ZeroER: Entity Resolution using Zero Labeled Examples

Renzhi Wu
Georgia Institute of Technology
renzhiwu@gatech.edu

Sanya Chaba
Georgia Institute of Technology
sanyachaba@gatech.edu

Saurabh Sawlani
Georgia Institute of Technology
sawlani@gatech.edu

Xu Chu
Georgia Institute of Technology
xu.chu@cc.gatech.edu

Saravanan Thirumuruganathan
QCRI, HBKU
sthirumuruganathan@hbku.edu.qa

ABSTRACT

Entity resolution (ER) refers to the problem of matching records in one or more relations that refer to the same real-world entity. While supervised machine learning (ML) approaches achieve the state-of-the-art results, they require a large amount of labeled examples that are expensive to obtain and often times infeasible. We investigate an important problem that vexes practitioners: is it possible to design an effective algorithm for ER that requires *Zero* labeled examples, yet can achieve performance comparable to supervised approaches? In this paper, we answer in the affirmative through our proposed approach dubbed ZeroER.

Our approach is based on a simple observation — *the similarity vectors for matches should look different from that of unmatches.* Operationalizing this insight requires a number of technical innovations. First, we propose a simple yet powerful generative model based on Gaussian Mixture Models for learning the match and unmatch distributions. Second, we propose an adaptive regularization technique customized for ER that ameliorates the issue of feature overfitting. Finally, we incorporate the transitivity property into the generative model in a novel way resulting in improved accuracy. On five benchmark ER datasets, we show that ZeroER greatly outperforms existing unsupervised approaches and achieves comparable performance to supervised approaches.

SIGMOD'20, June 14–19,2020, Portland, OR, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6735-6/20/06…$15.00
https://doi.org/10.1145/3318464.3389743

CCS CONCEPTS

• Information systems → Entity resolution; **• Computing methodologies → Unsupervised learning**.

KEYWORDS

entity resolution; entity matching; unsupervised learning

ACM Reference Format:
Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity Resolution using Zero Labeled Examples. In *Proceedings of the 2020 ACM SIGMOD International Conference onManagement of Data (SIGMOD'20), June 14–19, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3318464.3389743

1 INTRODUCTION

Entity resolution (ER) – also known as duplicate detection or record matching – refers to the problem of identifying tuples in one or more relations that refer to the same real world entity. For example, an e-commerce website would want to identify duplicate products (such as from different suppliers) so that they could all be listed in the same product page. ER has been extensively studied in many research communities, including databases, statistics, NLP, and data mining (e.g., see multiple surveys [25, 29, 31, 34, 41, 45]).

The standard approach for entity resolution is to use textual similarities between two tuples to determine if they are a match or unmatch [29]. Informally, in Figure 1(a)(b), the matching pair (fd1, zg2) is textually similar, while the non-matching pair (fd1, zg1) is not. Typically, a vector of similarity scores (aka, feature vector) is computed for a tuple pair by applying various similarity functions (such as edit distance, or jaccard similarity) on corresponding attributes. Multiple similarity scores can be used for a single attribute. Figure 1(c) shows a subset of the similarity scores computed by the open-source ER package Magellan[39]. For example, the feature cos_qgm_3_qgm_3 represents the cosine similarity between 3-grams on the name attribute between a pair of tuples. Given the computed feature vectors, ER is typically formulated as a binary classification problem where a classifier is trained to determine whether a tuple pair is a match or an unmatch based on its similarity vector.

The Need for ZeroER. Supervised machine Learning (ML) approaches provide the state-of-the-art results for ER [26, 28, 40, 44]. Powerful techniques such as deep learning have a voracious appetite for large amounts of labeled training data. For example, both DeepER [28] and DeepMatcher [44] require hundreds to thousands of labeled examples. Even non-deep learning based methods require hundreds of labeled examples [40]. It has been reported [26] that achieving F-measures of ~99% with random forests can require up to 1.5M labels even for relatively clean datasets. Hence, generating large amounts of training data is a time consuming and challenging task even for domain experts.

Furthermore, organisations often have many relations to be de-duplicated (e.g., General Electric has 75 different procurement systems [52]), and generating labeled data becomes a huge issue. Due to this, there has been intense interest from the database community in self-service data preparation that requires limited efforts from domain scientists. Motivated by these considerations, we develop a ZeroER approach that can perform ER with comparable performance to supervised approaches, but with zero labeled examples.

Key Insights and Challenges. Designing a ZeroER approach that achieves competitive performance across different datasets is extremely challenging. We highlight three fundamental ideas that make this possible.
(1) Generative Modelling. Supervised approaches learn to tell apart matches and unmatches from the large amount of labeled training data. However, ZeroER does not have this useful information and has to design a mechanism to distinguish the matches from unmatches. ZeroER leverages a blindingly simple yet powerful observation: *the similarity vectors (aka feature vectors) for matches should look different from the similarity vectors for unmatches.*

We plot in Figure 2 the distribution of feature values for the fodors-zagats dataset, using ground-truth matches and unmatches. For ease of visualization, we only show the distribution along two feature dimensions. As we can see, the similarity values of these two features for the matches concentrate in the upper right corner, while the values for unmatches in the bottom left corner.

This observation inspires us to use *generative modelling for the ER problem* — the feature vector $\mathbf{x}$ of a match can be assumed to be generated according to one distribution (termed the M-Distribution), and the feature vector $\mathbf{x}$ of an unmatch can be assumed to be generated according to a different distribution (termed the U-Distribution).

If we can learn the probability density functions (PDFs) of two distributions, i.e. $p(\mathbf{x}|y = M)$ and $p(\mathbf{x}|y = U)$, and the prior probability of a tuple pair being a match $\pi_M = p(y = M)$, or a non-match $\pi_U = p(y = U) = 1 - \pi_M$, then determining whether a given tuple pair should be classified as match is equivalent to computing the posterior probability $p(y = M|\mathbf{x})$ for its feature vector $\mathbf{x}$ using the Bayes rule:

$$p(y = M|\mathbf{x}) = \frac{p(\mathbf{x}, y = M)}{p(\mathbf{x})} = \frac{p(\mathbf{x}, y = M)}{p(\mathbf{x}, y = M) + p(\mathbf{x}, y = U)}$$
$$= \frac{\pi_M \times p(\mathbf{x}|y = M)}{\pi_M \times p(\mathbf{x}|y = M) + \pi_U \times p(\mathbf{x}|y = U)} \quad (1)$$

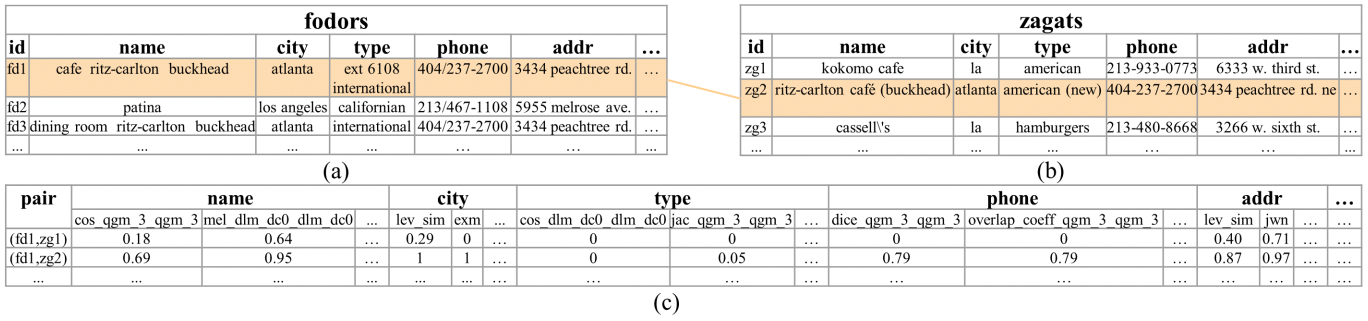
Accurately learning the prior π_M, the M-Distribution $p(\mathbf{x}|y = M)$, and the U-Distribution $p(\mathbf{x}|y = U)$ – with Zero labeled data – is the key to achieve high-quality predictions.

Let θ_M (resp. θ_U) denote the parameters of the M-Distribution (resp. U-Distribution). Different choices of θ_M and θ_U correspond to different generative models. We choose to use Gaussian distributions for both the M- and the U-Distribution, as the Gaussian distribution is known to be the default distribution used in various areas to represent real-valued random variables with unknown distributions [42]. In our case, the two distributions would be multi-dimensional Gaussian distributions parameterized by a mean vector and a co-variance matrix $\theta_C = \{\mu_C, \Sigma_C\}$, where $C \in \{M, U\}$.

This reduces the generative model to the Gaussian Mixture Model (GMM) with two components, and all parameters $\{\pi_M, \mu_C, \Sigma_C\}, C \in \{M, U\}$ can then be learned by maximizing the data likelihood via an expectation-maximization algorithm without any labeled data [24]. However, using the naive GMM as the generative model for ER produces inferior results, since it fails to recognize one important characteristic of the ER problem, namely, the number of matches is usually a tiny fraction of the total number of tuple pairs. The lack of matching pairs makes learning θ_M particularly difficult.

Example 1.1. In the fodors-zagats dataset, we have 112 matches in the ground truth, and we have seven attributes. An out-of-box invocation of the open-source Magellan package [39] generated 68 features for those seven attributes, which means we have 2346 ($= 68 + 68 * 67/2$) parameters in Σ_M. Using a small number of examples (e.g., 112) to estimate a large co-variance matrix (e.g., 2346 parameters) causes inaccurate estimates [23, 54].

In Section 3, we show how existing approaches to reduce the number of parameters in the naive GMM is not suitable for ER, and we will introduce ZeroER's generative model that includes two novel and ER-specific adaptations to the naive GMM. Our generative model would only require $(4d + 1)$ total number of parameters for a dataset with d-dimensional features.
(2) Feature Regularization. In supervised ML settings, overfitting is a well-known problem that causes an ML model to fit too closely or exactly to a particular set of data (usually



fodors						
id	**name**	**city**	**type**	**phone**	**addr**	**…**
fd1	cafe ritz-carlton buckhead	atlanta	ext 6108 international	404/237-2700	3434 peachtree rd.	…
fd2	patina	los angeles	californian	213/467-1108	5955 melrose ave.	…
fd3	dining room ritz-carlton buckhead	atlanta	international	404/237-2700	3434 peachtree rd.	…
...	...	...	...	…	…	...

(a)

zagats						
id	**name**	**city**	**type**	**phone**	**addr**	**…**
zg1	kokomo cafe	la	american	213-933-0773	6333 w. third st.	…
zg2	ritz-carlton café (buckhead)	atlanta	american (new)	404-237-2700	3434 peachtree rd. ne	…
zg3	cassell\'s	la	hamburgers	213-480-8668	3266 w. sixth st.	…
...	...	...	...	…	…	...

