`d-blink`: Distributed End-to-End Bayesian Entity Resolution

Neil G. Marchant[a] Andee Kaplan[b] Daniel N. Elazar[c]
Benjamin I. P. Rubinstein[a] Rebecca C. Steorts[d]

[a]School of Computing and Information Systems, University of Melbourne
[b]Department of Statistics, Colorado State University
[c]Methodology Division, Australian Bureau of Statistics
[d]Department of Statistical Science and Computer Science, Duke University
Principal Mathematical Statistician, United States Census Bureau
DRB #: CBDRB-FY20-309

September 14, 2020

Abstract

Entity resolution (ER; also known as record linkage or de-duplication) is the process of merging noisy databases, often in the absence of unique identifiers. A major advancement in ER methodology has been the application of Bayesian generative models, which provide a natural framework for inferring latent entities with rigorous quantification of uncertainty. Despite these advantages, existing models are severely limited in practice, as standard inference algorithms scale quadratically in the number of records. While scaling can be managed by fitting the model on separate blocks of the data, such a naïve approach may induce significant error in the posterior. In this paper, we propose a principled model for scalable Bayesian ER, called "distributed Bayesian linkage" or `d-blink`, which jointly performs blocking and ER without compromising posterior correctness. Our approach relies on several key ideas, including: (i) an auxiliary variable representation that induces a partition of the entities and records into blocks; (ii) a method for constructing well-balanced blocks based on k-d trees; (iii) a distributed partially-collapsed Gibbs sampler with improved mixing; and (iv) fast algorithms for performing Gibbs updates. Empirical studies on six data sets—including a case study on the 2010 Decennial Census—demonstrate the scalability and effectiveness of our approach.

Keywords: auxiliary variable, distributed computing, Markov chain Monte Carlo, partially-collapsed Gibbs sampling, record linkage

1 INTRODUCTION

When information about a statistical population is scattered across multiple databases, there may be immense value in combining them. A combined database can provide a more accurate and complete view of the population by improving coverage, bringing together analytic variables, and resolving erroneous and missing values. This allows statisticians to draw richer and more reliable conclusions. Among the types of questions that can be addressed by combining such databases are the following: How accurate are census enumerations for minority groups (Winkler, 2006)? How many of the elderly are at high risk for sepsis in different parts of the country (Saria, 2014)? How many people were victims of war crimes in recent conflicts in Syria (Price et al., 2013)?

An important step when combining databases is identifying records that refer to the same statistical unit. This is challenging in practice because consistent identifiers, such as social security numbers, are often not available. Identifiers may be omitted due to privacy concerns, they may be inconsistent across the databases, or they may have never been recorded. In such cases, practitioners must rely on *entity resolution* (ER) to infer the relationships between records and statistical units (entities) using linking variables in the observed data. This problem is studied in the statistics, machine learning, database and natural language processing communities, and is also known as entity disambiguation, merge-purge, record linkage, deduplication and co-reference resolution (Christen, 2012b; Dong and Srivastava, 2015; Soon et al., 2001).

ER is not only a crucial tool for statistical analysis, it is also a challenging statistical and computational problem in itself. This is because many databases lack reliable linking variables, the record comparison space scales quadratically in the number of records, and the number of parameters to be estimated grows with the number of records (Herzog et al., 2007; Lahiri and Larsen, 2005; Winkler, 1999, 2000). To meet present and near-future needs, ER methods must be flexible and scalable to large databases. Furthermore, they must be able to handle uncertainty and be easily integrated with post-ER statistical analyses, such as regression. All of this must be done while achieving low error rates.

Bayesian models offer a promising framework for ER as they support natural uncertainty

propagation, flexible modeling assumptions, and incorporation of prior information. However, existing Bayesian ER models either ignore scalability (Steorts, 2015; Zanella et al., 2016; Sadinle, 2017) or manage scalability in an unprincipled manner by applying blocking outside the Bayesian framework (Fortini et al., 2001; Larsen, 2005, 2012; Tancredi and Liseo, 2011; Gutman et al., 2013; Sadinle, 2014; Steorts et al., 2016). Blocking improves scalability by partitioning records into blocks and assuming records in different blocks do not refer to the same entity (Christen, 2012a). However, when blocking is performed as a separate deterministic step it is not possible to propagate the uncertainty. Moreover, since the blocks are fixed, a poor blocking design may compromise the accuracy of the entire ER process. In other words, one sacrifices uncertainty propagation and accuracy for scalability.

In this paper, we propose a principled approach to scaling Bayesian ER models, which does not suffer from the limitations of ad-hoc deterministic blocking. Using the `blink` ER model (Steorts, 2015) as a foundation, we propose a scalable and distributed extension called "distributed `blink`" or `d-blink` for short, which integrates probabilistic blocking in a fully Bayesian framework. To our knowledge, `d-blink` is the first Bayesian ER model which supports propagation of uncertainty between the blocking and matching/linking stages of ER, without compromising the correctness of the posterior. In addition, `d-blink` supports distributed/parallel inference at the block level to further improve scalability to large databases.

We make several contributions to the literature. First, we propose an auxiliary variable representation of `blink`, which induces a partitioning of the entities and records into blocks. These play a similar role as traditional deterministic blocks, however the assignments of latent entities and records to blocks are *random* and inferred *jointly* with the other model parameters. Second, we prove that our auxiliary variable representation preserves the marginal posterior distribution over the model parameters. This is a desirable property, as it means our inferences are theoretically independent of the blocking design. Third, we propose a method for constructing well-balanced blocks based on k-d trees. Fourth, we design a distributed partially-collapsed Gibbs sampler to perform inference, and demonstrate superior mixing times when compared to a standard Gibbs sampler. Fifth, we propose

algorithms for improving computational efficiency of the Gibbs updates which leverage indexing data structures and a novel perturbation sampling algorithm.

We implement our proposed methodology as an open-source Apache Spark package[1] and provide an R interface for broad accessibility[2]. We conduct empirical evaluations on two synthetic and three real data sets, demonstrating efficiency gains in excess of 300× compared to `blink`. To illustrate the effectiveness of our approach for realistic ER tasks, we present a case study using Census and administrative data from the U.S. state of Wyoming.

The paper is organized as follows. In Section 2 we review related work in ER methodology and approximate inference algorithms. We then formulate ER in a Bayesian setting in Section 3, and present the `d-blink` model with integrated probabilistic blocking. In Section 4 we provide guidelines for selecting blocking functions. We then discuss inference and propose a distributed partially-collapsed Gibbs sampler in Section 5. We suggest additional methods for improving computational efficiency of inference in Section 6. Section 7 provides a comprehensive empirical evaluation, and Section 8 presents a case study to U.S. Census and administrative data. We make closing remarks in Section 9.

2 RELATED WORK

We review related work across three main areas—ER methodology, inference for Bayesian ER models, and distributed Markov chain Monte Carlo (MCMC).

Entity resolution methodology. The first probabilistic approach to ER was due to Newcombe et al. (1959), who applied matching rules to pairs of records. This idea was later formalized in a seminal paper by Fellegi and Sunter (1969) within a decision-theoretic framework. Many variations of the Fellegi-Sunter (FS) approach have been proposed (for surveys, see Winkler, 2006, 2014), including a generalization to multiple databases (Sadinle and Fienberg, 2013). Others have addressed scalability of FS-type approaches using blocking/indexing methods (see Christen, 2012b; Steorts et al., 2014 for surveys) and

[1]Spark package source code available at `https://github.com/cleanzr/dblink`.

[2]R package source code available at `https://github.com/cleanzr/dblinkR`.

efficient data structures (Enamorado et al., 2019). However, traditional FS approaches do not naturally support propagation of ER uncertainty, and existing methods for scaling make approximations that sacrifice accuracy.

While the FS approach has been highly influential, it has also been criticized due to its lack of support for duplicates within databases; misspecified independence assumptions; and its dependence on subjective thresholds (Tancredi and Liseo, 2011). These limitations have prompted development of more sophisticated Bayesian models, including models for bipartite matching (Fortini et al., 2001; Larsen, 2005, 2012; Tancredi and Liseo, 2011; Gutman et al., 2013; Sadinle, 2017; McVeigh et al., 2019), deduplication (Sadinle, 2014; Tancredi et al., 2020) and matching across multiple databases (Steorts, 2015; Steorts et al., 2016). Several of these models operate on attribute-level comparisons between pairs of records in a similar vein as the FS approach (Larsen, 2005, 2012; Gutman et al., 2013; Sadinle, 2014, 2017; McVeigh et al., 2019). This contrasts with entity-centric generative models which assume the records arise as distortions to some latent entity attributes (Tancredi and Liseo, 2011; Steorts, 2015; Steorts et al., 2016; Tancredi et al., 2020).

In scenarios where training data is scarce or unavailable, Bayesian generative models tend to be more robust than discriminative or likelihood-based methods, as the priors have a regularizing effect. Bayesian generative models are also amenable to theoretical analysis: recent work has obtained lower bounds on the probability of misclassifying the entity associated with a record (Steorts et al., 2017). However, a major downside of Bayesian ER models is the computational cost of performing inference (see discussion below).

Apart from these advances in Bayesian models for ER (largely undertaken in statistics), there have been an abundance of contributions from the database and machine learning communities (see surveys by Getoor and Machanavajjhala, 2012; Christen, 2012b). Their focus has typically been on rule-based approaches (Fan et al., 2009; Singh et al., 2017), supervised learning approaches (Mudgal et al., 2018), hybrid human-machine approaches (Wang et al., 2012; Gokhale et al., 2014), and scalability (Papadakis et al., 2016). Broadly speaking, all of these approaches rely on either humans in-the-loop or large amounts of labelled training data, which is not generally the case in the Bayesian setting.

Inference for Bayesian ER models. Most prior work on Bayesian generative models for ER (e.g. Tancredi and Liseo, 2011; Gutman et al., 2013; Steorts, 2015) has relied on Gibbs sampling for inference. Compared to other Markov chain Monte Carlo (MCMC) algorithms, Gibbs sampling is relatively easy to implement, however it may suffer from slow convergence and poor mixing owing to its highly local moves (Liu, 2004). Scalability is also a challenge, as a naïve Gibbs update for the linkage structure requires all-to-all comparisons between records (or between records and entities for entity-centric models). This issue is often managed by applying deterministic blocking prior to Gibbs sampling, thereby sacrificing accuracy and proper treatment of uncertainty (Larsen, 2005, 2012; Tancredi and Liseo, 2011; Gutman et al., 2013; Sadinle, 2014).

In the broader context of clustering models, the *split-merge algorithm* (Jain and Neal, 2004) has been proposed as an alternative to Gibbs sampling. It is a Metropolis-Hastings algorithm, which traverses the space of clusterings via proposals that split individual clusters or merge pairs of clusters. Since multiple cluster items are updated in a single move, it is less susceptible to becoming trapped in local modes. Steorts et al. (2016) applied this algorithm, in combination with deterministic blocking, to update the linkage structure in an ER model similar to `blink`. A close relative of the split-merge algorithm is the *chaperones algorithm*, which was proposed for inference in microclustering models (Zanella et al., 2016). The chaperones algorithm is expected to be more efficient, as it preferentially focuses on more likely cluster reassignments, through a user-specified biased distribution on the product space of cluster items. However, the biased distribution must be designed so that random item pairs can be drawn efficiently, without explicitly constructing the product space.

More recently, Zanella (2020) proposed a general framework for designing informative proposals in a Metropolis-Hastings setting, which is suited for discrete spaces (e.g. the space of possible linkage structures). They show that *locally-balanced proposals* are asymptotically-optimal within the class of pointwise informative proposals, and demonstrate significant improvements in efficiency when compared to a split-merge-type algorithm. However, computing a locally-balanced proposal for the linkage structure is computationally challenging due to quadratic scaling. This can be mitigated to some extent by running locally-balanced

updates within randomly-selected sub-blocks of records. However to avoid poor mixing, care must be taken to ensure that randomly-selected sub-blocks contain likely matching records.

