A Variation on the Hit-Miss Model for Data Deduplication

Bryan Ek Emily Nystrom Chris Williams Lucas A. Overbey[*]

Abstract
The use of data from multiple information sources is common in many real world applications. As a result, both the aggregation of multi-sourced data and the handling of aggregate data have been discussed in the statistical literature for decades. This report describes a novel approach to deduplication of records from aggregate compiled data. Adapted from Noren's hit-miss model [*Data Mining and Knowledge Discovery, 2007*], our approach has two main contributions. First, we extended the binary match/mismatch treatment of strings, by incorporating a string distance, which allows for a more granular quantification of string similarity. Second, we improved computational speed of the algorithm by implementing an alternative method to account for correlations between fields.

Key Words: deduplication, data cleaning, record linkage, aggregation, hit-miss

1. Background

1.1 Motivation for Advanced Deduplication Methods

Multiple data sources are sometimes combined when goals of a study are better served by and/or necessitate the use of aggregate data. Such aggregation may achieve field augmentation (providing more information about existing records) and/or sample augmentation (including information from additional entities). In practice, these aggregated data repositories often require additional data cleaning processes (e.g., to remove or merge records that refer to the same entity) before they can be analyzed. In this report, we will focus on one particular data cleaning task: *duplicate detection*.

The process of deduplicating aggregate data is nontrivial *if* it is possible, within record transcription, that observed records from the same underlying entity may differ from one another. Such differences in record transcription may result from data entry errors (e.g. typos or noise), or systematic differences may be present from source-specific entry procedures or systems (e.g. one airline may use the full "Charleston International Airport" while another uses the airport code "CHS"). *Given these inconsistencies, how can such duplicates be effectively identified in a noisy dataset?* With respect to this deduplication problem, we wish to identify observed records (rows in data) that correspond to the same underlying entity. In studying the deduplication problem, we assume that every observed record is a representation of some underlying "true entity" and that information in a messy, aggregated set of records is an approximation of those events. To this end, we consider a model for quantifying the similarity of record pairs and thereafter classifying such pairs as "matches" or "nonmatches."

Within duplicate detection, we consider two related subtasks: (1) identification and removal of exact duplicates that refer to the same "event" and (2) identification and merging of records that represent the same "entity" but can contain both overlapping and disparate information. Either case assumes that each record is a reflection of a single underlying

[*]Naval Information Warfare Center Atlantic (NIWC LANT)

entity, for which fields considered in the deduplication task[1] are each fixed in "truth space" and each entity has a single true value for each field.

1.2 Related Work

Research related to duplicate detection has emerged in a variety of application domains, producing a fragmented array of literature in which it has also been referred to as *record linkage*, *entity resolution*, *object identification*, and *data matching* . It is also sometimes embedded in the *data cleaning* literature [1–5].

A disperse focus on duplicate detection has also yielded models prioritizing different aspects of the problem. For example, research emphasizing data compression has developed efficient algorithms for storage savings when considering primarily exact duplicates [6, 7]. Others have emphasized free text matching in the presence of typographical errors and common abbreviation schemes via measures of string distance [8–10]. Following the publication of Halbert Dunn's "Record Linkage" paper in December 1946 [11], early methods that have employed the record linkage terminology appeared in the public health arena [12–14], followed shortly by census applications [15, 16], providing a general framework for matches or mismatches in categorical fields. Fellegi and Sunter [17] formalized a statistical model using statistical record linkage rules introduced by Newcombe [14]. These methods rely on odds-ratios of frequencies applying to different types of data. More recently, Christen extended the model using scalable indexing [3, 18]. However, this general framework does not account for relative nearness of numeric fields, and matches are rewarded equally regardless of rarity of respective feature values.

Initially proposed by Copas and Hilton [19], the *hit-miss model* is a latent variable model that accounts for both event frequency and errors in numeric field without being overly sensitive to limitations in the amount of available labeled training data. Copas and Hilton's model was modified by Noren et al. [20] to account for correlation between fields.

Nonetheless, the Noren et al. hit-miss model variants still has some limitations. The model is more computationally expensive than Christen's algorithm for the Fellegi record linkage model [18], particularly in its requirement for estimation of the Information Criterion (IC). And although it numerically quantifies deviation in numeric fields, Noren's hit-miss model treats character fields as categorical variables, assigning discrete comparison labels – match, a mismatch, or empty (one or both values are blank) – based on a field's values in a pair of records. This is a limiting granularity to the possibility of small errors in a free text field. Tancredi et al. [21] proposed several extensions to Copas and Hilton's hit-miss model, including Bayesian variants and expanded the model's capability to work with population estimation, though not to address the above-mentioned issues.

1.3 Our Contribution

We introduce modifications to the Noren model that (1) improves computational efficiencies (while maintaining comparable performance) by replacing the IC with a simplification; and (2) utilizes string distance to quantify differences between records for character fields. Overall, our report describes an extended hit-miss model and discusses its performance in a

[1]This method would ignore fields that are not constant over different sources. A record would have a blank when that field is not present and so that field would not contribute positively or negatively to any comparisons for matching. Since the fields being split by source is not *a priori* known, this method can handle that situation, though does not explicitly illuminate that knowledge.

designed experiment. Additional contributions are our simulation and use of a dataset with more fields than typically used in benchmarking and our secondary analysis of performance results.

Noren's hit-miss model and our modifications are described in Section 2. Section 3 describes our experiment for evaluating the efficacy of these modifications against the original Noren model. Experimental results are summarized in Section 4. Discussion is provided in Section 5.

2. Methods

2.1 The Hit-Miss Model

2.1.1 Record Transcription as a Stochastic Process

The hit-miss model includes a stochastic process for transcription of record pairs. First, we will provide a brief description of the record transcription process, and then we will discuss a related process for record pairs.

Suppose a truth T in truth space is transcribed, producing an observation X in observation space. Consider three transcription states: no transcription (X is blank), other transcription with error (X nonblank, with $X \neq T$), and perfect transcription ($X = T$). The term "hit" in the hit-miss model refers to perfect transcription, and the term "miss" refers to imperfect transcription.

Now suppose that a second observation Y is transcribed from the same truth T, where the transciption states and transciption probabilities are assumed to be identical to those for the transcription of X. Under this framework, the transciption of two records from one truth can result in four events: two hits (HH), one miss and one hit (HM or MH), two misses (MM), or at least one blank. Now suppose we are given two observations, which may or may not match. We can start to answer, *"How likely is it that these observations came frome the same truth?"* by considering each of the aforementioned event types.

Similarly, the hit-miss model conducts element-wise (field-wise) comparisons between observed values in a pair of records and in light of the record transcription process. Under the hit-miss model [20], records $\mathbf{X}$ and $\mathbf{Y}$ contain observed values for a set of properties (fields) based on a true, unobserved event $\mathbf{T} = \mathbf{t}$, where $\mathbf{X}$ and $\mathbf{Y}$ are assumed to have been generated in independent random processes.[2] Four record transcription states are considered: hit, miss, blank, and (for numeric fields only) deviation (Fig. 1). For a given categorical field, the truth value can be transcribed correctly (hit), transcribed incorrectly (miss), or not recorded (blank). For a given numeric field, an additional deviation state, which is a variation of a "miss", is also possible. For simplicity, we assume that for a given field, both the probability of a deviation and the variance of deviation are constant and independent of the truth value. For the purposes of the model, we do not need to know the "true" value for a record, but rather, we attempt to identify whether a given pair of records came from the same underlying "truth."

[2]As a matter of notation, $\mathbf{X} = \left(X^{(1)}, X^{(2)}, ..., X^{(K)}\right)^T$ denotes a vector of random variables, where $X^{(k)}$ denotes the random variable for the field k in the first transcribed record, and let the observed values of a transcribed record be represented by lowercase equivalent: $\mathbf{x} = \left(x^{(1)}, x^{(2)}, ..., x^{(K)}\right)^T$. And the same notation is used for the second transcribed record ($\mathbf{Y}$). Here, we use capital X notation to represent a random variable from which a realization is denoted by lower case x.

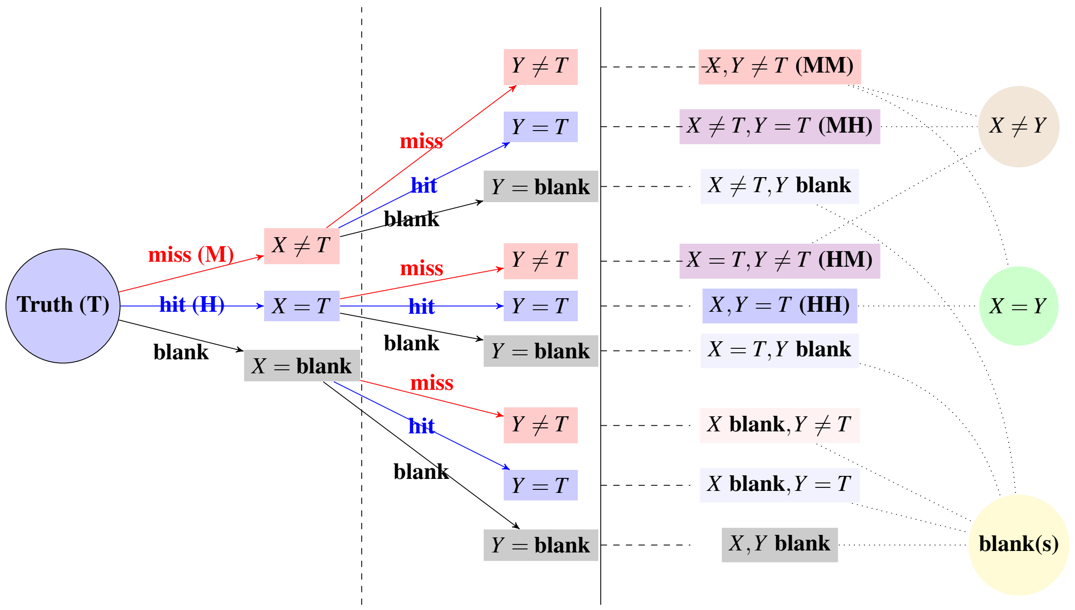

Figure 1: Transcription state diagrams for independent transcription of two records, X and Y, from a single truth T.

In the hit-miss model, deviations in numeric fields are quantified by numerical distance according to a chosen measurement type, such as linear, step, and squared; a distance metric can also be customized for scenarios where the numeric field represents a unique property for which distance has a specific meaning. Noren et al.'s hit-miss model does not include a deviation category for categorical fields (including character fields). Because deviation in character strings has a meaningful interpretation[3], we extended Noren's model by incorporating deviation for character fields (see Section 2.2).

