UCLA
UCLA Previously Published Works

Title
Unsupervised record matching with noisy and incomplete data

Permalink

Journal

ISSN

Authors
van Gennip, Yves
Hunter, Blake
Ma, Anna
et al.

Publication Date

DOI

Unsupervised record matching with noisy and incomplete data

Yves van Gennip · Blake Hunter · Anna Ma · Daniel Moyer ·
Ryan de Vera · Andrea L. Bertozzi

arXiv:1704.02955v1 [cs.DB] 10 Apr 2017

Abstract We consider the problem of duplicate detection: given a large data set in which each entry has multiple attributes, detect which distinct entries refer to the same real world entity. Our method consists of three main steps: creating a similarity score between entries, grouping entries together into 'unique entities', and refining the groups. We compare various methods for creating similarity scores, considering different combinations of string matching, term frequency-inverse document frequency methods, and n-gram techniques. In particular, we introduce a vectorized soft term frequency-inverse document frequency method, with an optional refinement step.

We test our method on the Los Angeles Police Department Field Interview Card data set, the Cora Citation Matching data set, and two sets of restaurant review data. The results show that in certain parameter ranges soft term frequency-inverse document frequency methods can outperform the standard term frequency-inverse document frequency method; they also confirm that our method for automatically determining the number of groups typically works well in many cases and allows for accurate results in the absence of a priori knowledge of the number of unique entities in the data set.

Y. van Gennip
University of Nottingham
E-mail: Y.VanGennip@nottingham.ac.uk

B. Hunter
Claremont McKenna College
E-mail: bhunter@cmc.edu

A. Ma
Claremont Graduate University
E-mail: anna.ma@cgu.edu

D. Moyer
University of Southern California
E-mail: moyerd@usc.edu

R. de Vera
formerly California State University, Long Beach
E-mail: ryan.devera.03@gmail.com

A. L. Bertozzi
University of California, Los Angeles
E-mail: bertozzi@math.ucla.edu

1 Introduction

Fast methods for matching records in databases that are similar or identical have growing importance as database sizes increase [37,39,11,21,1]. Slight errors in observation, processing, or entering data may cause multiple unlinked nearly duplicated records to be created for a single real world entity. Furthermore, records are often made up of multiple attributes, or fields; a small error or missing entry for any one of these fields could cause duplication.

For example, one of the data sets we consider in this paper is a database of personal information generated by the Los Angeles Police Department (LAPD). Each record contains information such as first name, last name, and address. Misspellings, different ways of writing names, and even address changes over time, can all lead to duplicate entries in the database for the same person.

Duplicate detection problems do not scale well. The number of comparisons which are required grows quadratically with the number of records, and the

number of possible subsets grows exponentially. Unlinked duplicate records bloat the storage size of the database and make compression into other formats difficult. Duplicates also make analyses of the data much more complicated, much less accurate, and may render many forms of analyses impossible, as the data is no longer a true representation of the real world. After a detailed description of the problem in Section 2 and a review of previous methods in Section 3, we present in Section 4 a vectorized *soft term frequency-inverse document frequency* (soft TF-IDF) solution for string and record comparison. In addition to creating a vectorized version of the soft TF-IDF scheme we also present an automated thresholding and refinement method, which uses the computed soft TF-IDF similarity scores to cluster together likely duplicate. In Section 5 we explore the performances of different variations of our method on four duplicates containing text databases.