In contrast to much of the literature on Bayesian ER models, McVeigh et al. (2019) proposed a method that combines deterministic blocking and restricted MCMC (based on earlier work by McVeigh and Murray, 2017). They balance approximation error by performing coarse-grained deterministic blocking/indexing as an initial step, followed by data-dependent post-hoc blocking. During inference, the linkage structure is updated using locally-balanced proposals, restricted to the post-hoc blocks. They demonstrate improved scalability—to data sets with several hundred thousand of record—with minimal risk of approximation error. However, their approach is not directly compatible with distributed inference (see below) and may require modification for use with an entity-centric model.

Parallel/distributed MCMC. Recent literature has focused on using parallel and distributed computing to scale up MCMC algorithms, where applications have included Bayesian topic models (Newman et al., 2009; Smola and Narayanamurthy, 2010; Ahn et al., 2014) and mixture models (Williamson et al., 2013; Chang and Fisher, 2013; Lovell et al., 2013; Ge et al., 2015). We review the application to mixture models, as they are conceptually similar to ER models.

Existing work has concentrated on Dirichlet process (DP) mixture models and hierarchical DP mixture models. The key to enabling distributed inference for these models is the realization that a DP mixture model can be reparameterized as a mixture of DPs. Put simply, the reparameterized model induces a *partitioning* of the clusters into blocks, such that clusters assigned to *distinct blocks* are conditionally independent. As a result, variables within blocks can be updated in parallel. Williamson et al. (2013) exploited this idea at the thread level to parallelize inference for a DP mixture model. Chang and Fisher (2013) followed a similar approach, but included an additional level of parallelization within blocks using a parallelized version of the split-merge algorithm. Others (Lovell et al., 2013; Ge et al., 2015) have developed distributed implementations in the MapReduce framework.

We do not consider DP mixture models in our work, as their behavior is ill-suited

for ER applications.[3] However we do borrow the reparameterization idea, albeit with a more flexible partition specification which permits similar entities to be co-blocked, while facilitating load balancing. It would be interesting to see whether similar ideas can be applied to microclustering models (Zanella et al., 2016), however preserving the marginal posterior distribution seems challenging in this case.

3 A SCALABLE MODEL FOR BAYESIAN ER

In this section, we present our scalable ER model called `d-blink`, which integrates probabilistic blocking in a fully Bayesian framework. Our model can be viewed as an extension of the `blink` model (Steorts, 2015) that incorporates an auxiliary partition of the latent entity parameter space into blocks. Unlike ad-hoc blocking approaches used previously in the literature (Larsen, 2005, 2012; Tancredi and Liseo, 2011; Gutman et al., 2013; Sadinle, 2014; Steorts et al., 2016), the blocks in `d-blink` are random, and inferred jointly with the other model parameters. This enables propagation of uncertainty between the blocking and ER stages. In addition, `d-blink` extends `blink` with support for missing values and user-defined attribute similarity measures.

We describe notation and assumptions in Section 3.1, before presenting `d-blink` in Section 3.2. We define attribute similarity measures in Section 3.3, including an optional truncation approximation which can improve scalability. In Section 3.4, we prove that the marginal posterior of `d-blink` (integrated over the blocks) reduces to `blink` under certain conditions. This is a desirable property, as it means our inferences are theoretically independent of the blocking design. Finally, in Section 3.5 we explain how the auxiliary blocks are beneficial in scaling and distributing inference.

[3]With a DP prior, the number of clusters grows logarithmically in the number of records, but empirical observations call for near-linear growth (Zanella et al., 2016).

3.1 Notation and problem formulation

In this section, we define notation and formulate ER in a Bayesian setting. Consider a collection of T tables[4] (databases) indexed by t, each with R_t records (rows) indexed by r and A aligned attributes (columns) indexed by a. Associated with the records is a fixed population of entities of size E indexed by e. Each entity e is described by a set of attributes $\boldsymbol{y}_e = [y_{ea}]_{a=1...A}$, which are aligned with the record attributes. The population of entities is partitioned into B blocks for computational convenience, using a blocking function `BlockFn` that maps an entity e to a block based on its attributes $\boldsymbol{y}_e$. We assume each record (t, r) belongs to a block γ_{tr} and is associated with an entity λ_{tr} within that block. The value of the a-th attribute for record (t, r) is denoted by x_{tra}, and is assumed to be a noisy observation of the associated entity's true attribute value $y_{\lambda_{tr}a}$. We allow for the fact that some attributes x_{tra} may be missing completely at random through a corresponding indicator variable o_{tra} (Little and Rubin, 2002, p. 12).

Table 1 summarizes our notation, including model-specific parameters which will be introduced shortly. We adopt the following rules to compactly refer to sets of variables:

- A boldface lower-case variable denotes the set of *all attributes*: e.g. $\boldsymbol{x}_{tr} = [x_{tra}]_{a=1...A}$.
- A boldface capital variable denotes the set of *all index combinations*: e.g. $\boldsymbol{X} = [x_{tra}]_{t=1...T;r=1...R_t;a=1...A}$.

We also define notation to separate the record attributes $\mathbf{X}$ into an observed part $\mathbf{X}^{(o)}$ (those x_{tra}'s for which $o_{tra} = 1$) and a missing part $\mathbf{X}^{(m)}$ (those x_{tra}'s for which $o_{tra} = 0$).

After specifying a generative model (see next section), we perform ER by inferring the *joint* posterior distribution over:

- the block assignments $\mathbf{\Gamma} = [\gamma_{tr}]_{t=1...T;r=1...R_t}$,
- the linkage structure $\mathbf{\Lambda} = [\lambda_{tr}]_{t=1...T;r=1...R_t}$, and
- the true entity attribute values $\boldsymbol{Y} = [y_{ea}]_{e=1...E;a=1...A}$,

[4]We define a *table* as an ordered (indexed) collection of records, which may contain duplicates (records for which all attributes are identical).

Table 1: Summary of notation.

Symbol	Description	Symbol	Description
$t \in 1 \dots T$	index over tables	γ_{tr}	assigned block for record r in table t
$r \in 1 \dots R_t$	index over records in table t	λ_{tr}	assigned entity for record r in table t
$e \in 1 \dots E$	index over entities	θ_{ta}	prob. attribute a in table t is distorted
$b \in 1 \dots B$	index over block	α_a, β_a	distortion hyperparams. for attribute a
$a \in 1 \dots A$	index over attributes	η_{ta}	prob. attribute a in table t is observed
$v \in 1 \dots \lvert\mathcal{V}_a\rvert$	index over domain of attribute a	$\mathcal{V}_a$	domain of attribute a
$R = \sum_t R_t$	total number of records	$\phi_a(\cdot)$	distribution over domain of attribute a
x_{tra}	attribute a for record r in table t	$\texttt{sim}_a(\cdot, \cdot)$	similarity measure for attribute a
z_{tra}	distortion indicator for x_{tra}	$\mathcal{R}_e$	set of records assigned to entity e
o_{tra}	observed indicator for x_{tra}	$\mathcal{E}_b$	set of entities assigned to block b
y_{ea}	attribute a for entity e	$\texttt{BlockFn}(\cdot)$	block assignment function

conditional on the observed record attribute values $\mathbf{X}^{(o)}$. Note that we operate in a fully unsupervised setting, since we do not condition on ground truth data for the links or entities. Inferring $\boldsymbol{\Gamma}$ is equivalent to the *blocking* stage of ER, where the records are partitioning into blocks to limit the comparison space. Inferring $\boldsymbol{\Lambda}$ is equivalent to the *matching/linking* stage of ER, where records that refer to the same entities are linked together. Inferring $\boldsymbol{Y}$ is equivalent to the *merging* stage, where linked records are combined to produce a single representative record. By inferring $\boldsymbol{\Gamma}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{Y}$ jointly, we are able to propagate uncertainty between the three stages.

3.2 Model specification

We now present our proposed model `d-blink` by describing the generative process. We provide a visual representation of the model in Figure 1, with key differences from `blink` highlighted in a dashed blue line style.

Entities. The population of entities is assumed to be of fixed size E. Each entity e is described by a vector of "true" attributes $\boldsymbol{y}_e \in \bigotimes_{a=1}^{A} \mathcal{V}_a$. The value of the a-th attribute y_{ea} is assumed to be drawn independently from a distribution ϕ_a over the attribute domain $\mathcal{V}_a$:

$$y_{ea} \overset{\text{ind.}}{\sim} \text{Discrete}_{v \in \mathcal{V}_a}[\phi_a(v)]. \tag{1}$$

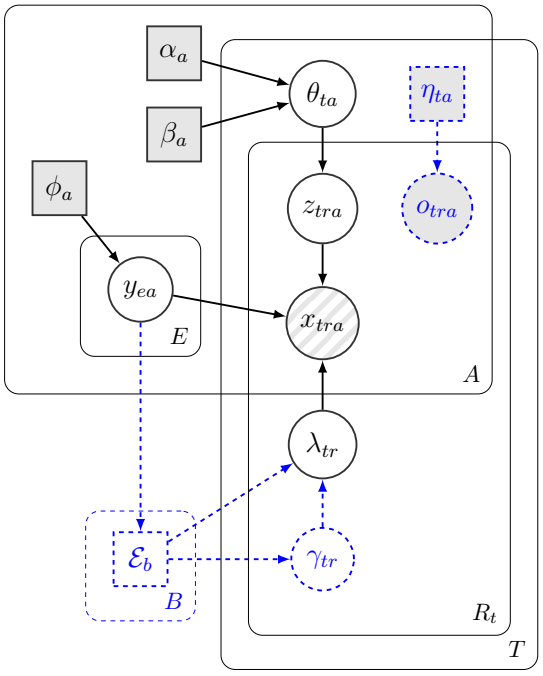

Figure 1: Plate diagram for `d-blink`. Extensions to `blink` are highlighted in a dashed blue line style. Circular nodes represent random variables; square nodes represent deterministic variables; (un)shaded nodes represent (un)observed variables; arrows represent conditional dependence; and plates represent replication over an index.

Following the `blink` model, we set the population size E and the distributions over the attribute domains ϕ_a empirically. Recommendations for setting these parameters are provided in Appendix F.3.

Blocks. The parameter space associated with the entities $\bigotimes_{a=1}^{A} \mathcal{V}_a$ is partitioned into B blocks. The partition is parameterized using a deterministic *blocking function*:

$$\texttt{BlockFn} : \bigotimes_{a} \mathcal{V}_a \to \{1, \ldots, B\}, \tag{2}$$

which is a free parameter and may be selected for inferential convenience. We provide recommendations for selecting the blocking function in Section 4, including an example based on k-d trees.

We shall often need to refer to the entities assigned to a particular block. To do this concisely, we introduce the notation $\mathcal{E}_b(\boldsymbol{Y}) = \{e : \texttt{BlockFn}(\boldsymbol{y}_e) = b\}$ to denote the set of entities assigned to block b. This is random due to the dependence on $\boldsymbol{Y}$, however we shall often omit the dependence for brevity.

Distortion. Associated with each table t and attribute a is a distortion probability θ_{ta}, with assumed prior distribution:

$$\theta_{ta} | \alpha_a, \beta_a \overset{\text{ind.}}{\sim} \text{Beta}[\alpha_a, \beta_a], \tag{3}$$

where α_a and β_a are hyperparameters. We provide recommendations for setting α_a and β_a in Appendix F. The distortion probabilities feed into the record-generation process below.

Records. We assume a record is generated by selecting an entity uniformly at random and copying the entity's attributes subject to distortion. The process for generating record r in table t is outlined below. Steps (i), (ii), and (v) deviate from `blink`.