The hit-miss algorithm is a supervised learning approach that uses an ensemble of hit-miss models, one per field in the database. Parameter values used in the hit-miss algorithm, including field-wise record transcription probabilities, are estimated from labeled[4] training data. Resulting models are referred to as "fitted" or "trained" models, and we will denote the ensemble of trained models as the "trained ensemble".

The trained ensemble can be used to provide information about new, unlabeled records. In particular, the trained model estimates "matching score" for a given pair of records. On a larger scale, a matching score can be calculated between a new record and each previous record, and resulting scores can be ranked to identify the previous records with the highest match scores. Unlike *correlation*, which has a well-known range (i.e., (-1,1)) and interpretation for values within that range, the hit-miss model's match score does not have a standardized range or value-based interpretation. Instead, a relative "high" score threshold (above which record pairs are clasified as expected duplicates) can determined for each dataset or chosen *a priori* based on a similar dataset (as discussed in [20]). Section 2.1.5 describes threshold estimation in more detail.

[3]In particular, character strings can deviate as a result of typographical errors, abbreviations, or other forms of transcription errors.

[4]Here, labels indicate underlying entity correspondence and allow for separation of training data into groups of duplicates (triplicates, or size-k equivalents).

2.1.2 Algorithm

The Noren algorithm is distilled in the pseudocode shown in Table 1. Given labeled ground truth training data indicating matching events for given pairs of observations, pairs are uniformly randomly selected without replacement that match for desired training data fields (i.e., if one record matches ten other records in the desired field and another record matches five, the former record is twice as likely to be chosen in a pair; but each individual pair has the same chance of being chosen). Training data is used to determine parameter values for each field. This can be parallelized since fields are treated independently at this stage.

The parameters depend on whether the field is categorical or numerical. For categorical fields, the hit-miss model uses the probabilities, a, b, and $1-a-b$ for miss, blank, and hit, respectively and β_j is the proportion of events with true value j. We compute b directly by counting the proportion of entries in the total database that are blank. For each j, β_j is estimated by the percentage of entries with value j among non-blank entries in the entire database. a is computed indirectly[5] by examining the frequency of discordant non-blank pairs in the training data.

For numeric fields, the hit-miss model is implemented with a_1, a_2, b, and $1-a_1-a_2-b$ for the probability of deviation, miss, blank, and hit, respectively. b is counted directly as with categorical fields. At this point we tacitly make an assumption about numeric fields that all numbers are equally likely.[6] We care about the *difference* between two observed numerical values. We need to estimate $f(d)$, the probability density function for the difference d between two independent random events that follow the distribution of $\mathbf{T}$. This has a mean of zero by symmetry. Noren et al. assumes $f(d)$ is normally distributed (and provides some empirical evidence), but this could be changed if more about the field is known. The variance for f (σ_2^2) is directly estimated from the entire database. You can either compute every pairwise difference or sample if the database is too large. The deviations (σ_1^2) are also assumed to be normally distributed. Finally, it is assumed that $\sigma_2^2 \gg \sigma_1^2$. An expectation-maximization (EM) algorithm iterates over the remaining parameters (a_1, σ_1^2, a_2) until convergence occurs.

When estimating parameters, Noren et al. also uses a chosen minimum value allowed to ensure that deviation and miss error types are allowable (not given a $-\infty$ score) even if training data does not contain an example. To avoid over-reliance on a few highly-correlated fields, Noren et al. computes pairwise correlations between entries of all fields using the IC.

2.1.3 Scoring

For each field, the field's weight is the log-likelihood ratio of the two possibilities (the records approximate the same versus different events) for each field. The match score includes weight contributions from each field and a correction based on correlations between fields.

For categorical fields, let $c^{(k)} = a^{(k)}(2-a^{(k)}-2b^{(k)})$. The weight of the comparison in

[5]Technically a different value is computed and a itself is never used. See Noren et al. [20] and its use of c.

[6]One could incorporate a "rarity" weighting as with categorical fields but this would have to account for record pairs that do not *exactly* match. We leave this extension for future research.

Table 1: Hit-Miss (Noren [20] version) Algorithm Summary

1. Obtain training data (which consists of pairs of known matching records).
2. Input precision values for each numeric field.
3. For each field (k, which we will denote with a (k) superscript), estimate transcription state probabilities and related process variances, where applicable.
   (a) For categorial fields, compute $b^{(k)}$, $\beta^{(k)}$, and $c^{(k)}$.
      i. Calculate the percentage of blanks ($b^{(k)}$) for field k in the database.
      ii. Calculate the proportion of nonblank entries in field k of the database that correspond to category j ($\beta_j^{(k)}$), for each category encountered for field k in the database.
      iii. Calculate the discord constant ($c^{(k)} = a^{(k)}(2 - a^{(k)} - 2b^{(k)})$), which is the relative frequency of discordant pairs in training data[a], divided by $1 - \sum_j \beta_j^{(k)2}$.[b]
   (b) For numeric fields, compute $b^{(k)}$, $\sigma_2^{2(k)}$, $a_1^{(k)}$, $\sigma_1^{2(k)}$, and $a_2^{(k)}$.
      i. Calculate the percentage of blanks in the database ($b^{(k)}$).
      ii. Calculate the variance of nonblank entries ($\sigma_2^{2(k)}$) for field k in the database.
      iii. Use the EM algorithm to estimate $a_1^{(k)}$, $\sigma_1^{2(k)}$, and $a_2^{(k)}$ from the training data.[c][d]
4. Combine field-wise hit miss models to create a hit-miss ensemble.
5. Use the fitted hit-miss ensemble to score record pairs.
   (a) Compute the score between a record pair for each field k.
   (b) For each matched field, compute all pairwise ICs between those fields.
   (c) Order those fields by maximal IC value.
   (d) Subtract a correction term for each match after the new “first”.
6. Noren then produced a threshold (based on a data set separate from the training set) for making a match/nonmatch decision. The threshold was created for a specific false discovery rate.

[a]That is, the frequency of training record pairs that are matches but differ in their observed values for field (k).

[b]Note that $1 - a^{(k)} - b^{(k)}$ is the probability of approximating the event exactly.

[c]Here, the EM algorithm maximizes the likelihood of training data appearances based on the 6 possible outcomes: {(H,H), (H,D), (H,M), (D,D), (D,M), (M,M)}, where H=Hit, M=Miss, D=Deviation.

[d]Here, the probability of approximating the event exactly is $1 - a_1^{(k)} - a_2^{(k)} - b_{(k)}$.

field k is defined as

$$W^{(k)}_{x^{(k)}_{i_1},x^{(k)}_{i_2}} = \begin{cases} \log\left(c^{(k)}\right) - 2\log\left(1-b^{(k)}\right) & x^{(k)}_{i_1} \neq x^{(k)}_{i_2} \\ \log\left[1 - c^{(k)}\left(1-\beta^{(k)}_j\right)\left(1-b^{(k)}\right)^{-2}\right] - \log\left(\beta^{(k)}_j\right) \quad \text{if} & x^{(k)}_{i_1} = x^{(k)}_{i_2} \\ 0 & x^{(k)}_{i_1} \text{ or } x^{(k)}_{i_2} \text{ blank} \end{cases}, \tag{1}$$

where i_1 and i_2 denote indices for records being compared, and $x^{(k)}_{i_1}$ and $x^{(k)}_{i_2}$ denote their respective values for field k.[7] For numeric fields, the weight of the difference d between $x^{(k)}_{i_1}$ and $x^{(k)}_{i_2}$ is given by[8]

$$W^{(k)}_d = \log \int_{d^{(k)}-P^{(k)}}^{d^{(k)}+P^{(k)}} p^{(k)}_m(t)dt - \log \int_{d^{(k)}-P^{(k)}}^{d^{(k)}+P^{(k)}} p^{(k)}_u(t)dt, \tag{2}$$

where $d^{(k)}$ denotes the difference between field k values in the two records and $P^{(k)}$ is the precision of this numeric field.[9] $p^{(k)}_m$ represents the probability density function (pdf) for the difference in field k between duplicate records, and $p^{(k)}_u$ represents the pdf for the difference in field k between non-duplicate records in field (i.e., given that 2 records are a match, their difference pdf would follow $p^{(k)}_m$, whereas if they don't match, their difference pdf would follow $p^{(k)}_u$).

In the hit-miss model, the difference between records from a pair is modeled as mixture of four components: (1) 2 hits, (2) at least one miss (and no blanks), (3) one hit and one deviation, and (4) two deviations. Thus, the corresponding pdf (p_m) is a sum of component densities, weighted by component prevalences. The difference between nonmatching records simply follows $p^{(k)}_u$, scaled by a factor for the probability that they are both not blank.

$$\begin{aligned} p_m(d) &= (1-a_1-a_2-b)^2\delta(d) + a_2(2-a_2-2b)f(d) \\ &\quad + 2a_1(1-a_1-a_2-b)\phi(d;0,\sigma_1^2) + a_1^2\phi(d;0,2\sigma_1^2), \\ \text{and } p_u(d) &= (1-b)^2 \cdot f(d), \end{aligned} \tag{3}$$

where $\delta(d)$ is the 0 indicator function (i.e., $\delta(d) = I(d=0)$) and $f(d)$ is the density function for the difference of 2 random variables chosen from the distribution of the entire field.

The total score between a pair of records is then the sum of $W^{(k)}_d$ (where $d = x^{(k)}_{i_1} - x^{(k)}_{i_2}$ is the difference between the two values in field k) for all numeric fields and $W^{(k)}_{x^{(k)}_{i_1},x^{(k)}_{i_2}}$ for all categorical fields. See Figure 2 for a visualization of the scoring process.

2.1.4 Correlation Correction

Noren et al. incorporates a mitigating factor to discount matches in fields that are highly correlated. If the database has fields that are essentially copies of each other, this mitigation

[7]We adapted Noren's original notation to include field-specific indices.

[8]Only the magnitude of the difference matters since the difference density distributions (for match/deviation/nonmatch) are symmetric about 0.

[9]$P^{(k)}$ is an input to the algorithm chosen by the user, e.g. if the field is age, you may pick a precision value of one year; if the field is a weight, you may pick the resolution of the scale (if known).

