges. Correlation loss is defined as the sum of edges labeled zero within a cluster plus the number of edges labeled one across different clusters. However, solving this optimization problem is NP-Hard [2].

Recently, [1] introduced the framework of restricted correlation clustering (RCC) to model de-duplication problems. This framework has two important differences from the standard correlation clustering formulation. Firstly, the goal is to find a clustering from a given class $\mathcal{F}$ of clusterings. In correlation clustering, the optimization problem was to find a clustering over the set of all possible clusterings of the dataset. In the framework of RCC, the optimization problem is restricted over a given class $\mathcal{F}$ of clusterings. Secondly, the goal of correlation clustering was to find a clustering which correlates as much as possible with the given edges. In this framework, the goal is to find a clustering which correlates as much as possible with the unknown target clustering. Formally, Given a set X, an unknown target clustering C^* of X and a class $\mathcal{F}$ of clusterings of X, the goal is to find a clustering C from the set $\mathcal{F}$ which minimizes the correlation loss w.r.t the unknown target clustering; the sum of different-cluster edges (w.r.t C^*) within a C-cluster plus the sum of same-cluster edges across different C-clusters.

This framework is highly suitable for de-duplication applications. The target clustering C^* should be understood as the ground truth clustering. That is, C^* is the clustering where only records corresponding to same entity are in the same cluster. The more interesting aspect of the framework is introduction of the class $\mathcal{F}$. For many real-world applications, the optimal solution to the correlation clustering is the desired solution. However, the negative NP-Hardness results make it in feasible to find that solution. In such scenarios, a reasonable objective could be the following. Run a variety of efficient clustering algorithms available and obtain different clusterings of the dataset. Then choose the clustering which is 'closest to' (defined formally in Section II) the ground truth clustering. In Section III we provide more detailed examples and scenarios where the framework of restricted correlation clustering could be applicable.

The framework of RCC tries to find a clustering which is closest to the ground truth clustering. In the absence of any information about the ground truth, one can not hope to solve the problem. The framework provides 'indirect' information about the ground truth in the following two ways. Firstly, we allow the algorithm to make *same-cluster* queries to an oracle. The oracle should be understood as a human expert who has knowledge about the ground truth clustering. Given two records from a dataset the expert answers yes or no depending upon whether the two records refer to the same entity or not. Secondly, the user designed or learned similarity (or distance) function provides indirect information about the ground truth. Pairs of records whose distance is below a certain threshold are likely to belong to the same cluster (correspond to the same entity).

To solve the RCC problem, we adopt the following strategy. We get a small set of labelled samples of pairs of points with the help of our oracle. Our sampling procedure uses two sub-procedures. One for sampling negative or different-cluster pairs and one for positive or same-cluster pairs. We then choose the clustering which makes the smallest number of 'mistakes' on the sample. We show that this strategy is theoretically sound. Our method only finds a clustering which *performs best* on the sampled pairs of points. But we prove that this sampling based approach is guaranteed to find a clustering which is 'close' to the best clustering in $\mathcal{F}$. Informally, the best performer on the sample is guaranteed to be close to the best true performer or the optimal solution of the RCC problem. A more formal description of the results is deferred to Section V.

Another important contribution of this work is the sampling sub-procedure for positive pairs. In many datasets, the similarity function is such that it supports locality sensitive hashing (LSH). We use this fact to obtain a procedure which requires only linear pre-processing time and can sample according to the distribution P^+ (the uniform distribution over the same-cluster pairs). We also prove that the number of queries to the same-cluster oracle (to sample one positive pair) is upper bounded by a small constant and is independent of the size of the dataset. We carry extensive experimental evaluation of our framework on a diverse class of clustering algorithms and across multiple real world datasets.

A. Related Work

The problem of evaluating clustering algorithms for the de-duplication problem was considered by [3]. They carried out extensive experimental evaluation of different graph-based clustering algorithms on a simulated dataset of strings. They showed that these algorithms perform "extremely well in terms of both accuracy and scalability." Our framework differs from theirs in many crucial ways. To evaluate a clustering algorithm, we do not need access to the complete ground truth clustering. As we prove in Section V, we can find clustering closest to the best clustering even when given access to a small number of oracle answers. Our framework is generic and can work for any class of clustering algorithms, be it graph-based (as was considered by [3]) or hierarchical clustering or any other de-duplication heuristic, we show this in Section VI.

Another class of work related to ours is related to the problem of correlation clustering [2] and its many variants. For example, [4] considered the problem of weighted correlation clustering. In this framework, the edge labels are allowed to be any real number in $[0, 1]$ instead of just zero or one. They showed that this problem is NP-Hard and gave a $O(\log |X|)$ approximation algorithm for the same. Besides correlation clustering, some application oriented works have also modelled de-duplication as a clustering problem. For example, [5] assumed that the set of duplicate records are transitive. Finding the clustering of the given graph G is now equivalent to computing the connected components of G.

Many application oriented works have also tried to address the problem of data de-duplication. Many works have focused on designing the right metric (or similarity measure) for the given dataset. Once this measure is defined, pairs of points whose distance are below a certain threshold (or whose similarity is above a certain threhhold) are deemed to be duplicates (or belonging to the same cluster). For example, to capture duplicates in the data generated due to typographical errors (like spelling mistakes), edit distance is used [6]. Jaro distance [7] is another measure which tries to capture typographical errors. Phonetic-based similarity measures tries to capture words which are similar sounding. Other measures try to capture the similarity in numerical data. A nice overview of such techniques can be found in [8]. While hand-designing a similarity measure for the given domain is quite popular. Some works also try to learn this function from labelled examples. For example, [9] and [10] use supervised learning techniques (like SVM) to try to learn the weights in the distance function.

Another theme in our work is the notion of human supervision for the clustering task. Many works have tried to introduce the supervision into the clustering problem. For example, [11], [12] and [13] introduced the concept of *link/don't link* constraints. Here, besides the usual input the clustering algorithm also receives a set of pairs of points which should belong to the same cluster and a set of pairs of points which should not belong to the same cluster. The algorithm then finds

a clustering which respects these constraints. Continuing this line of work, [14] introduced an interactive version of these constraints called *same-cluster queries*. Here, the clustering algorithm interacts with an oracle by asking whether two points belong to the same or different cluster. The oracle responds by answering 'yes' or 'no' depending on whether the two points belonged to the same clustering according to some ground truth clustering.

II. Preliminaries

Given X, a clustering C of the set X partitions it into k disjoint subsets or clusters. The clustering C can also be viewed as a $\{0,1\}$-function over the domain $X^{[2]} := \{(x_1,x_2) : x_1 \neq x_2\}$. Here, $C(x_1,x_2) = 1$ iff x_1, x_2 belong to the same cluster according to C.

We allow a clustering algorithm to make queries to a human oracle in the following way.

Definition 1 (Same-cluster oracle [14])**.** *Given a set X and an unknown target clustering C^*. A same-cluster C^*-oracle receives a pair $(x_1,x_2) \in X^{[2]}$ as input and outputs 1 if and only if x_1, x_2 belong to the same cluster according to C^*.*

From the perspective of de-duplication, a same-cluster oracle receives two records x_1 and x_2. The oracle returns 1 if x_1 and x_2 correspond to the same real-world entity. Otherwise, the oracle responds 0.

Definition 2 (Correlation loss [2])**.** *Given graph $G = (X,E)$ where X is the set of vertices (the given dataset to be clustered) and E is the set of edges. The correlation loss of a clustering C w.r.t the edges E is defined as*

$$\begin{aligned} corrL_E(C) &= corrN_E(C) + corrP_E(C), \text{ where} \\ corrN_E(C) &= |\{(x,y) : C(x,y) = 1 \text{ and } E(x,y) = 0\}|, \\ corrP_E(C) &= |\{(x,y) : C(x,y) = 0 \text{ and } E(x,y) = 1\}| \end{aligned} \tag{1}$$

A weighted version of the loss function places weights of w_1 and w_2 on the two terms and is defined as

$$corrL_E^{w_1,w_2}(C) = w_1\, corrN_E(C) + w_2\, corrP_E(C) \tag{2}$$

The goal of correlation clustering is to find a clustering which minimizes the (weighted) correlation loss. For our purposes, it is more relevant to consider a loss function which takes values in the range $[0,1]$. Also, we consider the correlation loss w.r.t a target clustering C^* rather than the edges E.

Definition 3 (Normalized correlation loss [1])**.** *Given domain X and a target clustering C^*. The loss of a clustering C w.r.t the target C^* is defined as*

$$\begin{aligned} L_{C^*}(C) = \;& \mu\; L_{P^+}(C) \;+\; (1-\mu)\; L_{P^-}(C), \text{ where} \\ L_{P^+}(C) = \;& \mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x,y) = 0\big], \\ L_{P^-}(C) = \;& \mathop{\mathbf{P}}_{(x,y)\sim P^-}\big[C(x,y) = 1\big] \end{aligned} \tag{3}$$

where P^+ is the uniform distribution over $X_+^{[2]} = \{(x,y) : C^*(x,y) = 1\}$ and P^- is the uniform distribution over $X_-^{[2]} = \{(x,y) : C^*(x,y) = 0\}$.

