Evaluating Methods for Efficient Entity Count Estimation

Jerin George Mathew
Sapienza Università di Roma
Rome, Italy
mathew@diag.uniroma1.it

Donatella Firmani
Sapienza Università di Roma
Rome, Italy
donatella.firmani@uniroma1.it

Divesh Srivastava
AT&T Chief Data Office
Bedminster, NJ, USA
divesh@research.att.com

ABSTRACT

The problem of estimating the size of a query result has a long history in data management. When the query performs entity resolution (aka record linkage or deduplication), the problem is that of estimating the number of distinct entities, referred to as the *entity count*. This problem has received attention from the statistics community but it has been largely overlooked in the data management literature. In this work, we formally define the entity count problem from a data management perspective and decompose it into a framework of fundamental steps. We explore approaches from both statistics and data management, systematically identifying a design space for different pipelines that address this problem. Finally, we provide extensive experiments to highlight the strengths and weaknesses of these approaches on real-world benchmarks.

PVLDB Reference Format:
Jerin George Mathew, Donatella Firmani, and Divesh Srivastava. Evaluating Methods for Efficient Entity Count Estimation. PVLDB, 18(8): 2589 - 2601, 2025.
doi:10.14778/3742728.3742750

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/jermathew/entity-count-estimation.

1 INTRODUCTION

The problem of estimating the size of a query has a long history in data management, ranging from early works on selectivity estimation (e.g., [29]) to more recent approaches for join query size estimation (e.g., [4]). In particular, the literature is rich with methods to estimate the size of a *select* or a *join* query. When applications need to work with the set of distinct entities contained in a dataset, the query performs entity resolution (aka record linkage or deduplication) on the dataset [47]. In this case, the query size estimation problem is that of estimating the number of distinct entities, referred in the following as the *entity count*.

Datasets often contain duplicate records that refer to the same entity, making the number of rows a poor estimate of its entity count. Consider the example dataset in Table 1: the number of rows is 6 but it only contains 2 distinct songs. The problem of recognizing duplicates, also known as Entity Resolution (ER), has received extensive attention from the the statistics, machine learning, NLP, and data management communities, from the seminal work of Fellegi-Sunter [12] to the recent breakthroughs based on language models [25], showing an unprecedented ability to identify how humans represent and misrepresent information. On the other hand, the problem of estimating the entity count has only received attention from the statistics community, with a variety of works such as capture-recapture methods [13] and Bayesian population estimation techniques [43].

ID	Title	Length	Artist	Year
A	Astronomy Domine - The Piper at the Gates of Dawn	4.202	Pink Floyd	'87
B	The Jimi Hendrix Experience - Ain't No Telling	112	NaN	10
C	Astronomy Domine	252093	Pink Floyd	1987
D	Ain't No Telling (Axis: Bold as Love)	01:52	Jimi Hendrix Experience	2010
E	001-Astronomy Domine	4m 12sec	Pink Floyd	NaN
F	Pink Floyd - Astronomy Domine	252	NaN	87

Table 1: Exemplary set of records from the *Music Brainz 20k* dataset [39]

Motivation. Estimating the number of distinct entities in a dataset has several practical applications. One key use case is query planning, where knowing the approximate entity count helps guide execution strategies. Similar to techniques in approximate query processing [38], entity count estimation could provide a computationally efficient way to assess dataset characteristics without performing full ER. This can assist query optimizers in deciding whether to apply aggressive blocking strategies when redundancy is high or opt for lighter deduplication when entity counts are close to the total number of records. Another important application is data cleaning. Many datasets contain varying levels of duplication, and knowing the approximate entity count allows users to estimate dataset quality before applying expensive ER. A high estimated duplication level suggests the need for targeted cleaning, while a low-duplication estimate may indicate that ER is unnecessary. An analogy can be drawn from unseen species estimation in statistics, where the goal is to infer the number of undiscovered species from a limited sample. Similarly, entity count estimation provides insights into dataset structure without explicitly resolving all duplicates. Despite its practical benefits, this problem has received limited attention outside the statistics community, leaving space for efficient solutions tailored to data management applications.

Challenges. A natural baseline for entity count is to pipeline ER and cluster counting. For example, ER might return the matching pairs $M = \{(A, C), (A, E), (C, E), (B, D)\}$, yielding the clustering

Proceedings of the VLDB Endowment, Vol. 18, No. 8 ISSN 2150-8097.
doi:10.14778/3742728.3742750

$\{\{A, C, E\}, \{B, D\}\}$, and cluster counting would return "2". This baseline can be extended by sampling s records for ER and applying upscaling techniques [15] in the counting step. However, it requires up to $O(s^2)$ matching queries, which must be issued to a machine learning classifier [44], a human crowd [46], or a large language model [25], incurring substantial latency and cost. Even progressive ER methods [34], while designed for low-latency high-F-score results, still need to be run exhaustively before the entity count can be derived.

Entity count approaches from the statistics community take a different route, relying on probabilistic and Bayesian models for ER—such as the frameworks of Fellegi and Sunter [12], Blink [42], and d-Blink [27]. These methods estimate matching probabilities, which can inform entity count. However, they are less compatible with modern ML-based matching, as they leave most matching decisions to statistical inference. As a result, efficient and practical solutions for entity count remain an open challenge, with significant room for improvement.

Our contribution. In this work we formally define the entity count problem from a data management perspective and decompose it into a framework of fundamental steps. Then, we consider works in both the statistics and data management literature, and systematically identify a design space for pipelines that can solve this problem. Finally, we provide extensive experiments to highlight their strengths and weaknesses on popular real-world benchmarks. Specifically,

- We propose a framework that integrates key steps for entity count estimation, combining methods from ML-based entity resolution (ER), statistical approaches and clustering;
- We introduce a sampling-based variant of the entity count pipeline, which reduces computational overhead by computing entity counts from a sampled subset and upscaling the result to approximate the full dataset;
- We systematically evaluate pipelines from both the data management and statistical literatures, testing their effectiveness and efficiency on real-world datasets. Our experiments show that clustering-based and efficient ER approximation-based methods achieve strong accuracy and efficiency on smaller datasets but struggle with scalability as the data size increases. Additionally, incorporating sampling into the pipeline improves scalability for larger datasets, though the accuracy can vary significantly, depending on the proportion of duplicate records and errors from the upscaling procedure.

The remainder of this paper is structured as follows. In Section 2, we review related work on tasks related to the entity count problem. Section 3 defines the entity count problem and Section 4 introduces the general framework. Section 5 describes a taxonomy of pipelines for entity count. Section 6 describes the proposed sampling-based approach and its implementation. We present experimental results in Section 7, evaluating the performance of different pipelines across several datasets. Finally, Sections 8 and 9 summarize the results and conclude the paper, outlining future directions for research.

2 RELATED WORK

In this section, we identify methods that are strongly related to the entity count problem and discuss how they can be leveraged to address our problem.

Join size estimation. Represents a key task for query optimization [21], and involves predicting at query time the size of the join between two tables after applying selection predicates to each. Several sampling-based approaches have been developed to address this task. For instance, Correlated Sampling [45] constructs small space synopses for quick join size estimates subject to dynamically specified predicate filter conditions. We mention also [5], which introduces a sampling algorithm called two-level sampling, which combines advantages of previous sampling methods, outperforming them on various join types.

Entity Resolution. Entity resolution (ER) aims at identifying records that refer to the same real-world entity. The task has been widely studied for over 50 years, beginning with the seminal probabilistic model by Fellegi and Sunter [12], and later extended through Bayesian approaches such as Blink [42] and its scalable variant d-Blink [27]. In recent years, several Machine Learning (ML) and Deep Learning (DL) methods have achieved state-of-the-art results. DeepER [8] combines GloVe [35] embeddings with an LSTM [20]-based DL model. DeepMatcher [44] generalizes this framework, supporting multiple embedding options. Ditto [25] integrates pretrained BERT [7] models with domain-specific features and data augmentation to generate synthetic training instances. ZeroER [48] proposes an unsupervised method based on Gaussian Mixture Models that performs competitively with supervised approaches.

Clustering. Clustering is closely related to estimating the number of unique entities in a dataset. It aims to group similar objects together, with high intra-cluster similarity and low inter-cluster similarity. Assuming each cluster corresponds to one entity, the entity count task can be seen as counting the number of clusters. Common techniques include correlation clustering [2], BIRCH [50], and DBSCAN [10], all of which assume full access to the dataset in an offline setting. In contrast, streaming clustering algorithms [49] process data in a single pass using limited memory, adapting to evolving input and dynamically forming new clusters.

Number of connected components estimation. This problem is closely related to entity count and involves estimating the number of connected components in a so-called parent graph, using only a sampled subgraph. In this context, the entity count task can be framed as estimating the number of connected components in a graph where nodes represent records and edges link records that refer to the same entity. These edges are not known in advance and must be inferred. The problem dates back to the seminal work by Frank [15], and has since been extended in several directions. For example, [22] introduces an estimator based on network motif counts, with guarantees on the mean squared error for graphs with bounded spectral gaps. Another line of work [23] focuses on chordal graphs, characterizing optimal sample complexity in the sublinear regime, and providing linear-time estimators along with minimax lower bounds.