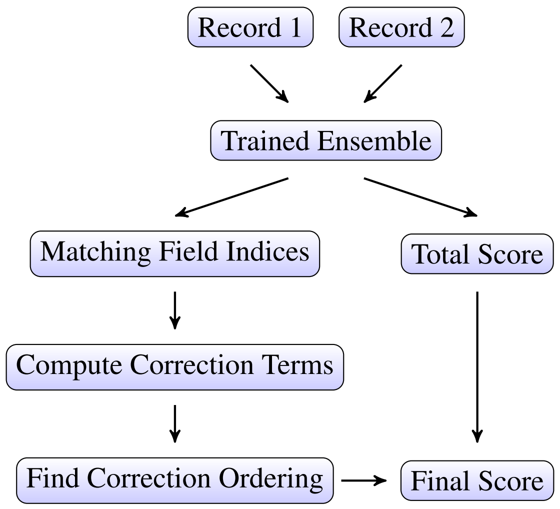

Figure 2: Visualization of workflow for scoring data from a trained hit-miss ensemble.

will prevent them from dominating the total score contributions. Noren et al. accomplishes this with IC values.

Suppose that two records match in field k_1 with value i and match in field k_2 with value j. The IC is the log-likelihood probability of getting a match of j in field k_2 given that the records already matched with i in field k_1 divided by the probability of matching with j if the two matching fields were independent,

$$IC_{i,j} = \log_2 \frac{P(j|i)}{P(j)}. \tag{4}$$

The IC is subtracted from the score for field k_2; we already had a highly correlated match in field k_1, so the match in k_2 is not worth as much. $IC_{i,j}$ can be extended to three or more matching fields $IC_{i,j,m,n,...}$ but that requires significantly more data to accurately estimate. Instead Noren et al. discounts a new match based on the highest IC to all previous matches (i.e. if two records have already matched in fields $k_1, \ldots, k_r$ with values $i_1, \ldots, i_r$, and now match in field k_{r+1} with value i_{r+1}, then the score discount is $\max\{IC_{i_n,i_{r+1}} | n = 1, \ldots, r\}$, in other words, discounted by the "maximum of previous knowledge obtained"). The correction terms should be included by iterating over the matched fields in descending order of the pairwise IC_{i_a,i_b} values, such that the most "new" knowledge is obtained in each successive iteration.

2.1.5 Threshold Estimation

Threshold estimation can be done using a parametric model, nonparametric estimation from another dataset, or a hybrid combination of data-driven estimation and parametric assumptions (as used by Noren et al.). Parametric, nonparametric, and hybrid approaches have different advantages and disadvantages with respect to sensitivity to prior assumptions (e.g., assumed match proportion), use of match/nonmatch labels from training data, and robustness to irregular testing data distribution. Noren et al.'s hybrid approach for determining score threshold uses component parameter estimates (e.g., the mean and variance for each of the match and nonmatch components) from scores and assumed a two-component mixture distribution, where match and nonmatch components were modeled as normal distributions.The cumulative distribution functions (cdfs) from the component and mixture distributions are used in formulas that relate the distribution of record scores to the threshold value.

Noren et al. uses a theoretical model for false discovery rate (FDR) (Eq. 26, [20]), or equivalently the positive predictive value (PPV). Adapted from Noren et al.'s equation, the value of the threshold that corresponds to a specified FDR (under the assumed model) can be calculated by solving for θ:

$$\begin{aligned} FDR(\theta) &= P\left(\text{nonmatch}|\widehat{\text{match}}\right) = P(\text{nonmatch}|\text{score} \geq \theta) \\ &= \frac{P(\text{score} \geq \theta|\text{nonmatch})\,P(\text{nonmatch})}{P(\text{score} \geq \theta)} \\ &= \frac{(1-F_{\text{non}}(\theta))\,\pi_{\text{non}}}{1-F_{\text{Mix}}(\theta)} = \frac{\pi_{\text{non}}S_{\text{non}}(\theta)}{S_{\text{Mix}}(\theta)} = \frac{\pi_{\text{non}}S_{\text{non}}(\theta)}{\pi_{\text{non}}S_{\text{non}}(\theta)+\pi_{\text{match}}S_{\text{match}}(\theta)}, \end{aligned} \tag{5}$$

where S denotes the survival function (i.e., $S_X(x) = P(X > x) = 1 - F_X(x)$), and the cumulative distribution function for the mixture function is given by $F_{\text{Mix}}(x) = \pi_{non}F_{\text{non}}(x) + \pi_{\text{match}}F_{\text{match}}(x)$. In our formulation, we use π_{match} to represent the proportion of record pairs that are duplicates.[10]

Solving for the positive predictive value (PPV) is equivalent to solving for one minus FDR:

$$\begin{aligned} PPV(\theta) &= 1 - FDR(\theta) = P\left(\text{match}|\widehat{\text{match}}\right) \\ &= P(\text{match}|\text{score} \geq \theta) = \frac{\pi_{\text{match}}S_{\text{match}}(\theta)}{S_{\text{Mix}}(\theta)}. \end{aligned} \tag{6}$$

Equation 6 can be used to estimate a threshold score above which record pairs are classified (predicted) as "matches" and below which record pairs are classified as "nonmatches."

Thresholds can also be estimated from other rate types (e.g., accuracy or positive predictive value). See Table 5 for our generalization of Equation 5 to other accuracy-related rate types (e.g., false ommission rate, accuracy). We also note that Equations 5 and 6 are written in terms of general distributions for match and nonmatch scores, whereas Noren's Equation 26 specifically uses the normal distributions for both the match component and the nonmatch component.[11]

Noren et al. estimates the probability that a pair is a match based on a prior record duplication rate and the number of records in the database, estimating parameters for the component distributions based on available data. In particular, Noren et al. uses match scores from the training data to estimate the mean, μ_{match}, and variance, σ^2_{match}, for the match distribution. Parameters for the distribution of nonmatch scores (μ_{non} and σ^2_{non}) were estimated from randomly chosen pairs of records from the entire database.[12] In

[10]We estimated the proportion of duplicate records as follows. From the full dataset, we reserved a subset of data to use for threshold estimation. We estimated the proportion of duplicate records from this subset and assumed that this subset's proportion of duplicate records was similar to the proportion of duplicates in the rest of the data to which the estimated threshold would be applied.

[11]In our experiments, we did follow Noren et al.'s implementation and assumed normal approximations for the match and nonmatch distributions. Considering more generalized threshold estimation approaches (e.g., including other rate types, component distributions, and estimation techniques for distributions based on different cases where training data may or may not be available) is an area for future research.

[12]Technically a matching pair could be chosen by chance but assuming the database is varied enough, this probability is low. And a few incorrectly labeled scores should not have a large effect.

practice, we found this approach to be sensitive to the duplication rate in the database. For example, a random sample drawn from mixed data is likely to contain more matches than nonmatches if the mixed data had more matches than nonmatches. Instead, for our implementation, we reserve a portion of our data (with true match/nonmatch status labels) for threshold estimation from which we estimated means and standard deviations for match and nonmatch scores and the proportion of matched pairs (π_{match}) (see Table 2 for more details).

2.2 Our Modifications

2.2.1 Incorporating String Distance

Datasets with free text fields allow for a diversity of variations to inputs that could potentially represent the same or similar items. For example, maritime shipping data may have free text fields for information such as the booking, shipping, payor, or notification party names, carrier information, or cargo descriptions that may not be annotated the same in each record. These types of variations are common in other datasets as well, including many hand-written or manually annotated forms, surveys, and information collection processes. Official and unofficial terms, phrases, acronyms, initializations, transliterations, and typos are all common occurrences in free form text of this sort. Unifying or finding a measurable distance between like terms across these variations is manually time-intensive and can still be prone to errors. The Noren et al. hit-miss model treated all of these fields as categorical, with no representation for deviations as in numeric fields.

String similarity measures can be employed to measure distance between potential matches. We are particularly interested in entity (proper name person, organization, location, etc.) toponym conflation; however, the choice of the appropriate string similarity measure may be language- and task-dependent. Recchia and Louwerse [22] perform a comparison of such string similarity measures on a similar toponym matching task, focused on location-based toponyms. Their results indicate that ideal toponym matching algorithms may be both context and language-dependent.

With how numeric fields are handled, incorporating a string distance to a field instead of treating it as purely categorical is relatively easy. All parameter estimation and final scoring of numeric fields is based on the difference between the two entries. Instead of using the distance between two numeric values, we use the string distance between the two observed strings. This string distance can also be easily changed to whichever is appropriate for the specific field and database. The only change from numeric is that we no longer necessarily have a deviation distribution that is symmetric around 0, let alone normal. This is because string distance is typically a non-negative value. Similarly for the distribution of f, we don't know that it is normal. As exemplar cases, we utilize the Levenshtein edit distance [23] and the Jaro distance [24]. We consider the deviation and random difference distributions as being half-normal curves.[13] This does lead to some edge case analysis when the string distance is closer to 0 than the precision but that is mostly not an issue.

[13]This is similar to what we do for numeric fields because we always take the negative value distance. The only difference is for the numeric case there is no edge-case analysis near a difference of 0.

2.2.2 *Correction Modification*

The correlation correction that Noren et al. implements has a few drawbacks, primarily, the computational cost. Specifically, computing the IC for each new scoring takes $O(n)$ time, where n is the number of records in the dataset, because pairwise occurrence probabilities must be estimated for all fields in which records match. Precomputing these probabilities would require $O(m^2 f^2)$ memory, where m is the maximum number of unique values for a field and f is the number of fields. Thus, when the number of fields and/or the diversity of values within fields (e.g., the number of unique values encountered for categorical fields) is large, the memory required is prohibitive, making this precomputation of probabilities impractical, and possibly, infeasible. For example, for a data set with only 30 fields and 2000 unique values per field, $\binom{30}{2}2000^2 = 1,740,000,000$ precomputed probabilities would be saved. This also requires making $\binom{f}{2}$ tables each of which takes $O(n)$ time, for a total of $O(nf^2)$ time to precompute the IC values. For databases with fewer fields and fewer unique values, this could be a worthwhile effort. We split the difference and precompute IC values for pairs of fields that have less than a certain bound of unique values.[14]

The other drawback is that IC only accounts for matching categorical fields. It does not handle numeric fields well, partly because of the wide diversity; most values come with some noise, creating many unique values. The IC correction then finds pairwise occurrence probabilities for each of these unique values when a pair of records ”matches,” including for numeric fields[15], which treats the numeric fields as categorical for the IC. These corrections then miss cases where numeric fields could have an even extremely high correlation (e.g. if the database has fields weight (kg) and weight (lb)).