The normalized correlation loss measures two quantities for the clustering C. The first is the fraction of the true positive pairs that C gets wrong (or loss over the positive pairs). The second is the fraction of true negative pairs that C gets wrong (or the loss over the negative pairs). It then obtains a weighted sum of the two losses.

Lets observe the relation between Defns. 2 and 3. Define $\gamma_0 := \mathbf{P}[C^*(x,y) = 1]$, that is the probability of true positive (or same-cluster pairs) in the dataset. Using the notation of Defn. 2, we see that $corrP_{C^*}(C) = \gamma_0 |X^{[2]}| L_{P^+}(C)$ and $corrN_{C^*}(C) = (1-\gamma_0)|X^{[2]}| L_{P^-}(C)$. Normalising by $|X^{[2]}|$ and choosing $\mu = \frac{w_2\gamma_0}{w_1(1-\gamma_0)+w_2\gamma_0}$ gives us the normalized version of the loss function.

Definition 4 (Informative metric [1])**.** *Given a metric space (X,d), a target clustering C^* and a parameter λ. We say that the metric d is (α,β)-informative w.r.t C^* and λ if*

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2}\big[d(x,y) > \lambda \mid C^*(x,y) = 1\big] \;\leq\; \alpha \tag{4}$$

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2}\big[C^*(x,y) = 1 \mid d(x,y) \leq \lambda\big] \;\geq\; \beta \tag{5}$$

Here U^2 is the uniform distribution over $X^{[2]}$.

In deduplication applications, often the distance function is such that pairs with distance within a certain threshold are likely to be in the same cluster. The definition of an informative metric formalizes this intuition. It says that most of the true positive pairs have a distance of atmost λ between them. Also, amongst all pairs with distance $\leq \lambda$, atleast a β fraction of them belong to the same cluster.

Definition 5 (γ-skewed)**.** *Given X and a target clustering C^*. We say that X is γ-skewed w.r.t C^* if*

$$\mathop{\mathbf{P}}_{(x,y)\sim U^2}\big[\; C^*(x,y) = 1\big] \;\leq\; \gamma$$

In de-duplication applications, most of the pairs are negative (or belong to different clusters). The above definition states this property formally. We are now ready to introduce the framework of *restricted correlation clustering*.

III. Restricted Correlation Clustering (RCC)

We know that finding the clustering which minimizes the correlation loss is NP-Hard. Moreover, it is NP-Hard even when we are allowed access to a same-cluster oracle [15].

Observe that the requirement of correlation clustering is very demanding. The algorithm is required to find a clustering over the set of all possible clusterings of the domain X. In the restricted framework, we change the goalpost slightly. The algorithm is now required to find a clustering C from a finite class $\mathcal{F}$ (of clusterings of X).

Definition 6 (Restricted correlation clustering (RCC))**.** *Given a clustering instance (X,d), an unknown target clustering*

C^* and weighting parameter μ. Given a finite class $\mathcal{F}$ of clusterings of the set X. Find $C \in \mathcal{F}$ such that

$$\hat{C} = \arg\min_{C \in \mathcal{F}} \; L_{C^*}(C) \tag{6}$$

A. Relation to practical applications

Consider the following scenario from the practitioner's point of view. The practitioner wants to implement correlation clustering. However, he/she knows that the problem is NP-Hard. The practitioner has prior knowledge that one of the many hierarchical clustering algorithms (like single-linkage or max-linkage or average-linkage or complete-linkage) is suitable for his/her dataset[1]. A hierarchical clustering algorithm outputs a clustering tree. Every pruning of the tree is a clustering of the original dataset. He/she would now like to know which amongst these clustering algorithms is suitable for his task. After having fixed the algorithm, the practitioner would then like to know which amongst these many prunings he/she should chose.

The framework of restricted correlation clustering is applicable in such scenarios. When $\mathcal{F} = \{T\}$ where T is a hierarchical clustering of X, the goal of RCC is to find the pruning from the tree T which has minimum normalized correlation loss. When $\mathcal{F} = \{T_1, \ldots, T_s\}$ where each T_i is a hierarchical clustering of X. Then the goal of RCC is to find a pruning with minimum loss amongst the prunings of all the s trees. Note that finding the pruning of the tree is the same as choosing the stopping point criteria when running linkage-based algorithms. Hence, the framework can help us choose the right stopping point for a particular hierarchical clustering algorithm.

If $\mathcal{F} = \{C_1, \ldots, C_s\}$ where each C_i is a clustering of the set X then the goal is to find a clustering with minimum loss. Note that $\mathcal{F}$ can be any of the examples as defined above or a union of these or some other finite class.

B. Solution strategy

In the RCC framework, we wish to minimize the loss which depends on the unknown target clustering C^*. However, in the absence of any information about C^*, there is no hope to find a clustering that minimizes L_{C^*}. Hence, to solve the RCC problem we allow the clustering (or learning) algorithm to make queries to a C^*-oracle.

Our goal is to calculate quantities $L_{P^+}(C)$ and $L_{P^-}(C)$ (Defn. 3) for each of the clusterings $C \in \mathcal{F}$ and then choose the clustering with minimum loss. To calculate both these quantities exactly, for each pair of points in our dataset, we would need to know whether they belong to the same-cluster or different-cluster. In other words, we would need access to the complete ground truth clustering C^*. Thus, instead of calculating these two quantities exactly we want to estimate them from a small sample, sampled according to the distributions P^+ and P^-.

One strategy to estimate $L_{P^+}(C)$ (and L_{P^-}) could be the following. Sample a set S_+ (and S_-) of pairs using the distribution P^+ (and P^-). Compute the fraction of mistakes made by each clustering C on S_+ (and S_-). Using the standard results from vc-dimension theory (Thm. 18), it is known that using this procedure we can estimate L_{P^+} for each of the clusterings $C \in \mathcal{F}$. Similarly, we could also estimate L_{P^-}. Using the two estimates, we could then estimate the loss L_{C^*} for each of the clusterings in our class and choose the clustering which has the smallest loss.

The main problem in this approach is that the distributions P^+ and P^- are unknown (as the target clustering C^* is not known). In Section IV, we discuss two approaches which (approximately) sample according to these distributions. Then in Section V, we show how these sampling procedures can be used to estimate L_{C^*} for all the clusterings in our class $\mathcal{F}$.

IV. Sampling for Restricted Correlation Clustering

We first describe the procedure $\mathcal{P}_0$ which samples according to P^-. Then we describe the procedure $\mathcal{P}_1$ which samples approximately according to the distribution P^+. The procedure $\mathcal{P}_0$ is the same as described in [1]. However, we state the algorithm here for completeness.

Algorithm 1: Procedure $\mathcal{P}_0$ for negative pairs

Input: A set X and a C^*-oracle.
Output: (x, y) such that $C^*(x, y) = 0$

```
while TRUE do
    Sample (x, y) using U^2
    if C*(x, y) = 0 then
        Output (x, y)
    end
end
```

The procedure samples a pair uniformly at random. Then using the oracle it checks if the sampled pair is negative and terminates if such a pair is found. If not then the process is repeated again. It is an easy exercise (proof in appendix Lemma 16) to see that the procedure $\mathcal{P}_0$ samples a pair according to the distribution P^-. From the algorithm description it is clear that to sample one negative pair we might need to make more than one query to the C^*-oracle. However, since the number of negative pairs is much greater than the number of positive pairs (γ-skewed) the number of 'wasted' queries to the oracle is small. The proof of this result is in the appendix (Lemma 17).

A. Sampling positive pairs

We now discuss our procedure $\mathcal{P}_1$ which approximates the distribution P^+. We show that the procedure samples according to a distribution T which has the following property. The loss $L_{\mathcal{T}}$ (Defn. 3) and the loss L_{P^+} for any clustering are close to one another. Hence, estimating the loss of a clustering w.r.t the distribution $\mathcal{T}$ also gives an estimate of the loss of that clustering w.r.t P^+. Now, we discuss the details of the sampling procedure.

[1]A nice overview of hierarchical clustering techniques can be found in [16]

Our metric d is (α, β)-informative w.r.t the target clustering C^*. That is, amongst all pairs with distance λ atleast β-fraction are positive. The sampling strategy of [1] was the following. Construct a set $K = \{(x, y) : d(x, y) \leq \lambda\}$ and then sample uniformly from the set K till a positive sample is found. Since most of the positive pairs have distance $\leq \lambda$, this sampling procedure approximates P^+ (the uniform distribution over the set of true positives). However, constructing the set K requires $\Theta(|X|^2)$ time. This makes the sampling procedure impractical for many situations. In this section, we will use techniques from locality sensitive hashing (LSH) combined with rejective sampling to develop a sampling procedure $\mathcal{P}_1$. We will show that $\mathcal{P}_1$ needs only linear pre-processing time (to build the hash maps) and outputs a positive pair sampled approximately according to P^+.