2.1 Discussion

The above works can be broadly classified along two key dimensions: duplication type and sampling usage. Duplication type refers to whether methods assume exact duplicates (identical records) or support approximate duplicates (with typos, schema mismatches, or missing values). Sampling usage captures whether methods use sampling for efficiency or operate on the full dataset. We observe that none of the above lines of work focus on a setting where approximate duplicates are present and sampling is used for efficiency. For instance, join size estimation assumes exact duplicates, making it unsuitable for datasets with approximate duplicates. ER and clustering methods can handle approximate duplicates but do not employ sampling. Even scalable approaches like d-Blink [27] rely on partitioning and parallelism rather than estimation from samples. Our work fills this gap by (i) introducing a general framework for entity count estimation under sampling and (ii) empirically evaluating ML and statistical methods to assess their accuracy and efficiency across datasets with varying characteristics.

3 PROBLEM STATEMENT

Let D be a dataset of record entries r. Records can be either structured (e.g. set of key-value pairs describing the technical specs of a mobile phone) or unstructured data entries (e.g. reviews about that same mobile phone model). Each record $r \in D$ is associated to a specific real-world entity $e \in E$. Let $f(r) \in E$ denote the underlying entity $e \in E$ associated to r. The *entity count* of D, denoted with $c(D)$, is the number of unique entities present in D, i.e. $c(D) = |\{f(r) \mid r \in D\}|$.

Example 3.1. Table 1 contains a set of exemplary records from the Music Brainz 20k dataset [39]. Each record in Table 1 represents a specific song, with some noticeable (noisy) duplicated records. For instance, records A, C, E and F refer to the song *"Astronomy Domine"* from Pink Floyd, while records B and D refer to the song *"Ain't No Telling"* from Jimi Hendrix Experience. The underlying number of unique entities in Table 1 is thus $c(D) = 2$.

We now introduce the entity count estimation problem.

Problem 1. *Given a dataset D, the entity count estimation (ECE) problem consists in providing an estimate $\hat{c}(D)$ of the true entity count $c(D)$.*

Note that the problem formulation only takes into account the entities featured in the input dataset, thus $1 \leq \hat{c}(D) \leq |D|$, i.e. the estimate cannot be larger that the input dataset.

Given an ECE method $\hat{c}(\cdot)$, its accuracy can be evaluated in terms of *approximation error* δ, which is defined as follows:

$$\delta(D) = \frac{\hat{c}(D) - c(D)}{c(D)} \tag{1}$$

where an approximation error of 0 implies a perfect estimate, while large negative or positive values denote substantial underestimation or overestimation of the result respectively.

Discussion. While ER methods can be used for ECE, they are typically optimized for F-score rather than entity count accuracy. This misalignment can result in ER models achieving high F-scores while still producing poor entity count estimates.

Evaluating Ditto for ECE. To illustrate the challenges in entity count estimation, we evaluate Ditto [25], a state-of-the-art ER model, using benchmark datasets (Table 2) with DistilBERT [40] and RoBERTa [26]. To improve efficiency, we apply token blocking, discard low-similarity pairs, and prioritize the most similar pairs using edge ordering [14]. Transitivity is used to infer matches and reduce redundant Ditto queries. Entity counts are estimated by clustering records based on predicted matches. Experiments were conducted on an NVIDIA GeForce RTX 3090 GPU. Figure 1 presents the results: the left plot shows the relationship between F-score and approximation error, the center compares precision and approximation error, and the right shows execution time across dataset sizes.

Observation 1 (Underestimation bias)**.** Across all datasets, Ditto systematically underestimates entity count, as indicated by negative approximation errors. This bias stems from the asymmetry in how false positives and false negatives affect clustering. In fact, false positives merge clusters, reducing the count, while false negatives do not necessarily increase it, since clusters remain intact if sufficient connectivity exists. The effect is particularly strong in high-duplication datasets (Cars, WDC-xlarge-computers, and Alaska-monitor), where merging errors have a greater impact (Figure 1, left). Precision, which reflects the rate of correctly identified matches, is lowest in WDC-xlarge-computers and Alaska-monitor, leading to more false positives and worsening underestimation (Figure 1, center). Interestingly, while RoBERTa achieves similar precision and F-score on Cars and Music-Brainz-200k, the approximation errors differ. The higher duplication ratio in Cars leads to more severe underestimation, whereas in Music-Brainz-200k, fewer duplicates limit the impact of false positives.

Observation 2 (Impact of duplication rate)**.** A high F-score can correspond to low approximation error, as seen with RoBERTa on Music-Brainz-20k (f-score 0.98, error 0). However, this is not always the case. DistilBERT achieves an F-score of 0.92 on DBLP-Scholar with an approximation error of -0.19, but a slight drop in F-score to 0.88 on Cars more than doubles the error to -0.46. Datasets with high duplication rates are more sensitive to false positives. In Cars, where ∼99% of records are duplicates, entity count depends on the number of clusters, so merging even a few clusters drastically lowers the count. In contrast, RoBERTa on Music-Brainz-200k, despite a similar F-score and slightly lower precision than DistilBERT on Cars, has a moderate error (-10%) due to its lower duplication rate. With more unique entities, false positives are distributed across many clusters, minimizing their impact. Conversely, in Cars, where most records belong to a handful of entities, even a few false positives significantly lower the estimate.

Observation 3 (Scalability)**.** Figure 1 (right) shows execution time as a function of dataset size. For datasets under 20k records, execution remains under 30 minutes. However, scalability becomes a concern for larger datasets. On Music-Brainz-200k, execution time jumps to nearly 6 hours with DistilBERT and ∼18 hours with RoBERTa, highlighting the computational burden of Ditto. This raises concerns about applying such models to even larger datasets, where execution times could become prohibitive.

In this work we study several families of methods for efficiently estimating the entity count of a dataset by using samples $D' \subseteq D$

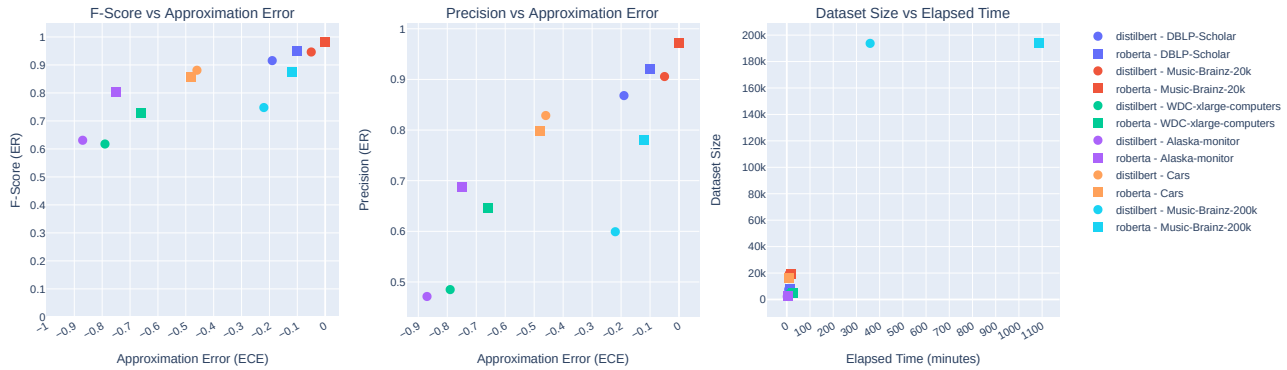

Figure 1: Using Ditto for ECE: approximation error vs. f-score (left), approximation error vs. precision (center), execution time vs. dataset size (right)

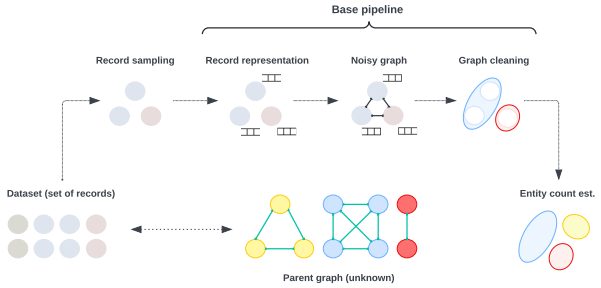

Figure 2: Overview of the main stages in our ECE framework

of the original dataset D. Note that this also includes the case where the original dataset is fully sampled. We unify these methods under a common framework by modeling D as an underlying (unknown) parent graph, a union of cliques where each clique represents records referring to the same entity. Given one or more samples $D' \subseteq D$, the goal is to estimate the total number of connected components in the parent graph using the observed number of components in the sampled subgraph. This framework enables scalable techniques that balance accuracy and computational cost, particularly for large datasets. For a sampled subset D', we denote the entity count estimate as $\hat{c}(D'|D)$, where $\hat{c}$ represents the estimation method.