(i) Choose a block assignment γ_{tr} at random in proportion to the block sizes:

$$\gamma_{tr} | \boldsymbol{Y} \overset{\text{ind.}}{\sim} \text{Discrete}_{b \in \{1 \ldots B\}}[|\mathcal{E}_b| / E]. \tag{4}$$

(ii) Choose an entity assignment λ_{tr} uniformly at random from block γ_{tr}:

$$\lambda_{tr}|\gamma_{tr}, \boldsymbol{Y} \overset{\text{ind.}}{\sim} \text{DiscreteUniform}[\mathcal{E}_{\gamma_{tr}}]. \tag{5}$$

(iii) For each attribute a, draw a distortion indicator z_{tra}:

$$z_{tra}|\theta_{ta} \overset{\text{ind.}}{\sim} \text{Bernoulli}[\theta_{ta}]. \tag{6}$$

(iv) For each attribute a, draw a record value x_{tra}:

$$x_{tra}|z_{tra}, y_{\lambda_{tr}a} \overset{\text{ind.}}{\sim} (1 - z_{tra})\delta(y_{\lambda_{tr}a}) + z_{tra}\,\text{Discrete}_{v\in\mathcal{V}_a}[\psi_a(v|y_{\lambda_{tr}a})] \tag{7}$$

where $\delta(\cdot)$ represents a point mass. If $z_{tra} = 0$, x_{tra} is copied directly from the entity. Otherwise, x_{tra} is drawn from the domain $\mathcal{V}_a$ according to the distortion distribution ψ_a. In the literature, this is known as a hit-miss model (Copas and Hilton, 1990).

(v) For each attribute a, draw an observed indicator o_{tra}:

$$o_{tra} \overset{\text{ind.}}{\sim} \text{Bernoulli}[\eta_{ta}]. \tag{8}$$

If $o_{tra} = 1$, x_{tra} is observed, otherwise it is missing.

Detail on the distortion distribution. $\psi_a(\cdot|w)$ chooses a distorted value for attribute a conditional on the true value w. In our parameterization of the model, it is defined as

$$\psi_a(v|w) = h_a(w)\phi_a(v)\mathrm{e}^{\texttt{sim}_a(v,w)}, \tag{9}$$

where $h_a(w) = 1/\sum_{v\in\mathcal{V}_a}\phi_a(v)\mathrm{e}^{\texttt{sim}_a(v,w)}$ is a normalization constant and $\texttt{sim}_a$ is the similarity measure for attribute a (see Section 3.3). Intuitively, this distribution chooses values in proportion to their empirical frequency, while placing more weight on those that are "similar" to w. This reflects the notion that distorted values are likely to be close to the truth, as is the case when modeling typographical errors.

Posterior distribution. The generative process described above corresponds to a posterior distribution over the model parameters, conditioned on the observed records. By reading the conditional dependence structure off the plate diagram (Figure 1) and marginalizing over the missing record attributes $\boldsymbol{X}^{(m)}$, one can show that the posterior distribution is of the following form:

$$p(\boldsymbol{\Gamma}, \boldsymbol{\Lambda}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{\Theta} | \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto \prod_{e,a} p(y_{ea}|\phi_a) \times \prod_{t,a} p(\theta_{ta}|\alpha_a, \beta_a) \times \prod_{\substack{t,r,a \\ o_{tra}=1}} p(x_{tra}|z_{tra}, \lambda_{tr}, y_{\lambda_{tr}a}) \\ \times \prod_{t,r} \Big\{ p(\gamma_{tr}|\boldsymbol{Y}) p(\lambda_{tr}|\gamma_{tr}, \boldsymbol{Y}) \prod_a p(z_{tra}|\theta_{ta}) \Big\}. \tag{10}$$

For further detail on the derivation and an expanded form of the posterior, we refer the reader to Appendix A.

3.3 Attribute similarity measures

We now discuss the attribute similarity measures that appear in the distortion distribution of Equation 9. The purpose of these measures is to quantify the propensity that some value v in the attribute domain is chosen as a distorted alternative to the true value w.

Definition (Attribute similarity measure)**.** *Let $\mathcal{V}$ be the domain of an attribute. An* attribute similarity measure *on $\mathcal{V}$ is a function $\texttt{sim} : \mathcal{V} \times \mathcal{V} \to [0, s_{\max}]$ that satisfies $0 \leq s_{\max} < \infty$ and $\texttt{sim}(v, w) = \texttt{sim}(w, v)$ for all $v, w \in \mathcal{V}$.*

Note that our parameterization in terms of attribute *similarity* measures differs from `blink`, which uses *distance* measures. This allows us to make use of a more efficient sampling method, as described in Section 6.3. The next proposition states that the two parameterizations are equivalent, so long as the distance measure is bounded and symmetric (a proof is provided in Appendix B.1).

Proposition 1. *Let $\texttt{dist}_a : \mathcal{V} \times \mathcal{V} \to [0, d_{\max;a}]$ be the attribute distance measure that appears in* `blink`*, and assume that $0 \leq d_{\max;a} < \infty$ and $\texttt{dist}_a(v, w) = \texttt{dist}_a(w, v)$ for all $v, w \in \mathcal{V}$. Define the corresponding attribute similarity measure for* `d-blink` *as*

$$\texttt{sim}_a(v, w) := d_{\max;a} - \texttt{dist}_a(v, w). \tag{11}$$

Then the parameterization of ψ_a used in `d-blink` *is equivalent to* `blink`.

In this paper, we restrict our attention to the following similarity measures for simplicity:

- *Constant similarity measure.* This measure is appropriate for categorical attributes, where there is no reason to believe one value is more likely than any other as a distortion to the true value w. Without loss of generality, it may be defined as $\texttt{sim}_{\text{const}}(v, w) = s_{\max}$ for all $v, w \in \mathcal{V}$.

- *Normalized edit similarity measure.* This measure is based on the edit distance metric, and is suitable for modeling distortion in generic string attributes. Following Yujian and Bo (2007), we define a normalized edit distance metric,

$$\texttt{dist}_{\text{nEd}}(v, w) = \frac{2\ \texttt{dist}_{\text{Ed}}(v, w)}{|v| + |w| + \texttt{dist}_{\text{Ed}}(v, w)},$$

  where $\texttt{dist}_{\text{Ed}}$ denotes the regular edit distance and $|v|$ denotes the length of string v. Note that alternative definitions of the normalized edit distance could be used (see references in Yujian and Bo, 2007), however the above definition is unique in that it yields a proper metric. Since the normalized edit distance is bounded on the interval $[0, 1]$ we can define a corresponding normalized edit similarity measure:

$$\texttt{sim}_{\text{nEd}}(v, w) = 1 - \texttt{dist}_{\text{nEd}}(v, w). \tag{12}$$

Ideally, one should select attribute similarity measures based on the data at hand. There are many possibilities to consider, such as Jaccard similarity, numeric similarity measures (Lesot et al., 2008) and other domain-specific measures (Bilenko and Mooney, 2003).

3.4 Model equivalence

We have purposely constructed `d-blink` so that it reduces to `blink` under certain conditions. Assuming the records are fully observed, the posterior distribution of `d-blink` as specified in Equation 10 is similar to `blink`. The difference lies in the factors involving the block assignments γ_{tr} and the entity assignments λ_{tr}. However, if one marginalizes out the auxiliary block assignments—as is done automatically in Markov chain Monte Carlo—the

posterior distributions are identical. This statement is made precise below (proof provided in Appendix B.2):

Proposition 2. *Suppose the conditions of Proposition 1 hold and that* $\alpha_a = \alpha$ *and* $\beta_a = \beta$ *for all* a*. Assume furthermore that all record attributes are observed, i.e.* $o_{tra} = 1$ *for all* t, r, a*. Then the marginal posterior of* $\boldsymbol{\Lambda}$*,* $\boldsymbol{Y}$*,* $\boldsymbol{Z}$ *and* $\boldsymbol{\Theta}$ *for* `d-blink` *(i.e. marginalized over* $\boldsymbol{\Gamma} = [\gamma_{tr}]_{t=1\ldots T;r=1\ldots R_t}$*) is identical to the posterior for* `blink`*.*

This is an important result, as it shows our inferences for the meaningful model parameters are the same as we would obtain from `blink`. Thus we are able to apply blocking to scale the model, without compromising the correctness of the posterior distribution.

3.5 Rationale for introducing block

We now briefly explain the role of the auxiliary block in `d-blink`. First, we note that without the block ($B = 1$), the Markov blanket for λ_{tr} includes the attribute values for *all* of the entities $\boldsymbol{Y}$. This presents a major obstacle when it comes to distributing the inference on a compute cluster, as the data is not separable. By incorporating block, we restrict the Markov blanket for λ_{tr} to include only a subset of the entity attribute values—those in the same block as record (t, r). As a result, it becomes natural to distribute the inference so that each compute node is responsible for a single block (see Section 5.2 for details). Secondly, we can interpret the block as performing probabilistic blocking in the context of MCMC sampling (introduced in Section 5), which improves computational efficiency. In a given iteration, the possible links for a record are restricted to the entities residing in the same block. However, unlike conventional blocking, the block assignments are not fixed—between iterations the entities and linked records may move between blocks.

4 BLOCKING FUNCTIONS

In Section 3.2 we introduced a generic blocking function (Equation 2) that is responsible for assigning entities to blocks. This function may be regarded as a free parameter, since it has no bearing on model equivalence according to Proposition 2. However, from a practical

perspective the blocking function ought to be chosen carefully, as it can impact inferential efficiency—both in terms of computational and mixing time. We suggest some guidelines for choosing a blocking function in Section 4.1, before presenting an example based on k-d trees in Section 4.2.

4.1 Interpretation and guidelines

Recall that the blocking function assigns an entity to a block according to its attributes $\boldsymbol{y}_e = [y_{ea}]_{a=1\ldots A}$. Since $\boldsymbol{y}_e$ is *unobserved*, it must be treated as a random variable over the space of possible attributes $\mathcal{V}_\otimes := \bigotimes_{a=1}^{A} \mathcal{V}_a$. This means the blocking function should not be interpreted as partitioning the entities directly. Rather, it should be interpreted as partitioning the space $\mathcal{V}_\otimes$ in which the entities reside, while taking the distribution over $\mathcal{V}_\otimes$ into account. With this interpretation in mind, we argue that the blocking function should ideally satisfy the following properties:

(i) *Balanced weight.* The blocks should have equal weight (probability mass) under the distribution over $\mathcal{V}_\otimes$, thereby ensuring the entities are distributed evenly (in expectation) among the blocks. This is a desirable property, as it ensures proper load balancing for our distributed inference algorithm (see Section 5.2).

(ii) *Entity separation.* A pair of entities drawn at random from the same block should have a high degree of similarity, while entities drawn from different blocks should have a low degree of similarity. This improves the likelihood that similar records will end up in the same block, and allows them to more readily form likely entities.

These properties need not be satisfied strictly: the extent to which they are satisfied is merely expected to improve the efficiency of the inference. For example, satisfying the first property requires knowledge of the marginal posterior distribution over $\boldsymbol{y}_e$, which is infeasible to calculate. We note that there is likely to be tension between the two properties, so that a balance must be struck between them.

4.2 Example: k-d tree blocking function

We now describe a blocking function based on k-d trees, which is used in our experiments in Section 7.

Background. A k-d tree is a binary tree that recursively partitions a k-dimensional affine space (Bentley, 1975; Friedman et al., 1977). In the standard set-up, each node of the tree is associated with a data point that implicitly splits the input space into two half-spaces along a particular dimension. Owing to its ability to hierarchically group nearby points, it is commonly used to speed up nearest-neighbor search. This makes a k-d tree a good candidate for a blocking function, since it can be balanced while grouping similar points.

Setup. Our setup differs from a standard k-d tree in several aspects. First, we consider a discrete space $\mathcal{V}_{\otimes}$ (not an affine space), where the "k dimensions" are the A attributes. Second, we do not store data points in the tree. We only require that the tree implicitly stores the boundaries of the blocks, so that it can assign an arbitrary $\boldsymbol{y} \in \mathcal{V}_{\otimes}$ to the correct partition (a leaf node). Finally, since we are working in a discrete space, the input space to a node is a countable set. The node must split the input set into two parts based on the values of one of the attributes.