Locality Sensitive Hashing (LSH)

Before we describe our technique, we introduce some relevant notation. A hash function $h : X \to \mathbf{N}$ maps the set X onto the set of natural numbers. Thus, a hashing function partitions the input of size n into $m \leq n$ different buckets (or blocks) $B_1, \ldots, B_m$ where each $B_i = \{x : h(x) = b_i\}$ for some b_i. Given (X, d), a Locality Sensitive Hashing (LSH) scheme w.r.t the distance metric d (or a similarity metric) aims to partition X into buckets such that 'similar' items map to the same bucket with high probability and 'dissimilar' items end up in different buckets with high probability. For example, MinHash scheme w.r.t Jaccard similarity measure [17], [18] is a common LSH-based hashing scheme. Another example is SimHash scheme w.r.t hamming similarity measure [19].

Definition 7 (LSH-based hashing algorithm)**.** *Given a set (X, d) and parameter s. An LSH-based hashing algorithm (or scheme) $\mathcal{A}$ outputs s different partitions $P_1, \ldots, P_s$ of X. Denote $P_i = \{B_{i1}, \ldots, B_{in_i}\}$. We say that $\mathcal{A}$ is (ϵ, ϵ')-tight w.r.t d and λ, λ' if*

- *If $d(x, y) \leq \lambda$ then $\mathbf{P}[b(x, y) = 1] > 1 - \epsilon$*
- *If $d(x, y) > \lambda'$ then $\mathbf{P}[b(x, y) = 1] < \epsilon'$*

where $b(x, y) = 1$ if and only if x, y are together in atleast one of the blocks B_{ij}.

Infact, we show that by choosing s (and other parameters) appropriately, we can construct LSH schemes which are $(\epsilon, \epsilon' = s\ln(1 + \epsilon))$-tight w.r.t λ and $\lambda' = 2\lambda\ln(1 + 1/\epsilon)$. Thus, for simplicity of notation, we say that $\mathcal{A}$ is ϵ-tight w.r.t λ to mean that it is (ϵ, ϵ')-tight w.r.t λ, λ' as chosen above.

Throughout the remainder of this section, we will assume that the hashing scheme satisfies ϵ-tightness. In the appendix, we provide details about why this assumption is justified. However, these results are orthogonal to the current discussion. Hence, we omit it here and only include it in the appendix (Thm. 15).

We now describe our sampling procedure. Let $\mathcal{B} := \{P_1, \ldots, P_s\} = \{B_{ij} : 1 \leq i \leq s, 1 \leq j \leq |P_i|\}$ be the set of blocks outputted by the hashing scheme and let $Q := \{(x, y) \in B_{ij}\}$. We first choose a block $B \in \mathcal{B}$ with probability proportional to $|B|^2$ (the number of pairs in the block). Then we sample a pair uniformly at random from this block B. Note that this strategy doesn't give us a uniform sample from Q. This is because a pair (x, y) may be present in multiple blocks. To get the uniform sample, we reject the pair with probability inversely proportional to $a(x, y)$ (the number of blocks in which x, y are together). This approach based on rejection sampling ensures that we have a uniform sample from Q.

Next, we check if the pair satisfies $d(x, y) \leq \lambda$. Note that the LSH-based scheme tries to put similar points in the same bucket, hence the probability of success at this step is 'high'. Finally, we check if $C^*(x, y) = 1$. Our sampling procedure $\mathcal{P}_1$ is described in Alg. 2.

Algorithm 2: Sampling procedure $\mathcal{P}_1$ for positive pairs

Input: A set $\mathcal{X}$, a hashing algorithm $\mathcal{A}$, a C^*-oracle and parameter λ.
Output: (x, y) such that $C^*(x, y) = 1$

Pre-compute:
Use an LSH-based hashing scheme $\mathcal{A}$ to obtain partitions $\{P_1, \ldots, P_s\}$.
$\mathcal{B} := \{P_1, \ldots, P_s\} = \{B_{ij} : 1 \leq i \leq s, 1 \leq j \leq |P_i|\}$.

Sampling:
while *TRUE* **do**
  Sample a block B from $\mathcal{B}$ with probability $\propto |B|^2$.
  Sample (x, y) uniformly at random from B^2.
  Let $a(x, y) = \{(x, y) \in B^2 : B \in \mathcal{B}\}$.
  Sample u uniformly at random from $[0, 1]$.
  if $u > \frac{1}{|a(x,y)|}$ **then**
    continue.
  end
  if $d(x, y) \leq \lambda$ *and* $C^*(x, y) = 1$ **then**
    Output (x, y).
  end
end

Thm. 8 shows that with high probability the procedure $\mathcal{P}_1$ samples a pair according to a distribution $\mathcal{T}$ which approximates P^+.

Theorem 8. *Given (X, d), a C^*-oracle and parameter λ. Let d satisfy (α, β)-informative w.r.t C^*. Let the hashing algorithm $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ. Then with probability atleast $1 - \exp(-2(\nu(1-\alpha)|X_+^2|)^2)$ (over the randomness in the hashing algorithm), $\mathcal{P}_1$ samples pairs (x, y) according to distribution $\mathcal{T}$ over $X^{[2]}$ such that for any labelling function $C : X^{[2]} \to \{0, 1\}$, we have that*

$$\mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x, y) = 0\big] - \alpha - \epsilon - \nu \leq \mathop{\mathbf{P}}_{(x,y)\sim T}\big[C(x, y) = 0\big] \leq (1 + 2\nu)(1 + 2\alpha) \mathop{\mathbf{P}}_{(x,y)\sim P^+}\big[C(x, y) = 0\big]$$

To sample one same-cluster pair, we might need to make more than one same-cluster query to the C^*-oracle. Lemma 9 shows that with high probability, the number of queries made by $\mathcal{P}_1$ to sample one positive pair is upper bounded by a small constant.

Lemma 9. *Given set X, a C^*-oracle and parameter λ. Let d be (α, β)-informative w.r.t λ and C^*. Let $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ. Let q be the number of same-cluster queries made by $\mathcal{P}_1$. Then with probability atleast $1-\exp(-\nu^2(1-\alpha)^2|X_+^2|^2)$ (over the randomness in the hashing algorithm)*

$$\mathbf{E}[q] \leq \frac{1}{\beta(1-\epsilon-\nu)}$$

The pre-compute stage uses a hashing algorithm to obtain s different partitions. From the discussion in the appendix (Thm. 14), we its easy to see that this runs in $O(n)$ time. Next, we analyse the time taken to sample one same-cluster pair. Thm. 10 shows that under reasonable assumptions, the time taken is upper bounded by a constant with high probability.

Theorem 10. *Given set X, a C^*-oracle and parameter λ. Let d be (α, β)-informative w.r.t λ and C^*. Let $\mathcal{A}$ satisfy ϵ-tightness w.r.t λ.*

Define $\lambda' = 2\lambda\log(1+\frac{1}{\epsilon})$ and $\epsilon' = \lceil\log(\frac{1}{\epsilon})\rceil(1+\log(\frac{1}{\epsilon}))$. Let $K = \{(x,y) : d(x,y) \leq \lambda\}$ is the set of all pairs of points with distance $\leq \lambda$. Similarly, define sets $K_1 = \{(x,y) : \lambda < d(x,y) \leq \lambda'\}$ and $K_2 = \{(x,y) : d(x,y) > \lambda'\}$. Let $|K_1| \leq \rho_1|K|$ and $\epsilon'|K_2| \leq \rho_2|K|$.

Let t be the time taken to sample one point. Then with probability atleast $1-\exp\left(\frac{-\nu^2(1-\epsilon)(1-\alpha)|X_+^2|}{2}\right)-\exp\left(\frac{-\nu^2\rho_2|K|}{3}\right)$ (over the randomness in the hashing algorithm), we have that

$$\mathbf{E}[t] \leq s^2(1+\frac{1}{\eta})$$

where $\eta := \frac{(1-\nu)(1-\epsilon)\beta}{(1+\nu)(1+\rho_1+\rho_2)}$.

V. Sample and query complexity of RCC

In the previous section we saw how to sample (approximately) according to the distributions P^+ and P^-. We sample a 'small' set of true positive (or same-cluster) and true negative (or different-cluster) pairs using our distributions. We then choose the clustering $\hat{C} \in \mathcal{F}$ with the minimum number of mistakes on the sampled pairs. We prove that the true normalized correlation loss $L_{C^*}(C)$ is close to the loss of $\hat{C}^*$ (the clustering with minimum loss in $\mathcal{F}$). Thus, our solution strategy shows that by only having a small amount of information about C^* (making a small number of queries) we can find a clustering which is close (in terms of loss) to the optimal clustering in $\mathcal{F}$. We describe this procedure in Alg. 3.