4 A GENERAL FRAMEWORK FOR ECE

We illustrate in Figure 2 a general framework to estimate the entity count of a dataset D using samples of records $D' \subseteq D$. Conceptually, the framework consists of three main steps. The first step consists in sampling a set of records $D' \subseteq D$. The following step is estimating the number of unique entities in D', which we frame as estimating the number of connected components induced by the subgraph associated by D'. Finally the last step is estimating the number of unique entities in D, which we frame as upscaling the number of connected components found in the subgraph.

We now describe the individual steps in the devised framework for ECE below:

Record sampling. This step consists of selecting a representative sample $D' \in D$ of the dataset to be used to estimate the entity count of the original dataset D. Note that this also includes the case where the entire dataset is sampled. This step aims at providing a trade-off between accurate and efficient estimation of the entity count of a dataset. In fact, small sample sizes can provide fast estimation of the entity count but the accuracy might be low. On the other hand, larger samples can lead to more accurate results but can be expensive in terms of running time and might suffer from scalability issues when it comes to large datasets.

Record representation. This step consists of processing the records in D' and modeling them in a meaningful format for the subsequent step of generating a graph from D'. In general, different techniques can be used to model the records, ranging from standard bag-of-words (BoW) model to more recent dense vector (embedding) based approaches. Ideally the selected format should capture the similarity between records, such that the records that are similar to each other have similar representations.

Noisy graph generation. This step uses the sampled records and their representations to generate a weighted graph $G(V, E)$ where each node $v \in V$ corresponds to a record $r \in D'$ and edge weights measure the similarity of pairs of records, based on their representation. Ideally, this graph should result in a union of disjoint cliques, where only records referring to the same underlying entities are linked to each other. However, since records are approximate duplicates instead of exact duplicates, the resulting graph might contain spurious edges that link records associated with different entities, as well as missing links between records that refer to the same entity. These errors can result in overlapping maximal cliques, where records incorrectly belong to multiple clusters, or in fragmented clusters, where records that should be connected remain isolated or form smaller disconnected subgroups. To address these inconsistencies, the subsequent *Graph cleaning* step can refine the structure of the graph, ensuring a more accurate clustering of records.

Graph cleaning. This step refines the noisy graph to better approximate a union of disjoint cliques. The goal is to remove incorrect edges and reconstruct missing ones, ensuring that the cleaned graph aligns more closely with an induced subgraph of the unknown parent graph associated with D, given the set of sampled records D'. Graph cleaning can be implemented using a range of techniques, including ML-based ER models to statistical ER and clustering methods. For instance, an ML-based ER method can classify edges as

matching or non-matching, removing spurious connections while retaining correct ones. Similarly, clustering-based methods can help separate mistakenly merged cliques or merge smaller clusters that should be connected, addressing the issue of overlapping maximal cliques.

Entity count estimation. In this step, we use the union of cliques generated in the previous phase to estimate the number of entities in the original dataset, D, by scaling the number of cliques. Various scaling techniques have been proposed in the literature, particularly for estimating the number of connected components in a graph based on an induced subgraph, such as [15, 23]. These techniques involve statistical estimators, which can be either biased or unbiased, and aim to estimate the total number of connected components in the original graph by analyzing the number of nodes within each clique in the induced subgraph.

Among the steps in the pipeline illustrated in Figure 2, we identify the combination of *Record representation*, *Noisy graph generation* and *Graph cleaning* as being the core components in estimating the entity count of D. In fact, those three steps can be readily run to estimate the entity count of the dataset without the need to sample the original dataset. Moreover, these steps allow space for a variety of techniques, ranging from ER-based ones to possibly new techniques for ECE. We will call the combination of those three steps *base pipeline*. The joint use of *Record sampling* and *Entity count estimation* on top of a base pipeline allow to scale to larger datasets and offer a trade-off between accuracy and efficiency.

In this work, we focus on four main families of base pipelines and study their performance in terms of accuracy and efficiency for ECE. These pipelines draw from established methods in entity resolution (ER), clustering, and statistics, as well as recent advancements in machine learning and embedding models.

i) **ML ER** This base pipeline represents the adaptation of ER methods for the entity count task. This pipeline uses popular blocking algorithms (e.g. [3, 17]) to generate a noisy graph followed by the use of ML-based ER methods (e.g. [25]) to filter out noisy edges and generate a union of cliques.
ii) **Simulation** This base pipeline provides a faster approximation of the ML ER approach by simplifying the ER process for entity count estimation. Instead of exhaustively applying the ML ER model, it approximates the model's output by mapping record pair similarities to calibrated matching probabilities. These probabilities are derived by querying the ML ER model on a subset of records to speed up the process while still grouping similar records into cliques for entity counting.
iii) **Statistical ER** This base pipeline allows traditional statistical ER approaches (e.g. [12, 27, 42]) to be repurposed for the ECE task by using the linkage structure provided by these methods to induce a union of cliques that groups records that refer to the same entity.
iv) **LLM embeddings** This family frames the entity count task as a clustering problem. It utilizes large language models (LLMs) (e.g., [30]) to embed the records into dense vector representations. These embeddings are then fed into clustering algorithms (e.g., [10]) to form a union of cliques, where each clique corresponds to records representing the same entity.

The family of base pipelines that we consider in our study represent a comprehensive set of methods that can be used to address the ECE problem. In general, the outlined framework can be extended with new methods for ECE as long as they generate a union of cliques from a (sample of) records. Additionally, in all of these pipelines, sampling and upscaling techniques can be applied to improve efficiency and scalability, allowing for quicker approximations of the entity count without the need for exhaustive computation.

5 BASE PIPELINES

We now describe in more detail the base pipelines by breaking down their inner workings in terms of their implementation of the *Record representation*, *Noisy graph generation* and *Graph cleaning* steps.

5.1 ML ER pipeline

The ML ER pipeline adapts machine learning-based ER techniques to address the entity count estimation task.

Record representation. A common approach in ER is to represent records using a Bag-of-Words (BoW) model, where the semantic similarity between records is measured based on shared tokens. Techniques like TF-IDF weighting are frequently employed to give more weight to informative terms, helping to distinguish between similar records. Other methods, such as embedding-based representations can also be applied when deeper semantic similarities are required to handle more complex data relationships.

Noisy graph generation. In ER, reducing the number of pairwise comparisons is crucial, particularly for large datasets. Blocking algorithms, such as adaptive blocking [3], are commonly used to group records into blocks and limit the number of record pairs that need to be compared. From the output of blocking, a noisy graph can be constructed by connecting records within the same block, with edges representing potential matches. Alternatively, metablocking techniques [33] can be employed to directly create a *metablocking graph*, where edges between records are weighted based on their co-occurrence across multiple blocks. These approaches allow for a significant reduction in the number of comparisons while ensuring that relevant record pairs are retained for further analysis. Similarity between records can then be measured using metrics like cosine similarity, depending on the chosen blocking approach and assigned as weights to the edges.

Graph cleaning. The final step groups records referring to the same entity by classifying edges in the noisy graph as matches or non-matches using an ER model. To improve efficiency, edges can be processed in a prioritized order such as node ordering [14], ensuring the most promising ones are evaluated first. Transitivity can also be applied to infer additional matches (e.g., if A matches B and B matches C, then A matches C), reducing redundant queries. Finally, to further refine entity groupings, additional queries can be made for matching edges to improve robustness [16]. The result is a union of cliques, where each clique represents a unique entity.

5.2 Simulation pipeline

The Simulation pipeline provides an approximation of the ML-based ER process for entity count estimation, focusing on improving computational efficiency by simulating the edge classification process. The *Record Representation* and *Noisy Graph Generation* steps are the same as in the ML ER pipeline, where records are represented and a noisy graph is constructed based on blocking techniques.

Graph cleaning. The Simulation pipeline approximates edge classification by mapping similarity scores to calibrated matching probabilities, avoiding explicit classification for every edge. Similarity scores are partitioned into bins (e.g., equiwidth histogram with 10 bins of size 0.1), and a sample of edges from each bin is classified using an ER model. The observed match rate within each bin defines its calibrated matching probability, which is then applied to all edges in that range. For example, if 80 of 100 sampled edges in the 0.8–0.9 similarity range are matches, the bin's probability is set to 0.8, and remaining edges in that bin are assigned matching outcomes by sampling from a Bernoulli distribution. This process reduces computational overhead while maintaining fidelity to the ER model's decision patterns. Key hyperparameters, including bin width, sample size, and probability aggregation method, can be adjusted to optimize trade-offs between accuracy and efficiency. Once probabilities are established, edges are processed similarly to ML ER, iterating in a predefined or random order [14], with transitivity applied to infer additional matches and minimize queries. To enhance robustness, edges within the same cluster as previously matched pairs may be queried [16], ensuring consistency in classification.

5.3 Statistical ER pipeline

The Statistical pipeline uses ER techniques from the statistical literature to derive the linkage structure of records and generate a union of cliques.

Record representation. In this pipeline, records are treated as raw textual representations without requiring preprocessing steps, such as transforming them into vectorized formats. Statistical ER methods like Blink [42], d-Blink [27], and other Bayesian models grounded in Fellegi and Sunter's record linkage theory [12] are capable of working with the original records while computing similarities internally. These models can incorporate embedding-based similarities or rely on empirical priors, as described by Steorts [42], to estimate the likelihood of records belonging to the same entity.