Fitting the tree. Since it is infeasible to calculate the marginal posterior distribution over $\boldsymbol{y}_e$ exactly, we use the empirical distribution from the tables as an approximation. As a result, we treat the records (tables) as a sample from the distribution over $\boldsymbol{y}_e$, and fit the tree so that it remains balanced with respect to this sample. The depth of the tree d determines the number of blocks (2^d).

Achieving balanced splits. When fitting the tree, each node receives an input set of samples and a rule must be found that splits the set into two roughly equal (balanced) parts based on an attribute. We consider two types of splitting rules: the *ordered median* and the *reference set* (see Appendix C). We allow the practitioner to specify an ordered list of attributes to be used for splitting. To ensure balanced splits, we recommend selecting

attributes with a large domain. If possible, we recommend preferencing attributes which are known a priori to be reliable (low distortion), as this will reduce the shuffling of entities/ records between blocks. In principle, it is possible to automate the process of fitting a tree: one could grow several trees with randomly-selected splits and use the one that is most balanced. We examine balance empirically in Appendix H.

5 INFERENCE

We now turn to approximating the full joint posterior distribution over the unobserved variables $\boldsymbol{Z}$, $\boldsymbol{Y}$, $\boldsymbol{\Theta}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{\Lambda}$, as given in Equation 10. Since it is infeasible to sample from this distribution directly, we design MCMC algorithms based on partially-collapsed Gibbs (PCG) sampling (van Dyk and Park, 2008). In addition, we show how to exploit the conditional independence induced by the blocks to distribute the PCG sampling across multiple threads or machines.

5.1 Partially-collapsed Gibbs sampling

Following the `blink` paper (Steorts, 2015), we initially experimented with regular Gibbs sampling.[5] However, the resulting Markov chains exhibited slow convergence and poor mixing. This is a known shortcoming of Gibbs sampling which may be remedied by collapsing variables and/or updating correlated variables in groups (Liu, 2004). These ideas form the basis for a framework called *partially-collapsed Gibbs (PCG) sampling*—a generalization of Gibbs sampling that has better convergence properties (van Dyk and Park, 2008).

Under the PCG framework, variables are updated in groups by sampling from their conditional distributions. These conditional distributions may be taken with respect to the joint posterior (like regular Gibbs), or with respect to *marginal distributions* of the joint posterior (unique to PCG). The latter case is called *trimming* and must be handled with care so as not to alter the stationary distribution of the Markov chain.

[5] We define *regular* Gibbs sampling as the most basic variation where variables are updated iteratively one-at-a-time by sampling from their conditional distributions.

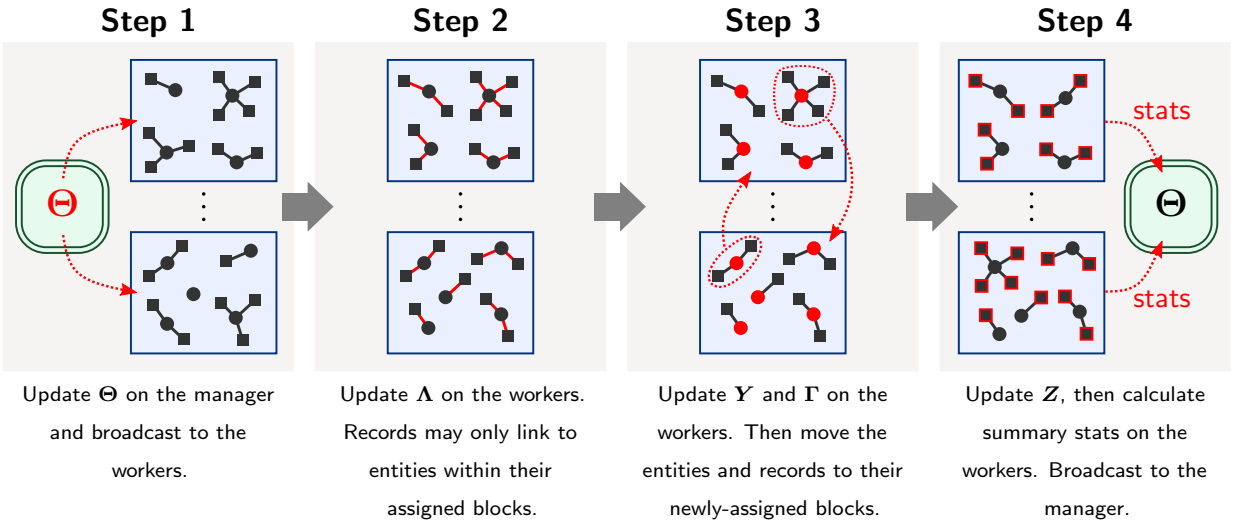

Figure 2: Schematic depicting a single iteration of distributed PCG sampling. The entity attributes ($\boldsymbol{Y}$—circular nodes), record attributes and their distortion indicators ($\boldsymbol{X}$, $\boldsymbol{Z}$—square nodes), and links from records to entities ($\boldsymbol{\Lambda}$—node connectors) are distributed across the workers (blue rectangular plates) according to their assigned blocks. The distortion probabilities ($\boldsymbol{\Theta}$) reside on the manager (green rounded-rectangular plate).

In applying PCG sampling to `d-blink`, we must decide how to apply the three tools: *marginalization* (equivalent to grouping), *permutation* (changing the order of the updates) and *trimming* (removing marginalized variables). In theory, the convergence rate should improve with more marginalization and trimming, however this must be balanced with the following: (i) whether the resulting conditionals can be sampled from efficiently, and (ii) whether the resulting dependence structure is compatible with our distributed set-up (see Section 5.2). We consider two samplers, PCG-I and PCG-II, described below. Of the two, we recommend PCG-I as it is more efficient in our empirical evaluations (see Section 7.1). We include the PCG-II sampler, as one would expect the PCG-II sampler to perform better than the PCG-I sampler in terms of mixing, however when computational efficiency is taken into account the performance is worse (see Figure 6).

5.1.1 PCG-I sampler

The PCG-I sampler uses regular Gibbs updates for θ_{ta}, λ_{tr} and z_{tra} for all t, r and a. The conditional distributions for these updates are listed in Appendix D. When updating the entity attributes y_{ea} and the block assignments γ_{tr}, marginalization and trimming are used. Specifically, we apply marginalization by jointly updating $\boldsymbol{y}_e$ and $\{\gamma_{tr}, \boldsymbol{z}_{tr}\}_{\mathcal{R}_e}$ (the set of γ_{tr}'s and $\boldsymbol{z}_{tr}$'s for records (t, r) linked to entity e). We then trim (analytically integrate over) $\{\boldsymbol{z}_{tr}\}_{\mathcal{R}_e}$.

We shall now derive this update. Referring to Equation 10, the joint posterior of $\boldsymbol{y}_e$, $\{\gamma_{tr}, \boldsymbol{z}_{tr}\}_{\mathcal{R}_e}$ conditioned on the other parameters has the form

$$p\big(\boldsymbol{y}_e, \{\gamma_{tr}, \boldsymbol{z}_{tr}\}_{\mathcal{R}_e} \big| \boldsymbol{Z}^{\neg\mathcal{R}_e}, \boldsymbol{\Gamma}^{\neg\mathcal{R}_e}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}, \boldsymbol{X}^{(o)}, \boldsymbol{O}\big) \propto$$
$$\prod_a \Big\{ p(y_{ea}|\phi_a) \times \prod_{(t,r)\in\mathcal{R}_e} p(\gamma_{tr}|\boldsymbol{Y}) p(\lambda_{tr}|\gamma_{tr}, \boldsymbol{Y}) p(z_{tra}|\theta_{ta}) \times \prod_{\substack{(t,r)\in\mathcal{R}_e \\ o_{tra}=1}} p(x_{tra}|z_{tra}, \lambda_{tr}, y_{ea}) \Big\},$$

where superscript $\neg\mathcal{R}_e$ denotes exclusion of any records $(t, r) \in \mathcal{R}_e$ (those currently linked to entity e). Substituting the distributions and trimming $\{\boldsymbol{z}_{tr}\}_{\mathcal{R}_e}$ yields

$$p\big(\boldsymbol{y}_e, \{\gamma_{tr}\}_{\mathcal{R}_e} \big| \boldsymbol{Z}^{\neg\mathcal{R}_e}, \boldsymbol{\Gamma}^{\neg\mathcal{R}_e}, \boldsymbol{\Theta}, \boldsymbol{\Lambda}, \boldsymbol{X}^{(o)}, \boldsymbol{O}\big) = p(\{\gamma_{tr}\}_{\mathcal{R}_e} | \mathcal{R}_e, \boldsymbol{y}_e) \prod_a p(y_{ea}|\mathcal{R}_e, \boldsymbol{\Theta}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \tag{13}$$

where

$$p(y_{ea}|\mathcal{R}_e, \boldsymbol{\Theta}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto \phi_a(y_{ea}) \prod_{\substack{(t,r)\in\mathcal{R}_e \\ o_{tra}=1}} \{(1-\theta_{ta})\mathbb{I}[x_{tra} = y_{ea}] + \theta_{ta}\psi_a(x_{tra}|y_{ea})\}$$

and

$$p(\{\gamma_{tr}\}_{\mathcal{R}_e}|\mathcal{R}_e, \boldsymbol{y}_e) \propto \prod_{(t,r)\in\mathcal{R}_e} \mathbb{I}[\gamma_{tr} = \texttt{BlockFn}(\boldsymbol{y}_e)]\,.$$

Note that the update for $\{\gamma_{tr}\}_{\mathcal{R}_e}$ is deterministic, conditional on $\boldsymbol{y}_e$ and $\mathcal{R}_e$.

Since we have applied trimming, we must permute the updates so that the trimmed variables $\boldsymbol{Z}$ are not conditioned on in later updates. This means the updates for $\boldsymbol{y}_e$ and $\{\gamma_{tr}, \boldsymbol{z}_{tr}\}_{\mathcal{R}_e}$ must come *after* the updates for θ_{ta} and λ_{tr}, but *before* the updates for z_{tra}.

Table 2: Dependencies for the conditional updates used in the PCG-I sampler.

Update variables	Dependencies
θ_{ta}	$z_{t\cdot a} = \sum_r z_{tra}$
λ_{tr}	$\boldsymbol{z}_{tr}$, $\boldsymbol{x}_{tr}$, γ_{tr}, $\mathcal{E}_{\gamma_{tr}}$, $\{\boldsymbol{y}_e\}_{e\in\mathcal{E}_{\gamma_{tr}}}$
y_{ea}, $\{\gamma_{tr}, z_{tra}\}_{(t,r)\in\mathcal{R}_e}$	$\mathcal{R}_e$, $\{x_{tra}\}_{(t,r)\in\mathcal{R}_e}$, $\{\theta_{ta}\}_{(t,r)\in\mathcal{R}_e}$
z_{tra}	x_{tra}, λ_{tr}, $y_{\lambda_{tr}a}$, θ_{ta}

5.1.2 PCG-II sampler

The PCG-II sampler is identical to PCG-I, except that it replaces the regular Gibbs update for λ_{tr} with an update that marginalizes and trims $\boldsymbol{z}_{tr}$. To derive the distribution for this update, we first consider the joint posterior of λ_{tr} and $\boldsymbol{z}_{tr}$ conditioned on the other parameters:

$$p(\lambda_{tr}, \boldsymbol{z}_{tr}|\boldsymbol{\Gamma}, \boldsymbol{Y}, \boldsymbol{\Theta}, \boldsymbol{Z}^{\neg(t,r)}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto$$
$$p(\lambda_{tr}|\gamma_{tr}, \boldsymbol{Y}) \times \prod_a p(z_{tra}|\theta_{ta}) \times \prod_{\substack{a \\ o_{tra}=1}} p(x_{tra}|z_{tra}, \lambda_{tr}, y_{\lambda_{tr}a})$$

where superscript $\neg(t,r)$ denotes exclusion of record (t,r). Substituting the distributions and trimming $\boldsymbol{z}_{tr}$ yields

$$p(\lambda_{tr}|\boldsymbol{\Gamma}, \boldsymbol{Y}, \boldsymbol{\Theta}, \boldsymbol{Z}^{\neg(t,r)}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto$$
$$\mathbb{I}[\lambda_{tr} \in \mathcal{E}_{\gamma_{tr}}(\boldsymbol{Y})] \times \prod_{\substack{a \\ o_{tra}=1}} \Big\{(1-\theta_{ta})\mathbb{I}[x_{tra} = y_{\lambda_{tr}a}] + \theta_{ta}\psi_a(x_{tra}|y_{\lambda_{tr}a})\Big\}. \tag{14}$$

5.2 Distributing the sampling

By examining the conditional distributions derived in the previous section and those listed in Appendix D, one can show that the updates for the variables associated with entities and records (z_{tra}, λ_{tr}, γ_{tr} and y_{ea}) only depend on variables associated with entities and records assigned to the *same* block (excluding $\boldsymbol{\Theta}$). These dependencies are summarized in Table 2 for the PCG-I sampler. The distortion probability θ_{ta} is an exception—it is not associated with any block and may depend on z_{tra}'s across *all* blocks.