(b)

pair	**name**			**city**			**type**			**phone**			**addr**			**…**
	cos_qgm_3_qgm_3	mel_dlm_dc0_dlm_dc0	…	lev_sim	exm	…	cos_dlm_dc0_dlm_dc0	jac_qgm_3_qgm_3	…	dice_qgm_3_qgm_3	overlap_coeff_qgm_3_qgm_3	…	lev_sim	jwn	…	…
(fd1,zg1)	0.18	0.64	…	0.29	0	…	0	0	…	0	0	…	0.40	0.71	…	…
(fd1,zg2)	0.69	0.95	…	1	1	…	0	0.05	…	0.79	0.79	…	0.87	0.97	…	…
...	...	...	…	...	...	…	…	…	…	…	…	…	…	…	…	…

(c)

Figure 1: ER on a benchmark restaurant dataset [2] with two tables (a) fodors table and (b) zagats table. (c) shows similarity vectors (aka feature vectors) generated by the open-source ER package Magellan [39].

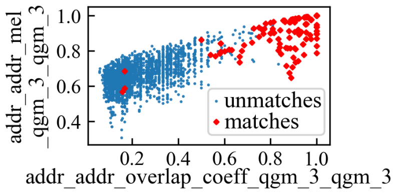


Figure 2: Distribution of matches and unmatches on two features in the fodors-zagats dataset.

the training set), and may therefore fail to fit additional data or predict future observations reliably well (usually the test set). As we were developing and testing generative models for ZeroER, we discovered a similar "overfitting" problem that causes the generative model to fit too closely or exactly to some particular features, such that other features play almost no role in making predictions.

Example 1.2. For simplicity, let us illustrate the overfitting problem with two features in the fodors-zagats dataset. The first feature f_1 : *city_city_exm* represents whether the city attribute of a tuple pair is the same or not, and the second feature f_2 : *phone_phone_jar* represents the jaccard similarity between the phone attribute of a tuple pair. Figure 3 shows the frequency of tuple pairs taking different values of these two features using ground-truth matches and unmatches. As we can see, matches and unmatches may have the same or different city attribute; and matches usually have a much higher jaccard similarity on phone attribute than unmatches.

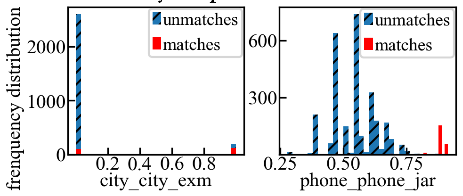


Figure 3: Distribution of two features in the fodors zagats dataset.

For ease of illustration, let us assume that f_1 and f_2 are independent (though the same overfitting occurs even if features are dependent). Therefore, the M-Distribution (similarly, the U-Distribution) can be written as $p(\mathbf{x}|y = M) = p(\mathbf{x}[f_1]|y = M) \times p(\mathbf{x}[f_2]|y = M)$, i.e., we can fit the two features independently.

Fitting a GMM *without labeled data* on f_1 would give the following two components: the first component has a mean of 1.0 and a variance of 0, and the second component has a mean of 0.0 and a variance of 0, as f_1 has two clearly separated density regions. As we know matches generally have higher similarities than unmatches, we conclude that the first component corresponds to $p(\mathbf{x}[f_1]|y = M)$, and the second component corresponds to $p(\mathbf{x}[f_1]|y = U)$. Similarly, suppose fitting a GMM *without labeled data* on f_2 gives the following: $p(\mathbf{x}[f_2]|y = M)$ has a mean of 0.9 and a variance of 0.1. $p(\mathbf{x}[f_2]|y = U)$ has a mean of 0.6 and a variance of 0.2.

If we use the above learned distributions to make predictions, a feature vector $\mathbf{x}$ with $x[f_1] = 1.0$ will be surely predicted as a match, regardless of the value of $x[f_2]$. This is because $p(\mathbf{x}[f_1]|y = M)$ will have an infinite density when $x[f_1] = 1.0$ due to its 0 variance. Therefore, $p(\mathbf{x}|y = M)$ will be infinity, which gives $p(y = M|\mathbf{x}) = 1.0$ using Equation (1). Similarly, a feature vector $\mathbf{x}$ with $x[f_1] = 0$ will be surely predicted as a unmatch, regardless of the value of $x[f_2]$.

The GMM will predict matches or unmatches solely based on f_1, and f_2 plays no role. This is problematic, as we know in the ground truth (see Figure 3), there are matches with $x[f_1] = 0$ and unmatches with $x[f_1] = 1$. If the model took into account f_2 properly, then we may be able to prevent these erroneous predictions.

While the above example shows an extreme case where one feature entirely dictates the prediction, in practice, we see many features with extremely small variance that essentially dominate the computation, leaving other features to have little impact on the prediction. Note that this feature overfitting problem is caused by degenerate features, so it happens regardless of the amount of data.

Our observed feature overfitting in the extreme case, where one feature entirely dictates the prediction, is similar to the known *singularity* problem in pattern recognition where the determinant of a covariance matrix is zero [13]. A common solution to address the singularity problem is to add a small constant value to the diagonal entries of the covariance matrix [5]. While this avoids having infinite values in the computation, it does not address our overfitting problem and certain features may still dominate the prediction.

In Section 3.2, we propose an ER-specific adaptive regularization strategy that regularizes each feature differently, depending on the overlap between the M- and U-Distribution in each feature. Our proposed regularization strategy ensures that every feature can contribute to making predictions.

(3) Incorporating Transitivity. ER exhibits the *transitivity* property: if both (t_1, t_2) and (t_1, t_3) are matches, then (t_2, t_3) is also a match. Intuitively, incorporating transitivity into the generative model should increase its performance, since it provides additional information.

Example 1.3. In Figure 1, both (fd1,zg2) and (fd3, zg2) have high similarity, so a model tends to classify both as matches. In the ground truth, (fd1,zg2) is indeed a true match, while (fd3, zg2) is a false positive. We can see that fd1 and zg2 are the same cafe in a hotel, while fd3 is the dining room in that hotel. Without such domain knowledge, it is very difficult for a model to recognize that (fd3, zg2) is not a match.

However, this false positive can be detected using transitivity. Classifying (fd1,zg2) and (fd3, zg2) as matches will make (fd1, fd3) a match too through transitivity. If we know the fodors table is duplicate free (as we show later, we can leverage transitivity without assuming either input table to be duplicate free), then (fd1, fd3) cannot be a match. This means that one of (fd1,zg2) and (fd3, zg2) has to be a false positive. Since (fd1,zg2) is more similar than (fd3, zg2), it will be easy for a model to conclude that (fd3, zg2) is not a match.

The supervised approaches make predictions on each unlabeled tuple pair independently and can only address transitivity violations as a postprocessing step. In contrast, we incorporate transitivity in the model learning process, which gives a significant boost in model performance. Intuitively, transitivity serves as constraints on the labeling possibilities of all tuple pairs. Solving the "constrained optimization" problem is highly non-trivial as it is a non-convex problem with quadratic number of constraints with respect to the number of tuple pairs.

In Section 4, we introduce a novel way of incorporating transitivity in the generative model, and propose optimizations to solve it with minimal additional computation cost.

Contributions. We propose ZeroER for entity resolution using zero labeled examples with the following features:

(1) Unsupervised: ZeroER requires minimal human input. Users only need to provide two input tables L and R, similarity measures used to create the feature matrix, and optionally a blocking function — ZeroER is then able to determine matches and non-matches automatically with zero labeled training data.

(2) High Quality: As ZeroER includes several innovations that exploit ER-specific properties, it produces high-quality results across several distinct domains. Specifically, we show that ZeroER i) greatly outperforms unsupervised baseline methods; and ii) achieves comparable results against supervised and active learning methods, but saves substantial human labeling efforts.

(3) Efficient: We design an efficient iterative expectation-maximization based algorithm for learning the parameters of M- and U-distributions. We ensure that materializing our insights, including feature regularization and incorporating transitivity, do not come at the expense of computational performance.

2 PRELIMINARIES

We formally define entity resolution in Section 2.1 and describe the generative modelling idea for ER and how to train a generative model in Section 2.2.

2.1 Entity Resolution Problem Definition

An entity is a distinct real-world object such as a customer, an organization, a publication etc. We are given two relations T and T' with attributes $\{A_1, A_2, \ldots, A_m\}$. The entity resolution (ER) problem [29] seeks to identify all pairs of tuples (t, t'), where $t \in T, t' \in T'$, that refer to same real-world entity. A pair of tuples is said to be a *match* (denoted as M) (resp. *unmatch* (denoted as U)) when they refer to the same (resp. different) real-world entity. When $T = T'$, this is also known as *data deduplication*; and when $T \neq T'$, this is also known as *record linkage*. Figure 1 provides an illustration where we wish to identify which pairs of records from Fodors and Zagat refer to the same restaurants.

To reduce the quadratic number of tuple pairs an ER problem has $(|T| \times |T'|)$, a *blocking function* $\mathcal{B}$ typically is used to filter out those tuple pairs that are highly unlikely to be matches (e.g., tuple pairs with different values in the state attribute will be not be compared), and obtain a candidate set $C_s \subseteq T \times T'$ that is usually much smaller. Designing an effective blocking strategy is an orthogonal research problem, and many prior work has been dedicated to this [47, 48, 56]. The blocking function $\mathcal{B}$ is an optional input to our system, and $C_s = T \times T'$ if no blocking is provided.

To determine the M or U label for every tuple pair, one needs to generate similarity vectors (aka feature vectors) for every tuple pair in C_s by applying a set of similarity functions $\mathcal{F}$. Given an ER task, typically, $\mathcal{F}$ is either provided by domain experts, or generated according to some default feature engineering strategy implemented in ER packages. ZeroER can work with any $\mathcal{F}$. Figure 1(c) shows a sample of features generated by the open-source Magellan package[39].

Formal Problem Definition. Given two relations T and T', a set of similarity functions $\mathcal{F}$ for this dataset, and an optional blocking function $\mathcal{B}$, the goal of ZeroER is to assign a binary label $y \in \{M, U\}$ for every pair in $T \times T'$ without any labeled data.

We apply $\mathcal{B}$ (if available) to obtain a set of N tuple pairs $C_s \subseteq T \times T'$, and then apply $\mathcal{F}$ to obtain a feature vector $\mathbf{x} = (x^1, x^2, \ldots, x^d)$ for every tuple pair in C_s. We use y to denote the unknown label for every feature vector $\mathbf{x}$.

2.2 Generative Modelling for ER

As discussed, we propose to solve the ER problem using generative modelling in the absence of labeled examples. Specifically, we consider the features and label $(\mathbf{x}, y)$ of a tuple pair to be generated as follows:

(1) Generate the value for y (M or U) based on the Bernoulli distribution parameterized by π_M. Intuitively, the process tosses a coin that comes heads with probability π_M. If it comes heads, it chooses $y = M$, else $y = U$.
(2) Generate the feature vector $\mathbf{x}$ from the selected distribution. For example, if $y = M$, then $\mathbf{x}$ is sampled according to the M-Distribution $p(\mathbf{x}|y = M)$, parameterized by θ_M; else $\mathbf{x}$ is sampled according to the U-Distribution $p(\mathbf{x}|y = U)$, parameterized by θ_U.