Noisy graph generation. Unlike the previous pipelines, the Statistical ER pipeline does not explicitly construct a noisy graph. Instead, the statistical ER model directly generates a union of cliques based on probabilistic assignments. Methods like Fellegi and Sunter's [12] model assign a matching status to record pairs based on likelihood ratios, while d-Blink and Bayesian models use empirically motivated priors to probabilistically assign entity identifiers to records. Although the graph is not explicitly constructed, it is implicitly represented through the computation of similarity scores between records. For instance, Blink precomputes similarity matrices for each attribute, where each matrix can be viewed as an independent noisy graph, encoding pairwise similarities between records based on a specific similarity function. These per-attribute graphs contribute to the overall linkage process, where statistical models infer relationships without directly filtering edges. Instead of applying a classifier to filter edges, statistical models probabilistically infer entity assignments using techniques such as Gibbs sampling or expectation-maximization.

Graph cleaning. The statistical models treat records as input and assign entity identifiers (e.g., integers) probabilistically, based on the likelihood of records representing the same entity. Once records are assigned to entities, the union of cliques is generated by grouping records with the same identifier into the same clique. Statistical methods such as Blink and d-Blink apply Bayesian principles to estimate the linkage structure, making the process highly adaptable to different datasets and priors.

5.4 LLM embedding pipeline

The LLM Embedding pipeline uses embeddings generated by LLMs and clustering techniques to estimate entity counts. In this approach, records are transformed into dense vector representations, which are then clustered to identify distinct entities. Depending on the clustering algorithm used, a similarity graph may be computed explicitly or implicitly.

Record representation. Each record is transformed into a dense vector using LLM-based embedding models, such as BERT [7] or other large language models. These embeddings capture the semantic content of the records in a high-dimensional vector space, enabling more accurate comparisons between records than traditional vector-based methods.

Noisy graph generation. In this pipeline, the generation of a noisy graph is sometimes implicit, depending on the clustering algorithm. While some clustering algorithms may rely on the computation of pairwise similarities (resulting in an explicit similarity graph), others may cluster records directly based on their distances in the embedding space without materializing a graph. For example, algorithms such as DBSCAN [10] or OPTICS [1] do not require a prior graph, but still effectively group records based on their relative proximity in the embedding space. A critical feature of the chosen clustering algorithm is that it should not require the number of clusters to be specified in advance, as the purpose of the pipeline is to estimate this value (the entity count).

Graph cleaning. The result of clustering the records based on their embeddings is treated as a union of cliques, where each cluster corresponds to a set of records representing the same entity. High-dimensional clustering techniques, such as DBSCAN or OPTICS, are particularly suited for this task, as they can handle varying densities and do not require pre-specifying the number of clusters. The clusters produced by these algorithms form the union of cliques, implicitly grouping records that are similar in the embedding space.

6 SAMPLING-BASED PIPELINES FOR ECE

While the previously mentioned pipelines can estimate the entity count of D, their computational complexity may limit scalability for datasets containing millions of records. For instance, the ML ER pipeline exhibits a time complexity that is quadratic in the number of records in the worst case, making it impractical for large-scale datasets. To address this, the framework in Figure 2 introduces two steps: *Record sampling* and *Entity count estimation*. These steps improve the efficiency of the core method by (i) selecting a subset

of records $D' \subseteq D$ and (ii) scaling the number of cliques detected in D' to estimate the total entity count in D. We focus on Bernoulli sampling paired with the *Klusowski Bernoulli estimator*. Bernoulli sampling selects each record independently with probability $p < 1$, resulting in an expected sample size of $N \cdot p$, where $N = |D|$. After obtaining the sample, the Klusowski Bernoulli estimator is applied to upscale the result. The *Klusowski Bernoulli estimator* [23] refines the *Frank Bernoulli estimator* [15]. The Frank Bernoulli estimator provides an unbiased estimate of the number of connected components in the parent graph based on an induced subgraph sampled via Bernoulli sampling. However, the Frank estimator, while unbiased, exhibits high variance and can yield negative estimates due to its alternating series formulation. The Klusowski estimator, while slightly biased, reduces variance and ensures more stable estimates, particularly on large datasets. Although we primarily focus on Bernoulli sampling, other methods such as Simple Random Sampling (SRS) can also be used. SRS selects a fixed number of records D', where every subset of size k has equal probability of being chosen. The Frank SRS estimator can upscale entity counts from SRS samples but, like the Frank Bernoulli estimator, can suffer from high variance and yield negative estimates.

7 EXPERIMENTS

7.1 Datasets

We conducted experiments using a total of 10 datasets, which are listed in Table 2. We classify these datasets based on their number of rows and the number of unique entities. Given that the number of unique entities can vary significantly across different datasets, we compute a score to measure the duplication level of each dataset, defined as follows, where D' is the sample dataset:

$$\text{Duplication factor}(D) = \frac{|D| - c(D)}{|D| - 1}$$

To distinguish between datasets with low and high levels of duplication, we use a 50% threshold. Datasets with a duplication factor above 0.50 are classified as *high-duplication*, where a majority of records correspond to duplicate entities, while those below this threshold are classified as *low-duplication*, indicating relatively sparse duplication. Based on this categorization and dataset size, we classify datasets into four different groups, which are further described below.

Small datasets with few duplicates. For this category we consider the *Music-Brainz-20k*, a dataset of roughly 20.000 records from [24] which contains song records from the MusicBrainz database. This dataset spans five sources and includes duplicates for half of the original records. Each variant is generated using the DAPO data generator [19] to create records with modified attribute values.

Small dataset with multiple duplicates. We consider 4 datasets for this category. `DBLP-Scholar` is a dataset from the ER benchmark [24] and consists of entries from the DBLP and Google Scholar bibliographic sources. `WDC xlarge (computers)` is a dataset from the WDC benchmark [36] and contains duplicated computer listings from several web sources. `Alaska (monitor)` is a dataset from the Alaska benchmark [6] and consists in duplicated specification of monitors from more than 20 data sources. Since the set of attributes can significantly vary among records from different sources, we attached each record an additional synthetic attribute consisting of the concatenation of its original attributes. Finally, `Cars` is a dataset generated by scraping textual descriptions of vehicles, such as make and model, from various online sources [16].

Large datasets with few duplicates. For this category we consider two splits of the `Music-Brainz` dataset, containing 200,000 and 2 million records respectively. We also consider the 5 million split of the `North Carolina Voters` dataset, from [24], which is based on real records from the North Carolina voter registry. The datasets comprises records from 5 sources and contains both exact duplicates and noisy duplicates across all sources.

Large datasets with multiple duplicates. For this category, we augmented the `Cars` dataset and the `WDC xlarge (computers)` dataset with synthetic records until they reached at least 1 million records. Each synthetic record was generated by combining different attribute values from records referring to the same entity, while also randomly dropping tokens from both the original and synthetically generated records.

7.2 Metrics

We evaluate ECE approaches in terms of accuracy and efficiency. Accuracy is measured using approximation error (Equation 1), which for sampling-based pipelines is:

$$\delta(D') = \frac{\hat{c}(D|D') - c(D)}{c(D)}$$

Efficiency is measured in terms of sample size $|D\prime|$ and running time to compute $\hat{c}(D|D')$, reflecting two aspects: sample efficiency and computational efficiency. A method is sample-efficient if it achieves low approximation error even with a small fraction of the dataset, i.e., $|D'| = o(|D|)$. Computational efficiency considers how runtime scales with dataset size—an ideal approach should not only work with small samples but also maintain reasonable execution time as datasets grow.

7.3 Implementation details

We now describe the implementation details of the four base pipelines used in our experiments. For each pipeline, we evaluate two model variants to assess their impact on entity count estimation. These pipelines follow the core steps outlined in Figure 2.