Thm. 11 analyses the sample complexity of our approach. That is, the number of labelled positive and negative pairs our algorithm needs as input, so that the estimates of the loss based on this sample are close to their true values. We show that as long as the number of sampled pairs are in $O(\frac{\text{VC-Dim}(\mathcal{F})}{\epsilon^2})$ then our algorithm finds a clustering $\hat{C}$ which is close to the best clustering in $\mathcal{F}$. Here, VC-Dim is a combinatorial property which measures how 'complex' or rich the class of clusterings is. Note that the number of samples needed is independent of the size of the dataset X.

Algorithm 3: Empirical Risk Minimization

Input: (X,d), a set of clusterings $\mathcal{F}$, a C^*-oracle, parameter λ and sizes m_+ and m_-.
Output: $C \in \mathcal{F}$

1. Sample a sets S_+ and S_- of sizes m_+ and m_- using procedures $\mathcal{P}_1$ and $\mathcal{P}_0$ respectively.
2. For every $C \in \mathcal{F}$, compute
$$\hat{P}(C) = \frac{|\{(x,y) \in S_+ : C(x,y) = 0\}|}{|S_+|}$$
$$\hat{N}(C) = \frac{|\{(x,y) \in S_- : C(x,y) = 0\}|}{|S_-|}$$
3. Define $\hat{L}_{C^*}(C) = \mu\hat{P}(C) + (1-\mu)\hat{N}(C)$.
4. Output $\arg\min_{C\in\mathcal{F}} \hat{L}_{C^*}(C)$

For common classes, like $\mathcal{F} = \{T_1, \ldots, T_s\}$ where each T_i is a hierarchical clustering of X, [1] showed that the VC-Dim$(\mathcal{F})$ is in $o(\log s)$. Thus for such classes a small number of samples suffice to find a clustering which is close to the best clustering in $\mathcal{F}$.

Theorem 11. *Given metric space (X,d), a class of clusterings $\mathcal{F}$ of X and a threshold parameter λ. Given $\epsilon, \delta \in (0,1)$ and a C^*-oracle. Let d be (α,β)-informative w.r.t C^* and λ and let X satisfy γ-skewed property. Let $\mathcal{A}$ be the ERM-based approach as described in Alg. 3 (where the LSH-based scheme is ζ-tight) and $\hat{C}$ be the output of $\mathcal{A}$. If*

$$m_-, m_+ \ \geq a\frac{\text{VC-Dim}(\mathcal{F}) + \log(\frac{2}{\delta})}{\epsilon^2} \tag{7}$$

where a is a global constant then with probability atleast $1-\delta-\exp\left(\frac{-2\nu^2(1-\alpha)^2|X_2^+|^2}{49}\right)$ (over the randomness in $\mathcal{A}$), we have that

$$L_{C^*}(\hat{C}) \ \leq \ \min_{C\in\mathcal{F}} L_{C^*}(C) + 3\alpha + \zeta + \epsilon + \nu$$

Proof. Let T_0 be the distribution induced by $\mathcal{P}_0$ and T_1 be the distribution induced by $\mathcal{P}_1$. Using Thm. 18, we know that if $m_+ > a\frac{\text{VC-Dim}(\mathcal{F})+\log(\frac{1}{\delta})}{\epsilon^2}$ then with probability atleast $1-\delta-e^{-2\nu^2(1-\alpha)^2|X_2^+|^2}$, we have that for all C

$$\begin{aligned} &|\hat{P}(C) - \mathop{\mathbf{P}}_{(x,y)\sim T_1}[C(x,y)=0]| \leq \epsilon \\ \implies &\hat{P}(C) \leq \epsilon + (1+2\nu)(1+2\alpha)L_{P^+}(C) \quad \text{and} \\ &L_{P^+}(C) - \epsilon - \alpha - \zeta - \nu \leq \hat{P}(C) \end{aligned} \tag{8}$$

Note that we obtain upper and lower bounds for $\mathop{\mathbf{P}}_{(x,y)\sim T_1}[h(x,y) = 0]$ using Thm. 8. Similarly, if

$m_- > a\frac{\text{VC-Dim}(\mathcal{F})+\log(\frac{1}{\delta})}{\epsilon^2}$, then with probability atleast $1-\delta$, we have that for all h,

$$|\hat{N}(C) - \mathop{\mathbf{P}}_{(x,y)\sim T_0}[C(x,y)=1]| \leq \epsilon$$
$$\implies \hat{N}(C) \leq \epsilon + L_{P^-}(C) \quad \text{and } L_{P^-}(C) - \epsilon \leq \hat{N}(C) \quad (9)$$

Combining Eqns. 8 and 9, we get that with probability atleast $1-2\delta-e^{-2\nu^2(1-\alpha)^2|X_2^+|^2}$, we have that for all $C \in \mathcal{F}$

$$\begin{aligned}\hat{L}(C) &\leq \mu[\epsilon + (1+2\nu)(1+2\alpha)L_{P^+}(C)] \\ &\quad + (1-\mu)(\epsilon + L_{P^-}(C)) \\ &\leq L_{C^*}(C) + \epsilon + 2\alpha + 2\nu + 4\alpha\nu\end{aligned}$$

$$\begin{aligned}\text{And } \hat{L}(C) &\geq \mu(L_{P^+}(h) - \epsilon - \alpha - \nu - \zeta) + (1-\mu)(L_{P^-}(h) - \epsilon) \\ &\geq L_{C^*}(C) - \epsilon - \alpha - \nu - \zeta\end{aligned}$$

Now, let $\hat{C}$ be the output of $\mathcal{A}$ and let $\hat{C}^*$ be $\arg\min_{C\in\mathcal{F}} L_{C^*}(C)$. Then, we have that

$$\begin{aligned}L_{C^*}(\hat{C}) &\leq \hat{L}(\hat{C}) + \alpha + \epsilon + \nu + \zeta \leq \hat{L}(\hat{C}^*) + \alpha + \epsilon + \nu + \zeta \\ &\leq L_{C^*}(\hat{C}^*) + 2\epsilon + 3\alpha + 3\nu + 4\alpha\nu + \zeta\end{aligned}$$

Choosing $\epsilon = \frac{\epsilon}{2}$ and $\delta = \frac{\delta}{2}$ and $\nu = \frac{\nu}{7}$ throughout gives the result of the theorem, and this completes the proof of the theorem. □

Finally, we analyse the query complexity of our approach. That is the number of queries that our algorithm makes to the C^*-oracle. Our algorithm makes queries during the sampling procedure. We see that to sample m_- negative and m_+ positive pairs the number of queries is 'close' to $m_+ + m_-$ with very high probability. Thus, the number of 'wasted' queries is small.

Theorem 12. *[Query Complexity] Let the framework be as in Thm. 11. With probability atleast* $1 - \exp\left(-\frac{\nu^2 m_-}{4}\right) - \exp\left(-\frac{\nu^2 m_+}{4}\right) - \exp(-\nu^2(1-\alpha)^2|\mathcal{X}_+^2|^2)$ *over the randomness in the sampling procedure, the number of same-cluster queries* q *made by* $\mathcal{A}$ *is*

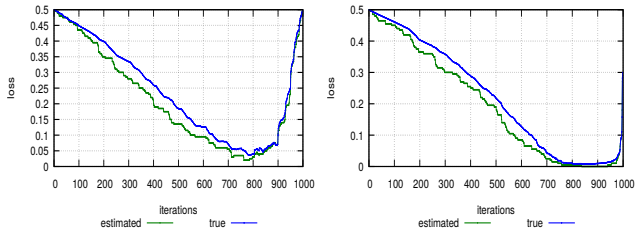
$$q \leq (1+\nu)\left(\frac{m_-}{(1-\gamma)} + \frac{m_+}{\beta(1-\zeta-\nu)}\right)$$

Proof. The number of queries made to sample the set S_g is $q_g = m_g$. Let q_+ denote the number queries to sample the set S_+. Using Lemma 9, we know that $\mathbf{E}[q_+] \leq \frac{1}{\beta(1-\zeta-\nu)}$ with probability atleast $1 - \exp(-\nu^2(1-\alpha)^2|\mathcal{X}_+^2|^2)$ over the randomness in the hashing procedure. Given that the expectation is bounded as above, using Thm. 19, we get that $q_+ \leq \frac{(1+\nu)m_+}{\beta(1-\zeta-\nu)}$ with probability atleast $1 - \exp(\frac{-\nu^2 m_+}{4})$. Similarly, combining Lemma 17 with Thm. 19, we get that with probability atleast $1-\exp(\frac{-\nu^2 m_-}{4})$, $q_- \leq \frac{(1+\nu)m_-}{(1-\gamma)}$. □

VI. Evaluation

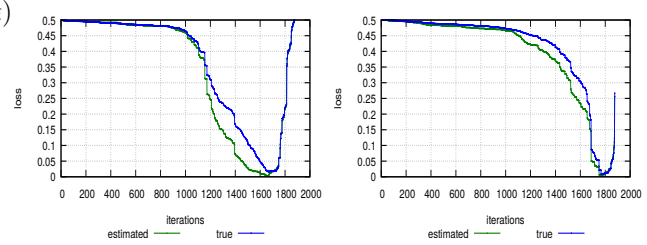
We now present the evaluation of our framework on a simulated and four real world datasets. In Section VI-B we show that our framework is generic and can be used to choose amongst many of the classes of algorithms for de-duplication. We also show that our framework can always choose a clustering which is close to the best clustering (algorithm) from a given class of clustering (algorithms) and our estimated loss for each of the clustering is very close to the true loss of these clustering algorithms. In Section VI-C we show that our framework is robust to upto 10% of oracle mistakes, which far exceeds the intended settings dealing with human experts. Finally, in Section VI-D we show that in our framework a relatively small number of samples are enough to accurately estimate the loss of a clustering.