The generative model is governed by its set of parameters $\Theta = \{\theta_M, \theta_U, \pi_M\}$. If we can learn Θ, then we can classify any tuple pair based on its similarity vector $\mathbf{x}$. Specifically, the classification problem is to determine whether $p(y = M|\mathbf{x}) > 0.5$. In supervised learning, a functional mapping between the feature space and label space is learned using labeled examples, which directly gives $p(y = M|\mathbf{x})$. In ZeroER, instead of learning $p(y = M|\mathbf{x})$ directly, we aim to learn the prior probability $p(y = M)$, the M-Distribution $p(\mathbf{x}|y = M)$, and the U-Distribution $p(\mathbf{x}|y = U)$, which we can use to compute $p(y = M|\mathbf{x})$ via Equation (1).

Learning Parameters of Generative Model. Given $N = |C_s|$ tuple pairs $(\mathbf{x}_i, y_i)$, $i \in [1, N]$, where $\mathbf{x}_i$ is observed and y_i is unknown, a common way to estimate Θ is by maximizing the *log data likelihood function* (the log is taken for computational convenience):

$$\begin{aligned} L(\Theta) &= \log \Pi_{i=1}^{N} p(\mathbf{x}_i, y_i|\Theta) = \Sigma_{i=1}^{N} \log \left(p(y_i|\Theta) p(\mathbf{x}_i|\Theta, y_i)\right) \\ &= \Sigma_{i=1}^{N} \Sigma_{C \in \{M,U\}} \mathbb{1}_{y_i = C} \log \left(\pi_C p(\mathbf{x}_i|\theta_C)\right) \end{aligned} \tag{2}$$

where $\mathbb{1}_.$ is the identity function that evaluates to 1 if the condition is true and 0 otherwise; and $p(\mathbf{x}_i|\theta_C)$ is short for $p(\mathbf{x}_i|y_i = C)$ parameterized by θ_C.

Since y_i is unknown, Equation (2) cannot be optimized directly. The EM algorithm is the canonical algorithm to use in the case of unobserved variables [24]. Each iteration of the EM algorithm consists of two steps: an Expectation (E)-step and a Maximization (M)-step. Intuitively, the E-step determines what is the (soft) class assignment y_i for every tuple pair based on the parameter estimates from last iteration Θ^{t-1}. In other words, E-step computes the posterior probability $\gamma_{i,M} = p(y_i = M|\mathbf{x}_i, \Theta^{t-1})$. The M-step takes the new soft class assignments and re-estimates all parameters Θ^t. Formally, the two steps are as follows:

(1) **E Step.** Given the parameter estimates from the previous iteration Θ^{t-1}, compute the posterior probabilities:

$$\begin{aligned} \gamma_{i,M}^{t} &= p(y_i = M|\mathbf{x}_i, \Theta^{t-1}) \\ &= \frac{\pi_M^{t-1} \times p(\mathbf{x}_i|\theta_M^{t-1})}{\pi_M^{t-1} \times p(\mathbf{x}_i|\theta_M^{t-1}) + (1 - \pi_M^{t-1}) \times p(\mathbf{x}_i|\theta_U^{t-1})} \end{aligned} \tag{3}$$

(2) **M Step.**Given the new soft class assignments as defined by $\gamma_{i,M}^{t}$, re-estimate Θ by maximizing the following expected log likelihood function:

$$\mathbb{E}\{L(\Theta)\} = \Sigma_{i=1}^{N} \Sigma_{C \in \{M,U\}} \gamma_{i,C}^{t} \log \left(\pi_C p(\mathbf{x}_i|\theta_C)\right) \tag{4}$$

In other words, $\Theta^t = \arg\max_{\Theta} \mathbb{E}\{L(\Theta)\}$.

GMM for ER. The generative model becomes the Gaussian Mixture Model (GMM) with two mixture components when choose the M and U distribution ($p(\mathbf{x}|y = M)$ and $p(\mathbf{x}|y = U)$) to be Gaussian. Gaussian distribution often serves as an effective default distribution for numerical random variables whose distributions are not known [42]. Furthermore, there exist closed-form solutions for estimating the parameters of a Gaussian distribution given a set of data points. Under GMM, the parameters for M-distribution and U-distribution are $\theta_C = \{\mu_C, \Sigma_C\}$, where $C \in \{M, U\}$, μ_C is the mean vector, and Σ_C is the covariance matrix.

The EM algorithm can now be instantiated with the Gaussian PDF $p(\mathbf{x}_i|\theta_C) = \frac{1}{(2\pi)^{d/2}} \frac{1}{\det(\Sigma_C)^{1/2}} \exp\{-\frac{1}{2}(\mathbf{x}_i - \mu_C)^T \Sigma_C^{-1} (\mathbf{x}_i - \mu_C)\}$. The E-step can be directly calculated. The M-step has a closed-form solution [13] as follows:

$$\begin{aligned} N_C &= \Sigma_{i=1}^{N} \gamma_{i,C};\ \pi_C = N_C/N;\ \mu_C = \bar{\mathbf{x}}_C = 1/N_C \Sigma_{i=1}^{N} \gamma_{i,C} \mathbf{x}_i \\ \Sigma_C &= S_C = 1/N_C \Sigma_{i=1}^{N} \gamma_{i,C} (\mathbf{x}_i - \bar{\mathbf{x}}_C)(\mathbf{x}_i - \bar{\mathbf{x}}_C)^T \end{aligned} \tag{5}$$

where $C \in \{M, U\}$. In other words, μ_C and Σ_C are assigned to be the weighted sample mean $\bar{\mathbf{x}}_C$ and weighted sample covariance S_C respectively in the M-step.

3 THE ZEROER GENERATIVE MODEL

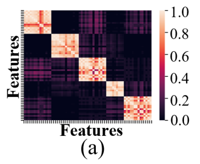
GMM as the generative model for ER described in Section 2 is a promising start. However, as we shall show, blindly applying it for ER produces sub-optimal results. The key issue is that GMM is not adapted to handle properties unique to ER, which we address next.

3.1 Data Deficiency

In a typical ER dataset, the number of matches is dramatically less than the number of unmatches by many orders of magnitude. This impacts the accuracy of parameter estimation for the generative model. Specifically, the number of parameters to be learned for the covariance matrix Σ_M is $\binom{d}{2}$ under GMM. Recall from Section 1 that for many datasets such as Fodors-Zagat, the value of $\binom{d}{2}$ can even be higher than the number of matches, which would lead to an inferior estimate of Σ_M. Specifically, this would cause a large systematic distortion on the eigenstructure of Σ_M [23, 54], and hence affecting the accuracy of the M-Distribution.

The existing approaches. Existing approaches for tackling data deficiency makes simplifying assumptions to reduce the number of parameters. As provided in the well-known sklearn package [5], the two most common approaches are (1) assuming feature independence,i.e. Σ_M(and also Σ_U) is diagonal (non-diagonal entries are 0). (2) assuming the covariance matrix can be shared across all components, i.e. $\Sigma_M = \Sigma_U$. While these two approaches reduce the number of parameters significantly, as we shall show next and also in experiments, they are unsuitable for ER. We propose two ER-specific ways to reduce the number of parameters next.

Feature grouping. Naively assuming feature independence ignores the ER feature engineering process where multiple features may be created for each aligned attribute. Clearly, each of these features are dependent on each other. Consider Figure 4(a) that shows the heat map of the correlation matrix of all features for ground-truth matches from the Fodors Zagats dataset (the heat map for unmatches is similar). We can clearly see that some blocks have higher values while other values are closer to 0. Not surprisingly, each block corresponds to the set of features generated by the same attribute. This suggests a natural modification to naive GMM: (1) *features generated from the same attribute are dependent*; and (2) *features generated by different attributes are independent.*



(a)

$$\Sigma_C = \begin{bmatrix} \Sigma_C^1 & 0 & \dots & 0 \\ 0 & \Sigma_C^2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \Sigma_C^m \end{bmatrix}$$

(b)

Figure 4: (a) Heat map of correlation between features using ground-truth matches in the fozors-zagats dataset.(b) Covariance matrix after feature grouping.

Conceptually, the covariance matrix Σ_C is now a block diagonal matrix where each block corresponds to the covariance matrix of the features obtained from the same attribute, as shown in Figure 4(b), where $C \in \{M, U\}$ and Σ_C^g is the covariance matrix for features of the g-th attribute. This results in a dramatically reduced number of parameters.

Correlation sharing. The traditional approach for parameter sharing in GMM is to assume that covariance values are the same for both M and U distributions,i.e. $\Sigma_M = \Sigma_U$. To empirically verify if this can be used in ER, for every dataset, we calculate the sample covariance matrix for matches S_M and unmatches S_U using the ground-truth, and we calculate the cosine similarity between S_M and S_U by flattening both matrices to vectors. If $\Sigma_M = \Sigma_U$, one would expect cosine(S_M,S_U) is close to 1. As shown in the first row in Table 1, cosine(S_M,S_U) deviates a lot from 1, which suggests that $\Sigma_M = \Sigma_U$ is not a valid assumption for ER.

	FZ	DA	DS	AB	AG
cosine(S_M,S_U)	0.76	0.69	0.74	0.92	0.73
cosine(R_M,R_U)	0.97	0.94	0.98	0.99	0.99

Table 1: Cosine similarity between covariance matrix (first row) and correlation matrix (second row) using ground-truth matches and unmatches after feature grouping on five real-world datasets.

Instead of naively sharing the entire covariance matrix (like prior approaches), we decompose the covariance matrix and then share only the part that is consistent over M and U. Specifically, we decompose Σ_C where $C \in \{M, U\}$ based on the well-known relationship between covariance and Pearson correlation of two random variables A and B: correlation$(A, B) = \frac{\text{covariance}(A,B)}{\sigma_A \sigma_B}$, where σ_A and σ_B are the standard deviation of A and B, respectively. Correlation can be seen as a normalized measure of covariance – while covariance$(A, B) \in [-\infty, \infty]$, correlation$(A, B) \in [-1, 1]$. We decompose Σ_C ($C \in \{M, U\}$) as follows:

$$\Sigma_C = \Lambda_C R_C \Lambda_C \qquad (6)$$

where Λ_C is a diagonal matrix with $\Lambda_C[j, j]$ represents the standard deviation of feature j in class C, and R_C is the Pearson correlation matrix of pairs of features in class C.

Our key observation is that, in the ER context, *the Pearson correlation matrices R_M and R_U should be very similar.* This is because features in a same group are obtained by applying different similarity functions on the same attribute, so the Pearson correlation between two features reflects the correlation of their corresponding similarity functions on the same attribute, regardless of matches or unmatches.

To verify our intuition, we compute the cosine similarity between R_M and R_U using ground-truth matches and unmatches. As shown in the second row in Table 1, *cosine*(R_M, R_U) is close to 1 consistently across all datasets. This ER specific property allows to rewrite Equation (6) as:

$$\Sigma_M = \Lambda_M R \Lambda_M \text{ and } \Sigma_U = \Lambda_U R \Lambda_U \qquad (7)$$

where R is the shared correlation matrix. Since R is the same for both classes, we can estimate it using the entire dataset in a preprocessing step and completely eliminate it from the set of parameters Θ.