This dependence structure—in particular, the conditional independence of entities and records across blocks—makes the PCG sampling amenable to distributed computing. As such, we propose a manager-worker architecture where:

- the *manager* is responsible for storing and updating variables *not* associated with any block (i.e. $\boldsymbol{\Theta}$); and
- each *worker* represents a block, and is responsible for storing and updating variables associated with the entities and records assigned to it.

The manager/workers may be processes running on a single machine or on machines in a cluster. If using a cluster, we recommend that the nodes be tightly coupled, as frequent communication between them is required.

Figure 2 depicts a single iteration of PCG sampling using our proposed manager-worker architecture. Of the four steps depicted, steps 2 and 3—where the links, entity attributes and block assignments are updated—are the most computationally intensive. We therefore expect to achieve a significant speed-up by distributing these steps across the workers.

To ensure good load balancing of these steps it is important that the blocks are well-balanced (see Section 4.1), otherwise workers responsible for smaller blocks must wait idly for other workers to finish before the next iteration can begin. This is because step 1 requires global synchronization of state across the workers. The blocks also have an effect on communication costs, which are most significant in step 3, where the entities and linked records are shuffled to their newly-assigned blocks. A well-chosen blocking function can minimize this cost, by ensuring similar records/entities are co-blocked.

6 COMPUTATIONAL EFFICIENCY CONSIDERATIONS

6.1 Efficient pruning of candidate links

In this section, we describe a trick that is aimed at improving the computational efficiency of the Gibbs update for λ_{tr} (used in the Gibbs and PCG-I samplers). This particular trick does *not* apply to the joint PCG update for λ_{tr} and $\boldsymbol{z}_{tr}$ (used in the PCG-II sampler).

Consider the conditional distribution for the λ_{tr} update in Equation S4 of Appendix D:

$$p(\lambda_{tr} = e|\boldsymbol{\Gamma}, \boldsymbol{Y}, \boldsymbol{Z}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto \\ \mathbb{I}[e \in \mathcal{E}_{\gamma_{tr}}(\boldsymbol{Y})] \times \prod_{\substack{a \\ o_{tra}=1}} \Big\{(1 - z_{tra})\mathbb{I}[x_{tra} = y_{ea}] + z_{tra}\psi_a(x_{tra}|y_{ea})\Big\}. \tag{15}$$

The support of this distribution is the *set of candidate links* for record (t, r), which we denote by $\mathcal{L}_{tr}$. Looking at the first indicator function above, we see that $\mathcal{L}_{tr} \subseteq \mathcal{E}_{\gamma_{tr}}$, i.e. the candidate links are restricted to the entities in the *same block* as record (t, r). Thus, a naïve sampling approach for this distribution takes $O(|\mathcal{E}_{\gamma_{tr}}|)$ time.

We can improve upon the naïve approach by exploiting the fact that $\mathcal{L}_{tr}$ is often considerably smaller than $\mathcal{E}_{\gamma_{tr}}$. To see why this is the case, note that the second indicator function in Equation 15 further restricts $\mathcal{L}_{tr}$ if any of the distortion indicators for the observed record attributes are zero. Specifically, if $z_{tra} = 0$ and $o_{tra} = 1$, $\mathcal{L}_{tr}$ cannot contain any entity whose a-th attribute y_{ea} does not match the record's a-th attribute x_{tra}. This implies $\mathcal{L}_{tr}$ is likely to be small in the case of low distortion.

Putting aside the computation of $\mathcal{L}_{tr}$ for the moment, this means we can reduce the time required to update λ_{tr} to $O(|\mathcal{L}_{tr}|)$. To compute $\mathcal{L}_{tr}$ efficiently, we propose maintaining an inverted index over the entity attributes within each block. Specifically, the index for the a-th attribute in block b should accept a query value $v \in \mathcal{V}_a$ and return the set of entities that match on v:

$$\mathcal{M}_{pa}(v) = \{n \in \mathcal{E}_p : y_{ea} = v\}. \tag{16}$$

Once the index is constructed, we can efficiently retrieve the set of candidate links for record (t, r) by computing a multiple set intersection:

$$\mathcal{L}_{tr} = \bigcap_{\{a: z_{tra}=0 \wedge o_{tra}=1\}} \mathcal{M}_{\gamma_{tr}a}(x_{tra}). \tag{17}$$

This assumes at least one of the observed record attributes is not distorted. Otherwise $\mathcal{L}_{tr} = \mathcal{E}_{\gamma_{tr}}$.

Since the sizes of the sets $\mathcal{M}_{\gamma_{tr}a}(x_{tra})$ are likely to vary significantly, we advise computing the intersection iteratively in increasing order of size. That is, we begin with the smallest set and retain the elements that are also in the next largest set, and so on. With a hash-based set implementation, this scales linearly in the size of the first (smallest) set.

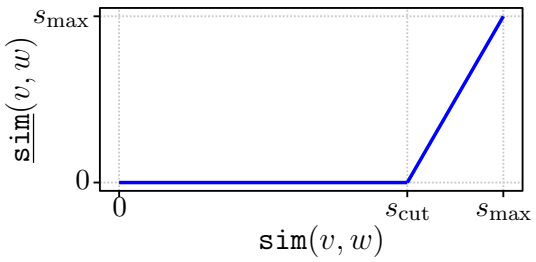

Figure 3: Transformation from a raw similarity function ($\texttt{sim}$) to a truncated similarity function ($\underline{\texttt{sim}}$).

6.2 Caching and truncation of attribute similarities

We have not yet emphasized that the updates for $\boldsymbol{\Lambda}$, $\boldsymbol{Y}$ and $\boldsymbol{\Gamma}$ depend on the attribute similarities between pairs of values in the attribute domains. Specifically, for each attribute a, we need to access the indexed set $\mathcal{S}_a = \{\texttt{sim}_a(v, w) : v, w \in \mathcal{V}_a \times \mathcal{V}_a\}$. These similarities may be expensive to evaluate on-the-fly, so we cache the results in memory on the workers.

To manage the quadratic scaling of $\mathcal{S}_a$, and in anticipation of another trick introduced in Section 6.3, we transform the similarities so that those *below* a cut-off $s_{\text{cut};a}$ are regarded as completely disagreeing. We achieve this by applying the following truncation transformation to the raw attribute similarity $\texttt{sim}_a(v, w)$:

$$\underline{\texttt{sim}}_a(v, w) = \max\left(0,\ \frac{\texttt{sim}_a(v, w) - s_{\text{cut};a}}{1 - s_{\text{cut};a}/s_{\max;a}}\right). \tag{18}$$

as illustrated in Figure 3. Whenever a raw attribute similarity is called for, we replace it with this truncated version. Only pairs of values with positive truncated similarity are stored in the cache—those not stored in the cache have a truncated similarity of zero by default. Note that attributes with a constant similarity function $\texttt{sim}_{\text{const}}$ are treated specially—there is no need to cache the index set of similarities, since they are all identical.

It is important to acknowledge that the truncated similarities are an approximation to the original model. We claim that the approximation is reasonable on the following grounds:

- *Low loss of information.* Below a certain cut-off, the attribute similarity function is unlikely to encode much useful information for modeling the distortion process. For example, the fact that $\texttt{sim}_{\text{nEd}}(\text{“Smith”}, \text{“Chiu”}) = 0.385$ whereas

$\texttt{sim}_{\text{nEd}}$("Smith", "Chen") = 0.286, doesn't necessarily suggest that "Chiu" is more likely than "Chen" as a distorted alternative to "Smith".

- *Precedent.* In the record linkage literature, value pairs with similarities below a cut-off are regarded as completely disagreeing (Winkler, 2002; Enamorado et al., 2019).

- *Efficiency gains.* As we shall soon see in Section 6.3, we can perform the combined $\boldsymbol{Y}$, $\boldsymbol{\Gamma}$, $\boldsymbol{Z}$ update more efficiently by eliminating pairs below the cut-off from consideration.

6.3 Fast updates of entity attributes using perturbation sampling

We now present a novel sampling algorithm that allows us to efficiently perform the PCG update for y_{ea} and $\{\gamma_{tr}, z_{tra}\}_{\mathcal{R}_e}$. The algorithm relies on the observation that the conditional distribution for y_{ea} can be expressed as a mixture over two components:

(i) a *base distribution* over $\mathcal{V}_a$ which is ideally constant for all entities; and

(ii) a *perturbation distribution* which varies for each entity, but has a much smaller support than $\mathcal{V}_a$.

With this representation, we can avoid computing and sampling from the full distribution over $\mathcal{V}_a$, which varies for each y_{ea} update. Rather, we only need to compute the perturbation distribution over a much smaller support, and then sample from the mixture, which can be done efficiently using the Vose-Alias method (Vose, 1991). We refer to this algorithm as *perturbation sampling*.

6.3.1 Perturbation sampling

Although we're interested in applying perturbation sampling to a specific conditional distribution, we describe the idea in generality below.

Consider a target probability mass function (pmf) $p(x|\omega)$ with finite support $\mathcal{X}$, which varies as a function of parameters $\omega \in \Omega$. In general, one must recompute the probability tables to draw a new variate whenever ω changes—a computation that takes $O(|\mathcal{X}|)$ time. However, if the dependence on ω is of a certain restricted form, we show that it is possible

to achieve better scalability by expressing the target as a mixture. This is made precise in the following result.

Proposition 3. *Let $p(x|\omega)$ be a pmf with finite support $\mathcal{X}$, which depends on parameters $\omega \in \Omega$. Suppose there exists a "base" pmf $q(x)$ over $\mathcal{X}$ which is independent of ω and a non-negative bounded perturbation term $\epsilon(x|\omega)$, such that $p(x|\omega)$ can be factorized as $p(x|\omega) \propto q(x)(1+\epsilon(x|\omega))$. Then $p(x|\omega)$ can be expressed as a mixture over the base pmf $q(x)$ and a "perturbation" pmf $v(x|\omega) := c\, q(x)\epsilon(x|\omega)$ over $\mathcal{X}^\star = \{x \in \mathcal{X} : \epsilon(x|\omega) > 0\}$ as follows:*

$$p(x|\omega) = \frac{c}{1+c} q(x) + \frac{1}{1+c} v(x|\omega) \tag{19}$$

where $c^{-1} := \sum_{x \in \mathcal{X}^\star} q(x)\epsilon(x|\omega)$.