We reduce the computational cost as well as generalize to matching numeric fields by considering simple linear correlations between fields. The cost of finding the correlation between any pair of fields is $O(n)$ so the total precomputing time is $O(nf^2)$, the same as for precomputing IC values. However, each pair of fields only records a single value between -1 and 1 versus an entire table so the memory savings are significant.[16]

A correlation for a pair of categorical fields is found using Cramer's V correlation. Between a categorical field and a numeric field we use an adjusted R^2 assuming each categorical entry should have a fixed numerical value. We then compute the standard errors from each categorical entry's numeric values to determine an adjusted R^2. Finally for two numeric fields, we use the Pearson correlation coefficient. If a field contains a match between the two records, the increased score from the hit-miss model is scaled by $(1-M)$ where M is the maximum correlation to previously computed fields in which the records matched. So if one field is perfectly predicted by a previous, $M = 1$ and the score is not boosted at all.

To compute the iterative order for checking fields, we begin by rewriting the correlations in a forest graph like structure, such that each field is connected with a directed arrow to the other field with which it has the highest correlation. We then use a topological sort of the graph; starting with the root[17] of the tree, we order the remaining fields by distance (in a graph sense) from the root, breaking ties at random. If there are multiple disconnected trees (composing a forest), those individual orderings are intertwined. The goal of the ordering

[14]This bound was chosen based on the memory capabilities of the machine we used for testing.

[15]A pair of records is considered matching in a numeric field if they are more likely a match (from a deviation or hit) than a nonmatch.

[16]Also, reading/writing to giant tables is a non-trivial computational task.

[17]Technically the "root" of this tree will be a pair of fields with arrows to each other. Pick one at random.

is the same as with IC values: end with the fields that provide the least amount of "new" information. This is precomputed and done once, whereas the ordering for IC values could change for each scoring. Overall, in small informal tests, the ordering of the fields tended not to affect the scores in a substantial way.

Our alternative correction method listed above is referred to later as *cor*. We also consider an extra correction term for mismatches. We discount the penalty from a mismatch by scaling the penalty by $(1-m)$, where m is the maximum correlation to previously computed fields in which the records mismatched. The alternative correction method with both match and mismatch corrections is referred to later as *cor2*. Noren et al.'s correction method is referred to as *IC*, and no correction method is *control*.

3. Experiments

3.1 Simulated Data

We conducted two experiments, Experiment 1 (H-M Algorithm Performance; Section 3.3) and Experiment 2 (Data Messification; Section 3.4), using "messified" datasets that were simulated from a Kaggle dataset about Crimes in Boston from 2015 to 2018 [25]. Reasons for using the Kaggle Boston Crime dataset for our experiments include its availability for public use, its size (having 319k observations), and the variety of data types represented in 17 different fields, including geographical, temporal, and categorical string-based fields. We treated the original Kaggle dataset as the "ground truth," from which we simulated derived datasets to represent aggregation of records with the possibility of duplication and transcription errors. Then these derived datasets were used for our experiments. The exp Additional details about our simulated messification process are described in Section 3.4.

3.2 Conducting Algorithm Performance Experiments

Steps for evaluating algorithm performance for both experiments are described in Table 2. The details for each individual experiment are described in Sections 3.3 and 3.4, respectively. Results of the experiments are outlined in Section 4, with an additional linear model analysis in the Appendix.

3.3 Experiment 1: Assessing the Impact Hit-Miss Choices

3.3.1 Experimental Factors for Experiment 1

Our first experiment is a two-factor experiment to study the impact of correction type and string method on the hit-miss model performance (Fig. 3). Field dependency correction types considered in our experiment include the following levels:

- no correction ("control"),
- the IC correction method (used by Noren),
- our correlation-based corrections (2 variations, *cor* and *cor2*, described in Section 2.2).

String methods considered in our experiment included the following levels:

- exact string comparison (used by Noren et al.) (E),

Table 2: Algorithm Performance Experiment Evaluation Steps

Perform the following steps for each dataset in the experiment (*with specifics in footnotes*).

1. Locate or assign a unique identifier for the database of records to represent a "match", in terms of having been transcribed from a single entity in truth space.[a]

2. Partition data into train, test, and validation subsets.

   (a) Group the records by the unique identifier.

   (b) Randomly assign each of the group identifiers to exactly one of the train/test/validation partitions using a pre-determined relative partition size.

   - Training Set Creation Method 1: Randomly select two records from each group with replacement.[b] This method gives equal representation to each group, regardless of group size.[c]
   - Training Set Creation Method 2: For each group, create a set of all possible record pairs. Combine sets from each group into a pool set. Then randomly select the desired number of total pairs from the pool set. This method will weight group representation by group size.

   (c) For the testing set, further divide the set into a portion used for matches and a portion used for non-matches.[d]

   - For each group assigned to the matching portion, randomly pick two records with replacement. The resulting self-matches are a check that scoring places them appropriately high.
   - For the non-matching portion, for each group, randomly pick a different group to create a group pair. Then for each group in that pair, pick a record from that group.

   (d) Repeat the testing set division and record pair creation for the validation set.

[a] We used the field "Incident Number" as the unique identifier field.

[b] This could obviously (especially for groups with a single record) lead to self-matches being selected in the training data. We chose to keep these self-matches for training to bias the model towards being stricter on matching criteria. If most groups only have a single record (i.e. there are no duplicates for that record), then the majority of matches we come across would actually be self-matches.

[c] We chose Training Set Creation Method 1.

[d] We chose to have the proportion of matches in our testing set be equal to the proportion of non-self-matches in the training set, bounded between 1% and 10%. Bounding between these percentages was so that we had some matches to score against, but yet not be overwhelmed by matches because likely implementations would have a low proportion of matches in testing.

Table 2: Algorithm Performance Experiment Evaluation Steps (continued)

3. For each desired string distance and correction method combination, train a hit-miss algorithm using the training subset (See Table 1) and score the testing and validation sets using the chosen correction method.
    (a) Use training data to estimate parameters for the hit-miss model ensemble.
    (b) Use the trained hit-miss ensemble to calculate scores with correction for all record pairs in the test and validation subsets.
4. From the scored testing set, estimate the threshold value based on a chosen threshold method.[a]
5. Use the estimated threshold to classify pairs in validation data. Record pairs with scores above the estimated threshold are classified as "matches" and pairs with scores below the threshold are classified as "nonmatches."
6. Compute achieved accuracy-related rates (e.g., true positive rate (TPR)) by comparing predicted classifications from Step 5 to true classification labels (See Table 5).
7. Assess hit-miss algorithm performance by analyzing achieved rates results (e.g., using linear models or graphical visualizations).

[a]We estimated a threshold value based on Equation 6. Although Noren et al. used "random" selection scored records to simulate nonmatches, we found that resulting threshold estimates are heavily dependent on the prevalence of matches in training data. Thus, we reserved a "validation" subset, which includes both matches and nonmatches, to estimate the threshold.)

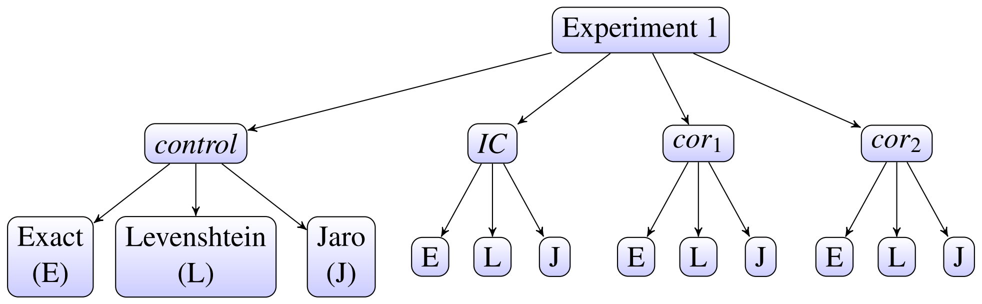

Figure 3: Visualization of Experiment 1's full factorial design with factors correction type and string method.

- Levenshtein distance (L),
- Jaro distance (J).

3.3.2 Data for Experiment 1

For each (correction type, string method) combination, five independent (differently seeded in the input data) trials were used for a total of 5 x 4 x 3 = 60 experimental runs, from which results are recorded for each simulated dataset. This experiment was conducted for two different simulated datasets, for a total of 60 x 2 = 120 runs.

Simulated datasets 3 and 9 (Table 3) were selected for Experiment 1 because they incorporate both types of missingess that were considered in our second experiment and at two scales of duplication. Datasets 3 and 9 correspond to $(p_{\text{Noise}}, p_{\text{Remove}}, N_D) = (0.5, 0.3, 2)$ and (0.5, 0.3, 4) respectively, where p_{Noise}, p_{Remove}, and N_D are described as in Section 3.4.2. From the full Kaggle Boston Crime dataset, a subset of 50,000 of the original records was selected. Thus, an $N_D = 2$ will involve 100,000 records, and an $N_D = 4$ will involve 200,000 records.

3.4 Experiment 2: Assessment of the Impact of Data Messification

Our second experiment studied the impact of several data "messiness" factors on hit-miss algorithm performance. These messiness factors were simulated for the Kaggle Boston Crime dataset, specifically, to build intuition about changes in our datasets and corresponding hit-miss ensemble algorithm outputs.

3.4.1 Data Messification Experimental Factors

The data messification experiment is a three-factor experiment that considers factors related to duplication, noise, and missing data. The following three factors were considered in our data messification experiment:

- Duplication rate (N_D): the rate at which a row of the data is randomly duplicated (with replacement). The duplication process is applied before the messifications below occur.
- Random noise messification rate (p_{Noise}): the probability that the random noise formula is applied to any row in the data

- Missing messification parameter (p_{Remove}): the probability that missing completely at random (MCAR) removal will be applied to a given cell in the dataset. This probability is the same for all cells. The same parameter, p_{Remove}, will also represent the probability that context-based missingness will be applied, as described below. Thus, context-based missingness and MCAR cell removals (both described in more detail below) will occur at the same rates.

A full factorial design was used for our three-factor experiment; thus, all 2 x 3 x 2 = 12 combinations of p_{Noise}, p_{Remove}, and N_D levels were considered. Levels for each of the experimental factors are shown below:

- Noise propensity: $p_{\text{Noise}} \in \{0.1, 0.5\}$;
- Missingness propensity: $p_{\text{Remove}} \in \{0, 0.1, 0.3\}$;
- Duplicate rate: $N_D \in \{2, 4\}$.

3.4.2 Data & Data Messification Procedure

A messification dataset was simulated for each of these (duplication, noise, missing data) rate combinations, for a total of 12 messification datasets (Table 4). Additional messification procedure details are described in Table 3.