Parameters in the ZeroER generative model. After feature grouping and correlation sharing, the generative model for ZeroER is parameterized by $\Theta = \{\pi_M, \Lambda_M, \Lambda_U, \mu_M, \mu_U\}$, where Λ_M (resp. Λ_U) is a diagonal matrix and $\Lambda_M[j, j]$ (resp. $\Lambda_U[j, j]$) is the standard deviation of the jth feature of class M (resp. class U). Therefore, we have a total number of $4d + 1$ parameters, a significant reduction from $d^2 + 3d + 1$ parameters of the naive GMM.

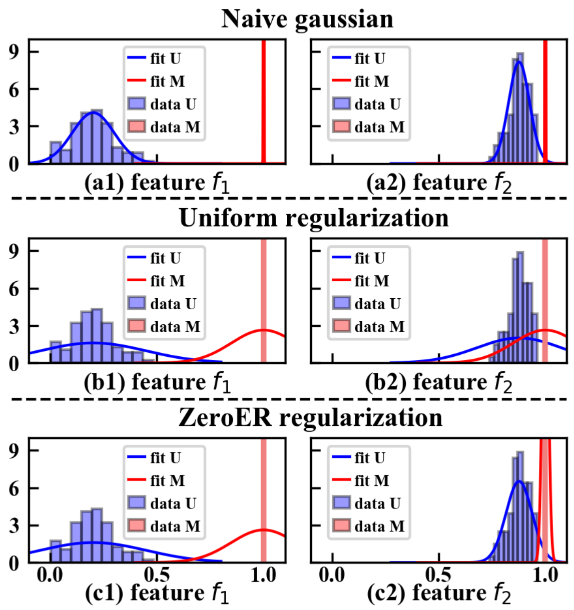


Figure 5: Two features f_1 and f_2 suffer from overfitting problem shown in (a1) and (a2). An uniform regularization works for f_1 (b1), but does not work for f_2 (b2). The adaptive regularization by ZeroER works for both (c1 and c2).

3.2 Feature Regularization

Another unique challenge of ER is that many features can have extremely small variance and can dominate the computation, effectively rendering other features to have little impact on the predictions. In the extreme case, as shown in Example 1.2, a feature with 0 variance in any component $C \in \{M, U\}$ will make $\det(\Sigma_C) = 0$ (determinant of the covariance matrix), and that feature will solely determine the prediction, known as the singularity problem [13].

Existing Approach: Uniform Regularization. The traditional approach for addressing the singularity problem is to add a small constant κ to the diagonal entries of Σ_C, i.e., the variance of every feature. This is known formally as the uniform/Tikhonov regularization [27, 35], in which each feature is regularized uniformly. This approach is widely used including in the popular sklearn package [5]. While this uniform regularization avoids the singularity problem, i.e., having one feature dictating the predictions, it still does not address the overfitting problem, i.e. some features have much more influence than others. In addition, a constant κ that works for one feature might be unsuitable for another.

Example 3.1. Consider two features f_1 in Figure 5(a1) and f_2 in Figure 5(a2) that suffer from the overfitting problem. A κ is chosen to appropriately regularize f_1 is shown in Figure 5(b1). The two distributions are now well separated, do not have extremely small variances and f_1 could be used for making predictions. However, applying the same κ to f_2 results in Figure 5(b2), which is clearly an inferior fit. There is too much overlap between the M-distribution and the U-distribution after regularization. This undermines the model's ability to distinguish matches and unmatches.

Our Solution: Adaptive Regularization. Before giving our solution, we first examine how uniform regularization works in principle. Regularization can be seen as adding an "punishing" term to the objective function. Formally, the uniform regularization modifies the expected data likelihood objective function in the M-Step (c.f. Equation (4)) as:

$$\mathbb{E}'\{L(\Theta)\} = \mathbb{E}\{L(\Theta)\} - 1/2 \times \mathrm{tr}(K(N_M\Sigma_M^{-1} + N_U\Sigma_U^{-1})) \quad (8)$$

where $K = \kappa I$ is a diagonal matrix acting as regularization parameter; tr() denotes the trace operation on a matrix – the sum of all diagonal elements of the matrix; N_M and N_U ensure the regularization term is "normalized" to the amount of data N. Without it, the first term will dominate when N is large. Intuitively, this regularization term punishes the variances of M and U being small.

To maximize Equation (8), the optimal values of μ_C and π_C are still given by Equation (5) but the optimal Σ_C is now:

$$\Sigma_C = S_C + K \text{ where } C \in \{M, U\} \quad (9)$$

In other words, designing a regularization strategy is essentially about designing the diagonal regularization matrix K. In uniform regularization, K is chosen to be $K = \kappa I$, so Equation (9) adds a number κ on every diagonal entry of the covariance matrix. As shown in Example 3.1, using a too large κ for feature f_2 causes much overlap of the M and U distributions, rendering f_2 ineffective in making predictions.

A good regularization needs to meet two goals: (1) the relative influences of dominating features are decreased, so that all features can have a chance to contribute to making predictions; and (2) the more influential features before regularization should still be more influential after regularization, so as not to decrease the model's predictive capabilities. Intuitively, the influence of a feature can be measured by the overlap between its M-Distribution and U-Distribution — the smaller the overlap, the more influential a feature is. We propose to design a diagonal regularization matrix $K = diag(\kappa_1, \kappa_2, \ldots, \kappa_d)$ such that all features have the same amount of *overlap increase*, which would achieve the above two goals. To see this, consider a hypothetical scenario where features f_3 and f_4 have overlaps of 0.001 and 0.3, respectively, before regularization, so f_3 is dominating f_4 in making predictions. After we add the same overlap, say 0.2, f_3 and f_4 now have overlaps of 0.201 and 0.5, respectively. Clearly, f_3 is still more influential than f_4, but the relative influences of f_3 is decreased significantly (e.g., $\frac{0.3}{0.001} \gg \frac{0.5}{0.201}$).

Specifically, we use Bhattacharyya coefficient (BC) to measure the overlap, as it is a widely accepted measure of the overlap of two distribution and offers a tight upper-bound to

Bayes error [11], the error rate of a optimal classifier on examples generated by the two distributions. The Bhattacharyya coefficient of the M and U distribution on the jth feature dimension is as follows [3]:

$$BC_j = \exp\Bigg[-\frac{1}{4}\ln\big(\frac{1}{4}(\frac{\Sigma_M[j,j]}{\Sigma_U[j,j]} + \frac{\Sigma_U[j,j]}{\Sigma_M[j,j]} + 2)\big) - \frac{1}{4}\frac{(\mu_M[j] - \mu_U[j])^2}{\Sigma_M[j,j] + \Sigma_U[j,j]}\Bigg] \tag{10}$$

Let BC_j denote the BC coefficient of feature j before regularization, and BC'_j denote the BC coefficient of feature j after regularization, i.e. after adding κ_j to the variance of feature j ($\Sigma_C[j,j], C \in \{M, U\}$). For every feature j, we would like to design κ_j such that $BC'_j - BC_j$ is the same for all features. Let $k' \in [0,1]$ be the regularization parameter that denotes the constant amount of increase of BC in all features. Then we can compute k_j by solving the following equation:

$$BC'_j - BC_j = \kappa', \forall j \in [1, d] \tag{11}$$

Equation (11) has only one unknown variable k_j and can be solved efficiently with the newton raphson method [55].

Example 3.2. Continuing with the previous example on regularizing f_1 and f_2. Using our adaptive regularization strategy, f_1 is fitted in Figure 5(c1) and f_2 is fitted in Figure 5(c2). As we can see, both features are now well separated and well spread out to avoid the overfitting problem.

3.3 ZeroER Algorithm So Far

We now present the algorithm of ZeroER without transitivity in Algorithm 1. As we can see, the updates to the parameters Θ happen in the M-Step (Lines 8 to 12), and the E-Step re-estimates the posterior probability of every tuple pair (Line 14). EM algorithm requires an initialization of posterior probabilities $\gamma_{i,C}$ (Line 4). To do this, we first use a min-max scaler to normalize every feature into [0, 1]. We set $\gamma_{i,M} = 1$ (hence $\gamma_{i,U} = 0$) if $||\mathbf{x}_i|| > \epsilon$; otherwise, $\gamma_{i,M} = 0$. We choose 0.5 as the default value for ϵ, and show experimentally that ZeroER is robust to the choice of ϵ.

The EM algorithm also requires a termination condition (Line 6). The EM iteration is terminated when the difference of Equation (4) between two consecutive iterations is less than a threshold (10^{-5}). We also set a limit the number of EM iterations to 200, a common practice also used in popular ML packages such as sklearn [5]. When EM is terminated due to the limit of the maximum number of iterations (instead of likelihood convergence), we average the likelihood results from the latest 20 iterations.

4 INCORPORATING TRANSITIVITY

Transitivity constraint stipulates that if tuple pairs (t_1, t_2) and (t_1, t_3) are matches, then (t_2, t_3) *must* be a match [31]. While one can enforce transitivity as a postprocessing step, incorporating transitivity into the ER algorithm achieves better performance, as we will verify empirically.

Algorithm 1: ZeroER without transitivity

```
   Input: two tables $T$ and $T'$, a set of similarity functions
          $\mathcal{F}$, an optional blocking function $\mathcal{B}$
   Output: matching pairs in $T \times T'$
1  $C_s \leftarrow$ all pairs in $T \times T'$ after applying $\mathcal{B}$
2  similarity vector $\mathbf{x} \leftarrow$ apply $\mathcal{F}$ on every pair in $C_s$
3  $R \leftarrow$ the correlation matrix for feature pairs using all
     similarity vectors $\mathbf{x}'s$.
4  Initialize $\gamma_{i,C}$ ($C \in \{M, U\}$) for all tuple pairs.
5  Create variables of ZeroER's generative model
     $\Theta = \{\pi_M, \Lambda_M, \Lambda_U, \mu_M, \mu_U\}$ that will be updated in the
     EM loops.
6  while Not Converged do
7      M Step
8          Update $\mu_M$,$\mu_U$,$\pi_M$ by Equation (5)
9          $\Lambda_C[j,j] = \sqrt{\frac{1}{N_C}\sum_{i=1}^{N}\gamma_{i,C}(\mathbf{x}_i[j] - \bar{\mathbf{x}}_C[j])^2}$
10         $\Sigma_C \leftarrow \Lambda_C R \Lambda_C$
11         $K \leftarrow \mathrm{diag}\{\kappa_1, \dots, \kappa_d\}$, where $\kappa_j$ is obtained by
             solving Equation (11).
12         $\Sigma_C \leftarrow \Sigma_C + K$
13     E Step
14         Update $\gamma_{i,M}$ by Equation (3).
15 end
16 return all pairs such that $\gamma_{i,M} > 0.5$
```

4.1 Transitivity as Posterior Constraint

Let $\gamma_{i,j,M}$ denote the posterior probability of a tuple pair (t_i, t_j) being a match, namely $\gamma_{i,j,M} = P(y_{i,j} = M|\mathbf{x}_{i,j})$. Note that when we write $\gamma_{i,j,M}$, the two subscripts i, j are used to denote a tuple pair (t_i, t_j); and when we write $\gamma_{i,M}$ (as used in previous sections), the one subscript i is used to denote the i^{th} tuple pair.