Proof. The result is straightforward to verify by substitution. □

Algorithm S1 (in Appendix E) shows how to apply this result to draw random variates from a target pmf. Briefly, it consists of three steps: (i) the perturbation pmf v and its normalization constant c are computed; (ii) a biased coin is tossed to choose between the base pmf q and the perturbation pmf v; and (iii) a random variate is drawn from the selected pmf. If q is selected, a pre-initialized Alias sampler is used to draw the random variate (reused for all ω). Otherwise if v is selected, a new Alias sampler is instantiated. The result below states the time complexity of this algorithm (see Appendix E for a proof).

Proposition 4. *Algorithm S1 (in Appendix E) returns a random variate from the target pmf $p(x|\omega)$ for any $\omega \in \Omega$ in $O(|\mathcal{X}^\star|)$ time.*

This is a promising result, since the size of the perturbation support $|\mathcal{X}^\star|$ is typically of order 10 for our application, while the size of the full support $|\mathcal{X}|$ may be as large as 10^5. Hence, we expect a significant speed-up over the naïve approach.

6.3.2 Application of perturbation sampling

We now return to our original objective: performing the joint PCG update for y_{ea} and $\{\gamma_{tr}, z_{tra}\}_{\mathcal{R}_e}$. Referring to Equation 13, we can express the conditional distribution for y_{ea}

(i.e. the target distribution) as

$$p(y_{ea} = v|\mathcal{R}_e, \boldsymbol{\Theta}, \boldsymbol{X}^{(o)}, \boldsymbol{O}) \propto q_a(v|\mathcal{R}_e, \mathbf{O}) \left(1 + \epsilon_a(v|\mathcal{R}_e, \boldsymbol{\Theta}, \boldsymbol{X}^{(o)}, \boldsymbol{O})\right). \tag{20}$$

The base distribution is given by

$$q_a(v|\mathcal{R}_e, \boldsymbol{O}) \propto \phi_a(v) \left(h_a(v)\right)^{n_a(\mathcal{R}_e, \boldsymbol{O})} \tag{21}$$

where $n_a(\mathcal{R}_e, \boldsymbol{O}) = |\{(t, r) \in \mathcal{R}_e : o_{tra} = 1\}|$ is the number of records linked to entity e with observed values for attribute a; and the perturbation term is given by

$$\epsilon_a(v|\mathcal{R}_e, \boldsymbol{\Theta}, \{x_{tra}\}_{\mathcal{R}_e}) = \prod_{\substack{(t,r)\in\mathcal{R}_e \\ o_{tra}=1}} \left\{ \mathrm{e}^{\underline{\mathtt{sim}}_a(x_{tra}, v)} + \frac{(\theta_{ta}^{-1} - 1)\, \mathbb{I}[x_{tra} = v]}{\phi_a(x_{tra}) h_a(x_{tra})} \right\} - 1. \tag{22}$$

The full support of the target pmf is $\mathcal{V}_a$, while the perturbation support is given by

$$\{x_{tra} : (t, r) \in \mathcal{R}_e \wedge o_{tra} = 1\} \cup \{v \in \mathcal{V}_a : \underline{\mathtt{sim}}_a(v, x_{tra}) > 0 \wedge o_{tra} = 1 \text{ for any } (t, r) \in \mathcal{R}_e\}.$$

In words, this set consists of the observed values for attribute a in the records linked to entity e, plus any sufficiently similar values from the attribute domain (for which the truncated similarity is non-zero). The size of the perturbation set will vary depending on the cut-off used for the truncation transformation—the higher the cut-off, the smaller the set. This implies that there is a trade-off between efficiency (small perturbation set) and accuracy (lower cut-off).

Remark. *The astute reader may have noticed that the base distribution q_a given in Equation 21 is not completely independent of the conditioned parameters, as is required by Proposition 3. In particular, q_a depends on $n_a(\mathcal{R}_e, \boldsymbol{O})$—roughly the size of entity e. Fortunately, we expect the range of regularly encountered entity sizes to be small, so we sacrifice some memory by instantiating multiple Alias samplers for each $n_a(\mathcal{R}_e, \boldsymbol{O})$ in some expected range. In the worst case, when a value is encountered outside the expected range and the base distribution is required (unlikely since the weight on the base component is typically small), we instantiate the base distribution on-the-fly (same asymptotic cost as the naïve approach).*

Table 3: Summary of data sets. Those marked with a '⋆' are synthetic.

Data set	# records (R)	# tables (T)	# entities	# attributes (A)	
				categorical	string
⋆ ABSEmployee	660,000	3	400,000	4	0
NCVR	448,134	2	296,433	3	3
NLTCS	57,077	3	34,945	6	0
SHIW0810	39,743	2	28,584	8	0
⋆ RLdata10000	10,000	1	9,000	2	3

7 EMPIRICAL EVALUATION

We present an evaluation of `d-blink` using two synthetic and three real data sets, as summarized in Table 3. The data sets include applications such as ER of employees in administrative and survey data (`ABSEmployee`), ER of voters in registration databases (`NCVR`) and ER of respondents in anonymized survey data (`SHIW0810`). All results presented here were obtained using a local server in pseudocluster mode, however some were replicated on a cluster in the Amazon public cloud (see Appendix G) to test the effect of higher communication costs. Further details about the data sets, hardware, implementation and parameter settings are provided in Appendix F.

7.1 Computational and sampling efficiency

Following Turek et al. (2016), we measured the efficiency using the rate of effective samples produced per unit time (ESS rate), which balances sampling efficiency (related to mixing/ autocorrelation) and computational efficiency. We used the `mcmcse` R package (Flegal et al., 2017) to compute the effective sample size (ESS), which implements a multivariate method proposed by Vats et al. (2019).

Since the number of variables in the model is unwieldy (there are at least $(E+R+T)A+R$ unobserved variables) we computed the ESS for the following summary statistics:

- the number of observed entities (scalar);
- the aggregate distortion for each attribute (vector); and

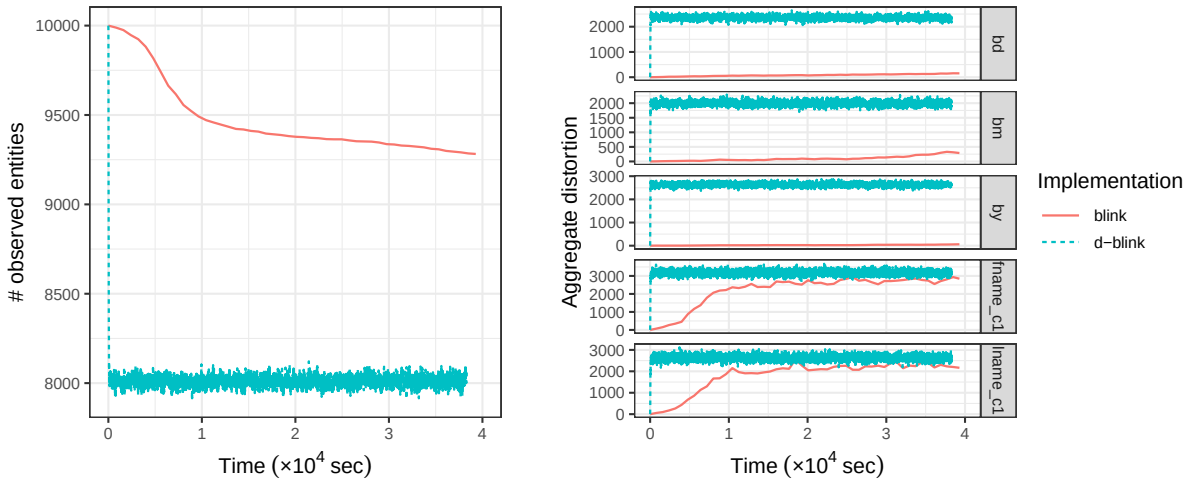

Figure 4: Comparison of convergence rates for `d-blink` and `blink`. The summary statistics for `d-blink` (number of observed entities on the left and attribute distortions on the right) rapidly converge to equilibrium, while those for `blink` fail to converge within 11 hours.

- the cluster size distribution (vector containing frequency of 0-clusters, 1-clusters, 2-clusters, etc.).

`d-blink` versus `blink`. We compared `d-blink` (using the PCG-I sampler) to our own implementation of `blink` (i.e. a Gibbs sampler without any of the tricks described in Section 6). For a fair comparison, we switched off blocking in `d-blink`. We used the relatively small `RLdata10000` data set, as `blink` cannot cope with larger data sets. Figure 4 contains trace plots for two summary statistics as a function of running time. It is evident that `blink` has not converged to the equilibrium distribution within the allotted time of 11 hours, while `d-blink` converges to equilibrium in 100 seconds. Looking solely at the time per iteration, `d-blink` is at least 200× faster than `blink`.

Blocking and efficiency. We tested the effect of varying the number of blocks B on the efficiency of `d-blink`. For each value of B, we computed the ESS rate averaged over 3000 iterations. We used the `NLTCS` data set and the PCG-I sampler. Figure 5 presents the results in terms of the speed-up relative to the ESS rate for $B = 1$. We observe a near-linear speed-up in B, with the exception of $B = 32$. The speed-up is expected to taper off with

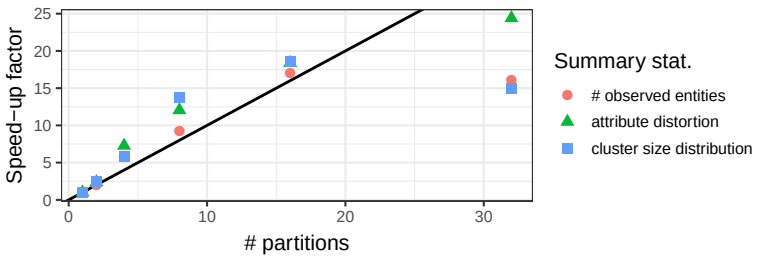

Figure 5: Efficiency of `d-blink` as a function of the number of blocks B and summary statistic of interest (larger is better). The speed-up measures the ESS rate relative to the ESS rate for $B = 1$ (no blocking) for the `NLTCS` data set.

increasing numbers of blocks, as parallel gains in efficiency are overcome by losses due to communication costs and/or poorer mixing. This tipping point seems difficult to predict for a given set up, as it depends on complex factors such as the data distribution, the splitting rules used, and the hardware characteristics.

Sampling methods and efficiency. We evaluated the efficiency of the three samplers introduced in Section 5.1 (Gibbs, PCG-I and PCG-II). As above, we computed the ESS rate as an average over 3000 iterations. We set $B = 16$ and used the `NLTCS` data set. The results, shown in Figure 6, indicate that the PCG-I sampler is considerably more efficient (by a factor of 1.5–2×) than the baseline Gibbs sampler for this data set. We also observe that the PCG-II sampler performs quite poorly in comparison: between 20–30× slower than the Gibbs sampler. This is because the marginalization and trimming for the $\mathbf{\Lambda}$ update for PCG-II prevents us from applying the trick described in Section 6.1. Thus although PCG-II is expected to be more efficient in terms of reducing autocorrelation, it is less efficient overall as each iteration is too computationally expensive.

7.2 Linkage quality

Though not our primary focus, we assessed the performance of `d-blink` in terms of its predictions for the linkage structure (the matching step) for the data sets in Table 3. This was not previously possible with `blink`, as it could only scale to small data sets of around

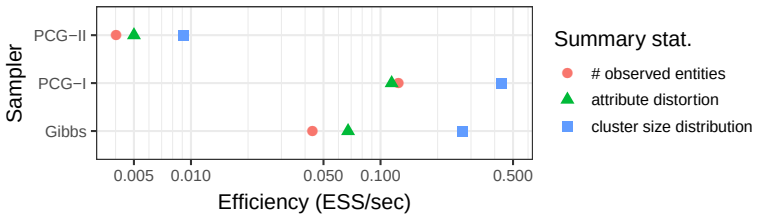

Figure 6: Efficiency of `d-blink` as a function of the sampler and summary statistic of interest (larger is better). All measurements are for the NLTCS data set with $B = 16$.