[2]Algorithm did not finish within a reasonable time limit because of the high computational cost.



(a) Single linkage (b) Complete linkage

Fig. 1: Simulated dataset: Loss reported for every iteration of hierarchical clustering



(a) Single linkage (b) Complete linkage

Fig. 2: Publications dataset: Loss reported for every iteration of hierarchical clustering

A. Evaluation setup

Algorithms In our evaluation we use four graph based clustering algorithms: (1) Articulation point clustering (*ArtPt*) [20], (2) Star clustering (*Star*) [21], (3) Approximate correlation clustering (*ApproxCorr*) [2], (4) Markov clustering (*Markov*) [22]. These graph based algorithms have been used for de-duplication problems as shown in previous work [3]. Hierarchical clustering algorithms are very effective and have been widely used to perform de-duplication. We consider 4 different linkage methods for hierarchical clustering: single linkage (C1), complete linkage (C2), weighted linkage (C3), and average linkage (C4). In addition to this we also implemented a heuristic based de-duplication algorithm (*NaïveDedup*) where any two data points are considered similar if their distance is below a certain threshold. The output of this algorithm is pairs of data points which are marked similar.

	simulated		publications		products I		products II		restaurants	
Clustering	true loss	estimated loss	true loss	estimated loss	true loss	estimated loss	true loss	estimated loss	true loss	estimated loss
ArtPt	0.091	0.105	0.023	0.005	0.206	0.170	0.153	0.160	0.094	0.110
Star	0.052	0.060	0.100	0.050	0.207	0.190	0.231	0.170	0.041	0.045
ApproxCorr	NA[2]	NA	0.180	0.145	0.380	0.310	0.373	0.340	0.094	0.065
Markov	0.011	**0.000**	0.017	0.010	0.159	0.130	0.125	0.085	0.045	0.030
NaïveDedup	0.397	0.365	0.497	0.495	0.413	0.405	0.394	0.380	0.094	0.080
C1 (single)	0.019	0.025	0.016	0.018	0.150	0.110	0.131	0.120	0.022	**0.015**
C2 (complete)	0.005	0.005	0.009	0.009	0.150	0.130	0.135	0.065	0.034	0.040
C3 (weighted)	0.002	**0.000**	**0.005**	**0.006**	**0.110**	0.110	0.107	0.070	**0.019**	0.020
C4 (average)	**0.001**	**0.000**	0.007	0.017	0.120	**0.100**	**0.099**	**0.060**	**0.019**	0.020
Mean loss difference		0.016		0.014		0.027		0.035		0.010

TABLE I: True loss and the loss estimated by our framework.

		25, 25 samples		100, 100 samples		500, 500 samples	
Clustering	true loss	# queries	estimated loss	# queries	estimated loss	# queries	estimated loss
C1 (single)	**0.06107**	51	0.06	204	0.025	1023	0.024
C2 (complete)	**0.04177**	50	0.02	210	0.005	1024	0.016
C3 (weighted)	**0.03831**	50	0.02	203	0.015	1027	0.016
C4 (average)	**0.03489**	52	0.02	207	0.020	1043	0.013

TABLE II: Simulated dataset: Impact of number of samples on the loss of the clustering

		25, 25 samples		100, 100 samples		500, 500 samples	
Clustering	true loss	# queries	estimated loss	# queries	estimated loss	# queries	estimated loss
C1 (single)	**0.11075**	51	0.08	208	0.055	1031	0.041
C2 (complete)	**0.37172**	50	0.34	204	0.315	1035	0.334
C3 (weighted)	**0.29622**	51	0.14	203	0.260	1037	0.239
C4 (average)	**0.26877**	50	0.20	204	0.195	1027	0.202

TABLE III: Publications dataset: Impact of number of samples on the loss of the clustering

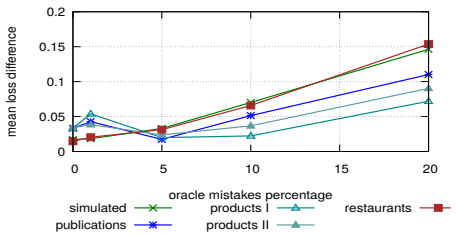


Fig. 3: Impact of oracle mistakes

Datasets For our evaluation we use five datasets. First dataset is a simulated dataset of ten thousand strings of length 20 where we simulate a clustering over the set of strings and use it as our ground truth. We use Jaro distance [7] as the distance metric for strings. To simulate a clustering we generate some seed strings and then for each seed string we generate multiple secondary strings by slightly editing the seed string. Each cluster of strings resembles a single entity. Second dataset is a real-world bibliographical information of scientific publications [23]. The dataset has 1879 publication records with duplicates. The ground truth of duplicates is available. To perform clustering on this dataset we first tokenized each publication record and extracted 3-grams from them. Then, on 3-grams we used Jaccard distance to define distance between two records. Next two datasets are lists of E-commerce products: First dataset contains 1,363 products from Amazon, and 3,226 products from Google, and the ground truth has 1,300 matching products. Second dataset contains 1,082 products from Abt, and 1,093 products from Buy, and the ground truth has 1,098 matching products. Both these products datasets are publicly available at [24]. The fifth dataset is a list of 864 restaurants from the Fodor's and Zagat's restaurant guides that contains 112 duplicates. This dataset is also publicly available at [25]. To perform clustering on the products and restaurants datasets we normalized the records (product or restaurant description) using standard techniques from natural language processing, namely; denoising text, word tokenization, normalization, and stemming and lemmatization. Given a record, this process gives us a list of word tokens. For each token, we first obtained a vector representation of the word using *Global Vectors* for word representations (GloVe [26]). We averaged this representation across word tokens to obtain the representation of a single record. We use cosine similarity to define the distance between two records. For the simulated and publications datasets, our distance metric was Jaccard and hence we use the MinHash [17] as the hashing scheme. For the rest of the datasets, we used SimHash [19] as the hashing scheme. For all the datasets we use ground truth as the oracle that can answer *same-cluster queries*. To calculate the true loss of a clustering (i.e. $L_{C^*}(C)$) we access all of the ground truth. Our framework uses only a sample of the ground truth to estimate the loss of a clustering. To judge the performance of our framework we compare the estimated loss $\hat{L}_{C^*}(C)$ against the true loss.

B. Clustering selection

In this experiment we demonstrate that our framework is generic and can be used to choose the best clustering algorithm amongst any of the classes of algorithms for deduplication. We used our framework on all the algorithms mentioned in Section VI-A. The results on five datasets are summarized in Table I. For each dataset we report the loss of the true-best clustering ($L_{C^*}(C)$) and the estimated loss of the best clustering selected by our framework $\hat{L}_{C^*}(C)$. This experiment highlights two main features of our framework: (i) our framework can always choose a clustering close to the best clustering algorithm from a given class of clustering algorithms using only a small number of samples, which is 200 (100 positive samples, and 100 negative samples) for all datasets and all the algorithms in Table I. (ii) Our estimated loss for each clustering is very close to the true loss of these clustering algorithms. At the bottom of the table we report the mean loss difference between estimated loss and true loss computed over all the algorithms.

We would also like to emphasize that in our framework we sample only once for each dataset and use that sample to estimate the loss of all the clusterings. In Figure 1 and 2 we show that our sample can very closely estimate the loss of every clustering generated at each iteration of the hierarchical clustering. Similarly, for Table I we sampled only once for each dataset and evaluated all the clusterings generated by each algorithm. Note that, each of the graph based algorithms have a hyper-parameter, i.e. the threshold on the edge weights. Edges with weights above this threshold represent dissimilar items and are pruned from the graph. For each of the graph based clustering algorithm we applied our framework on multiple values of the hyper-parameter and report only the ones with least true loss. However, for every choice of the hyper-parameter we observed that the estimated loss was very close to the true loss.

C. Effect of oracle mistakes

In this experiment we show that our framework is effective in real-world scenarios where the oracle may not be perfect and can make mistakes. Whenever the oracle classifies a similar pair as dissimilar or a dissimilar pair as similar we count it as a mistake. In our datasets we artificially introduce such mistakes and vary their ratio from from 0%, to 20%. In Figure 3 we show that our framework can closely estimate the clustering loss up to 10% of oracle mistakes, which, in real-world far exceeds the intended settings dealing with human experts. The Y-axis in Figure 3 reports the mean difference between true loss and estimated loss over all the clusterings selected in Table I.