Whereas Experiment 1 used all (string method, correction type) combinations and only two messification datasets, Experiment 2 used only two (string method, correction type) combinations and used all messification datasets. From the full list of 12 (string method, correction type) combinations considered in Experiment 1, two combinations were selected for our data messification experiment[18]: (1) Noren et al.'s implementation (i.e., (exact string method, IC correction type)); and (2) one of the new (string method, correction type) combinations that we proposed. This second combination was the combination in {(L, *cor*), (L, *cor2*), (J, *cor*), (J, *cor2*)} that had the highest average true positive rate (across all trials and datasets 3 and 9) in Experiment 1.[19]

Similar to the first experiment, the data messification experiment used five independent trials for each combination of experimental factors and for each of the two selected (correction type, string method) pairs, for a total of 5 x 12 x 2 = 120 experimental runs for the data messification experiment.

4. Results

4.1 Experiment 1 Results

Figures 4 and 5 display the achieved accuracy-related rates from Experiment 1, in which thresholds were estimated based on a desired, theoretical PPV. Within each figure, each plot corresponds to a different rate type, which is displayed on the vertical axis. The horizontal axis (and point color) of each plot corresponds to the correction type (*control*, *cor*, *cor2*, and *IC*). Point shape corresponds to string method (E, L, or J). Spacing within each correction type corresponds to trial number.

[18]Our decision to select two pairs is based on our desired to focus this experiment on data messification factor for a particular dataset, rather than including hit-miss model algorithm factors and data messification factors in a combined study.

[19]Other definitions of "best" could be used here. This definition aligns with the rest of our experiment.

Table 3: Experiment 2 Messification Procedure Details

- **Injection of Random Noise, by Field.**
  - Offense Code: Add a random integer (between 1 and 100) with probability p_{Noise}.
  - Offense Code Group: Add typos with probability p_{Noise}.
  - Offense Description: Garble text with probability p_{Noise}.
  - Latitude: Add a random integer (between 1 and 15) with probability p_{Noise}.
  - Longitude: Add random integer (between 1 and 15) with probability p_{Noise}.
- **MCAR Removal of Individual Cell Values.**
  Every cell (every field in every row) has an equal likelihood (p_{Remove}) of being removed. Cells are randomly removed independent of whether other cells in the same row or column have also been removed.
- **Context-Based Missingness.**
  Values in the Street and District fields were sometimes removed based on the value of the Offense Code field. For Offense Code values between 599 and 1500 (inclusive), street and district were removed with probability p_{Remove}. This scenario simulates removing detail data for crimes that could reflect poorly on neighborhoods.[a]

[a]This is a fake scenario to illustrate context-based missingness for illlustration purposes only, and scenario assumptions are not inspired by known practice.

Table 4: Data Messification dataset index, p_{Noise}, p_{Remove}, and N_D for each of the 12 messification simulated datasets

Index	p_{Noise}	p_{Remove}	N_D
1	0.5	0.0	2
2	0.5	0.1	2
3	0.5	0.3	2
4	0.1	0.0	2
5	0.1	0.1	2
6	0.1	0.3	2
7	0.5	0.0	4
8	0.5	0.1	4
9	0.5	0.3	4
10	0.1	0.0	4
11	0.1	0.1	4
12	0.1	0.3	4

PPV plots show the achieved PPV values for the testing dataset. These achieved PPV achieved values can be compared to the desired, theoretical PPV (0.95), which was used to estimate the thresholds that were used to classify pairs in the testing data. Differences between the desired and achieved PPV values may be caused by differences in the actual distribution family of the scores, compared to the assumed normal distribution (Eq. 6), and/or naturally occuring differences that result from sampling (since different data partitions were used to estimate the threshold than to train the threshold).[20] From the achived PPV values, (*cor2*, J) was the only (score type, string method) combination that did not have an achived PPV of at least 0.95. The superior achieved PPV with respect to the exact (E) string distance may be a product of differences in the theoretical distribution of scores for these combinations. Recall that the combinations with the exact string distance do not directly account for near, inexact matches in string fields. And in general, we expect that systematic differences seen in the achieve PPV plot may be related to systematic differences in the actual distribution of scores compared to the theoretical distribution.[21] The consistency of PPV results for both dataset 3 and 9 (Figures 4 and 5) indicate that the theoretical threshold computation does seem to appropriately account for differing duplication rates.

The TPR plots provide information about the achieved TPR (also known as "hit rate," "sensitivity," or "recall") for various string/correction combinations, where like the PPV plots, all results were generated using thresholds that were estimated from a desired, theoretical PPV of 0.95. With a lower duplication rate ($N_D = 2$, Figure 4), the higest achieved TPR tends to vary between string distance and correction types. For example, among string methods for correction type *cor*, the Jaro distance (J) has the highest TPR. However, among string methods for correction type *IC*, the Levenshtein distance (L) results in the highest TPR, and J and L are about equal for both *control* and *cor2* correction types. For the higher duplication rate ($N_D = 4$, Figure 5), the achieved TPR for the exact string

[20]In the former case, consideration of other distributions could improve the accuracy of the score, while the latter case is not something to be mitigated. Overall, we expect that a mix of distribution family (macro) and parameter estimate (micro) differences is likely present in our experiment, and there is room for additional research in considering additional methods for estimating the threshold.

[21]This is an area for future research.

method was always smaller than the achieved TPRs for other string distances; and this pattern was seen for all correction methods for the higher duplication rate. And overall, greater variation was seen among achieved TPRs for $N_D = 4$ (Figure 5) than for $N_D = 2$ (Figure 4).

The F1-score is the harmonic mean between TPR and PPV, so the achieved values in the F1-score plots are a form of a balance both achieved TPR and achieved PPV. Thus, for example, the (*control*, E) pair is not artificially punished by having a lower TPR because it corresponds to an actual that is better at the given theoretically-set threshold. At the lower duplication rate ($N_D = 2$, Figure 4), the exact string method had the highest F1-scores (among string methods); alternatively, for the higher duplication rate, other string methods had higher achieved F1-score. These differences make sense based on differences in TPR despite consistencies in PPV between the two duplication rates. Thus, the ideal hit-miss variant may depend on the specific application (see Appendix), the characteristics of duplicates, and the form of their "messiness." We explore this more in Experiment 2.

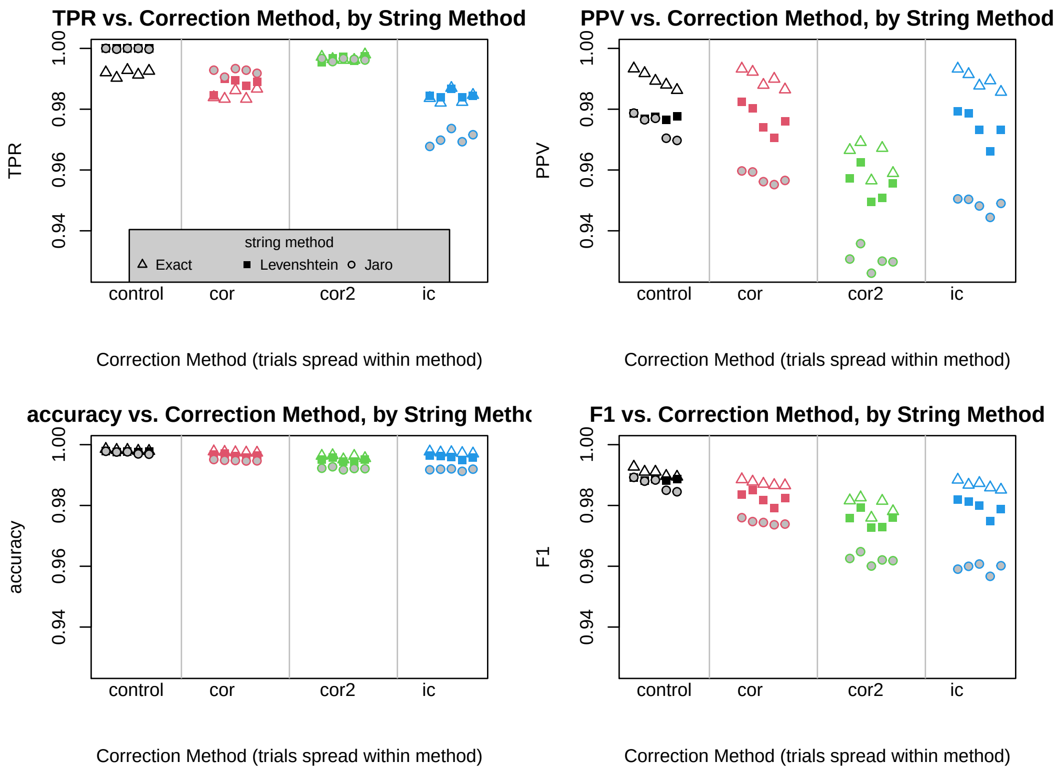

Figure 4: Achieved accuracy-related rates for Experiment 1, dataset 3

4.2 Experiment 2 Results

Figures 6 and 7 display the results for various messification types on Noren et al.'s hit-miss method and one of the variants to our method. Here, we choose the distance/correction pair (J, *cor2*) as a representative variant

We can see that the PPV actually reaches and subsequently exceeds the theoretical PPV threshold with Noren's method (Figure 6) as we increase the rate of field removals (p_{Remove}). However, the TPR is inversely impacted, and in particular, when N_D and p_{Noise} is

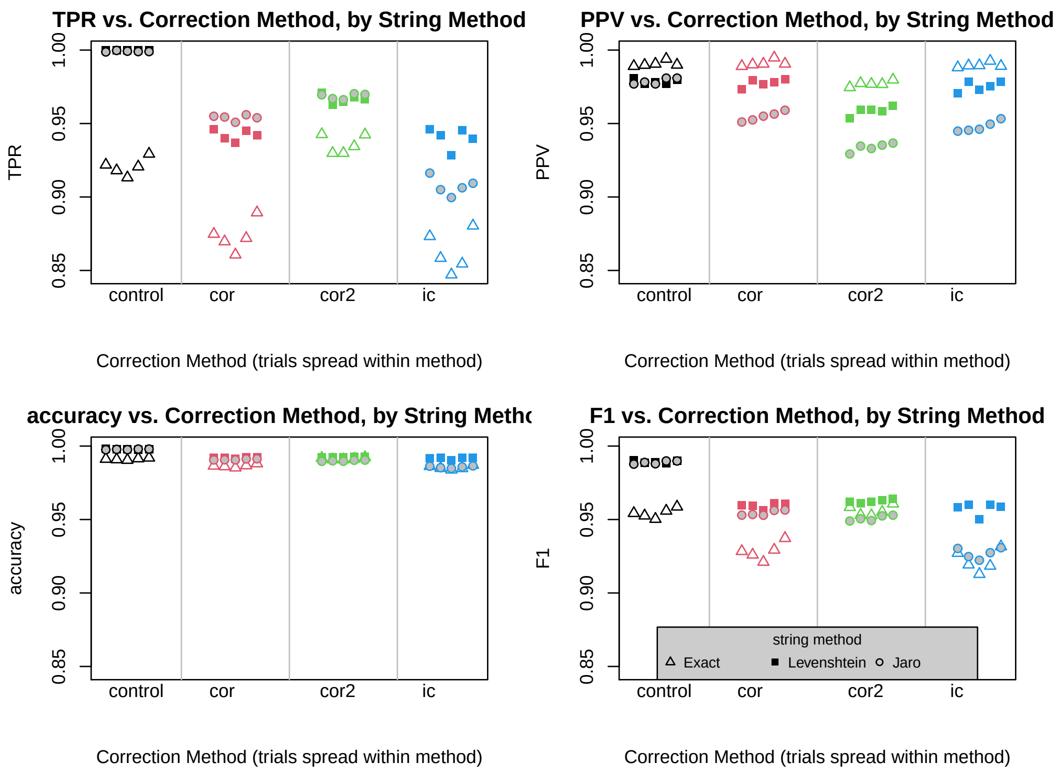

Figure 5: Achieved accuracy-related rates for Experiment 1, dataset 9

simultaneously increased. Because of this inverse relationship, this has a minimal impact on overall F1-score, except in our most extreme messification case ($p_{\text{Noise}} = 0.5$, $p_{\text{Remove}} = 0.3$, $N_D = 4$).