Consider three tuple pairs (t_1, t_2), (t_1, t_3), and (t_2, t_3). Transitivity property [31] states that if both (t_1, t_2) and (t_1, t_3) are matches, then (t_2, t_3) must be match. However, it is possible that (t_2, t_3) is a match, but not both of (t_1, t_2) and (t_1, t_3). We capture this observation as a probabilistic inequality that needs to be satisfied:

$$\gamma_{1,2,M} \times \gamma_{1,3,M} \leq \gamma_{2,3,M} \tag{12}$$

Equation (12) secures the transitivity property. Consider a deterministic scenario that (t_1, t_2) and (t_1, t_3) are known to be matches so that $\gamma_{1,2,M} = \gamma_{1,3,M} = 1$. By Equation (12) we have $\gamma_{2,3,M} \geq 1$, so (t_2, t_3) must also be a match. Now consider a probabilistic scenario that $\gamma_{1,2,M} = 0.7$ and $\gamma_{1,3,M} = 0.6$, which means in 70% of the cases (t_1, t_2) is a match and in 60% of the cases (t_1, t_3) is a match. Transitivity occurs when (t_1, t_2) and (t_1, t_3) are both matches which is $70\% \times 60\% = 42\%$ of the cases. Therefore, (t_2, t_3) has at least 42% chances of being a match due to transitivity, as captured by Equation (12).

Let $\boldsymbol{\gamma}$ denote the vector of posterior probabilities of being a match for all tuple pairs in $T \times T'$. For each trio of tuples t_i, t_j, t_k, we can define the probabilistic constraint from Equation (12). The set of all such constraints define a feasibility set for $\boldsymbol{\gamma}$:

$$Q = \{\boldsymbol{\gamma} | \gamma_{i,j,M}\gamma_{i,k,M} - \gamma_{j,k,M} \leq 0 \ \forall i,j,k\} \tag{13}$$

Recall from Section 2.2, the objective function of the generative model is data likelihood function $L(\Theta)$ parameterized by Θ, which is maximized via an EM algorithm. The posterior probabilities $\boldsymbol{\gamma}$ are computed based on Θ via Bayes rule (c.f. Equation (1)). On the surface, it seems very difficult to incorporate Q in the learning process, namely, finding the best Θ that maximizes $L(\Theta)$, while ensuring the posterior probabilities $\boldsymbol{\gamma}$ computed based on Θ satisfies Q.

Incorporating Posterior Constraints in EM. Fortunately, this conundrum could be solved by using the *free energy* view of the EM algorithm. In this view, the EM algorithm is seen to optimize a different objective function, the free energy function, that treats both Θ and $\boldsymbol{\gamma}$ as its parameters. Previously, free energy based approach was proposed to develop the incremental and sparse variants of the EM algorithm [46]. We adopt this approach as it allows us incorporate the transitivity constraints directly into the optimization process. The free energy function $F(\Theta, \boldsymbol{\gamma})$ [38, 46] is defined as:

$$F(\Theta, \boldsymbol{\gamma}) = \Sigma_{i=1}^{n} \Sigma_{C \in \{M,U\}} \left[\gamma_{i,C} \log\left(\pi_C p(\mathbf{x}_i|\theta_C)/\gamma_{i,C}\right)\right] \tag{14}$$

It turns out $F(\Theta, \boldsymbol{\gamma})$ can be optimized the exact same way as $L(\Theta)$ using the same EM algorithm, with a different view of the E-Step and the M-Step [46]. Specifically, the E-Step is now seen as maximizing $F(\Theta, \boldsymbol{\gamma})$ with respect to $\boldsymbol{\gamma}$ given Θ^{t-1} estimated from the previous iteration:

$$\boldsymbol{\gamma}^t = \arg\max_{\boldsymbol{\gamma}} F(\Theta^{t-1}, \boldsymbol{\gamma}) \tag{15}$$

and the M-Step is now seen as maximizing $F(\Theta, \boldsymbol{\gamma})$ with respect to Θ given $\boldsymbol{\gamma}^{t-1}$ estimated from the previous iteration:

$$\Theta^t = \arg\max_{\Theta} F(\Theta, \boldsymbol{\gamma}^t) \tag{16}$$

Once we adopt this free energy view of EM, the constraints in Equation (13) can be incorporated by adapting the E-step as a constrained optimization problem, namely:

$$\boldsymbol{\gamma}^t = \arg\max_{\boldsymbol{\gamma} \in Q} F(\Theta^{t-1}, \boldsymbol{\gamma}) \tag{17}$$

Lemma 4.1. *The constrained maximization problem in Equation* (17) *is a non-convex optimization problem.*

We leave the full proof of Lemma 4.1 to the full technical report [7]. Typically, heuristic methods such as projected gradient descent and alternating minimization are used to obtain an approximate solution for the non-convex optimization [36, 53]. Unfortunately, these methods involve an additional expensive iterative process at every E-step drastically increasing the overall running time. We introduce a much more efficient solution next.

4.2 Solving the Constrained Problem

Let $\boldsymbol{\gamma}^*$ be the optimal solution for the unconstrained optimization problem in Equation (15), which is given by Equation (3). Let $\boldsymbol{\gamma}^{**}$ be the unknown optimal solution for the constrained optimization problem in Equation (17). If $\boldsymbol{\gamma}^*$ already satisfies Q, then obviously $\boldsymbol{\gamma}^{**} = \boldsymbol{\gamma}^*$; otherwise, we need an efficient way of computing $\boldsymbol{\gamma}^{**}$. Though the optimization problem in Equation (17) is non-convex (due to non-convex constraints Q), the objective function $F(\Theta, \boldsymbol{\gamma})$ is convex with respect to $\boldsymbol{\gamma}$. Therefore, we know that the solution $\boldsymbol{\gamma}^{**}$ has to be on the boundary of Q [14]. Figure 6 shows different solutions in 2D, where $\boldsymbol{\gamma}^*$ is outside of Q and $\boldsymbol{\gamma}^{**}$ is on Q.

As discussed, finding the exact $\boldsymbol{\gamma}^{**}$ is expensive. However, it is possible to *approximate* $\boldsymbol{\gamma}^{**}$ by $\boldsymbol{\gamma}'$ on the boundary of Q that can be found much more efficiently. Our intuition is that the closer the approximate solution $\boldsymbol{\gamma}'$ is to the known optimal solution $\boldsymbol{\gamma}^*$ of the unconstrained problem, the better the approximation is.

The operation of finding $\boldsymbol{\gamma}'$ based on its distance with $\boldsymbol{\gamma}^*$ is known as projection [36]. Since there are many different ways to measure the "closeness" between $\boldsymbol{\gamma}'$ and $\boldsymbol{\gamma}^*$, there are many different ways of doing the projection. We propose an efficient way of projection that requires minimum changes on $\boldsymbol{\gamma}^*$, and at the same time, tries to maximize $F(\Theta, \boldsymbol{\gamma})$.

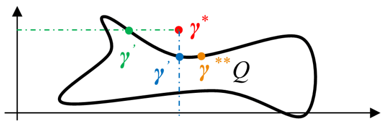


Figure 6: Illustrations of different solutions to the constrained maximization problem in 2-D space.

Handling one constraint. Consider a single constraint in Q: $\gamma_{i,j,M} \times \gamma_{i,k,M} \leq \gamma_{j,k,M}$. When the inequality is violated, one can adjust one of the three probabilities to make it hold: decrease $\gamma_{j,k,M}$, increase $\gamma_{i,j,M}$ or increase $\gamma_{i,k,M}$ so that $\gamma_{i,j,M} \times \gamma_{i,k,M} = \gamma_{j,k,M}$. In Figure 6, these adjustments correspond to projecting $\boldsymbol{\gamma}^*$ on the boundary of Q along the axises to be the green or blue points. Formally, the three ways of projection along the three axis are:

$$\gamma'_{i,j,M} = \frac{\gamma_{j,k,M}}{\gamma_{i,k,M}}, \ \gamma'_{i,k,M} = \frac{\gamma_{j,k,M}}{\gamma_{i,j,M}}, \ \gamma'_{j,k,M} = \gamma_{i,j,M}\gamma_{i,k,M} \tag{18}$$

Therefore, given the optimal solution $\boldsymbol{\gamma}^*$ to Equation (15) that does not satisfy the one constraint in Q, we obtain three possible approximate solutions by updating each one of the three posterior probabilities as in Equation (18), and we pick the one that has the maximal $F(\Theta, \boldsymbol{\gamma})$.

Handling multiple constraints. Now we consider multiple constraints $Q = \{g_1, g_2, \ldots\}$. A natural approach is to do the projection in Equation (18) for each of the constraints. However, a probability $\gamma_{i,j,M}$ might get involved in several constraints. For example, $\gamma_{i,j,M}$ is involved in two constraints $\{g_1, g_2\}$, specifically the following two cases can happen: 1) The projection by Equation (18) in constraint g_1 has increased/decreased $\gamma_{i,j,M}$ to $\gamma'_{i,j,M}$, and the projection in the current constraint g_2 requires $\gamma'_{i,j,M}$ to be decreased/increased, which would make g_1 unsatisfied. In this case, we adopt a greedy approach: When the best projection axis in Equation (18) causes conflicts with a previous constraint, we project along the second best axis. If it causes conflicts again we project along the third axis and if conflicts happen again we perform no projection on this constraint. 2) Constraint g_1 increased/decreased $\gamma_{i,j,M}$ to $\gamma'_{i,j,M}$. After adjusting $\gamma_{i,j,M}$ to be $\gamma'_{i,j,M}$, constraint g_2 also becomes satisfied. Or, constraint g_2 requires $\gamma'_{i,j,M}$ to be increased/decreased further. In this case, we just increase/decrease $\gamma'_{i,j,M}$ further.

Furthermore, there are two cases when a probability $\gamma_{i,j,M}$ involved in the constraints doesn't exist in the posterior vector $\boldsymbol{\gamma}^*$. First, the corresponding tuple pair (t_i, t_j) was excluded by blocking; in this case, we assume $\gamma_{i,j,M} = 0$. Second, $\gamma_{i,j,M}$ is not in $\boldsymbol{\gamma}^*$ when t_i and t_j are from the same table, as C_s only contains cross-table tuples pairs. This is not a problem when $T = T'$ because (t_i, t_j) is a cross-table tuple pair at the same time. We handle the case when $T \neq T'$ in Section 4.3.