D. Impact of sample size

In this experiment we show that even a small number of samples are enough to estimate the true loss ($L_{C^*}(C)$). We consider four different clusterings, each one picked at random from the four hierarchical clustering methods (C1 - C4). Table II and III reports the loss for simulated and publications dataset, respectively. For each dataset we increased the number of positive and negative samples and measured the loss. The table also shows the true loss of the clustering. It can be seen that the estimated loss calculated by our framework is close to the true loss even with 25 positive samples and 25 negative samples. In addition to this, the loss does not change much by increasing the number of samples. Which means that there is no incentive to sample more. We also show that the number of queries performed by our framework are close to the sample size (as claimed in Thm. 12), which are orders of magnitude less than $O(|X|^2)$. For example, in the simulated dataset and single linkage clustering (C1) with 25 positive and 25 negative samples our framework performed 51 queries, that means only one query was wasted. Similarly, 4 queries were wasted for 100 positive and 100 negative samples, and so on.

VII. Conclusion

We considered the framework of restricted correlation clustering to model the problem of detecting duplicate records in a database. The goal of the restricted variant is to choose a clustering from a given class of clusterings which minimizes the correlation loss. The clustering algorithm is allowed to interact with a domain expert by asking whether two points belong to the same or different cluster. We analyzed the sample and query complexity of finding the best clustering from the class $\mathcal{F}$. We showed that our novel hashing based sampling procedure requires only $O(\text{VC-Dim}(\mathcal{F}))$ labelled samples and finds a clustering which is close to the best clustering in $\mathcal{F}$. The pre-processing time of our algorithm is linear in the size of the dataset. Moreover, to sample one labelled example it only makes a constant number of queries to the domain expert (with high probability). We complement our theoretical results with an extensive empirical evaluation on a diverse class of clustering algorithms applied on multiple real-world datasets.

References

[1] S. Kushagra, S. Ben-David, and I. Ilyas, "Semi-supervised clustering for deduplication," in *AISTATS*, 2019.

[2] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering," *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004.

[3] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, "Framework for evaluating clustering algorithms in duplicate detection," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 1282–1293, Aug. 2009. [Online]. Available: https://doi.org/10.14778/1687627.1687771

[4] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, "Correlation clustering in general weighted graphs," *Theoretical Computer Science*, vol. 361, no. 2-3, pp. 172–187, 2006.

[5] M. A. Hernández and S. J. Stolfo, "The merge/purge problem for large databases," in *ACM Sigmod Record*, vol. 24, no. 2. ACM, 1995, pp. 127–138.

[6] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, no. 8, 1966, pp. 707–710.

[7] M. A. Jaro, *UNIMATCH, a Record Linkage System: Users Manual*. Bureau of the Census, 1980.

[8] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, "Duplicate record detection: A survey," *IEEE Transactions on knowledge and data engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[9] M. Cochinwala, V. Kurien, G. Lalk, and D. Shasha, "Efficient data reconciliation," *Information Sciences*, vol. 137, no. 1-4, pp. 1–15, 2001.

[10] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg, "Adaptive name matching in information integration," *IEEE Intelligent Systems*, vol. 18, no. 5, pp. 16–23, 2003.

[11] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine learning*, vol. 74, no. 1, pp. 1–22, 2009.

[12] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 59–68.

[13] S. Basu, A. Banerjee, and R. Mooney, "Semi-supervised clustering by seeding," in *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[14] H. Ashtiani, S. Kushagra, and S. Ben-David, "Clustering with same-cluster queries," in *Advances in neural information processing systems*, 2016, pp. 3216–3224.

[15] N. Ailon, A. Bhattacharya, and R. Jaiswal, "Approximate correlation clustering using same-cluster queries," in *Latin American Symposium on Theoretical Informatics*. Springer, 2018, pp. 14–27.

[16] O. Maimon and A. Browarnik, "Nhecd-nano health and environmental commented database," in *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 1221–1241.

[17] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher, "Min-wise independent permutations," *Journal of Computer and System Sciences*, vol. 60, no. 3, pp. 630–659, 2000.

[18] A. Z. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*. IEEE, 1997, pp. 21–29.

[19] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 380–388.
[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.
[21] J. Aslam, E. Pelekhov, and D. Rus, *The Star Clustering Algorithm for Information Organization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–23. [Online]. Available: https://doi.org/10.1007/3-540-28349-8_1
[22] S. van Dongen, "Graph clustering by flow simulation," Ph.D. dissertation, University of Utrecht, 2000.
[23] "Bibliography of scientific publications," https://www13.hpi.uni-potsdam.de/fileadmin/user_upload/fachgebiete/naumann/projekte/dude/CORA.xml.
[24] "E-commerce," https://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution.
[25] "Fodor's and zagat's restaurant guides," http://www.cs.utexas.edu/users/ml/riddle/data.html.
[26] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.
[27] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," in *Vldb*, vol. 99, no. 6, 1999, pp. 518–529.
[28] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.
[29] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," *Journal of the ACM (JACM)*, vol. 36, no. 4, pp. 929–965, 1989.
[30] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
[31] D. G. Brown, "How i wasted too long finding a concentration inequality for sums of geometric variables," *Found at https://cs. uwaterloo. ca/~ browndg/negbin. pdf*, vol. 6, 2011.

APPENDIX

A. Locality Sensitive Hashing

The technique of locality sensitive hashing was introduced by [27] to solve the problem of approximate nearest neighbour search. The LSH-based hashing schemes try to put similar points into the same bucket. Hence, to search for points which are 'similar' to a given point $x \in X$, one only needs to search within the same hash bucket instead of searching within the whole set X. Next, we describe a generic hashing based algorithm.

Definition 13 (LSH [19])**.** *Given a set X and a similarity function $f : X \times X \to [0, 1]$. Given a class of hash functions $H = \{h_1, h_2, \ldots\}$. An LSH w.r.t X and f is a probability distribution over H such that for each $x, y \in \mathcal{X}$,*

$$\mathop{\mathbf{P}}_{h \in H} [h(x) = h(y)] = f(x, y)$$

Some common examples of an LSH schemes are minhash scheme w.r.t jaccard similarity measure [17], [18], simhash scheme w.r.t hamming similarity measure [19]. We describe a generic locality sensitive hashing procedure in Alg. 4. Observe that, Alg. 4 outputs s different partitions of X. Assume without loss of generality, that a point x lies in the blocks $B_{11} \in P_1, B_{21} \in P_2, \ldots, B_{s1} \in P_s$. Now, to search for points y which are similar to x, we only need to search within the blocks B_{i1}. We say that these points lie in the same 'bucket' as x. We say that $b(x, y) = 1$ if and only if x and y lie together in the same block in atleast one of the partitions.

Algorithm 4: A generic LSH based hashing algorithm [19], [28]

Input: A set X, a similarity function f, a class of hash functions H over X, integers r, s and a perfect hash function p over the domain $\mathbf{N}^r$.
Output: Partitions $P_1, \ldots, P_s$ of the set X.

1 Let D be a distribution over H which satisfies Defn. 13 and let $k = rs$.
2 Sample hash functions $h_1, \ldots, h_k$ iid using D.
3 Group the hash functions into s bands. Each band contains r hash functions.
4 For all x and $1 \le i \le s$, let $g_i(x) = (h_{(i-1)s+1}(x), \ldots, h_{ir}(x))$. That is, $g_i(x)$ represents the i^{th} signature of x.
5 For all $1 \le i \le s$, obtain partitions P_i of $\mathcal{X}$. That is, $P_i = \{B_{i1}, \ldots, B_{im_i}\}$ where each $B_{ij} = \{x : p(g_i(x)) = b_{ij}\}$ for some b_{ij}.
6 Output $\{P_1, \ldots, P_s\}$.

Hence, to search for points which are similar to x, we need to search over y such that $b(x, y) = 1$. Our sampling procedure will sample pairs from the set $\mathcal{Q} := \{(x, y) : b(x, y) = 1\}$ (details in the next section). Hence, a requirement from the hashing scheme is that we should be able to construct the set $\mathcal{Q}$ in linear time. Thm. 14 shows that this is indeed the case.

Theorem 14. *Given X, a similarity function f, a class of hash functions H, integers r, s and perfect hash function p. Alg. 4 runs in $O(|X|rs \max_{ij} |B_{ij}|)$.*

Proof. Let $n = |X|$. Sampling k different hash function takes rs time. Computing the signatures for all x takes nrs time. Obtaining the different partition takes ns time. Now, computing R_x for all x takes time $t = \sum_{i=1}^{b} \sum_{j=1}^{m_j} |B_{ij}|^2$. Now, we know that for all i, $\sum_{j=1}^{m_j} |B_{ij}| = n$. Hence, $\sum_{j=1}^{m_j} |B_{ij}|^2 \le \max_j B_{ij} \ \sum_{j=1}^{m_j} |B_{ij}| = n \max_j B_{ij}$. Hence, $t \le ns \max_{ij} |B_{ij}|$. □

We see that the running time is dependent upon the block sizes. If the maximum block size is a constant then the hashing scheme runs in linear time. Even when the block sizes are bounded by $\log n$, then the scheme runs in $O(n \log n)$. Next, we show that a if $d(x, y) \le \lambda$ then $(x, y) \in \mathcal{Q}$ with very high probability. Also, if $d(x, y) > O(2\lambda)$ then with high probability $(x, y) \notin \mathcal{Q}$.