1000 records.

Point estimate methodology. To evaluate the matching performance of `d-blink` with respect to the ground truth, we extracted a point estimate of the linkage structure from the posterior using the *shared most probable maximal matching sets (sMPMMS)* method (Steorts et al., 2016). This method circumvents the problem of label switching (Jasra et al., 2005)—where the identities of the entities do not remain constant along the Markov chain.

The sMPMMS method involves two main steps. In the first step, the most probable entity cluster is computed for each record based on the posterior samples. In general, these entity clusters will conflict with one another—e.g. the most probable entity cluster for r_1 might be (r_1, r_2) while for r_2 it is (r_1, r_2, r_3). The second step resolves these conflicts by assigning precedence to links between records and their most probable entity clusters. The result is a globally-consistent estimate of the linkage structure—i.e. it satisfies transitivity.

We distributed the computation of the sMPMMS method in the Spark framework. We used 9000 approximate posterior samples which were derived from a Markov chain of length 10^5 by discarding the first 10^4 iterations as burn-in[6] and applying a thinning interval of 10. These parameters were chosen by inspection of trace plots, some of which are reported in Appendix L. By contrast to the point estimates reported here, we also examined full posterior estimation in Appendix I.

Baseline methods. We compared the linkage quality of `d-blink` with three baseline methods as described below. We focused on (scalable) unsupervised methods as we assumed

[6]We applied a burn-in of 210k iterations for `NCVR` as it was slow to converge.

Table 4: Comparison of matching quality. "ARI" stands for adjusted Rand index and "Err. # clust." is the percentage error in the number of clusters.

Data set	Method	Pairwise measures			Cluster measures	
		Precision	Recall	F1-score	ARI	Err. # clust.
ABSEmployee	`d-blink`	0.9763	0.8530	**0.9105**	**0.9105**	**+1.667%**
	Fellegi-Sunter (10)	**0.9963**	0.8346	0.9083	—	—
	Fellegi-Sunter (100)	**0.9963**	0.8346	0.9083	—	—
	Near Matching	0.0378	**0.9930**	0.0728	—	—
	Exact Matching	0.9939	0.8346	0.9074	0.9074	+9.661%
NCVR	`d-blink`	0.9146	**0.9654**	**0.9393**	**0.9392**	**–3.587%**
	Fellegi-Sunter (10)	0.9868	0.7874	0.9083	—	—
	Fellegi-Sunter (100)	0.9868	0.7874	0.9083	—	—
	Near Matching	0.9899	0.7443	0.8497	—	—
	Exact Matching	**0.9925**	0.0017	0.0034	0.0034	+51.09%
NLTCS	`d-blink`	0.8319	0.9103	0.8693	0.8693	–22.09%
	Fellegi-Sunter (10)	**0.9094**	0.9087	**0.9090**	—	—
	Fellegi-Sunter (100)	**0.9094**	0.9087	**0.9090**	—	—
	Near Matching	0.0600	**0.9563**	0.1129	—	—
	Exact Matching	0.8995	0.9087	0.9040	**0.9040**	**+2.026%**
SHIW0810	`d-blink`	**0.2514**	0.5396	**0.3430**	**0.3429**	–37.65%
	Fellegi-Sunter (10)	0.0028	0.9050	0.0056	—	—
	Fellegi-Sunter (100)	0.0025	**0.9161**	0.0050	—	—
	Near Matching	0.0043	0.9111	0.0086	—	—
	Exact Matching	0.1263	0.7608	0.2166	0.2166	**–37.40%**
RLdata10000	`d-blink`	0.6334	**0.9970**	0.7747	**0.7747**	**–10.97%**
	Fellegi-Sunter (10)	0.9957	0.6174	0.7622	—	—
	Fellegi-Sunter (100)	0.9364	0.8734	0.9038	—	—
	Near Matching	0.9176	0.9690	**0.9426**	—	—
	Exact Matching	**1.0000**	0.0080	0.0159	0.0159	+11.02%

very little to no training data was available.

- *Exact Matching.* Links records that match on all A attributes. It is unsupervised and ensures transitivity.

- *Near Matching.* Links records that match on at least $L-1$ attributes. It is unsupervised, but does not guarantee transitivity.

- *Fellegi-Sunter.* Links records according to a pairwise match score that is a weighted sum of attribute-level dis/agreements. The weights are specified by the Fellegi-

Sunter model (Fellegi and Sunter, 1969) and were estimated using the expectation-maximization algorithm, as implemented in the `RecordLinkage` R package (Sariyar and Borg, 2010). We chose the threshold on the match score to optimize the F1-score using a small amount of training data (size 10 and 100). This makes the method semi-supervised. Note that the training data was sampled in a biased manner to deal with the imbalance between the matches and non-matches (half with match scores above zero and half below). The method does not guarantee transitivity.

Results. Table 4 presents performance measures categorized by data set and method. The pairwise performance measures (precision, recall and F1-score) are provided for all methods, however the cluster performance measures (adjusted Rand Index, see Vinh et al., 2010, and percentage error in the number of clusters) are only valid for methods that guarantee transitivity of closure (`d-blink` and Exact Matching). Despite being fully unsupervised, `d-blink` achieves competitive performance when compared to the semi-supervised Fellegi-Sunter method. The two simple baselines, Near Matching and Exact Matching, are acceptable for data sets with low noise but perform poorly otherwise (e.g. `NCVR` and `RLdata10000`). We conducted an empirical sensitivity analysis for `d-blink` with respect to variations in the hyperparameters. The results for `RLdata10000` (included in Appendix J) show that `d-blink` is somewhat sensitive to all of the hyperparameters tested, however sensitivity is in general predictable, following clear and intuitive trends. One interesting observation is the fact that `d-blink` tends to overestimate the amount of distortion. This is perhaps not surprising given the absence of ground truth.

8 APPLICATION TO THE 2010 U.S. DECENNIAL CENSUS

National statistics agencies frequently need to link inter- or intra-agency data sets, for a number of purposes such as quality control. One critical problem in the United States (U.S.) occurs every ten years, when the U.S. Census Bureau must enumerate the population in each State as mandated under the U.S. Constitution, Article I, Section 2. The enumeration is used to apportion the representation of legislators, and to allocate resources for housing,

highways, schools, assistance programs, and other projects that are vital to the prosperity, welfare, and economic growth of the U.S. As the country grows and becomes more diverse, it becomes more challenging to produce an accurate enumeration. Many individuals elect not to fill out census forms, which results in them not being counted in the enumeration. Other individuals may be counted multiple times due to duplicate responses. For example, students attending universities or private schools (living in group quarters) are often double counted as they are legally required to be counted by their university/school, while also being counted by their parents/guardians as part of a household.

Motivated by these data duplication issues, we apply `d-blink` to conduct an enumeration in the state of Wyoming. In order to improve coverage, we combine records from the 2010 Decennial Census with administrative records from the Social Security Administration's Numerical Identification System (Numident).[7] In total, we consider 1,050,000 records representing the population of Wyoming: a subset of 494,000 records from the 2010 Decennial Census and 556,000 records from the Numident.[8] Our goal is to recover the unique individuals represented in these records using unsupervised ER. We apply `d-blink` using the overlapping attributes from the Census and the Numident: first and last name, date of birth, gender, and zip code. We treat first and last name as string-type attributes and the remaining attributes as categorical. To manage scalability, we utilize the k-d tree blocking function outlined in Section 4.2, splitting recursively on gender and birth year at each level of the tree. Inference and MCMC diagnostics are discussed in Appendix K.

After performing ER using `d-blink`, we are able to provide a posterior estimate of the total number of unique individuals represented in both data sets. Table 5 reports a point estimate based on the mean. The standard error is quite narrow, which is consistent with knowledge of the uniform prior (Steorts et al., 2016). We find that our estimate is significantly larger than the unadjusted count of 563,626 reported by Rastogi et al. (2012). The difference may be explained by several factors. Firstly, our approach may capture individuals who are not represented in the Census, but who are represented in the Numident

[7]The Numident is the Social Security Administration's computer database file of an abstract of the information contained in an application for a U.S. Social Security number.

[8]These figures have been rounded to the nearest thousand as they are protected under Title 13.

Table 5: Results for ER of 2010 Census and Numident data in Wyoming. Pairwise evaluation measures are computed using ground truth identifiers available for a subset of the records.

Pairwise measures			Posterior population size	
Precision	Recall	F1-score	Mean	Std. error
0.97	0.84	0.90	616,000	5,000

(assuming they have a Social Security number). Indeed, the participation rate for the Census is known to be lower in Wyoming than for other states (United States Census Bureau, n.d.). Secondly, there may be some double-counting for records that cannot be reliably linked—e.g. due to missing or unreliable attribute values. Thirdly, there may be minor differences in the Census data—e.g. whether blank forms are discarded or not.

To assess the reliability of ER, we report pairwise evaluation measures (precision, recall and F1-score) in Table 5. These measures are computed using ground truth identifiers, which are available for a limited subset of the records. To our knowledge, these are the first performance measures that have been published for ER of Census and administrative data at the state-level. However, we note that the measures should be interpreted with caution, as the limited ground truth may not be representative of all records (hence the need for unsupervised methods).

We believe that `d-blink` shows promise in producing enumerations at the state-level, while accounting for ER uncertainty. Moving forward, it would be beneficial to study the accuracy and scalability of `d-blink` in other states, to further assess the reliability of our methodology for conducting linkage tasks within national statistical agencies. While it is beyond the scope of this paper, we are also interested in incorporating additional sources of administrative data, such as tax records, in future work.

9 CONCLUDING REMARKS

In this paper we have proposed `d-blink`—a method for performing scalable ER with integrated blocking in a fully Bayesian framework. Our approach leverages an auxiliary variable representation, which partitions the latent entities and records into auxiliary blocks. Since the auxiliary blocks are not fixed, but inferred during inference, we are able to propagate uncertainty between the blocking and ER stages. This stands in contrast with the existing literature, where blocking and ER are performed in two separate stages without uncertainty propagation. In addition, we have shown that our approach does not compromise the correctness of the marginal posterior over the model parameters. In other words, approximate posterior samples produced by `d-blink` are independent of the blocking design in the asymptotic limit.

To further improve scalability, we discussed inference for `d-blink` in a distributed/ parallel setting. We proposed a blocking function based on k-d trees, which achieves good load balancing at the block level. We designed a distributed partially-collapsed Gibbs sampler, with superior mixing properties compared to a standard Gibbs sampler. We also presented fast algorithms for the Gibbs updates, which leverage indexing data structures and perturbation sampling. Our empirical evaluation on five data sets demonstrated efficiency gains for `d-blink` in excess of $300\times$ when compared to existing methods. We also demonstrated the potential of `d-blink` in a population enumeration case study, using data from the 2010 Decennial Census and Social Security Administration. The resulting enumeration was output with uncertainty, and achieved high precision and recall.

An implementation of `d-blink` is provided as an open-source Apache Spark package. We also provide an interface for R users, for broad accessibility. Our software has been put in place within the United States Census Bureau for research purposes.

SUPPLEMENTAL MATERIALS

Appendices: Includes proofs, further details about the experimental setup, and additional results. (PDF file)

Code: An implementation of `d-blink` in Apache Spark and a corresponding R interface. (Zip file)

Data: An archive containing data sets that we have permission to redistribute. (Zip file)

ACKNOWLEDGEMENTS

The authors would also like to thank the anonymous reviewers, Associate Editor and Editor for their valuable comments and helpful suggestions. N. Marchant acknowledges the support of an Australian Government Research Training Program Scholarship and the AMSIIntern program hosted by the Australian Bureau of Statistics. R. C. Steorts and A. Kaplan acknowledge the support of NSF SES-1534412 and CAREER-1652431. B. Rubinstein acknowledges the support of Australian Research Council grant DP150103710. N. Marchant and B. Rubinstein also acknowledge support of Australian Bureau of Statistics project ABS2018.363.