Our variant, ((J, *cor2*), Figure 7) is similarly impacted in both PPV and TPR. However, in many cases, the PPV does not reach the theoretical threshold, but the TPR does not decrease as much as in the Noren et al. method. Because of this smaller change in the TPR with the most extreme messification case ($p_{\text{Noise}} = 0.5$, $p_{\text{Remove}} = 0.3$, $N_D = 4$), this variant approach begins to exceed the Noren et al. F1-scores. This change in relative rates may suggest that as the characteristics of the messification change, the most appropriate method for detecting duplicates may also change. In addition, the more computationally efficient methods (i.e., those without (IC)) may actually perform better in certain data deduplication tasks. Note, some characteristics of F1-score variation may also be a product of these alternative distance and correction methods leading to varying degrees of deviance from the theoretical PPV thresholds, which yields a threshold which potentially occupies a different point on their respective precision-recall (PR) curves. As mentioned earlier, evaluating the performance of the threshold method is an area for future research.

5. Discussion

In general, with regards to duplication scoring methods, there are a few ways the hit-miss algorithm could be used in practice. For example, in a live database, it can be employed to determine if a newly added record or entity is actually new or a duplicate of an existing

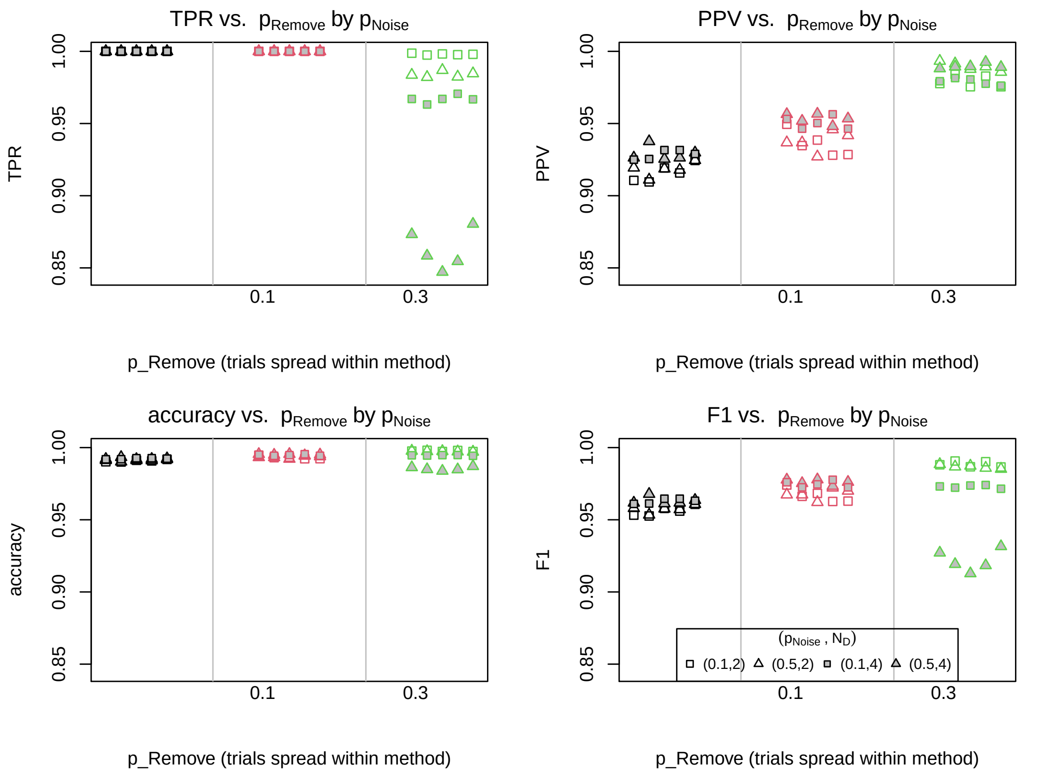

Figure 6: Achieved accuracy-related Rates for Experiment 2, Noren et al. [20] method only (i.e., (exact, *IC*)), shared vertical axis-scale. Separated horizontal axis sections and outline color and separated horizontal axis sections corresponds to p_{Remove}, shape corresponds to p_{Noise}, and shading corresponds to N_D. The vertical axis of the plot corresponds to the particular accuracy-related rate indicated in the axis label, and the rate type varies across plots within this figure. In this plot, the same limits are used for the vertical axis to allow comparability across rate types (plots).

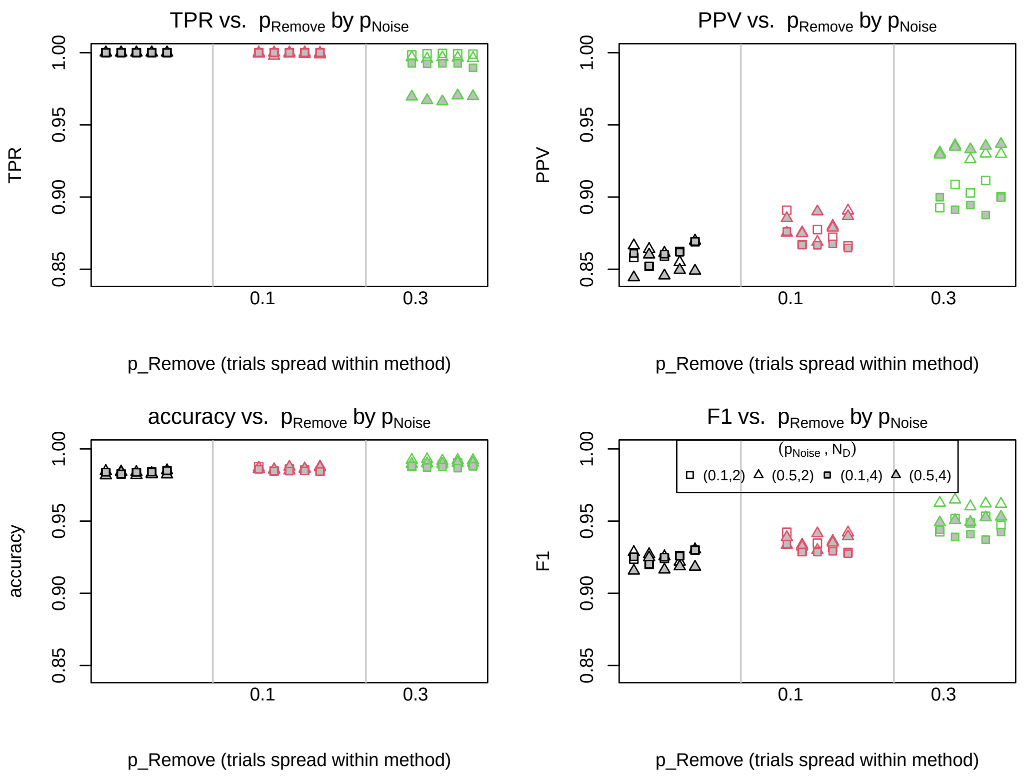

Figure 7: Achieved accuracy-related rates for Experiment 2, our method only (i.e., (J, *cor2*)).

entry. In this scenario, the current database could be used as training data to develop the ensemble, and a threshold could be determined from a held out, more recent subset of added data. Given a new record, the hit-miss algorithm could identify if the new record has a pairwise score with any existing record above the threshold score, indicating that it is likely a duplicate.

Alternatively, for a database with existing duplicates, a practitioner could first manually find some matching records in a subset of the database to develop a training set and determine a threshold. This manual method could be any other duplication detection technique that may work well but doesn't scale well to the size of the database. Then, additional likely duplicates could be identified by applying the hit-miss algorithm on the remaining portion of the database. An alternative to scoring with a match/nonmatch threshold is to score many pairs of records (possibly blocking to focus on more likely matches) and use these scores as weighted edges in a graph between records as vertices. Then clustering can be performed on the resulting graph to look for similar record groupings. Depending on how restrictive the clustering algorithm used is, a cluster could correspond to a single true event or a group of similar events. This method reduces the matches required to be found manually for training.

While the Noren et al. hit-miss model provides an effective means to identify these duplicates, there may be certain situations where alternative variants may perform better, for example, when a large portion of the variability in matches is a result of specific differences in character-based fields. Furthermore, while the IC correction factor introduced by Noren et al. fulfills a similar purpose for data with correlated fields, it adds a potentially prohibitive computational cost. Here, we introduce variants that may, in some situations, perform equal to or better than the Noren et al. model while reducing the computational burden. Ultimately, the choice of deduplication method may depend on the application, characteristics of the database, and requirements for computational efficiency.

Acknowledgements

Funding for this research was awarded through the Navy Innovative Science and Engineering (NISE) Program at the Naval Information Warfare Center Atlantic.

References

[1] D. Brizan and A. Tansel, “A survey of entity resolution and record linkage methodologies,” *Communications of the IIMA*, vol. 6, no. 11, 2006.

[2] A. Elmagarmid, P. Ipeirotis, and V. Verykios, “Duplicate record detection: a survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 1–16, 2007.

[3] P. Christen, “A survey of indexing techniques for scalable record linkage and deduplication,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 9, pp. 1537–1555, 2012.