Theorem 15. *Given a set X, a distance function $d : X \times X \to [0, 1]$, a class of hash functions H, threshold parameter λ and a parameter δ. Let the similarity function be $f(x, y) := 1 - d(x, y)$ and let $\mathcal{A}$ be a generic LSH based algorithm (Alg. 4) which outputs partitions $P_1, \ldots, P_s$. Let $b(x, y) = 1$ iff x, y are together in atleast one of these partitions. Choose r, s such that $\frac{1}{2\lambda} < r < \frac{1}{-\ln(1-\lambda)}$ and $s = \lceil 2.2 \ln(\frac{1}{\delta}) \rceil$. Define $\delta' := s \ln(1 + \delta)$. Then for $(x, y) \in \mathcal{X}^2$*

- *If $d(x, y) \le \lambda$ then $\mathbf{P}_{h \in H} [\ b(x, y) = 1] \ > \ 1 - \delta$*

- *If* $d(x,y) > 2\lambda \ln\left(1+\frac{1}{\delta}\right)$ *then* $\mathbf{P}_{h\in H}[b(x,y)=1] \;<\; \delta'$

Proof. Observe that

$$\begin{aligned}\mathbf{P}[b(x,y)=0] &= \mathbf{P}\;\;[\textstyle\bigcap_{i=1}^{s} g_i(x) \neq g_i(y)]\\ &= \prod_i \Big(1 - \prod_{j=1}^{r} \mathbf{P}[h_{(i-1)r+j}(x) = h_{(i-1)r+j}(y)]\Big)\\ &= \prod_{i=1}^{s}(1 - \prod_{j=1}^{r} f(x,y)) = (1-f(x,y)^r)^s\end{aligned}$$

Consider the case when $d(x,y) \leq \lambda$. From the choice of s, we know that $s \geq 2.2\ln(1/\delta) \implies s \geq \frac{\ln(1/\delta)}{1-\ln(e-1)} \iff 1-\frac{1}{e} \leq \delta^{1/s}$. From the choice of r, we know that $r < \frac{1}{-\ln(1-\lambda)} \iff r\ln(\frac{1}{1-\lambda}) < 1 \iff (1-\lambda)^r > \frac{1}{e}$. Hence, then we have that $\mathbf{P}[b(x,y)=0] = (1-(1-d(x,y))^r)^s \leq (1-(1-\lambda)^r)^s < \delta$. This proves the first part of the theorem.

For the second part, consider the case when $d(x,y) > \lambda'$ where λ' is such that $\ln(1+1/\delta) = \frac{\lambda'}{2\lambda}$.

$\frac{1}{2\lambda} < r \iff \frac{\ln(1+1/\delta)}{\lambda'} < r$. Now, $\lambda' \leq \ln\left(\frac{1}{1-\lambda'}\right)$. Hence,

$$\begin{aligned}&\implies \frac{\ln(1+1/\delta)}{\ln(\frac{1}{1-\lambda'})} < r \iff r\ln(1-\lambda') < \ln(\frac{\delta}{1+\delta})\\ &\iff 1-(1-\lambda')^r > e^{-\ln(1+\delta)} = e^{-\delta'/s} > (1-\delta')^{1/s}\\ &\implies \mathbf{P}[b(x,y)=0] > (1-(1-\lambda')^r)^s > 1-\delta'\end{aligned}$$

Now, the only thing that remains to be shown is that we can choose an integer r satisfying the conditions of the theorem. Consider the function $f(x) = -\frac{1}{2x} - \frac{1}{\ln(1-x)}$. Using elementary analysis, we see that for $x \to 0$, $f(x) \to \infty$. Infact, for $x \leq 0.32$, $f(x) > 1$. Hence, for $\lambda \leq 0.32$, r satisfying the conditions of the theorem exists. □

B. Proofs of theorems and lemmas

Lemma 16. *Given X and a C^*-oracle. The procedure $\mathcal{P}_0$ samples a pair (x,y) according to the distribution P^-.*

Proof. The probability that a negative pair is sampled during a trial is $U^2(X^{[2]-}) =: q$. Fix a negative pair (x,y) and let $U^2(x,y) = p$. Hence, the probability that the pair (x,y) is sampled $= p + (1-q)p + (1-q)^2p + \ldots = p\sum_{i=0}^{\infty}(1-q)^i = \frac{p}{q} = \frac{U^2(x,y)}{U^2(X^{[2]-})} = P^-(x,y)$. □

Lemma 17. *Given set X and a C^*-oracle. Let X be γ-skewed. Let q be the number of same-cluster queries made by $\mathcal{P}_0$ to the C^*-oracle. Then, $\mathbf{E}[q] \leq \frac{1}{1-\gamma}$.*

Proof. Let p denote the probability that a negative pair is sampled during an iteration. We know that $p \geq (1-\gamma)$. Let q be a random variable denoting the number of iterations (or trials) before a negative pair is sampled. Then, q is a geometric random variable. $\mathbf{E}[q] = \frac{1}{p} \leq \frac{1}{1-\gamma}$. □

Proof of Thm. 8

Let $Q := \{(x,y) : b(x,y)=1\} = \{(x,y) \in B^2 : B \in \mathcal{B}\}$. Let $K = \{(x,y) : d(x,y) \leq \lambda\}$, let $K_Q = K \cap Q$, let $K^+ := \{(x,y) \in K : C^*(x,y)=1\}$ and finally let $K_Q^+ = \{(x,y) \in K_Q \;:\; C^*(x,y)=1\}$. Note that the choice of Q depends upon the hashing algorithm $\mathcal{A}$. However, after the pre-compute stage, the set Q is fixed and the sampling procedure samples from the set Q. Our procedure works in four steps.

S.1 $\mathcal{P}_1$ samples a point (x,y) from the set Q and induces a distribution D_1 on Q.

$$D_1(x,y) = \sum_{B \in a(x,y)} \frac{|B|^2}{\sum_{B'\in\mathcal{B}}|B'|^2}\frac{1}{|B|^2} = \frac{|a(x,y)|}{\sum_{B'\in\mathcal{B}}|B'|^2}$$

S.2 Next, we reject the sampled point with some probability thereby inducing another distribution D_2 on Q. Now, $D_2(x,y)$ satisfies the following recurrence

$$\begin{aligned}D_2(x,y) &= D_1(x,y)\frac{1}{|a(x,y)|}\\ &+ \Big(1 - \sum_{(x',y')\in Q} D_1(x',y')\frac{1}{|a(x',y')|}\Big)D_2(x,y)\end{aligned}$$

The recurrence basically says that the probability that the pair (x,y) is sampled is equal to the probability that it is sampled during the current round or nothing is sampled during the current round and then (x,y) is sampled. Simplifying the above equation, we get that

$$D_2(x,y) = \frac{\frac{1}{\sum_{B'\in\mathcal{B}}|B'|^2}}{\sum_{(x',y')\in Q}\frac{1}{\sum_{B'\in\mathcal{B}}|B'|^2}} = \frac{1}{|Q|}$$

S.3 We reject the sampled point if $(x,y) \notin K_Q$. In this step, we induce a distribution D_3 on K_Q. It is easy to see that $D_3(x,y) = \frac{1}{|K_Q|}$

S.4 Next, we reject the sampled point if $(x,y) \notin K_Q^+$. After this step, we induce a distribution D_4 on K_Q^+. $T(x,y) := D_4(x,y) = \frac{1}{|K_Q^+|}$

Another observation, which will be useful later in the proof is that for any $(x,y) \in K^+$, $\mathbf{P}[(x,y) \notin K_Q^+] < \delta$ (Thm. 15). Hence, hoeffding's inequality we get that, $\mathbf{P}\big[|K^+ \setminus K_Q^+| < (\delta+\nu)|K^+|\big] \geq 1 - \exp(-2\nu^2|K^+|^2)$

Next, we will show that the distribution T is an approximation of P^+. First, observe that $\mathcal{X}$ satisfies μ-nazdeek property. Hence, we get that

$$1 - \alpha \leq \mathbf{P}[d(x,y) \leq \lambda | C^*(x,y)=1] = \frac{|K^+|}{|X_+^2|} \qquad (10)$$

Now, let h be any labelling function over $\mathcal{X}^2$.