Reducing the number of constraints. The efficiency of our approach is reliant on the number of constraints that are checked in each iteration, which is cubic w.r.t. the number of tuples. However, it is possible to prune most of these constraints. Specifically, transitivity property is only useful for tuple pairs that are predicted to be matches. Since the number of matches are orders of magnitude smaller than unmatches, the number of constraints also reduce significantly. Specifically, we only need to consider the following reduced set of constraints:

$$Q' = \{\boldsymbol{\gamma} | \gamma_{i,j,M}\gamma_{i,k,M} \leq \gamma_{j,k,M}, \text{where } \gamma_{i,j,M} \geq 0.5, \gamma_{i,k,M} \geq 0.5\} \tag{19}$$

4.3 Transitivity Across Two Tables

Incorporating transitivity in the general scenario when $T \neq T'$ is more involved. In Equation (12), the three involved tuple pairs are (t_1, t_2), (t_1, t_3), and (t_2, t_3); two of the three tuples t_1, t_2, t_3 must come from the same table, and the third tuple must come from the other table. Without loss of generality, let us assume t_2, t_3 are from the same table. Using one generative model for $T \times T'$ discussed before only gives the values of $\gamma_{1,2,M}$ and $\gamma_{1,3,M}$ but not $\gamma_{2,3,M}$. If we know both T and T' are duplicate-free, we can set $\gamma_{2,3,M} = 0$; however, we make no such assumption in this work.

One simple way to be able to compute the posterior probabilities of tuple pairs in the same table is to use the same generative model parameterized by Θ, and include in the data likelihood function $L(\Theta)$ (Equation (2)) not only the N tuple pairs in $T \times T'$, but also the N_l tuple pairs in $T \times T$ and the N_r tuple pairs in $T' \times T'$. However, this approach produces inferior results primarily because the distribution of similarities vectors for tuple pairs in $T \times T'$ is quite different from those in $T \times T$ or in $T' \times T'$. For example, as shown in Figure 1, the phone area code is separated by "/" in the fodors table while by "-" in the zagats table, and city name "los angeles" is written with full name in fodors while in zagats it is abbreviated as "la".

Given the above observation, we propose to use a generative model with three sets of parameters: the original set Θ for capturing similarity vectors of tuple pairs in $T \times T'$, and two new sets Θ_l and Θ_r for capturing similarity vectors of tuple pairs in $T \times T$ and $T' \times T'$, respectively. The new data likelihood function becomes:

$$\begin{aligned} L(\Theta, \Theta_l, \Theta_r) = & \Sigma_{i=1}^{N} \Sigma_{C \in \{M,U\}} \mathbb{1}_{y_i=C} \log\left(\pi_C p(\mathbf{x}_i|\theta_C)\right) \\ & + \Sigma_{i=N+1}^{N+N_l} \Sigma_{C \in \{M,U\}} \mathbb{1}_{y_i=C} \log\left(\pi_{l,C} p(\mathbf{x}_i|\theta_{l,C})\right) \\ & + \Sigma_{i=N+N_l+1}^{N+N_l+N_r} \Sigma_{C \in \{M,U\}} \mathbb{1}_{y_i=C} \log\left(\pi_{r,C} p(\mathbf{x}_i|\theta_{r,C})\right) \end{aligned} \tag{20}$$

Equation (20) is essentially the sum of Equation (2) with three different set of parameters:

$$L(\Theta_l, \Theta_r, \Theta) = L(\Theta_l) + L(\Theta_r) + L(\Theta) \tag{21}$$

Similarly, we can view the data likelihood function as the new free energy function that treats $\boldsymbol{\gamma}_l$, $\boldsymbol{\gamma}_r$, and $\boldsymbol{\gamma}$ as parameters:

$$F(\Theta_l, \boldsymbol{\gamma}_l, \Theta_r, \boldsymbol{\gamma}_r, \Theta, \boldsymbol{\gamma}) = F(\Theta_l, \boldsymbol{\gamma}_l) + F(\Theta_r, \boldsymbol{\gamma}_r) + F(\Theta, \boldsymbol{\gamma}) \tag{22}$$

where $F(\Theta_l, \boldsymbol{\gamma}_l)$, $F(\Theta_r, \boldsymbol{\gamma}_r)$ and $F(\Theta, \boldsymbol{\gamma})$ are the three free energy functions for the three types of similarity vectors.

Note that we still have one generative model with three sets of parameters, instead of three independent generative models, as the three sets of parameters are linked by transitivity constraints. Algorithm 2 describes the final ZeroER procedure incorporating transitivity. As we can see, the majority of the procedure is about running the corresponding part in Algorithm 1 three times for $T \times T'$, $T \times T$, and $T' \times T'$, including the initialization, the M-Step, and the updating posterior probabilities part of the E-Step. The only major change is in Line 7 that resolves transitivity violations once we have updated posterior probabilities using the method described in Section 4.2.

5 EXPERIMENTS

We conduct an extensive set of experiments to evaluate the efficacy of ZeroER along three dimensions:

- *Feasibility and Performance of ZeroER (Section 5.2)* Is it possible for an unsupervised method to achieve performance comparable to supervised ML algorithms? How does it compare with existing unsupervised methods?
- *Ablation Analysis (Section 5.3).* How does our innovations (generative modelling, feature regularization, and incorporating transitivity) in ZeroER contribute to its accuracy?
- *Sensitivity Analysis (Section 5.4).* Is ZeroER sensitive to the size of the dataset, the regularization hyperparameter, its initialization, the set of features used, and the strength of the blocking function used?

Algorithm 2: ZeroER

```
Input: two tables $T$ and $T'$, a set of similarity functions
    $\mathcal{F}$, an optional blocking method $\mathcal{B}$
Output: matching pairs in $T \times T'$
1 For each of $(T, T')$, $(T, T)$ and $(T', T')$, execute Line 1 to
    Line 5 in Algorithm 1
2 while Not Converged do
3     M Step
4         Run Line 8 to Line 12 in Algorithm 1 for
            left,right and cross table to update the three set
            of parameters $\Theta_l$, $\Theta_r$ and $\Theta$.
5     E Step
6         Obtain $\boldsymbol{\gamma}_l^*$, $\boldsymbol{\gamma}_r^*$ and $\boldsymbol{\gamma}^*$ using Equation (3) with $\Theta_l$,
            $\Theta_r$ and $\Theta$, respectively.
7         Resolve transitivity constraints, namely, update
            $\boldsymbol{\gamma}_l$, $\boldsymbol{\gamma}_r$ and $\boldsymbol{\gamma}$, based on $\boldsymbol{\gamma}_l^*$, $\boldsymbol{\gamma}_r^*$ and $\boldsymbol{\gamma}^*$, using the
            method in Section 4.2
8 end
9 return all pairs corresponding to entries in $\boldsymbol{\gamma}^*$ that are
    greater than 0.5
```

5.1 Experimental Setup

Hardware and Platform. All our experiments were performed on a machine with a 2.20GHz Intel Xeon(R) Gold 5120 CPU, a Tesla K80 GPU and with 96GB 2666MHz RAM.

Datasets. We focus on the ER task of structured datasets. We conducted extensive experiments on five structured datasets from three diverse domains including publication, e-commerce and restaurants. Table 2 provides statistics of these datasets. All are popular benchmark datasets extensively used in prior ER work.

Dataset	#Tuples	#Ms	#As
FZ, Fodors-Zagat [2]	533 - 331	112	7
DA, DBLP-ACM [1]	2,616 - 2,294	2,224	4
DS, DBLP-Scholar [1]	2,616 - 64,263	5,347	4
AB, Abt-Buy [1]	1,082 - 1,093	1,098	3
AG, Amazon-Google products [1]	1,363 - 3,226	1,300	4

Table 2: Datasets characteristics, Ms and As denote matches and attributes

Algorithms Evaluated. We compare ZeroER against 4 supervised methods (1-4) and 5 unsupervised methods (5-9), and 1 active learning method (10). The ten algorithms are:

(1) *Logistic Regression (LR):* This is a typical linear classifier. We use 5-fold cross validation to tune the ℓ_2 regularization parameter.

(2) *Random Forest (RF):* This is a typical tree-based classifier. The number of trees is set as 100 and the minimum number of samples required to be at a leaf node is tuned by a 5-fold cross validation to avoid overfitting.

(3) *Multi-layer Perceptron (MLP):* This is a typical deep learning classifier. We use two hidden layers with 50 and 10 hidden units and tuned ℓ_2 regularization parameter by 5-fold cross validation.

(4) *DeepMatcher (DM):* DeepMatcher [44] with hybrid attribute summarization is the state-of-the-art deep learning method for entity resolution. We use the open-source implementation [8] from the authors. Hyperparameters are tuned by grid search and a validation set is used to select the best model snapshot among all epochs to prevent overfitting [44].

(5) *K-Means (KM-SK):* This baseline applies the K-Means algorithm from Scikit-Learn with $K = 2$. If the features vectors for matches and unmatches are very different, then this method would provide good results.

(6) *K-Means (KM-RL):* This is an improved baseline from [22] that is calibrated for two-cluster ER task. Traditional K-Means often fails when the sizes of two clusters are very uneven [4] which is often the case in ER. This algorithm tackles the class imbalance through class weighting so that matches get a higher weight than unmatches.

(7) *GMM:* This applies the naive Gaussian Mixture Model from scikit-learn with 2 components.

(8) *ECM:* The Fellegi-Sunter (FS) model [30] is a seminal approach for probabilistic unsupervised ER that uses binary features and assumes feature independence. We used an improved implementation from [21, 22] that provides better results than the initial FS model. This approach implements a expectation conditional maximization algorithm that relaxes the simplistic feature independence assumption from FS.

(9) *PPjoin:* This is a similarity-join method that employs filtering techniques to improve computational efficiency [58]. Since PPJoin is a single-attribute joining algorithm, we apply PPJoin by concatenating all attributes. PPJoin implemented and evaluated [58] optimized version of two similarity functions (Jaccard and Cosine). PPjoin also requires a threshold to be set. For each dataset, we search the threshold from 0 to 1 by a step size of 0.2 and try both similarity function and report the best result (denoted by PP*).

(10) *Active learning based Random forest (AL-RF):* We use the implementation from modAL [20] and use the default query strategy (uncertainty sampling).

Setups for Various Algorithms. In this work, we treat feature engineering as a black-box, the default set of features for each dataset is generated by Magellan [39], which are used by all methods except DM that uses embedding features and PPjoin that uses its few supported similarity functions. Under the hood, Magellan infers a type for each aligned attribute, and applies a set of pre-defined similarity functions for each type to generate features. All possible attribute types and the similarity functions considered for each attribute type can be found at [6]. We also design a default blocking strategy for each dataset based on its most informative features (e.g., title) using locality sensitive hashing; the same blocking is again employed by all compared methods. For all methods that need a seed, the reported results are the average results of ten runs. We will conduct sensitivity analysis using different features and blocking strategies in Section 5.4.

We use the sklearn [49] implementation for LR, RF, and MLP. We randomly split each dataset to training and test set by 50%-50%. Note that using 50% of the data as training data is very generous as in practice the amount of labeled data available is much smaller. The match entries in the training set are over-sampled as is typically done in training supervised ML methods in the presence of class imbalance. For DM, we hold out $\frac{1}{3}$ of the training set as validation set. For all unsupervised methods, we use the whole dataset for fitting without train-test split. For active learning, it starts with 10 random examples and queries one example at every learning iteration and is terminated once it has queried 50% of the matches or 50% of all examples; We evaluate the performance on the remaining examples. By default, we set the feature regularization parameter κ' for ZeroER to 1%; we set the initialization threshold ϵ to be 0.5; We evaluate the sensitivity of ZeroER to these parameters in Section 5.4.