ML ER. The ML ER pipeline consists of record representation, noisy graph generation, and graph cleaning. Records are modeled using a TF-IDF-weighted Bag-of-Words (BoW) representation, where tokens are extracted from attributes, and informative terms receive higher weights. A noisy graph is generated via token blocking, grouping records sharing at least one token while filtering large blocks (>10% of the dataset). Records in the same block are connected in a blocking graph, with edges weighted by TF-IDF-based Jaccard similarity. Since block cleaning may separate matching records into different connected components, potentially overestimating the entity count, we measure this effect using *sparsity*, defined as $\sum_i^n c(CC_i)/c(D)$. For example, a sparsity of 1.20 indicates an overestimation of 20% by an oracle method. To improve efficiency, edges with Jaccard similarity below 0.1 are filtered out, though this may further contribute to overestimation by removing

Category	Name	Num attributes	Split	Num records	Num matches	Num entities	Duplication score
Small dataset with few duplicates	Music Brainz 20k [39]	5	20k	19,375	16,250	10,000	48.39%
Small dataset with multiple duplicates	DBLP-Scholar [28]	2	N/A	7,626	13,760	2352	69.17%
	WDC xlarge (computers)*	4	N/A	4,676	9,990	1,080	76.92%
	Alaska (monitor) [6]	1	N/A	2,273	12,949	231	89.88%
	Cars [18]	6	N/A	16,185	5,954,651	48	99.71%
Large dataset with few duplicates	Music Brainz 200k [39]	5	200k	193,750	162,500	100,000	48.39%
	Music Brainz 2M [39]	5	2M	1,937,500	1,624,503	1,000,000	48.39%
	North Carolina Voters 5M [39]	4	5M	5,000,000	3,331,384	3,500,840	29.98%
Large dataset with multiple duplicates	Cars 1M	6	1M	1,000,024	22,762,570,161	48	99.99%
	WDC xlarge (computers) 1M*	4	1M	1,042,500	5,462,382,825	1,080	99.90%

* http://webdatacommons.org/largescaleproductcorpus/v2

Table 2: List of datasets used for the experiments

Dataset	Max len	Green edges	Red edges
Music-Brainz-20k	256	2000	2000
DBLP-Scholar	256	1000	1000
WDC-xlarge-computers	512	4000	7000
Alaska-monitor	256	2500	2500
Cars	256	15000	15000
Music-Brainz-200k	256	5000	5000
Music-Brainz-2M	256	5000	5000
North Carolina Voters-5M	256	5000	5000
Cars-1M	256	20000	20000
WDC-xlarge-computers-1M	256	20000	20000

Table 3: Training hyperparameters and dataset-specific settings for Ditto

additional matching edges. For graph cleaning, we use Ditto [25], evaluating both DistilBERT [40] and RoBERTa [26]. To enhance efficiency, record pairs are processed in batches of 1024, and transitivity is applied to infer matches from classified pairs, reducing redundant queries. Ditto is trained consistently across datasets with 20 epochs, a batch size of 32, and a learning rate of $3 \cdot 10^{-5}$ (AdamW optimizer). Datasets are split into 70% training, 20% testing, and 10% validation, with a fixed number of sampled matching/non-matching edges. Dataset-specific parameters, such as sequence length and the number of sampled edges, are listed in Table 3.

Simulation. The Simulation pipeline approximates the ML ER pipeline by replacing direct ER model queries with a calibration process. We construct an equiwidth histogram with 10 bins covering similarity values from 0 to 1. For each bin, we sample up to 100 record pairs (small datasets) or 1000 pairs (large datasets) and classify them using Ditto, with separate calibration for DistilBERT and RoBERTa. The matching probability for each bin is computed as the average outcome of sampled pairs. When processing an edge, its probability is determined by its bin, and a matching outcome is sampled from a Bernoulli distribution. This avoids invoking the ER model for every edge, improving efficiency. To enhance robustness, additional edges within the same cluster as matched edges are queried [16]. Edges are processed randomly with a fixed seed, and transitivity is applied to maintain consistency in inferred matches.

Statistical ER. For this pipeline, we use Blink [42], an unsupervised Bayesian ER method that operates on strings and categorical features. In our implementation, all features across the datasets were treated as string attributes to ensure compatibility with Blink. We consider two variants of the pipeline, differing in the similarity function used: one based on an MPNet-based Sentence Transformer [37, 41] and the other using an edit distance-based similarity function (the default in Blink). The MPNet variant processes string features into dense representations, while the edit distance variant directly computes pairwise similarities based on character-level differences. The model processes these string features and handles both the record representation and noisy graph generation steps. Instead of explicitly constructing a graph, Blink assigns entity identifiers probabilistically during a Gibbs sampling process [42], which estimates posterior probabilities for linking records to latent entities. We used the same hyperparameters as in the original Blink paper ($a = 1, b = 99$). The output is a union of cliques, where records sharing the same entity identifier are grouped together.

LLM embedding. For this pipeline, we leverage two variants. The first variant uses OpenAI's text-embedding-3-large [31] to generate dense vector representations of records, while the second variant relies on an edit distance-based similarity function. For the clustering step, we use DBSCAN [10], a density-based algorithm that does not require the number of clusters to be pre-specified. In the OpenAI embedding variant, clustering is performed on the cosine distances between the generated embeddings, whereas in the edit distance variant, clustering operates on pairwise similarity scores derived from edit distance computations. The LLM Embedding pipeline does not explicitly construct a noisy graph. Instead, the clustering process operates directly on the computed distances. As DBSCAN groups records based on their relative distances, it implicitly generates a union of cliques, with each clique corresponding to a unique entity. We set the *min sample* parameter to 1 and used $eps = 0.65$ for the OpenAI variant and $eps = 0.35$ for the edit variant.

7.4 Methodology

We ran experiments on multiple datasets to evaluate the performance of the base and sampling pipelines in terms of both efficiency (running time) and accuracy (approximation error). The reported execution times exclude Ditto training and Simulation calibration,

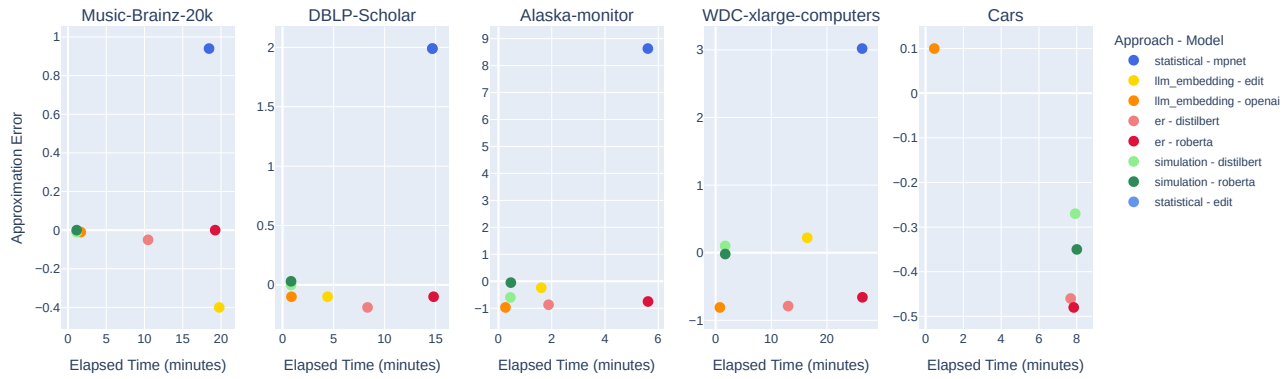

Figure 3: Base pipeline performance on small-sized datasets for low and high duplication factor datasets

as these are pre-inference steps specific to each dataset, and including them would blur the distinction between training and inference, making runtime comparisons with other pipelines misleading. For the LLM Embedding pipeline, embeddings were not precomputed, and their computation time is included in the total execution time. Requests to the OpenAI embedding model were batched with a batch size of 512, with an average time cost of 0.006 seconds per record. Sampling pipelines were evaluated at four sampling ratios: 1%, 5%, 10%, and 20%. Each was run five times with different random seeds. After each run, we applied the estimator from [23] to upscale the entity count, and calculated the average approximation error over the five runs. For datasets with over 1 million records, we applied embedding binarization in the base LLM Embedding pipeline to mitigate memory issues. For the statistical approach, in both the baseline and sampling pipelines, and across both similarity function variants, we introduced a time-based stopping condition. Specifically, Blink runs for at most the same duration as the ER pipeline with RoBERTa, ensuring that its execution remains within a comparable computational budget. In addition to that, Blink requires that records are organized into "files" (tables), where duplicated records are spread across different tables. Each record was assigned a file ID to separate duplicates. Since Blink requires at least two records per file, we resampled when the sample resulted in files with only one record, ensuring all tables had at least two records.

7.5 Experimental results

We evaluate the accuracy and efficiency of the base and sampling pipelines across datasets, running experiments on a server with an AMD Ryzen 9 5950X, 64 GB RAM, and an NVIDIA RTX 3090. Overall, LLM Embedding and Simulation pipelines perform well on small datasets, while Statistical and ML ER pipelines face scalability issues on larger ones. Sampling-based pipelines significantly reduce runtime, but their accuracy depends on the dataset's duplication rate and the upscaling process. Furthermore, we observe that while sampling-based pipelines offer considerable improvements in computational time, their accuracy is influenced by both the dataset's duplication factor and the upscaling process.