2 Terminology and problem statement

We define a data set D to be an $n \times a$ array where each element of the array contains a string (possibly the empty string). We refer to a column as a *field*, and denote the kth field c^k. A row is referred to as a *record*, with r_i denoting the ith record of the data set. An element of the array is referred to as an *entry*, denoted $e_{i,j}$ (referring to the ith entry in the jth field). Each entry can contain multiple features where a *feature* is a string of characters. There is significant freedom in choosing how to divide the string in entry $e_{i,j}$ into multiple features. In this paper in our implementations of soft TF-IDF (Section 3.3), we compare two different methods: (1) cutting the string at white spaces and (2) dividing the string into *N-grams*. For example, consider an entry $e_{i,j}$ containing the string "Albert Einstein". Following method (1) this entry has two features: "Albert" and ''Einstein". Method (2), the N-gram representation, creates features $f_1^k, \ldots, f_L^k$, corresponding to all possible substrings of $e_{i,j}$ containing N consecutive characters (if an entry contains N characters or fewer, the full entry is considered to be a single token). Hence L is equal to the length of the string minus $(N-1)$. In our example, if we use $N=3$, $e_{i,j}$ has 13 features. Ordered alphabetically (with white

space " " preceding "A"), the features are

$$f_1^k = \text{`` Ei''}, \ f_2^k = \text{``Alb''}, \ f_3^k = \text{``Ein''}, \ f_4^k = \text{``ber''},$$
$$f_5^k = \text{``ein''}, \ f_6^k = \text{``ert''}, \ f_7^k = \text{``ins''}, \ f_8^k = \text{``lbe''},$$
$$f_9^k = \text{``nst''}, \ f_{10}^k = \text{``rt ''}, \ f_{11}^k = \text{``ste''}, \ f_{12}^k = \text{``t E''},$$
$$f_{13}^k = \text{``tei''}.$$

In our applications we remove any N-grams that consist purely of white space.

When discussing our results in Figure 3 and Sections 5 and 6 we will specify where we have used method (1) and where we have used method (2), by indicating if we have used *word features* or *N-gram features* respectively.

For each field we create a dictionary of all features in that field and then remove stop words or words that are irrelevant, such as "and", "the", "or", "None", "NA", or " " (the empty string). We refer to such words collectively as "stop words" and to this reduced dictionary as the "set of features", f^k, where:

$$f^k := \left(f_1^k, f_2^k, \ldots, f_{m-1}^k, f_m^k\right),$$

for m features. The dictionary represents an ordered set of unique features found in field c^k.

Note that m, the number of features in the dictionary, depends on k, since a separate dictionary is constructed for each field. To keep the notation as simple as possible, we will not make this dependence explicit in our notation. Since, in this paper, m is always used in the context of a given, fixed k, this should not lead to confusion.

We will write $f_j^k \in e_{i,k}$ if the entry $e_{i,k}$ contains the feature f_j^k. Multiple copies of the same feature can be contained in any given entry. This will be explored further in Section 3.2. Note that an entry can be "empty" if it only contains stop words, since no features in the dictionary can represent it.

We refer to a subset of records as a *cluster* and denote it $R = \{r_{t_1}, \ldots, r_{t_p}\}$ where each $t_i \in \{1, 2, \ldots n\}$ is the index of a record in the data set.

The duplicate detection problem can then be stated as follows: given a data set containing duplicate records, find clusters of records that represent a single entity, i.e., the sets of records that are duplicates of each other. Duplicate records, in this sense, are not identical records but 'near identical' records. They are allowed to vary due to spelling errors or missing entries.

3 Existing methods

Numerous algorithms for duplicate detection already exist, including various probabilistic methods [18], string comparison metrics [17,36], feature frequency methods [29], and hybrid methods [9]. Here we present a brief overview of those methods which are related to the new method we introduce in Section 4. We review both the *Jaro* and *Jaro-Winkler* string metrics, the feature frequency based *term frequency-inverse document frequency* (TF-IDF) method, and the hybrid *soft TF-IDF* method.

3.1 Character-based similarity: Jaro and Jaro-Winkler

Typographical variations are a common cause of duplication among string data, and the prevalence of this type of error motivates string comparison as a method for duplicate detection. The Jaro distance [17] was originally devised for duplicate detection in government census data and modified by Winkler [36] to give more favorable similarities to strings with matching prefixes. This latter variant is now known as the Jaro-Winkler string metric and has been found to be comparable empirically with much more complex measures [9]. Despite their names, neither the Jaro distance, nor the Jaro-Winkler metric, are in fact distances or metrics in the mathematical sense, since they do not satisfy the triangle inequality, and exact matches have a score of 1, not 0. Rather, they are similarity scores.