Performance Measures. We used F-score as the performance measure as ER is a task with unbalanced labels.

5.2 ZeroER Performance

Overall comparison. Table 3 reports the results.

- *ZeroER vs unsupervised methods:* As we can see, though ZeroER builds upon GMM, ZeroER significantly outperforms GMM. ECM is simply not competitive as it binarizes all features and uses Bernoulli mixture model, which loses information. Out of the two K-Means clustering algorithms (KM-RL and KM-SK), KM-RL is obviously better as it addresses the cluster imbalance problem observed in ER tasks. ZeroER is on average 11% better than PP*(PPjoin with the best configuration obtained using the ground-truth) on every dataset. Note that in practice it is not possible to arrive at the best configuration for PPJoin without sufficient labeled data. We also observed that the performance of PPjoin is very sensitive to the threshold, and its F-score can decrease by as large as 0.67 with a 0.2 change in the threshold. Overall, ZeroER greatly outperforms all five unsupervised methods due to the ER-specific innovations.
- *ZeroER vs supervised methods:* The performance of ZeroER is comparable to those supervised methods. This is really impressive as they use the same set of features; while supervised methods have 50% of all data as training, ZeroER is completely unsupervised. DM shows better results on AB and AG because these two datasets contain large strings. However, as shown in Table 4, DM requires more than ten thousand labels to achieve this performance.
- *ZeroER vs active learning:* Active learning is able to even outperform supervised learning. This is because we give active learning the same labeling budget as supervised learning (50% of labeled data), and active learning is able to judiciously select which 50% data to use. In practice, though, active learning budget tends to be much lower. ZeroER is much worse than active learning on DS, because ZeroER relies on a hypothesis that may be satisfied to different degrees on different datasets. The hypothesis is that the matches and unmatches can be divided into two clusters under gaussian distributions.

		Unsupervised					Supervised				Active learning
	ZeroER	ECM	KM-RL	KM-SK	GMM	PP*	RF	LR	MLP	DM	AL-RF
FZ	**1**	0.07	0.30	0.30	0.30	0.97	0.97	0.98	0.99	0.93	**1**
DA	0.96	0.09	0.95	0.27	0.26	0.87	0.98	0.96	0.97	0.97	**0.99**
DS	0.86	0.07	0.85	0.43	0.07	0.83	0.93	0.88	0.92	0.95	**0.99**
AB	0.52	0.01	0.01	0.02	0.02	0.29	0.46	0.18	0.32	**0.63**	0.44
AG	0.48	0.01	0.02	0.02	0.02	0.30	0.51	0.18	0.35	**0.67**	0.46
average	0.76	0.05	0.43	0.19	0.13	0.65	0.77	0.64	0.71	**0.83**	0.78

Table 3: F-score for all methods.

Overall running time comparison. Figure 7 shows that the running time of ZeroER is comparable to other methods. Active learning and DM are typically 10 to 100 times slower than other methods. Active learning will be even slower in practice because we only reported machine time and human time should also be considered in practice. The relative running time of ZeroER varies across datasets. This is because the running time of ZeroER is affected by the number of matches (Equation (19)) and the time for solving the regularization parameter (Equation (11)) and these two vary across different datasets. Note the reported time in Figure 7 is for matching only, and does not include the time for blocking and feature generation.

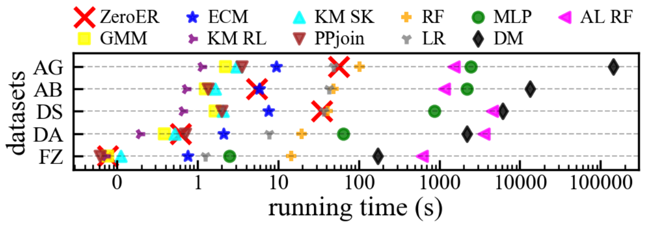


Figure 7: Running time for all algorithms.

	LR	RF	MLP	DM	AL-RF
FZ	2915*	2915*	2915*	2332	1572
DA	418	232	417	4647	26
DS	413	227	270	6768	33
AB	162981*	162981*	162981*	16865	162981*
AG	358281*	7589	358281*	17916	358281*

Table 4: # labeled examples needed for supervised and active learning methods to match ZeroER's F-score. Numbers with an asterisk denote all tuple pairs in a dataset since those methods are not able to match ZeroER's performance.

Labeling effort saved. We further investigate how much labeled data does the supervised and active learning methods need to match the performance of ZeroER. Table 4 shows the labeling efforts saved is quite substantial. Active learning in general is able to reduce the labeled examples required compared with supervised learning, though it requires human involvement as the algorithm runs. Though, DM achieves the best average performance, it requires thousands of labels. On easy datasets (FZ, DA and DS), DM can require up to 30 times more labels than traditional supervised methods.

5.3 Ablation Analysis.

We next perform a series of experiments to understand the contributions of different components of ZeroER and how do they compare with existing solutions to address the same problem. Results are summarized in Table 5. The right-most three columns are the results of replacing each of the three ZeroER innovations (for data deficiency, feature regularization and transitivity) to the naive ones respectively.

	ZeroER	group+share corr → diag+share cov	adaptive reg → uniform reg	posterior constraint → post-processing
FZ	**1**	0.97	0.95	0.99
DA	0.96	0.96	0.36	**0.97**
DS	**0.86**	0.78	0.59	0.41
AB	**0.52**	0.08	0.07	0.45
AG	**0.48**	0.09	0.04	0.42
average	**0.76**	0.57	0.38	0.65

Table 5: Ablation analysis for ZeroER.

Handling data deficiency. Our method of handling data deficiency (feature grouping and sharing correlation matrix) works better than existing commonly used approaches (assuming feature independence and sharing covariance matrix) on four datasets (FZ, DS, AB and AG).

Feature regularization. We compare our adaptive regularization to uniform regularization. Since there is no magic constant that works for all features in uniform regularization (c.f. Example 3.1), we use the default regularization constant provided in the sklearn package [5]. ZeroER adaptive regularization is significantly better than uniform regularization on all datasets.

Incorporating Transitivity. We compare incorporating transitivity in a post-processing step with ZeroER's approach of modelling it as constraints in the learning process. To make postprocessing possible, we need to assume that the left and right table are duplicate free and the matching probability of any tuple pair in the same table to be 0, i.e., $\gamma_{2,3,M} = 0$ in Equation (12), and we designate the one tuple pair out of the two cross-table tuple pairs with a higher posterior probability ($\max(\gamma_{1,2,M}, \gamma_{1,3,M})$) as the match. Our way of incorporating transitivity as constraints on posterior probabilities significantly produces better results on all datasets. The difference is particularly huge on the DS dataset, as the left and right tables on the DS dataset are actually not duplicate-free. ZeroER makes no such assumption.

5.4 Sensitivity Analysis

We show how ZeroER is robust to various hyperparameter settings, features used, and the blocking methods used.

Sensitivity to regularization hyperparameter. We vary the regularization parameter κ' to understand how it affects the results. The Bhattacharyya coefficient is in [0,1], so the regularization parameter has to be in [0,1]. When $\kappa' = 1$, the standard deviation of M and U distributions is infinity and the two distribution becomes identical. Therefore, from 0 to 10% of the maximum BC value, i.e. [0,0.1], is a reasonable region for κ'. As shown in Figure 8(a), the performance of ZeroER is fairly robust for κ' in [0,0.1]. Note that $\kappa' = 0$ means there is no feature regularization. As κ' increases, F1 score first increases (regularization helps), then becomes relatively stable for a period, and finally gradually decreases (too strong of a regularization). Regardless, $\kappa' = 0.01$ is a good and safe default choice that works well for all datasets.

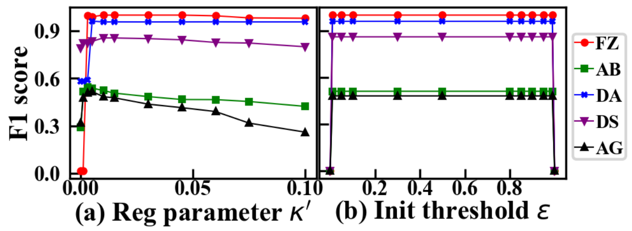


Figure 8: F1 score under different (a) regularization parameter κ, (b) initialization threshold ϵ.

Sensitivity to initialization. As shown in Figure 8(b), ZeroER is robust to initialization threshold ϵ with no changes on quality for all datasets. When $\epsilon = 0$ or 1, (which should not be used as initialization parameter value), no data is assigned to M or U class so that EM fails to recognize two clusters.

Sensitivity & Scalability to unlabeled training data size. We demonstrate that ZeroER can produce excellent results even when trained on a subset of dataset. Specifically, we vary the amount of data (without labels) used to fit ZeroER and evaluate how it impacts ZeroER's performance on the reminder of data. As shown in Figure 9(a), F1 score increases as the amount of unlabeled training data increases. Figure 9(a) also shows that ZeroER already gives good F1 score with only about 10% training data, which can save about 90% training time. By varying the training data size, we can also demonstrate the scalability of ZeroER. Figure 9(b) shows the running time per EM iteration is roughly linear to the amount of training data. This means ZeroER is scalable.

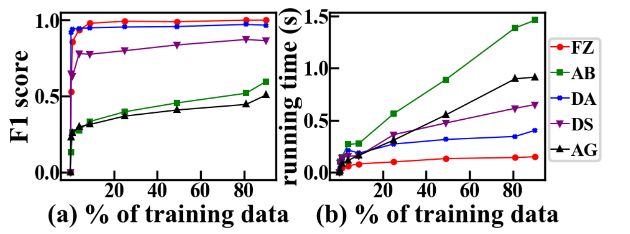


Figure 9: (a) F1 score and (b) running time per EM iteration when varying (%) of unlabeled training data.

Sensitivity to blocking. Though we consider blocking to be orthogonal to us, we vary the aggressiveness of blocking by changing the overlapping size used in the locality sensitive hashing blocking scheme. A more aggressive blocking setting will generate less tuple pairs, but potentially at the risk of missing matches. We vary the overlapping size from the minimum value (least aggressive) that still allows the algorithm to finish within a preset time limit to the maximum value that results in zero tuple pairs. As shown in Figure 10(a), both ZeroER and RF are robust to blocking. Overall, ZeroER is comparable to RF and is better than random forest on FZ and AB regardless of the overlapping size. By increasing the overlapping size, some of the difficult examples would be excluded in blocking. Dataset AB is affected more dramatically, as it has more difficult examples excluded by blocking.

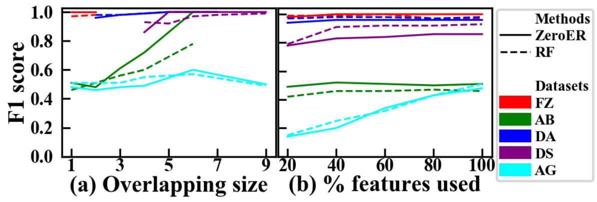


Figure 10: F1 score for ZeroER and RF (random forest) varying (a) degree of blocking and (b) (%) of features.