On small datasets simulation and LLM embedding perform better compared to the other base pipelines. Figure 3 shows that across small datasets, Simulation (RoBERTa, DistilBERT) and LLM Embedding (OpenAI) consistently achieve the lowest approximation errors. For instance, on Music-Brainz-20k, Simulation with DistilBERT has no error (0%), followed by LLM Embedding (OpenAI) and Simulation with RoBERTa at −1%. A similar trend holds for DBLP-Scholar, where Simulation (DistilBERT) remains at 0% error, with RoBERTa close behind at 3%, suggesting that similarity-based clustering works well even in moderately duplicated datasets. For highly duplicated datasets, however, results shift. On Alaska-monitor and WDC-xlarge-computers, Simulation with RoBERTa performs best, while LLM Embedding (OpenAI) struggles with errors of -97% and -81%. These datasets contain lengthy product descriptions (121 and 200 tokens on average, using *cl100k_base* encoding), whereas other datasets have at most 90 tokens per record. The increased text length makes it harder for OpenAI's model to generate effective representations. On the other hand, for the Cars dataset, where nearly all records are duplicates, LLM Embedding (OpenAI) provides the best estimate (10% error), while Simulation performs worse due to low precision (~36%), leading to excessive merging of clusters and errors of -27% (DistilBERT) and -35% (RoBERTa). LLM Embedding (edit) is omitted from Figure 4 due to its high error (72%) and long runtime (1.71 hours). Comparing pipeline types, Simulation consistently outperforms ER in both accuracy and runtime. The ER pipeline, though closer in accuracy to LLM Embedding (OpenAI), is significantly slower, particularly with RoBERTa. Still, all pipelines complete within 30 minutes on small datasets. The Statistical pipeline, however, performs the worst, often overestimating due to the limited number of Gibbs iterations (e.g., 5 on Music-Brainz-20k with MPNet), whereas prior work [42] suggests a large number of iterations (e.g. 10,000) are needed for reliable results. Pipeline variants also show clear differences. LLM Embedding (OpenAI) consistently outperforms its edit distance variant, reinforcing the advantage of embeddings over token-based approaches. Similarly, RoBERTa performs better than DistilBERT in both ER and Simulation, at the cost of longer runtimes. Finally, while MPNet and edit distance Statistical pipelines yield similar results on DBLP-Scholar, the edit variant fails to run within the time cap on other datasets due to its higher computational cost, highlighting its scalability limitations.

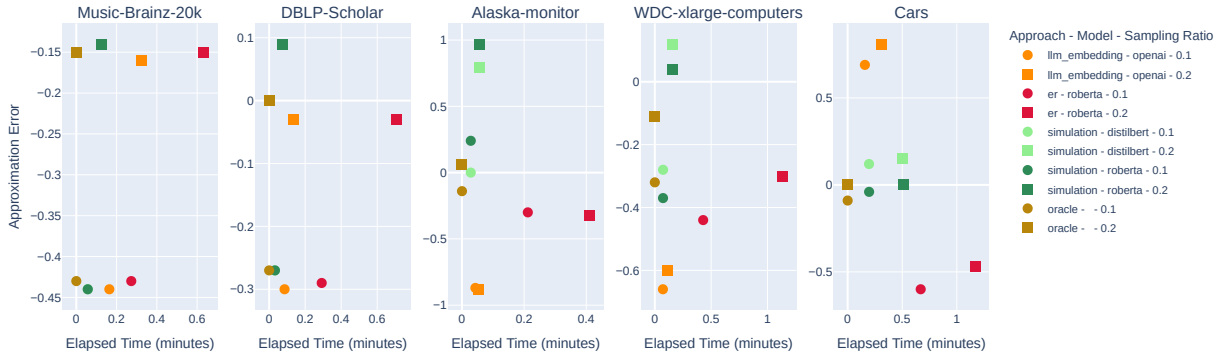

Figure 4: Sampling-based pipeline results on a selection of small datasets

Dataset name	Sampling ratio			
	1%	5%	10%	20%
DBLP-Scholar	1.67	9.02	15.43	25.77
Alaska-monitor	4.69	24.85	42.19	60.47
Music-Brainz 2M	0.78	4.03	7.91	14.83
North Carolina Voters 5M	0.66	3.2	6.12	11.21
Cars 1M	99.53	99.91	99.95	99.98
WDC-xlarge-computers	94.22	98.5	99.17	99.54

Table 4: Average sample data duplication factor across different sampling ratios for various datasets

Base pipelines fail to scale for million-scale datasets. We assess the base pipelines on larger datasets, starting with Music-Brainz 200k, where most pipelines run successfully except for the Statistical pipeline, which fails due to Blink's memory-intensive similarity matrix computation. The same performance patterns observed in small datasets hold: Simulation (RoBERTa) achieves the most accurate estimate (-2% error in ~70 min), followed by Simulation (DistilBERT) (-9% in ~70 min) and LLM Embedding (OpenAI) (-11% in 39 min). The ER pipeline (RoBERTa) performs comparably to LLM Embedding (OpenAI) but is far slower, while DistilBERT-based ER completes faster but with lower accuracy. The LLM Embedding (Edit) variant is the slowest, requiring ~33 hours and yielding a high error of -65%. For the million-scale datasets, scalability remains a significant challenge across all pipelines. The ER and Simulation pipelines fail due to the sheer size of the noisy graph, particularly for high-duplication datasets where the number of edges reaches hundreds of millions, making it infeasible to store and process within memory constraints. The Statistical pipeline does not scale due to Blink's reliance on precomputing similarity matrices for each attribute, which leads to prohibitive memory requirements. The LLM Embedding pipeline also encounters memory limitations, as embeddings for large datasets are stored in memory, quickly consuming available resources and causing the pipeline to run out of memory before clustering can be completed.

The upscaling estimator can introduce errors and is sensitive to the duplication factor. Before evaluating the sampling pipelines, we analyze the impact of the Klusowski Bernoulli estimator on entity count estimation. To isolate upscaling errors, we use an oracle sampling pipeline that assumes perfect clustering, meaning any discrepancy in entity count stems solely from the upscaling process. The results in Figures 4 and 5 reveal that upscaling introduces errors, particularly at smaller sampling ratios (e.g., 1% and 5%) on large datasets. The only exception was the Cars 1M dataset, consistently showing 0% approximation error even with just 1% sampling (~10,000 records). Since the dataset contains only 48 entities, even a small sample is likely to include at least one record per entity, which may help upscaling remain accurate. In contrast, low-duplication datasets are more prone to errors at small sampling ratios because the sampled data may cover only a fraction of the actual entities. As shown in Table 4, the duplication factor in the sample is often much lower than in the full dataset, making estimation more challenging. This is reflected in Figure 4, where the oracle pipeline shows higher approximation errors in low-duplication datasets like Music-Brainz-200k. Increasing the sampling ratio helps mitigate this issue, and in cases like DBLP-Scholar and WDC-xlarge-computers, the error approaches zero at 20% sampling. Interestingly, in high-duplication datasets like WDC-xlarge-computers-1M and Alaska, the upscaling error shifts from underestimation at low sampling ratios to overestimation at 20%, suggesting that duplication characteristics play a crucial role in upscaling accuracy.

Sampling-based pipelines can perform comparably to base pipelines on small datasets in significantly less time. Figure 4 presents the performance of sampling-based pipelines on small datasets. Due to high errors at lower sampling ratios, we focus on 10% and 20% sampling. For the ER pipeline, we focus on the RoBERTa variant, while for LLM Embedding, we consider the OpenAI variant. The statistical sampling pipeline did not complete within the time cap, set to match the execution time of the ER pipeline (RoBERTa). All sampling pipelines completed within two minutes across small datasets, with Simulation (RoBERTa) performing best in both accuracy and runtime. On low-duplication datasets like Music-Brainz-20k, the ER (RoBERTa), Simulation (RoBERTa), and LLM Embedding (OpenAI) pipelines had comparable approximation errors. At 20% sampling, Simulation (RoBERTa) had a -13% error in 14 seconds, while its base counterpart achieved near-zero error in 68 seconds (5× slower). Similarly, the ER pipeline (RoBERTa) at 20% sampling had a -15% error in 2.5 minutes, compared to its base pipeline, which also achived a near-zero error but required 19 minutes (7.6× slower). On DBLP-Scholar (20% sampling), the ER (RoBERTa), Simulation, and LLM Embedding (OpenAI) pipelines

performed comparably to their base counterparts. ER (RoBERTa) and LLM Embedding (OpenAI) achieved -3% error, compared to -10% for their base versions, with 21× and 6× speedups, respectively. The Simulation pipeline had a slightly higher error (-9% vs. -3%) but ran 11× faster. On high-duplication datasets, results varied. While increasing the sampling ratio to 20% improved accuracy on Music-Brainz-20k by reducing upscaling errors, the opposite was observed on Alaska. For this specific dataset, Simulation with DistilBERT performed best at 10% sampling, achieving 0% error in 3 seconds, whereas its base version had 10% error and took 1.71 minutes. On WDC-xlarge-computers and Cars, Simulation with RoBERTa (20% sampling) performed best, yielding 15% and 0% error in 7 and 30 seconds, respectively. The base counterparts were slower, with WDC-xlarge-computers taking 103 seconds (-2% error) and Cars requiring 8 minutes with -35% error.