To define the Jaro-Winkler metric, we must first define the Jaro distance. For two features f_i^k and f_j^k, we define the *character window size*

$$W_{i,j} := \left\lfloor \frac{\min(|f_i^k|, |f_j^k|)}{2} \right\rfloor,$$

where $|f_i^k|$ is the length of the string f_i^k, i.e., the number of characters in f_i^k counted according to multiplicity. The lth character of the string f_i^k is said to *match* the l'^{th} character of f_j^k, if both characters are identical and $l - W_{i,j} \leq l' \leq l + W_{i,j}$. Let M be the number of characters in string f_i^k that match with characters in string f_j^k (or, equivalently, the number of characters in f_j^k that match with characters in f_i^k), let $(a_1, \ldots, a_M)$ be the matched characters from f_i^k in the order they appear in the string f_i^k, and let $(b_1, \ldots, b_M)$ be the matched characters from f_j^k in order. Then t is defined to be half the number of *transpositions* between f_i^k and f_j^k, i.e.,

half the number of indices $l \in \{1, \ldots, M\}$ such that $a_l \neq b_l$. For this reason, each such pair (a_l, b_l) can be called a *transposition pair*. Now the Jaro distance [17] $J(f_i^k, f_j^k)$ is defined as

$$J(f_i^k, f_j^k) :=$$
$$\begin{cases} \frac{1}{3}\left(\frac{M}{|f_i^k|} + \frac{M}{|f_j^k|} + \frac{M-t}{M} \right), & \text{if } M \neq 0, \\ 0, & \text{if } M = 0. \end{cases}$$

Figure 1 shows an example of transpositions and matching character pairs.
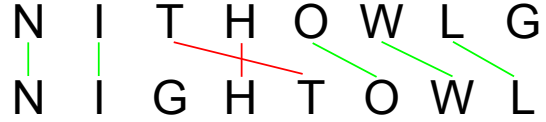

Fig. 1: Example of a comparison of two features in the computation of the Jaro distance, with character window size $W = 4$. The example has 7 matching character pairs, 2 of which are transposition pairs, represented by the red lines. The green lines indicate matching pairs that are not transpositions. Notice that "G" is not considered a matching character as "G" in "NITHOWLG" is the 8th character while "G" in "NIGHTOWL" is the 3rd character, which is out of the $W = 4$ window for this example. Here, $J = \frac{1}{3}(\frac{7}{8} + \frac{7}{8} + \frac{7-1}{7}) = 0.869$.

The Jaro-Winkler metric, $JW(f_i^k, f_j^k)$, modifies the original Jaro distance by giving extra weight to matching prefixes. It uses a fixed prefix factor p to give a higher similarity score to features that match from the beginning for a prefix length $\ell_{i,j}$. Given two features f_i^k and f_j^k, the Jaro-Winkler metric is

$$JW(f_i^k, f_j^k) := J(f_i^k, f_j^k) + p\,\ell_{i,j}\left(1 - J(f_i^k, f_j^k)\right), \quad (1)$$

where $J(f_i^k, f_j^k)$ is the Jaro distance between two features f_i^k and f_j^k, p is a given prefix factor, and $\ell_{i,j}$ is the number of prefix characters in f_i^k that match prefix characters in f_j^k. When we want to stress that, for fixed k, $JW(f_i^k, f_j^k)$ is an element of a matrix, we write $JW_{i,j}^k := JW(f_i^k, f_j^k)$, such that $JW^k \in \mathbb{R}^{m \times m}$.

In Winkler's original work he set $p = 0.1$ and restricted $\ell_{i,j} \leq 4$ (even when prefixes of five or more characters matched) [36]. We follow the same parameter choice and restriction in our applications in this paper. So long as $p\,\ell_{i,j} \leq 1$ for all i, j, the Jaro-Winkler metric ranges from 0 to 1, where 1 indicates

exact similarity between two features and 0 indicates no similarity between two features.