Sensitivity to features. We vary the amount (%) of features by sampling from the features generated by Magellan to see if the comparison between ZeroER and supervised methods is sensitive to the feature set used. As shown in Figure 10(b) The performance of ZeroER is robust to different (%) of features on most datasets except AG. Though on AG, both ZeroER and RF have degraded results.

6 RELATED WORK

Entity resolution is one of the fundamental and challenging problems in data curation. A good overview of ER can be found in surveys such as [25, 29, 31, 34, 41, 43, 45] and tutorials [31]. Previous work on ER can be broadly divided into three branches depending on how they relate to this paper: (1) identifying useful similarity functions; (2) tackling the efficiency challenges of ER; and (3) designing an algorithm to predict matches/unmatches.

A variety of similarity functions have been used for ER [29], including generic ones such as edit distance, jaccard distance, and cosine similarity, as well as some domain specific ones such as Jaro distance for people names [37]. A comparison of different similarity functions can be found in [15]. ER work usually either assumes that feature engineering for an ER dataset is done by a human, or employs a default set of similarity functions. ZeroER can work with any similarity functions, and by default, uses features generated by the open-source ER package Magellan [39].

A second branch of work aims at reducing the quadratic complexity of an ER task. Some of these algorithms are specific to a similarity function, such as edit distance [32] or jaccard distance [9, 33]. Chaudhuri et al [17] provides an efficient similarity join operator that can work with multiple similarity functions. Another main direction, called blocking, aims to avoid comparing tuples pairs that are unlikely to be matches, leveraging techniques such as locality sensitive hashing [47, 48]. ZeroER is orthogonal to this line of research, and takes an optional blocking function as input.

The third line of related work concerns with designing an algorithm to predict matches. Supervised ML approaches provide state-of-the-art results for ER. However they require substantial amount of training data in the form of matches and unmatches [26, 28, 40, 44]. This process is often time consuming, cumbersome and error prone. Popular approaches include binary classifiers such as Naive Bayes [57], decision tree [16] and SVM [12]. Active learning based techniques have also been proposed to reduce the amount of labeled examples [10, 51]. We have compared with three representative and best-performing supervised ML models as well as an active learning based approach, and showed that ZeroER achieves comparable results with zero labeled examples. General-purpose clustering algorithms (e.g., K-Means and GMM) can be used to perform ER in an unsupervised way. As we have shown, they usually produce much inferior results. The Fellegi-Sunter (FS) model [30] is generally regarded as the seminal work in probabilistic unsupervised ER, and was famously used for deduplicating the Census data that uses binary features and assumes feature independence [57]. Since then, specific adaptations to the FS model has been explored for specific domains, such as US homicides [50], traffic crashes [19], and hospitalizations [18]. ZeroER can be seen as a generalization of FS model that includes three major innovations (a reduced set of parameters, feature regularization, and incorporating transitivity), which together produces a much more performant unsupervised approach that works across datasets in different domains.

7 CONCLUSION AND FUTURE WORK

In this paper, we proposed ZeroER, an effective and efficient solution for solving entity resolution with *Zero* labeled examples. ZeroER provides excellent results across multiple datasets and is competitive with supervised ML approaches. This superior performance is achieved by a collection of novel techniques including generative modeling, adaptive feature regularization and incorporating transitivity. ZeroER dramatically reduces the labeling effort of domain scientists in ER tasks. We hope that ZeroER will play a vital role in the increasingly popular area of self-service data preparation. Currently, ZeroER treats feature engineering as a black box and uses features generated by Magellan. Designing automatic feature engineering solution for ER in general is an interesting future work.

REFERENCES

[1] Benchmark datasets for entity resolution. https://dbs.uni-leipzig.de/en/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution.

[2] Duplicate detection, record linkage, and identity uncertainty: Datasets. http://www.cs.utexas.edu/users/ml/riddle/data.html.

[3] Bhattacharyya distance - Wikipedia. https://en.wikipedia.org/wiki/Bhattacharyya_distance#Bhattacharyya_coefficient, Oct 2019. [Online; accessed 14. Oct. 2019].

[4] How to understand the drawbacks of k-means? https://stats.stackexchange.com/questions/133656/how-to-understand-the-drawbacks-of-k-means, Jun 2019. [Online; accessed 28 Jun 2019].

[5] sklearn.mixture.GaussianMixture — scikit-learn 0.21.2 documentation, May 2019. [Online; accessed 31. May 2019].

[6] User Manual for py_entitymatching — py_entitymatching 0.3.0 documentation, Jun 2019. [Online; accessed 9. Oct. 2019].

[7] ZeroER technical report. https://www.dropbox.com/s/aersjnp0gjmy5pz/ZeroER_technical_report.pdf?dl=0, April 2020.

[8] anhaidgroup. Deepmatcher. https://github.com/anhaidgroup/deepmatcher.

[9] A. Arasu, V. Ganti, and R. Kaushik. Efficient exact set-similarity joins. In *Proceedings of the 32nd international conference on Very large data bases*, pages 918–929. VLDB Endowment, 2006.

[10] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *SIGMOD*, pages 783–794, 2010.

[11] V. Berisha, A. Wisler, A. O. Hero, and A. Spanias. Empirically estimable classification bounds based on a nonparametric divergence measure. *IEEE Transactions on Signal Processing*, 64(3):580–591, 2015.

[12] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, 2003.

[13] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[14] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] A. Chandel, O. Hassanzadeh, N. Koudas, M. Sadoghi, and D. Srivastava. Benchmarking declarative approximate selection predicates. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 353–364. ACM, 2007.

[16] S. Chaudhuri, B. Chen, V. Ganti, and R. Kaushik. Example-driven design of efficient record matching queries. In *VLDB07*, pages 327–338, 2007.

[17] S. Chaudhuri, V. Ganti, and R. Kaushik. A primitive operator for similarity joins in data cleaning. In *22nd International Conference on Data Engineering (ICDE'06)*, pages 5–5. IEEE, 2006.

[18] D. E. Clark. Practical introduction to record linkage for injury research. *Injury Prevention*, 10(3):186–191, 2004.

[19] P. Cryer, S. Westrup, A. Cook, V. Ashwell, P. Bridger, and C. Clarke. Investigation of bias after data linkage of hospital admissions data to police road traffic crash reports. *Injury prevention*, 7(3):234–241, 2001.

[20] T. Danka and P. Horvath. modAL: A modular active learning framework for Python. available on arXiv at https://arxiv.org/abs/1805.00979.

[21] J. De Bruin. Probabilistic record linkage with the fellegi and sunter framework: Using probabilistic record linkage to link privacy preserved police and hospital road accident records. 2015.

[22] J. de Bruin. Python record linkage toolkit. https://github.com/J535D165/recordlinkage, 2018.

[23] A. P. Dempster. Covariance Selection. *Biometrics*, 28(1):157–175, Mar 1972.

[24] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[25] X. L. Dong and F. Naumann. Data fusion: resolving data conflicts for integration. *PVLDB*, 2(2):1654–1655, 2009.

[26] X. L. Dong and T. Rekatsinas. Data integration and machine learning: A natural synergy. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1645–1650. ACM, 2018.

[27] J. Duchi, S. Gould, and D. Koller. Projected subgradient methods for learning sparse gaussians. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, UAI'08, pages 153–160, Arlington, Virginia, United States, 2008. AUAI Press.

[28] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, and N. Tang. Distributed representations of tuples for entity resolution. In *PVLDB*, 2018.

[29] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *IEEETKDE*, 19(1):1–16, 2007.

[30] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[31] L. Getoor and A. Machanavajjhala. Entity resolution: theory, practice & open challenges. *PVLDB*, 5(12):2018–2019, 2012.

[32] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, D. Srivastava, et al. Approximate string joins in a database (almost) for free. In *VLDB*, volume 1, pages 491–500, 2001.

[33] M. Hadjieleftheriou, A. Chandel, N. Koudas, and D. Srivastava. Fast indexes and algorithms for set similarity selection queries. In *2008 IEEE 24th International Conference on Data Engineering*, pages 267–276. IEEE, 2008.

[34] T. N. Herzog, F. J. Scheuren, and W. E. Winkler. *Data Quality and Record Linkage Techniques*. Springer Science & Business Media, 2007.

[35] J. Honorio and T. Jaakkola. Inverse Covariance Estimation for High-Dimensional Data in Linear Time and Space: Spectral Methods for Riccati and Sparse Models. *Association for Uncertainty in Artificial Intelligence (AUAI)*, Jul 2013.

[36] P. Jain, P. Kar, et al. Non-convex optimization for machine learning. *Foundations and Trends® in Machine Learning*, 10(3-4):142–336, 2017.

[37] M. A. Jaro. Unimatch: A record linkage system: User's manual. *U.S. Bureau of the Census*, 1976.

[38] M. I. Jordan and C. Bishop. An introduction to graphical models, 2004.

[39] P. Konda, S. Das, P. Suganthan GC, A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. Naughton, et al. Magellan: Toward building entity matching management systems. *PVLDB*, 9(12):1197–1208, 2016.

[40] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, 3(1-2):484–493, 2010.

[41] N. Koudas, S. Sarawagi, and D. Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD*, pages 802–803, 2006.

[42] A. Lyon. Why are normal distributions normal? *The British Journal for the Philosophy of Science*, 65(3):621–649, 2013.

[43] F. Maggi. A survey of probabilistic record matching models, techniques and tools. *Scienti_c Report TR-2008*, 2008.

[44] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra. Deep learning for entity matching: A design space exploration. In *SIGMOD*, 2018.

[45] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Synthesis Lectures on Data Management. 2010.

[46] R. M. Neal and G. E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[47] G. Papadakis, G. Koutrika, T. Palpanas, and W. Nejdl. Meta-blocking: Taking entity resolutionto the next level. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1946–1960, 2013.

[48] G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. Comparative analysis of approximate blocking techniques for entity resolution. *Proceedings of the VLDB Endowment*, 9(9):684–695, 2016.

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

[50] M. Sadinle and S. E. Fienberg. 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.

[51] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *KDD*, 2002.

[52] M. Stonebraker and I. F. Ilyas. Data integration: The current status and the way forward. *IEEE Data Eng. Bull.*, 41(2):3–9, 2018.

[53] H. Tuy and N. Van Thuong. On the global minimization of a convex function under general nonconvex constraints. *Applied Mathematics and Optimization*, 18(1):119–142, 1988.

[54] S. Velasco-Forero, M. Chen, A. Goh, and S. K. Pang. Comparative Analysis of Covariance Matrix Estimation for Anomaly Detection in Hyperspectral Images. *IEEE J. Sel. Top. Signal Process.*, 9(6):1061–1073, Sep 2015.

[55] E. W. Weisstein. Newton's Method, Oct 2019. [Online; accessed 11. Oct. 2019].

[56] S. E. Whang, D. Menestrina, G. Koutrika, M. Theobald, and H. Garcia-Molina. Entity resolution with iterative blocking. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 219–232. ACM, 2009.

[57] W. E. Winkler. The state of record linkage and current research problems. In *Statistical Research Division, U.S. Census Bureau*, 1999.

[58] C. Xiao, W. Wang, X. Lin, J. X. Yu, and G. Wang. Efficient similarity joins for near-duplicate detection. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.