Sampling-based pipelines can scale but can provide reasonable estimates only on very highly duplicated datasets. We evaluate the performance of sampling-based pipelines on large datasets in Figure 5. To handle large-scale datasets, we focus on the 1% and 5% sampling levels and limit our evaluation to the to the Simulation (DistilBERT and RoBERTa), ER (RoBERTa), and LLM Embedding (OpenAI) pipelines. On low-duplication datasets, we see significant underestimation at the 1% sampling level, with approximation errors approaching -100%, driven by both the small sample size and errors introduced by the oracle. Increasing the sampling ratio to 5% mitigates some of the underestimation but still results in substantial errors, ranging from -60% to -70% on Music-Brainz-200k and Music-Brainz-2M. In terms of efficiency, sampling-based approaches are significantly faster but at the cost of accuracy. For instance, the ER pipeline with RoBERTa completes in 2.5 minutes with an approximation error of -63%, while its base counterpart achieves a much lower approximation error of -12% but requires nearly 18 hours to complete. For North Carolina Voters-5M, only LLM Embedding with OpenAI was able to run at 5% sampling, yielding an approximation error of -93%. The Simulation and ER pipelines encountered memory errors due to the large size of the blocking graph, which exceeded available memory capacity. On high-duplication datasets, such as Cars-1M and WDC-1M, sampling pipelines perform significantly better. The reduced impact of upscaling errors contributes to this improvement. On Cars-1M, LLM Embedding with OpenAI at 5% sampling achieves an approximation error of -12% in just 6 minutes, outperforming both ER and Simulation pipelines, which struggle due to the high sparsity of the blocking graph. For instance, with 5% of the data, the blocking graph of Cars-1M had an average sparsity of over 40, meaning that even with the oracle sampling pipeline, the entity count would be overestimated by a factor of 40 due to the fragmentation of true entities across multiple disconnected components. On WDC-1M, similarly to North Carolina Voters-5M, the ER and Simulation pipelines encountered memory errors at the 5% sampling level due to the excessive size of the blocking graph. Nevertheless, the best-performing approach was Simulation with DistilBERT using 1% of the data, which achieved an approximation error of -42%. LLM Embedding with OpenAI struggled despite using 5% of the data, yielding an approximation error of -58%. This is likely due to the lengthy record representations in WDC-1M, which contain detailed product descriptions, making it challenging for OpenAI embeddings to effectively differentiate records belonging to distinct entities.

Impact of sampling strategies on ECE. We examine how different sampling strategies affect entity count estimation. Our previous experiments use the estimator from [23], which assumes Bernoulli sampling. While SRS (simple random sampling) could be used instead, it would break these assumptions, making direct comparison difficult. To explore this, we turn to [15], which introduces separate estimators for Bernoulli sampling and SRS. These estimators, though unbiased, suffer from high variance, often producing inaccurate or even negative entity count estimates. For this reason, they were not included in our main experiments. Rather than focusing on accuracy, we use these estimators to compare the impact of sampling strategies. We evaluate the Oracle, ER (RoBERTa), and LLM Embedding pipelines on Music-Brainz-20K and DBLP-Scholar, using Bernoulli sampling ratios of 10% and 20% and selecting N in SRS to match these proportions (Table 5). Both estimators lead to high approximation errors and large standard deviations, making comparisons difficult. In some cases, mean approximation errors vary significantly (e.g., ER with RoBERTa on Music-Brainz-20K, all pipelines on DBLP-Scholar), while in others, they remain close (e.g., Oracle pipeline on Music-Brainz-20K). A promising direction for future work is adapting [23] to develop an estimator specifically for SRS, allowing for direct comparisons with its Bernoulli-based counterpart.

8 KEY TAKEAWAYS

Our experiments show that sampling-based pipelines scale well for entity count estimation, particularly in high-duplication datasets like Cars and Cars 1M. However, accuracy depends on factors such as the upscaling method, sampling ratio, and dataset characteristics. In low-duplication datasets (e.g., Music Brainz 200k), errors were higher, suggesting the need for refined sampling or adaptive upscaling. Table 6 summarizes the best-performing pipelines across different dataset conditions. Simulation (RoBERTa, DistilBERT) consistently achieved high accuracy. LLM Embedding (OpenAI) also performed well, providing accurate results on small datasets regardless of duplication levels, though with slightly lower accuracy than Simulation. It was particularly effective on large datasets with multiple duplicates (e.g., Cars 1M), except in cases where datasets contained very long records, such as WDC-xlarge-computers-1M. The ML ER pipeline (DistilBERT, RoBERTa) showed accuracy comparable to LLM Embedding (OpenAI) on some small datasets with both few and multiple duplicates (e.g., Music-Brainz-20k, DBLP-Scholar), but performed poorly on large datasets, regardless of their duplication score (e.g., Music-Brainz-2M, Cars 1M). The Statistical ER pipeline consistently overestimated, struggling to model duplication patterns when samples lacked diversity. Alternative estimators like Berg [15] showed extreme variance, leading to unreliable results despite being theoretically unbiased. While all pipelines are adapted from ER methodologies, their underlying strategies differ: ML ER explicitly resolves entities before counting, Simulation estimates counts via similarity-based probability mapping, Statistical ER probabilistically infers entity identifiers, and LLM Embedding clusters records via learned representations. These variations lead

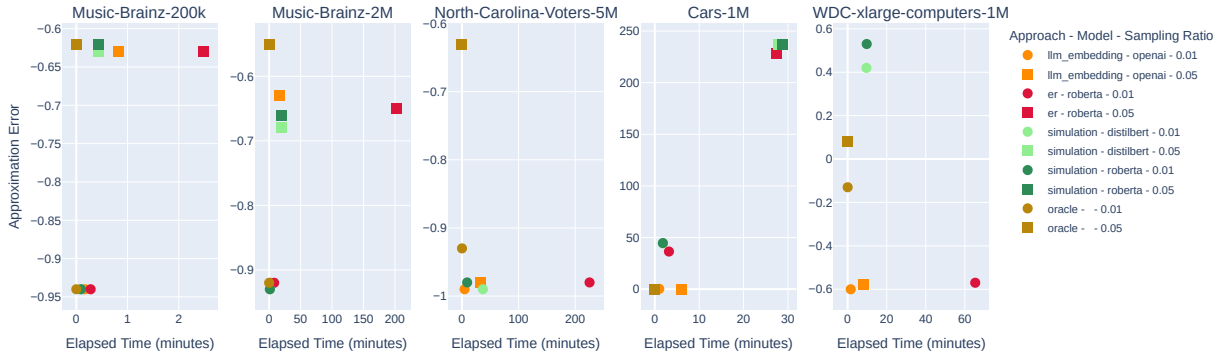

Figure 5: Sampling-based pipeline results on large datasets.

Pipeline	Music-Brainz-20k			
	Bernoulli (10%)	**SRS (10%)**	**Bernoulli (20%)**	**SRS (20%)**
Oracle	49.89 ± 8.56	23.59 ± 0.75	42.25 ± 5.83	36.01 ± 0.79
ER (RoBERTa)	47.12 ± 19.85	100.38 ± 107.54	$9.88 \cdot 10^5 \pm 1.83 \cdot 10^6$	$6.20 \cdot 10^3 \pm 8.71 \cdot 10^3$
LLM embedding (openai)	$-2.20 \cdot 10^8 \pm 5.04 \cdot 10^8$	$-2.30 \cdot 10^8 \pm 5.25 \cdot 10^8$	$-9.97 \cdot 10^8 \pm 1.89 \cdot 10^9$	$-1.06 \cdot 10^9 \pm 2.01 \cdot 10^9$

Pipeline	DBLP-Scholar			
	Bernoulli (10%)	**SRS (10%)**	**Bernoulli (20%)**	**SRS (20%)**
Oracle	$3.79 \cdot 10^3 \pm 8.59 \cdot 10^3$	$6.41 \pm 0.37 \cdot 10^{-1}$	$3.49 \cdot 10^3 \pm 7.32 \cdot 10^3$	9.14 ± 0.52
ER (RoBERTa)	$2.45 \cdot 10^7 \pm 5.60 \cdot 10^7$	$-2.07 \cdot 10^{14} \pm 3.58 \cdot 10^{14}$	$4.03 \cdot 10^9 \pm 7.52 \cdot 10^9$	$-3.02 \cdot 10^{16} \pm 5.23 \cdot 10^{16}$
LLM embedding (openai)	$-2.35 \cdot 10^6 \pm 6.42 \cdot 10^6$	$3.16 \cdot 10^5 \pm 7.37 \cdot 10^5$	$-3.99 \cdot 10^6 \pm 7.31 \cdot 10^6$	$-6.66 \cdot 10^7 \pm 1.36 \cdot 10^8$

Table 5: Approximation errors using the Bernoulli and SRS estimators in [15] for Music-Brainz-20k (top) and DBLP-Scholar (bottom)

Dataset size	Duplicates	
	Small duplication (≤ 0.5)	High duplication (> 0.5)
Small	Simulation (RoBERTa)	Simulation (DistilBERT) Simulation (RoBERTa)
Large	LLM Embedding (OpenAI), Simulation (DistilBERT) Simulation (RoBERTa)	LLM Embedding (OpenAI), Simulation (DistilBERT) Simulation (RoBERTa)

Table 6: Best-performing pipelines across different dataset conditions

to different accuracy-efficiency trade-offs, as reflected in our empirical findings.

Discussion. Most pipelines relied on in-memory techniques, limiting scalability on large datasets. The ML ER and Simulation pipelines used standard token-blocking, but adopting metablocking techniques [9, 32, 33] could improve scalability. Similarly, the LLM clustering pipeline required in-memory embeddings and clustering, which could be mitigated by scalable clustering algorithms [11]. The Statistical ER pipeline also faced in-memory issues with Blink.