In Figure 1 we have $\ell = 2$, as both features have identical first and second characters, but not a matching third character. This leads to $JW = 0.869 + 0.1 \cdot 2 \cdot (1 - 0.869) = 0.895$.

Because we remove stop words and irrelevant words from our set of features, it is possible for an entry to contain a feature that does not appear in the reduced set of features f^k. If a feature $\tilde{f} \in e_{i,k}$ does not appear in the dictionary f^k, we set, for all $f^k_q \in f^k$, $JW(f^k_q, \tilde{f}) := 0$. We call such features $\tilde{f}$ *null features*.

Algorithm 1: Jaro-Winkler Algorithm

> **Data:** c^k, an $n \times 1$ array of text
> **Result:** $JW^k \in \mathbb{R}^{m \times m}$
> Create the set of features $f^k = (f^k_1, \ldots, f^k_m)$
> **for** *each pair of features* (f^k_i, f^k_j) **do**
> > *Compute Jaro distance* $J_{i,j} = J(f^k_i, f^k_j)$
> > *Compute Jaro-Winkler similarity* $JW^k_{i,j} =$
> > $$\begin{cases} J_{i,j} + p\,\ell_{i,j}(1 - J_{i,j}), & \text{if neither feature} \\ & f^k_i \text{ or } f^k_j \text{ is a} \\ & \text{null feature,} \\ 0, & \text{else} \end{cases}$$
>
> **end**

3.2 Feature-based similarity: TF-IDF

Another approach to duplicate detection, generally used in big data record matching, looks at similar distributions of features across records. This feature based method considers entries to be similar if they share many of the same features, regardless of order; this compensates for errors such as changes in article usage and varying word order (e.g. "The Bistro", "Bistro, The", or "Bistro"), as well as the addition of information (e.g. "The Bistro" and "The Bistro Restaurant").

This form of duplicate detection is closely related to vector space models of text corpora [30], where a body of text is represented as a vector in some word vector space. The dimension of the space is the number of relevant words (other words are assumed to be meaningless), and, for a given record, each element of the vector representation is the frequency with which a word appears in the entry. (It should

be noted that these models also disregard word order.) A more powerful extension of these models is the term frequency-inverse document frequency (TF-IDF) scheme [29]. This scheme reweighs different features based on their frequency in a single field as well as in an entry.

Using the reduced set of features, f^k, we create the term frequency and inverse document frequency matrices. We define the *term frequency matrix* for the kth field, $TF^k \in \mathbb{R}^{n \times m}$, such that $TF^k_{i,j}$ is the number of times the feature f^k_j appears in the entry $e_{i,k}$ (possibly zero). A row of TF^k represents the frequency of every feature in an entry.

Next we define the diagonal *inverse document frequency matrix* $IDF^k \in \mathbb{R}^{m \times m}$ with diagonal elements[1]

$$IDF^k_{i,i} := \log \frac{n}{|\{e \in c^k : f^k_i \in e\}|},$$

where $|\{e \in c^k : f^k_i \in e\}|$ is the number of entries[2] in field c^k containing feature f^k_i, and where n is the number of records in the data set. The matrix IDF^k uses the number of entries in the field containing a feature to give features a more informative weight. The issue when using term frequency only, is that it gives features that appear frequently a higher weight than rare features, which often are empirically more informative than common features. The basic intuition is that a feature that occurs frequently in many entries is not a good discriminator.

The resulting weight matrix for field k is then defined with a logarithmic scaling for the term frequency as[3]

$$TFIDF^k := \log(TF^k + \mathbf{1})IDF^k, \qquad (2)$$

where $\mathbf{1}$ is an $n \times m$ matrix of ones and the log operation acts on each element of $TF^k + \mathbf{1}$ individually. The resulting matrix has dimension $n \times m$. Finally we normalize each row of $TFIDF^k$ by its ℓ^1 norm[4].