[4] L. Getoor and A. Machanavajjhala, “Entity resolution: theory, practice, and challenges,” *Proceedings of the VLDB Endowment*, vol. 5, no. 12, 2012.

[5] J. Malhotra and J. Bakal, “A survey and comparative study of data deduplication techniques,” *IEEE International Conference on Pervasive Computing (ICPC)*, 2015.

[6] N. Mandagere, P. Zhou, M. Smith, and S. Uttamchandani, “Demystifying data deduplication,” *Middleware*, Dec 2008.

[7] Q. He, Z. Li, and X. Zhang, “Data deduplication techniques,” *Int’l. Conf. on Future Information Technology and Management Engineering*, 2010.

[8] A. Monge and C. Elkan, “An efficient domain-independent algorithm for detecting approximately duplicate database records,” *Research Issues on Data Mining Knowledge Discovery*, 1997.

[9] S. Sarawagi and A. Bhamidipaty, “Interactive deduplication using active learning,” in *8th ACM Internation Conference on Knowledge Discovery and Data Mining*, (New York, NY), pp. 269–278, ACM Press, 2002.

[10] M. Bilenko and R. Mooney, “Adaptive duplicate detection using learning string similarity measures,” in *9th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 39–48, ACM Press, 2003.

[11] H. Dunn, “Record linkage,” *American Journal of Public Health and the Nations Health*, vol. 36, no. 12, pp. 1412–1416, 1946.

[12] J. Marshal, “Canada’s national vital statistics index,” *Pop. Studies*, vol. 1-2, pp. 204–211, 1947.

[13] H. Newcombe, J. Kennedy, S. Axford, and A. James, “Automatic linkage of vital records,” *Science*, vol. 130, no. 3381, pp. 954–959, 1959.

[14] H. Newcombe and J. Kennedy, “Record linkage: making maximum use of discriminating power of identifying information,” *Communications of the ACM*, vol. 5, no. 11, pp. 563–566, 1962.

[15] B. Tepping, “A model for optimum linkage of records,” *JASA*, vol. 63, pp. 1321–1332, 1968.

[16] W. Winkler, “Overview of record linkage and current research directions,” tech. rep., Technical Report, World Bureau of the Census, 2006.

[17] I. Fellegi and A. Sunter, “A theory for record linkage,” *Journal of the American Statistical Association*, vol. 64, no. 328, pp. 1183–1210, 1969.

[18] P. Christen, *Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection*. Springer Science & Business Media, 2012.

[19] J. Copas and F. Hilton, “Record linkage: statistical models for matching computer records,” *J. R. Stat Soc: Sers A*, vol. 153, no. 3, pp. 287–320, 1990.

[20] G. Noren, R. Orre, and A. Bate, “A hit-miss model for duplicate detection in the WHO drug safety database,” *Data Mining and Knowledge Discovery*, 2007. This article provides models for calculating match and mismatch probabilities.

[21] A. Tancredi, R. Steorts, and B. Liseo, “A unified framework for de-duplication and population size estimation,” *Bayesian Analysis*, vol. 15, no. 2, pp. 633–682, 2019.

[22] G. Recchia and M. Louwerse, “A comparison of string similarity measures for toponym matching,” *ACM SIGSPATIAL COMP’13*, 2013.

[23] V. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” *Soviet Physics Doklady*, vol. 10, pp. 707–710, 1965.

[24] M. Jaro, “Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida,” *Journal of American Statistical Association*, vol. 89, no. 414-420, 1989.

[25] Kaggle, “Crimes in boston dataset,” tech. rep., 2018. [dataset, www.kaggle.com/AnalyzeBoston/crimes-in-boston].

[26] R. Steorts, “Entity resolution with empirically motivated priors,” *Bayesian Analysis*, vol. 10, no. 4, pp. 849–875, 2015.

A. Mixture Distributions and Accuracy-Related Rates

Applying the law of total probability, the cdf for a K-component mixture random variable can be represented as a weighted sum of its individual components:

$$F_{\text{Mix}}(x) = P(X \le x) = \sum_{k=1}^{K} P(X \le x | X_k) P(X_k) = \sum_{k=1}^{K} F_{X_k}(x) \pi_{X_k}, \text{ where } \sum_{k=1}^{K} \pi_{X_k} = 1.$$

where π_{X_k} represents component prevalence. The correspond probability distribution function (f_{Mix}) for a K-component mixture random variable is given by

$$f_{\text{Mix}}(x) = \sum_{k=1}^{K} \pi_{X_1} f_{X_k}(x).$$

A two-component mixture distributions are a simple case with $K = 2$.

$$f_{\text{Mix}}(x) = \pi_{X_1} f_{X_1}(x) + \pi_{X_2} f_{X_2}(x),$$

which can also be represented in terms of majority and minority components (A and B, respectively):

$$f_{\text{Mix}}(x) = \pi_A f_A(x) + \pi_B f_B(x).$$

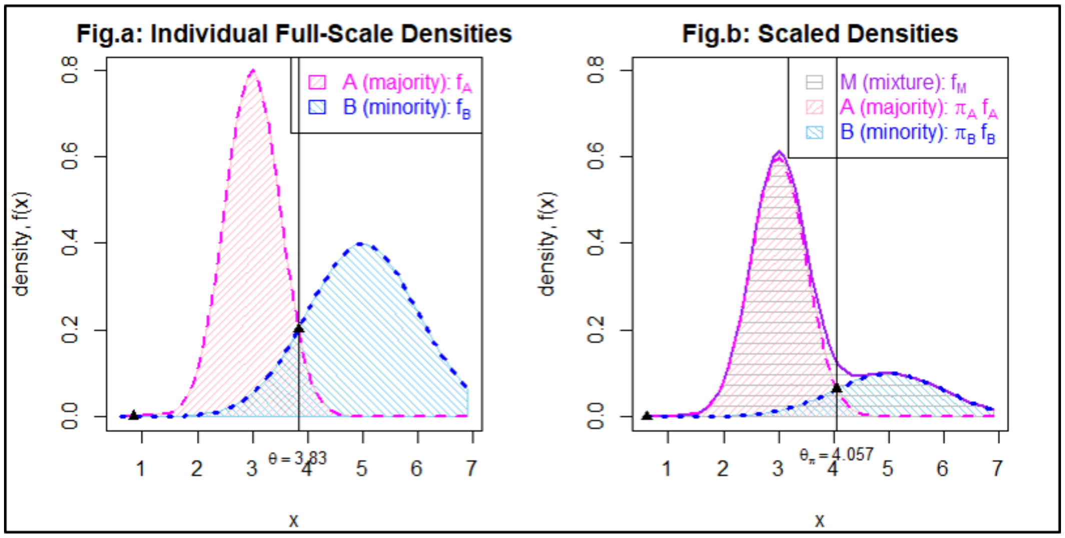

Figure 8: Illustration of individual component probability density functions (f_A and f_B for majority and minority components, respectively, with component mixture proportions π_A and π_B, respectively) and their proportional contribution to a two-component mixture distribution.

Table 5: Equations for Achieved and Theoretical Accuracy-Related Rates. As a matter of notation, FN represents false negatives, FP represents false positives, TN represents true negatives, and TP represents true positives; and $n_{<type>}$ denotes a count for each of these, where $n = n_{TP} + n_{FP} + n_{TN} + n_{FN}$. Predictions are indicated with hats (e.g., $\widehat{\text{match}}$). In the duplicate detection problem, we consider pairs of duplicates records to be positives and pairs of nonduplicate records to be negatives. We use F_X to denote the cumulative distribution of X ($F_X(x) = P(X \leq x)$, and S_X to denote the survival function ($S_X(x) = P(X > x) = 1 - F_X(x)$). θ denotes the treshold above which a record pair's score would be classified as a "match".

Rate Type	**Achieved Rates**	**Theoretical Rates** (for classifying record pairs)
FNR	$P(\widehat{-}\|+) = \frac{n_{FN}}{n_{FN}+n_{TP}}$	$P\left(\widehat{\text{non}}\|\text{match}\right) = F_{\text{match}}(\theta)$
FDR	$P(-\|\widehat{+}) = \frac{n_{FP}}{n_{TP}+n_{FP}}$	$P\left(\text{non}\|\widehat{\text{match}}\right) = \frac{\pi_{\text{non}} S_{\text{non}}(\theta)}{S_{\text{Mix}}(\theta)}$
FOR	$P(+\|\widehat{-}) = \frac{n_{FN}}{n_{TN}+n_{FN}}$	$P\left(\text{match}\|\widehat{\text{non}}\right) = \frac{\pi_{\text{match}} F_{\text{match}}(\theta)}{F_{\text{Mix}}(\theta)}$
FPR	$P(\widehat{+}\|-) = \frac{n_{FP}}{n_{TN}+n_{FP}}$	$P\left(\widehat{\text{match}}\|\text{non}\right) = S_{\text{non}}(\theta)$
NPV	$P(-\|\widehat{-}) = \frac{n_{TN}}{n_{TN}+n_{FN}}$	$P\left(\text{non}\|\widehat{\text{non}}\right) = \frac{\pi_{\text{non}} F_{\text{non}}(\theta)}{F_{\text{Mix}}(\theta)}$
PPV	$P(+\|\widehat{+}) = \frac{n_{TP}}{n_{TP}+n_{FP}}$	$P\left(\text{match}\|\widehat{\text{match}}\right) = \frac{\pi_{\text{match}} S_{\text{match}}(\theta)}{S_{\text{Mix}}(\theta)}$
TPR	$P(\widehat{+}\|+) = \frac{n_{TP}}{n_{TP}+n_{FN}}$	$P\left(\widehat{\text{match}}\|\text{match}\right) = S_{\text{match}}(\theta)$
TNR	$P(\widehat{-}\|-) = \frac{n_{TN}}{n_{TN}+n_{FP}}$	$P\left(\widehat{\text{non}}\|\text{non}\right) = F_{\text{non}}(\theta)$
Accuracy	$P(\text{correct classification}) = \frac{n_{TP}+n_{TN}}{n}$	$\pi_{\text{non}} F_{\text{non}}(\theta) + \pi_{\text{match}} S_{\text{match}}(\theta)$
F1-score	$\frac{2}{1/\text{TPR}+1/\text{PPV}} = \frac{n_{TP}}{n_{TP}+(n_{FN}+n_{FP})/2}$	$\frac{2\pi_{\text{match}} S_{\text{match}}(\theta)}{\pi_{\text{match}}+S_{\text{Mix}}(\theta)}$

A.1 Considerations for Selecting Accuracy-Related Rates

In practice, some applications of record linkage are more interested in one optimizing a particular rate type versus another. For example, sensitivity (TPR) and specificity (TNR) are commonly used in the evaluation of medical tests, and precision (PPV), recall (same as sensitivity), accuracy, and F1-score are commonly used in evaluating machine learning algorithm performance. And preferred presentation of results also differs by application and technical community (e.g., ROC Curves and precision-recall plots are common visualizations within some fields).