$$\begin{aligned}\mathop{\mathbf{P}}_{(x,y)\sim T}[C(x,y)=0] &= \frac{1}{|K_Q^+|}\sum_{(x,y)\in K_Q^+}\mathbf{1}_{[C(x,y)=0]}\\ &\leq \frac{1}{|K_Q^+|}\sum_{(x,y)\in X_2^+}\mathbf{1}_{[C(x,y)=0]}\end{aligned}$$

Now, with probability atleast $1 - \exp(-2\nu^2|K^+|^2) \geq 1 - \exp(-2\nu^2(1-\alpha)^2|X_+^2|^2)$ over the randomness in $\mathcal{A}$, we have

that $|K_Q^+| > (1-\nu-\delta)|K^+|$. Substituting this in the above equation gives

$$\mathop{\mathbf{P}}_{(x,y)\sim T}[C(x,y)=0] \leq \frac{1}{(1-\nu)(1-\delta)|K^+|}\sum_{(x,y)\in X_+^2}\mathbf{1}_{[C(x,y)=0]}$$
$$\leq \frac{\mathop{\mathbf{P}}_{(x,y)\sim P^+}[C(x,y)=0]}{(1-\nu-\delta)(1-\alpha)}$$

Now for the other direction, we have that

$$\mathop{\mathbf{P}}_{(x,y)\sim P^+}[C(x,y)=0] = \frac{1}{|X_+^2|}\sum_{(x,y)\in X_+^2}\mathbf{1}_{[C(x,y)=0]}$$
$$\leq \frac{1}{|K_Q^+|}\sum_{(x,y)\in K_Q^+}\mathbf{1}_{[C(x,y)=0]} + \frac{|X_+^2\setminus K_Q^+|}{|X_+^2|}$$
$$\leq \mathop{\mathbf{P}}_{(x,y)\sim T}[C(x,y)=0] + \frac{|X_+^2\setminus K^+|}{|X_+^2|} + \frac{|K^+\setminus K_Q^+|}{|K^+|}$$
$$\leq \mathop{\mathbf{P}}_{(x,y)\sim T}[C(x,y)=0] + \alpha + \nu + \delta$$

Now choosing, $\delta=\epsilon$ gives the result of the theorem. □

Proof of Lemma 9 Let K, Q, K^+, K_Q and K_Q^+ be as defined in the proof of Thm. 8. Also, let the distributions D_1, D_2, D_3 and D_4 be as defined in the same theorem. Also, from the analysis in bullet **S.4** in Thm. 8 with probability atleast $1-\exp(-2\nu^2(1-\alpha)^2|X_+^2|^2)$ we have that $\frac{|K_Q^+|}{|K^+|} \geq (1-\nu-\epsilon)$. Now, we have that X satisfies β-balanced property. Hence,

$$\beta \leq \mathbf{P}[h^*(x,y)=1 | d(x,y)\leq\lambda] = \frac{|K^+|}{|K|} \leq \frac{|K^+|}{|K_Q|}$$

Combining this we get that $\frac{|K_Q^+|}{|K_Q|} \geq \frac{\beta|K_Q^+|}{|K^+|} \geq (1-\nu-\epsilon)\beta$. Thus, given that a same-cluster query is made, the probability p that it succeeds is atleast $(1-\nu-\epsilon)\beta$. □

Proof of Thm. 10 Using Thm. 14, we know that one iteration of the pre-compute stage of $\mathcal{P}_1$ runs in $O(nrsm(\mathcal{B})) = O(n\log\left(\frac{2}{\epsilon}\right)\frac{-1}{\log(1-\lambda)}m(\mathcal{B})) = O(n)$. Next, we analyse the time taken to sample one point.

Let Q, K^+, K_Q^+ be as defined in the proof of Thm. 8. Let $\delta = \frac{\epsilon}{2}$. Using Thm. 15, we know that if $(x,y)\in K$ then $\mathbf{P}[(x,y)\in Q] > 1-\delta$. Also, if $(x,y)\in K_2$ then $\mathbf{P}[(x,y)\in Q] < \delta'$. For the purposes of analysing the time complexity of the sampling procedure, we can think of $\mathcal{P}_1$ as consisting of the following two steps.

- **T.1** $\mathcal{P}_1$ samples a pair (x,y) uniformly at random from Q. Thus, the probability of success at this step is $p_1 = \frac{1}{a(x,y)} \geq \frac{1}{s}$.
- **T.2** If the point also lies in K^+, that is $(x,y)\in K^+$ then it outputs that point. Thus, probability of success at this step is $p_2 := \frac{|K^+\cap Q|}{|Q|} = \frac{|K_Q^+|}{|Q|}$. Consider the following four sets, $K^+, K' := K\setminus K^+, K_1$ and K_2.
  Using multiplicative chernoff's bounds, we get that $\mathbf{P}[|K_Q^+| > (1-\nu)|K^+|(1-\delta)] \geq 1-\exp(\frac{-\nu^2(1-\delta)|K^+|}{2})$.
  Similarly, we have that $\mathbf{P}[|K_2\cap Q| < (1+\nu)|K|\rho_2] \geq 1-\exp(\frac{-\nu^2\rho_2|K|}{3})$. Also, note that $\mathbf{P}[|K'\cap Q| < (1+\nu)|K'|] = 1$ and $\mathbf{P}[|K_1\cap Q| < (1+\nu)|K_1|] = 1$. Using these results, we have that $\mathbf{P}[|K_Q^+| > (1-\nu)|K^+|(1-\delta)$ and $|(K_Q^+)^c\cap Q| < (1+\nu)(|K'|+|K_1|+|K|\rho_2)] \geq 1-\exp(\frac{-\nu^2(1-\delta)|K^+|}{2}) - \exp(\frac{-\nu^2\rho_2|K|}{3})$

$$\mathbf{P}\left[\frac{|K_Q^+|}{|(K_Q^+)^c\cap Q|} > \frac{(1-\nu)|K^+|(1-\delta)}{(1+\nu)(|K'|+|K_1|+|K_2|\delta')}\right]$$
$$\geq 1-\exp\left(\frac{-\nu^2(1-\delta)|K^+|}{2}\right) - \exp\left(\frac{-\nu^2\delta'|K_2|}{3}\right)$$

From the assumptions in the theorem, we get that $|K_1| \leq \rho_1|K|$. Now, $|K'| = |K| - |K^+| \leq (1-\beta)|K|$. Hence, $\frac{(1-\nu)|K^+|(1-\delta)}{(1+\nu)(|K'|+|K_1|+|K|\rho_2)} \geq \frac{(1-\nu)(1-\delta)\beta}{(1+\nu)(1-\beta+\rho_1+\rho_2)}$. Define $\eta := \frac{(1-\nu)(1-\delta)\beta}{(1+\nu)(1+\rho_1+\rho_2)}$. Then we get that,

$$\mathbf{P}\left[\frac{|K_Q^+|}{|Q|} > \frac{\eta}{\eta+1}\right] \geq 1-\exp\left(\frac{-\delta^2|K^+|}{(1-\delta)}\right)$$
$$-\exp\left(\frac{-\delta^2\rho_2|K|}{(1-\delta)^2}\right) =: \eta'$$

Hence, we see that with probability atleast η' over the randomness in the hashing procedure $p_2 \geq \eta$.

Using all the above, we get that $p \geq \frac{\eta}{(\eta+1)s}$. Recall that, p is the probability that the current iteration terminates. Thus, expected number of iterations to sample one point is $\leq \frac{s}{\eta}$. Note that the one iteration takes $\Theta(s)$ time (computing $|a(x,y)|$). The expected time to sample one point is $\leq s^2(1+\frac{1}{\eta})$ □

C. Technical lemmas and theorems

Theorem 18 (Fundamental theorem of learning [29])**.** *Here, we state the theorem as in the book [30]. Let H be a class of functions $h : \mathcal{X} \to \{0,1\}$ of finite VC-Dimension, that is* VC-Dim$(H) = d < \infty$. *Let D be a probability distribution over X and h^* be some unknown target function. Given $\epsilon, \delta \in (0,1)$. Let err_D be the $\{0,1\}$-loss function $err : H \to [0,1]$. That is $err_D(h) = \mathop{\mathbf{P}}_{x\in D}[h(x) \neq h^*(x)]$. Sample a set $S = \{(x_1,y_1),\ldots,(x_m,y_m)\}$ according to the distribution D. Define $err_S(h) = \sum_{i=1}^m \frac{\mathbf{1}_{[h(x_i)\neq h^*(x_i)]}}{m}$. If $m \geq a\frac{d+\log(1/\delta)}{\epsilon^2}$, then with probability atleast $1-\delta$ over the choice of S, we have that for all $h\in H$*

$$|err_D(h) - err_S(h)| \leq \epsilon$$

where a is an absolute global constant.

Theorem 19 (Concentration inequality for sum of geometric random variables [31])**.** *Let $X = X_1 + \ldots + X_n$ be n geometrically distributed random variables such that $\mathbf{E}[X_i] = \mu$. Then*

$$\mathbf{P}[X > (1+\nu)n\mu] \leq \exp\left(\frac{-\nu^2\mu n}{2(1+\nu)}\right)$$