REFERENCES

Ahn, S., Shahbaba, B., and Welling, M. "Distributed Stochastic Gradient MCMC." In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–1044–II–1052. Beijing, China: JMLR.org (2014).

Bentley, J. L. "Multidimensional Binary Search Trees Used for Associative Searching." *Commun. ACM*, 18(9):509–517 (1975).

Bilenko, M. and Mooney, R. J. "Adaptive Duplicate Detection Using Learnable String Similarity Measures." In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, 39–48. New York, NY, USA: ACM (2003).

Chang, J. and Fisher, J. W., III. "Parallel Sampling of DP Mixture Models Using Sub-clusters Splits." In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 1 of *NIPS'13*, 620–628. NY, USA: Curran Associates Inc. (2013).

Christen, P. "A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication." *IEEE Transactions on Knowledge and Data Engineering*, 24(9):1537–1555 (2012a).

—. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-Centric Systems and Applications. Berlin Heidelberg: Springer-Verlag (2012b).

Copas, J. B. and Hilton, F. J. "Record Linkage: Statistical Models for Matching Computer Records." *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 153(3):287–320 (1990).

Dong, X. L. and Srivastava, D. "Big Data Integration." *Synthesis Lectures on Data Management*, 7(1):1–198 (2015).

Enamorado, T., Fifield, B., and Imai, K. "Using a Probabilistic Model to Assist Merging of Large-Scale Administrative Records." *American Political Science Review*, 113(2):353–371 (2019).

Fan, W., Jia, X., Li, J., and Ma, S. "Reasoning About Record Matching Rules." *Proc. VLDB Endow.*, 2(1):407–418 (2009).

Fellegi, I. P. and Sunter, A. B. "A Theory for Record Linkage." *Journal of the American Statistical Association*, 64(328):1183–1210 (1969).

Flegal, J. M., Hughes, J., Vats, D., and Dai, N. *mcmcse: Monte Carlo Standard Errors for MCMC*. Riverside, CA, Denver, CO, Coventry, UK, and Minneapolis, MN (2017). R package version 1.3-2.

Fortini, M., Liseo, B., Nuccitelli, A., and Scanu, M. "On Bayesian Record Linkage." *Research in Official Statistics*, 4(1):185–198 (2001).

Friedman, J. H., Bentley, J. L., and Finkel, R. A. "An Algorithm for Finding Best Matches in Logarithmic Expected Time." *ACM Trans. Math. Softw.*, 3(3):209–226 (1977).

Ge, H., Chen, Y., Wan, M., and Ghahramani, Z. "Distributed Inference for Dirichlet Process Mixture Models." In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, 2276–2284. Lille, France: PMLR (2015).

Getoor, L. and Machanavajjhala, A. "Entity Resolution: Theory, Practice & Open Challenges." *Proc. VLDB Endow.*, 5(12):2018–2019 (2012).

Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., and Zhu, X. "Corleone: Hands-off Crowdsourcing for Entity Matching." In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, 601–612. New York, NY, USA: ACM (2014).

Gutman, R., Afendulis, C. C., and Zaslavsky, A. M. "A Bayesian Procedure for File Linking to Analyze End-of-Life Medical Costs." *Journal of the American Statistical Association*, 108(501):34–47 (2013).

Herzog, T. N., Scheuren, F. J., and Winkler, W. E. *Data Quality and Record Linkage Techniques*. New York: Springer-Verlag (2007).

Jain, S. and Neal, R. M. "A Split-Merge Markov chain Monte Carlo Procedure for the Dirichlet Process Mixture Model." *Journal of Computational and Graphical Statistics*, 13(1):158–182 (2004).

Jasra, A., Holmes, C. C., and Stephens, D. A. "Markov Chain Monte Carlo Methods and the Label Switching Problem in Bayesian Mixture Modeling." *Statistical Science*, 20(1):50–67 (2005).

Lahiri, P. and Larsen, M. D. "Regression Analysis With Linked Data." *Journal of the American Statistical Association*, 100(469):222–230 (2005).

Larsen, M. D. "Advances in Record Linkage Theory: Hierarchical Bayesian Record Linkage Theory." In *Proceedings of the Survey Research Methods Section*, 3277–3284. American Statistical Association (2005).

—. "An experiment with hierarchical Bayesian record linkage." (2012). arXiv:1212.5203.

Lesot, M.-J., Rifqi, M., and Benhadda, H. "Similarity measures for binary and numerical data: a survey." *International Journal of Knowledge Engineering and Soft Data Paradigms*, 1(1):63–84 (2008).

Little, R. J. A. and Rubin, D. B. *Statistical Analysis with Missing Data*. Wiley (2002).

Liu, J. S. *Monte Carlo Strategies in Scientific Computing*. Springer Series in Statistics. New York: Springer-Verlag (2004).

Lovell, D., Malmaud, J., Adams, R. P., and Mansinghka, V. K. "ClusterCluster: Parallel Markov Chain Monte Carlo for Dirichlet Process Mixtures." (2013). arXiv:1304.2302.

McVeigh, B. S. and Murray, J. S. "Practical Bayesian Inference for Record Linkage." (2017). arXiv:1710.10558.

McVeigh, B. S., Spahn, B. T., and Murray, J. S. "Scaling Bayesian Probabilistic Record Linkage with Post-Hoc Blocking: An Application to the California Great Registers." (2019). arXiv:1905.05337.

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., Deep, R., Arcaute, E., and Raghavendra, V. "Deep Learning for Entity Matching: A Design Space Exploration." In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, 19–34. New York, NY, USA: ACM (2018).

Newcombe, H. B., Kennedy, J. M., Axford, S. J., and James, A. P. "Automatic Linkage of Vital Records: Computers can be used to extract "follow-up" statistics of families from files of routine records." *Science*, 130(3381):954–959 (1959).

Newman, D., Asuncion, A., Smyth, P., and Welling, M. "Distributed algorithms for topic models." *Journal of Machine Learning Research*, 10(Aug):1801–1828 (2009).

Papadakis, G., Svirsky, J., Gal, A., and Palpanas, T. "Comparative Analysis of Approximate Blocking Techniques for Entity Resolution." *Proc. VLDB Endow.*, 9(9):684–695 (2016).

Price, M., Klinger, J., Qtiesh, A., and Ball, P. "Updated Statistical Analysis of Documentation of Killings in the Syrian Arab Repulic." (2013).

Rastogi, S., O'Hara, A., Noon, J., Zapata, E. A., Espinoza, C., Marshall, L. B., Schellhamer, T. A., and Brown, J. D. "2010 Census Match Study." Technical report, Center for Administrative Records Research and Applications, United States Census Bureau (2012).

Sadinle, M. "Detecting duplicates in a homicide registry using a Bayesian partitioning approach." *Ann. Appl. Stat.*, 8(4):2404–2434 (2014).

—. "Bayesian Estimation of Bipartite Matchings for Record Linkage." *Journal of the American Statistical Association*, 112(518):600–612 (2017).

Sadinle, M. and Fienberg, S. E. "A Generalized Fellegi-Sunter Framework for Multiple Record Linkage With Application to Homicide Record Systems." *Journal of the American Statistical Association*, 108(502):385–397 (2013).

Saria, S. "A $3 trillion challenge to computational scientists: Transforming healthcare delivery." *IEEE Intelligent Systems*, 29(4):82–87 (2014).

Sariyar, M. and Borg, A. "The RecordLinkage Package: Detecting Errors in Data." *The R Journal*, 2(2):61–67 (2010).

Singh, R., Meduri, V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J.-A., Solar-Lezama, A., and Tang, N. "Generating Concise Entity Matching Rules." In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, 1635–1638. New York, NY, USA: ACM (2017).

Smola, A. and Narayanamurthy, S. "An Architecture for Parallel Topic Models." *Proc. VLDB Endow.*, 3(1-2):703–710 (2010).

Soon, W. M., Ng, H. T., and Lim, D. C. Y. "A Machine Learning Approach to Coreference Resolution of Noun Phrases." *Computational linguistics*, 27(4):521–544 (2001).

Steorts, R. C. "Entity Resolution with Empirically Motivated Priors." *Bayesian Analysis*, 10(4):849–875 (2015).

Steorts, R. C., Barnes, M., and Neiswanger, W. "Performance Bounds for Graphical Record Linkage." In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, 298–306. Fort Lauderdale, FL, USA: PMLR (2017).

Steorts, R. C., Hall, R., and Fienberg, S. E. "A Bayesian Approach to Graphical Record Linkage and Deduplication." *Journal of the American Statistical Association*, 111(516):1660–1672 (2016).

Steorts, R. C., Ventura, S. L., Sadinle, M., and Fienberg, S. E. "A Comparison of Blocking Methods for Record Linkage." In Domingo-Ferrer, J. (ed.), *Privacy in Statistical Databases*, Lecture Notes in Computer Science, 253–268. Cham: Springer International Publishing (2014).

Tancredi, A. and Liseo, B. "A hierarchical Bayesian approach to record linkage and population size problems." *The Annals of Applied Statistics*, 5(2B):1553–1585 (2011).

Tancredi, A., Steorts, R., and Liseo, B. "A Unified Framework for De-Duplication and Population Size Estimation." *Bayesian Analysis* (2020).

Turek, D., Valpine, P. d., and Paciorek, C. J. "Efficient Markov chain Monte Carlo sampling for hierarchical hidden Markov models." *Environmental and Ecological Statistics*, 23(4):549–564 (2016).

United States Census Bureau. "2010 Census Participation Rates." https://www.census.gov/data/datasets/2010/dec/2010-participation-rates.html (n.d.). Accessed: 2020-05-25.

van Dyk, D. A. and Park, T. "Partially Collapsed Gibbs Samplers." *Journal of the American Statistical Association*, 103(482):790–796 (2008).

Vats, D., Flegal, J. M., and Jones, G. L. "Multivariate output analysis for Markov chain Monte Carlo." *Biometrika*, 106(2):321–337 (2019).

Vinh, N. X., Epps, J., and Bailey, J. "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance." *Journal of Machine Learning Research*, 11(Oct):2837–2854 (2010).

Vose, M. D. "A linear algorithm for generating random numbers with a given distribution." *IEEE Transactions on Software Engineering*, 17(9):972–975 (1991).

Wang, J., Kraska, T., Franklin, M. J., and Feng, J. "CrowdER: Crowdsourcing Entity Resolution." *Proc. VLDB Endow.*, 5(11):1483–1494 (2012).

Williamson, S., Dubey, A., and Xing, E. "Parallel Markov Chain Monte Carlo for Nonparametric Mixture Models." In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, 98–106. Atlanta, Georgia, USA: PMLR (2013).

Winkler, W. E. "The State of Record Linkage and Current Research Problems." Technical report, Statistical Research Division, U.S. Bureau of the Census (1999).

—. "Machine Learning, Information Retrieval, and Record Linkage." In *Proceedings of the Section on Survey Research Methods, 20–29*. American Statistical Association (2000).

—. "Methods for Record Linkage and Bayesian Networks." Technical Report Statistics #2002-05, U.S. Bureau of the Census (2002).

—. "Overview of Record Linkage and Current Research Directions." Technical Report Statistics #2006-2, Statistical Research Division, U.S. Census Bureau (2006).

—. "Matching and record linkage." *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(5):313–325 (2014).

Yujian, L. and Bo, L. "A Normalized Levenshtein Distance Metric." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1091–1095 (2007).

Zanella, G. "Informed Proposals for Local MCMC in Discrete Spaces." *Journal of the American Statistical Association*, 115(530):852–865 (2020).

Zanella, G., Betancourt, B., Wallach, H., Miller, J., Zaidi, A., and Steorts, R. C. "Flexible Models for Microclustering with Application to Entity Resolution." In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 1425–1433. NY, USA: Curran Associates Inc. (2016).