From the various rate types that can be calculated (e.g., such as those listed in Table 5), some rate types depend on the prior match proportion (π_{match}), whereas others are free of the match proportion. Outisde of application-area preferred or required rates, this dependence may be important to consider. In particular, the rate type used in [20] depends on the prior match proportion, and we found that if the assumed prior match proportion designated a majority class that differed from the majority class in the data to which the estimated threshold would be applied, then achieved rate value (actual rates when the threshold was applied to the test data) that depended on this prior match proportion may vary considerable from the desired rate value input into the threshold estimation algorithm. For example, if we assumed $\pi_{\text{match}} = 0.1$ but our testing data had more matches than nonmatches, then the desired rate types that depend on match proportion (e.g., FOR) was much different than if we had testing data with majority nonmatches.

B. Linear Models for Analyzing Results from Experiments 1 and 2

Linear models (LMs) could be used to enable a systematic analysis of results from our experiments (Experiments 1 and 2). In particular, our experiments are aimed at assessing the impact of specific experimental factors on hit-miss algorithm performance. Toward this end, we developed LMs models to relate accuracy-related rates from our experiments to experimental factors. A general equation for an LM that relates a particular rate type ($Rate$; e.g., true positive rate) to two primary factors of interest is given by

$$Rate_{i,j,k} = A_i + B_j + (AB)_{i,j} + \varepsilon_{i,j,k}, \quad \text{where } \varepsilon_{i,j,k} \overset{iid}{\sim} \text{Normal}\left(0, \sigma_E^2\right). \tag{7}$$

A and B denote the fixed effects of factors one and two, respectively, and their corresponding levels are indexed by subscripts i and j, respectively. The interaction is denoted by AB, and the trial index is represented by subscript k. The unexplained error term is represented by ε, and errors are assumed to be independent and identically distributed (iid) with zero mean and constant error variance.[22] This two-factor LM can be generalized to k factors (e.g., for a three-factor experiment as in Experiment 2).

The corresponding fitted linear regression equation that follows is given by Equation 8:

$$\widehat{Rate}_{i,j,k} = \widehat{A}_i + \widehat{B}_j + \widehat{(AB)}_{i,j}, \tag{8}$$

where $\widehat{}$ notation indicates estimates. Typical results from such fitted linear regression equations include summary tables with regression coefficient values, analysis of variance (ANOVA) tables, and/or model fit statistics.

From our experimental results, we fit LMs to the true positive rate (TPR). We chose TPR based on common practice from the statistics, record linkage, and machine learning literature [10, 26]. Applying the estimated threshold, the following achieved rate types were recorded in our experiments: true positive rate (TPR), positive predictive value (PPV), accuracy, and F1-score.[23]

B.1 LM for Experiment 1

We developed an LM (LM) to relate the achieved TPR from our two-factor algorithm performance experiment to two primary factors of interest (correction method and string method):

$$TPR^{(1)}_{i,j,k} = A_i + B_j + (AB)_{i,j} + \varepsilon_{i,j,k}. \tag{9}$$

The fixed effect of the field dependence correction method is represented by A, and its levels are indexed by i. The fixed effect of the string handling method is denoted by B, and its levels indexed by j. The interaction between correction method and string method is AB, and the trial index is k. The (1) superscript is used to distinguish this LM from the LM that we used to evaluate results for Experiment 2. The corresponding estimated regression equation is given by

$$\widehat{TPR}^{(1)}_{i,j,k} = \widehat{A}_i + \widehat{B}_j + \widehat{(AB)}_{i,j}. \tag{10}$$

Note that a separate model was fit to each of dataset 3 and dataset 9. Summary tables for Experiment 1's fitted models are provided in Table 6.

B.2 LM for Experiment 2

Similarly to Experiment 1, we fit an LM to relate the true positive rate from Experiment 2 to the three data messification experimental factors:

$$\widehat{TPR}^{(2)}_{i,j,k,l} = \widehat{D}_i + \widehat{E}_j + \widehat{G}_k + \widehat{DE}_{i,j} + \widehat{(DG)}_{i,k} + \widehat{(EG)}_{j,k} + \widehat{(DEG)}_{i,j,k}, \tag{11}$$

where D, E, and G represent the effects of p_{Noise}, p_{Remove}, and N_D, respectively. Corresponding levels for factors D, E, and G, are denoted by subscripts i, j, and k, respectively, and l denotes the trial index. The (2) superscript is used to distinguish this LM from the LM that we used to evaluate results for Experiment 1.

[22]We choose this traditional structure for its simplicity and clarity in explanation. Consideration of the validity of these assumptions and potential improvements to the model is an area for future consideration.

[23]Additional analysis of other rate types in our experimental data is an area for future research.

Table 6: LM Summary Tables for Experiment 1

(Dataset 3)	Estimate	Std. Error	Test Stat (t)	$P(T > \|t\|)$
score type = control	0.9918	0.0006	1655.75	$< 2.2 * 10^{-16}$
score type = cor	0.9847	0.0006	1643.87	$< 2.2 * 10^{-16}$
score type = cor2	0.9968	0.0006	1664.03	$< 2.2 * 10^{-16}$
score type = ID	0.9839	0.0006	1642.59	$< 2.2 * 10^{-16}$
string method = J	0.0081	0.0008	9.55	$1.1 * 10^{-12}$
string method = L	0.0082	0.0008	9.67	$7.6 * 10^{-13}$
(score, string) = (cor, J)	-0.0005	0.0012	-0.43	0.6712
(score, string) = (cor2, J)	-0.0085	0.0012	-7.13	$4.6 * 10^{-9}$
(score, string) = (IC, J)	-0.0216	0.0012	-18.03	$< 2.2 * 10^{-16}$
(score, string) = (cor, L)	-0.0047	0.0012	-3.93	0.0003
(score, string) = (cor2, L)	-0.0084	0.0012	-7.05	$6.2 * 10^{-9}$
(score, string) = (IC, L)	-0.0075	0.0012	-6.24	$1.1 * 10^{-7}$
(Dataset 9)	Estimate	Std. Error	Test Stat (t)	$P(T > \|t\|)$
score type = control	0.9206	0.0029	321.54	$< 2.2 * 10^{-16}$
score type = cor	0.8733	0.0029	305.01	$< 2.2 * 10^{-16}$
score type = cor2	0.9358	0.0029	326.86	$< 2.2 * 10^{-16}$
score type = IC	0.8627	0.0029	301.33	$< 2.2 * 10^{-16}$
string method = J	0.0785	0.0040	19.39	$< 2.2 * 10^{-16}$
string method = L	0.0793	0.0040	19.59	$< 2.2 * 10^{-16}$
(score, string) = (cor, J)	0.0022	0.0057	0.38	0.7022
(score, string) = (cor2, J)	-0.0458	0.0057	-8.00	$2.2 * 10^{-10}$
(score, string) = (IC, J)	-0.0340	0.0057	-5.93	$3.2 * 10^{-7}$
(score, string) = (cor, L)	-0.0107	0.0057	-1.86	0.0690
(score, string) = (cor2, L)	-0.0487	0.0057	-8.50	$4.0 * 10^{-11}$
(score, string) = (IC, L)	-0.0017	0.0057	-0.30	0.7691

Table 7: LM Summary Tables for Experiment 2

(Exact, IC)	Estimate	Std. Error	Test Stat (t)	$P(T > \lvert t \rvert)$
$p_{\text{Noise}} = 0.1$	1.0000	0.0018	548.86	0.0000
$p_{\text{Noise}} = 0.5$	1.0000	0.0018	548.86	0.0000
$p_{\text{Remove}} = 0.1$	-0.0000	0.0026	-0.00	1.0000
$p_{\text{Remove}} = 0.3$	-0.0020	0.0026	-0.78	0.4421
$N_D = 4$	0.0000	0.0026	0.00	1.0000
$(p_{\text{Noise}}, p_{\text{Remove}}) = (0.5, 0.1)$	0.0000	0.0036	0.00	1.0000
$(p_{\text{Noise}}, p_{\text{Remove}}) = (0.5, 0.3)$	-0.0141	0.0036	-3.86	0.0003
$(p_{\text{Noise}}, N_D) = (0.5, 4)$	0.0000	0.0036	0.00	1.0000
$(p_{\text{Remove}}, N_D) = (0.1, 4)$	0.0000	0.0036	0.00	1.0000
$(p_{\text{Remove}}, N_D) = (0.3, 4)$	-0.0311	0.0036	-8.53	0.0000
$(p_{\text{Noise}}, p_{\text{Remove}}, N_D) = (0.5, 0.1, 4)$	-0.0000	0.0052	-0.00	1.0000
$(p_{\text{Noise}}, p_{\text{Remove}}, N_D) = (0.5, 0.3, 4)$	-0.0901	0.0052	-17.49	0.0000

(Jaro, cor2)	Estimate	Std. Error	Test Stat (t)	$P(T > \lvert t \rvert)$
$p_{\text{Noise}} = 0.1$	1.0000	0.0003	3172.86	0.0000
$p_{\text{Noise}} = 0.5$	1.0000	0.0003	3172.86	0.0000
$p_{\text{Remove}} = 0.1$	-0.0000	0.0004	-0.00	1.0000
$p_{\text{Remove}} = 0.3$	-0.0009	0.0004	-2.07	0.0441
$N_D = 4$	-0.0000	0.0004	-0.00	1.0000
$(p_{\text{Noise}}, p_{\text{Remove}}) = (0.5, 0.1)$	-0.0001	0.0006	-0.08	0.9357
$(p_{\text{Noise}}, p_{\text{Remove}}) = (0.5, 0.3)$	-0.0028	0.0006	-4.38	0.0001
$(p_{\text{Noise}}, N_D) = (0.5, 4)$	0.0000	0.0006	0.00	1.0000
$(p_{\text{Remove}}, N_D) = (0.1, 4)$	0.0000	0.0006	0.00	1.0000
$(p_{\text{Remove}}, N_D) = (0.3, 4)$	-0.0072	0.0006	-11.35	0.0000
$(p_{\text{Noise}}, p_{\text{Remove}}, N_D) = (0.5, 0.1, 4)$	-0.0012	0.0009	-1.32	0.1929
$(p_{\text{Noise}}, p_{\text{Remove}}, N_D) = (0.5, 0.3, 4)$	-0.0206	0.0009	-23.12	0.0000