9 CONCLUSIONS AND FUTURE WORK

This paper presents the entity count estimation problem from a data management perspective and proposes a general framework that unifies techniques from machine learning-based ER, statistical record linkage, and clustering. A key contribution is the exploration of a sampling-based approach that estimates entity counts from a subset and upscales the result to the full dataset, improving scalability on large data. Our findings show that while sampling pipelines greatly improve efficiency, especially on high-duplication datasets like Cars 1M, accuracy suffers on low-duplication datasets due to upscaling errors. In some cases, these errors counterbalanced sample inaccuracies, highlighting both strengths and limitations. There are several directions for future work. Developing new estimators for Bernoulli sampling could enhance robustness. Adapting upscaling methods for alternative sampling strategies, such as subgraph-based sampling, would require new estimators with tailored statistical guarantees. Expanding the study to advanced statistical ER methods like d-Blink could improve generalizability. Beyond sampling, parallelism and distributed processing could further enhance scalability. Lastly, addressing memory constraints through streaming algorithms could extend applicability to even larger datasets.

ACKNOWLEDGMENTS

The authors thank the reviewers for providing insightful comments. This work was partially supported by the HORIZON Research and Innovation Action 101135576 INTEND "Intent-based data operation in the computing continuum", the SEED PNR Project "Frontiers in Linking records: knOWledge graphs, Explainability and tempoRal data", and the Sapienza Research Project B83C22007180001 "Trustworthy Technologies for Augmenting Knowledge Graphs". Jerin George Mathew was financed by the Italian National PhD Program in AI.

REFERENCES

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod record* 28, 2 (1999), 49–60.

[2] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine learning* 56 (2004), 89–113.

[3] Mikhail Bilenko, Beena Kamath, and Raymond J Mooney. 2006. Adaptive blocking: Learning to scale up record linkage. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 87–96.

[4] Nicolas Bruno, YongChul Kwon, and Ming-Chuan Wu. 2014. Advanced join strategies for large-scale distributed computation. *PVLDB* 7, 13 (2014), 1484–1495.

[5] Yu Chen and Ke Yi. 2017. Two-level sampling for join size estimation. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 759–774.

[6] Valter Crescenzi, Andrea De Angelis, Donatella Firmani, Maurizio Mazzei, Paolo Merialdo, Federico Piai, and Divesh Srivastava. 2021. Alaska: A flexible benchmark for data integration tasks. *arXiv preprint arXiv:2101.11259* (2021).

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[8] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *PVLDB* 11, 11 (2018), 1454–1467.

[9] Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, and Themis Palpanas. 2015. Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data. In *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 411–420.

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (Portland, Oregon) *(KDD'96)*. AAAI Press, 226–231.

[11] Adil Fahad, Najlaa Alshatri, Zahir Tari, Abdullah Alamri, Ibrahim Khalil, Albert Y Zomaya, Sebti Foufou, and Abdelaziz Bouras. 2014. A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE transactions on emerging topics in computing* 2, 3 (2014), 267–279.

[12] Ivan P Fellegi and Alan B Sunter. 1969. A theory for record linkage. *J. Amer. Statist. Assoc.* 64, 328 (1969), 1183–1210.

[13] Stephen E Fienberg. 1972. The multiple recapture census for closed populations and incomplete 2k contingency tables. *Biometrika* 59, 3 (1972), 591–603.

[14] Donatella Firmani, Barna Saha, and Divesh Srivastava. 2016. Online entity resolution using an oracle. *PVLDB* 9, 5 (2016), 384–395.

[15] Ove Frank. 1978. Estimation of the number of connected components in a graph by using a sampled subgraph. *Scandinavian Journal of Statistics* (1978), 177–188.

[16] Sainyam Galhotra, Donatella Firmani, Barna Saha, and Divesh Srivastava. 2018. Robust entity resolution using random graphs. In *Proceedings of the 2018 International Conference on Management of Data*. 3–18.

[17] Sainyam Galhotra, Donatella Firmani, Barna Saha, and Divesh Srivastava. 2021. Beer: blocking for effective entity resolution. In *Proceedings of the 2021 International Conference on Management of Data*. 2711–2715.

[18] Sainyam Galhotra, Donatella Firmani, Barna Saha, and Divesh Srivastava. 2021. Efficient and effective ER with progressive blocking. *The VLDB Journal* 30, 4 (2021), 537–557.

[19] Kai Hildebrandt, Fabian Panse, Niklas Wilcke, and Norbert Ritter. 2020. Large-Scale Data Pollution with Apache Spark. *IEEE Transactions on Big Data* 6, 2 (2020), 396–411. https://doi.org/10.1109/TBDATA.2016.2637378

[20] S Hochreiter. 1997. Long Short-term Memory. *Neural Computation MIT-Press* (1997).

[21] Yannis E Ioannidis and Stavros Christodoulakis. 1991. On the propagation of errors in the size of join results. In *Proceedings of the 1991 ACM SIGMOD International Conference on Management of data*. 268–277.

[22] Ashish Khetan, Harshay Shah, and Sewoong Oh. 2018. Number of Connected Components in a Graph: Estimation via Counting Patterns. *arXiv preprint arXiv:1812.00139* (2018).

[23] Jason M. Klusowski and Yihong Wu. 2020. Estimating the number of connected components in a graph via subgraph sampling. *Bernoulli* 26, 3 (2020), 1635 – 1664. https://doi.org/10.3150/19-BEJ1147

[24] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *PVLDB* 3, 1-2 (2010), 484–493.

[25] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *PVLDB* 14, 1 (2020), 50–60.

[26] Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* 364 (2019).

[27] Neil G Marchant, Andee Kaplan, Daniel N Elazar, Benjamin IP Rubinstein, and Rebecca C Steorts. 2021. d-blink: Distributed end-to-end Bayesian entity resolution. *Journal of Computational and Graphical Statistics* 30, 2 (2021), 406–421.

[28] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 international conference on management of data*. 19–34.

[29] Magnus Müller, Daniel Flachs, and Guido Moerkotte. 2021. Memory-efficient key/foreign-key join size estimation via multiplicity and intersection size. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 984–995.

[30] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pre-training. *arXiv preprint arXiv:2201.10005* (2022).

[31] OpenAI. 2024. New embedding models and API updates. https://openai.com/blog/new-embedding-models-and-api-updates Last accessed 2024-07-18.

[32] George Papadakis, Ekaterini Ioannou, Emanouil Thanos, and Themis Palpanas. 2021. *The four generations of entity resolution*. Springer.

[33] George Papadakis, Georgia Koutrika, Themis Palpanas, and Wolfgang Nejdl. 2013. Meta-blocking: Taking entity resolutionto the next level. *IEEE Transactions on Knowledge and Data Engineering* 26, 8 (2013), 1946–1960.

[34] Thorsten Papenbrock, Arvid Heise, and Felix Naumann. 2014. Progressive duplicate detection. *IEEE Transactions on knowledge and data engineering* 27, 5 (2014), 1316–1329.

[35] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[36] Anna Primpeli, Ralph Peeters, and Christian Bizer. 2019. The WDC training dataset and gold standard for large-scale product matching. In *Companion Proceedings of The 2019 World Wide Web Conference*. 381–386.

[37] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[38] Kexin Rong, Yao Lu, Peter Bailis, Srikanth Kandula, and Philip Levis. 2020. Approximate partition selection for big-data workloads using summary statistics. *PVLDB* 13, 12 (July 2020), 2606–2619. https://doi.org/10.14778/3407790.3407848

[39] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2017. Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In *Advances in Databases and Information Systems: 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings 21*. Springer, 278–293.

[40] V Sanh. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[41] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems* 33 (2020), 16857–16867.

[42] Rebecca C. Steorts. 2015. Entity Resolution with Empirically Motivated Priors. *Bayesian Analysis* 10, 4 (2015), 849 – 875. https://doi.org/10.1214/15-BA965SI

[43] Andrea Tancredi, Steorts Rebecca, Brunero Liseo, et al. 2020. A unified framework for de-duplication and population size estimation (with Discussion). *BAYESIAN ANALYSIS* 15, 2 (2020), 633–658.

[44] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep learning for blocking in entity matching: a design space exploration. *PVLDB* 14, 11 (2021), 2459–2472.

[45] David Vengerov, Andre Cavalheiro Menck, Mohamed Zait, and Sunil P Chakkappen. 2015. Join size estimation subject to filter conditions. *PVLDB* 8, 12 (2015), 1530–1541.

[46] Norases Vesdapunt, Kedar Bellare, and Nilesh Dalvi. 2014. Crowdsourcing algorithms for entity resolution. *PVLDB* 7, 12 (2014), 1071–1082.

[47] William E Winkler. 2014. Matching and record linkage. *Wiley interdisciplinary reviews: Computational statistics* 6, 5 (2014), 313–325.

[48] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1149–1164.

[49] Shinjae Yoo, Hao Huang, and Shiva Prasad Kasiviswanathan. 2016. Streaming spectral clustering. In *2016 IEEE 32nd international conference on data engineering (ICDE)*. IEEE, 637–648.

[50] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM sigmod record* 25, 2 (1996), 103–114.