---

[1] We use log to denote the natural logarithm in this paper.

[2] By the construction of our set of features in Section 2, this number of entries is always positive.

[3] Note that, following [9], we use a slightly different logarithmic scaling, than the more commonly used $TFIDF^k_{i,j} = \left(\log(TF^k_{i,j}) + 1\right)IDF^k_{i,i}$, if $TF^k_{i,j} \neq 0$, and $TFIDF^k_{i,j} = 0$, if $TF^k_{i,j} = 0$. This avoids having to deal with the case $TF^k_{i,j} = 0$ separately. The difference between $\log(TF^k_{i,j}) + 1$ and $\log(TF^k_{i,j} + 1)$ is bounded by 1 for $TF^k_{i,j} \geq 1$.

[4] Here we deviate from [9], in which the authors normalize by the ℓ^2 norm. We do this so that later in equation (3), we can guarantee that the soft TF-IDF values are upper bounded by 1.

Each entry $TFIDF_{i,j}^k$ represents the weight assigned to feature j in field k for record i. Note that each entry is nonnegative.

Algorithm 2: TF-IDF Algorithm

Data: c^k, an $n \times 1$ array of text
Result: $TFIDF^k \in \mathbb{R}^{n \times m}$
Create the set of features $f^k = (f_1^k, \ldots, f_m^k)$
for *each pair of features (f_i^k, f_j^k)* **do**
$\quad$ | $\quad$ *Compute term frequency $TF_{i,j}^k$*
end
for *each feature f_i^k* **do**
$\quad$ | $\quad$ *Compute inverse document frequency $IDF_{i,i}^k$*
end
Initialize $TFIDF^k = \log(TF^k + \boldsymbol{1})IDF^k$
Normalize rows of $TFIDF^k$

3.3 Hybrid similarity: soft TF-IDF

The previous two methods concentrate on two different causes of record duplication, namely typographical error and varying word order. It is easy to imagine, however, a case in which both types of error occur; this leads us to a third class of methods which combine the previous two. These *hybrid methods* measure the similarity between entries using character similarity between their features as well as weights of their features based on importance. Examples of these hybrid measures include the extended Jacard similarity and the Monge-Elkan measure [25]. In this section we will discuss another such method, soft TF-IDF [9], which combines TF-IDF with a character similarity measure. In our method, we use the Jaro-Winkler metric, discussed above in Section 3.1, as the character similarity measure in soft TF-IDF.

For $\theta \in [0,1)$, let $S(\theta, e_{i,k}, e_{j,k})$ be the set of feature pairs (f_p^k, f_q^k) such that $f_p^k \in e_{i,k}$, $f_q^k \in e_{j,k}$, and $JW(f_p^k, f_q^k) > \theta$, where JW is the Jaro-Winkler similarity metric from (1). The soft TF-IDF similarity between two entries $e_{i,k}$ and $e_{j,k}$ in field c^k is defined as

$$sTFIDF_{i,j}^k := \sum_{\substack{p,q \text{ such that} \\ (f_p^k, f_q^k) \in S(\theta, e_{i,k}, e_{j,k})}} TFIDF_{i,p}^k TFIDF_{j,q}^k JW(f_p^k, f_q^k). \quad (3)$$

The parameter θ allows for stronger control over the similarity of features, removing entirely pairs that do not have Jaro-Winkler similarity above a certain threshold. For the results presented in this paper, we set $\theta = 0.90$.

Note from (3) that for all i, j, and k, we have $sTFIDF_{i,j}^k \in [0,1]$. However, we do not necessarily have that $sTFIDF_{i,i}^k = 1$, even though this might be expected to hold for a similarity measure. Luckily, these diagonal elements of $sTFIDF^k$ will not be relevant in our method. For definiteness and computational ease[5], we redefine, for all k and all i, $sTFIDF_{i,i}^k := 1$.

In practice, this method's computational cost is greatly reduced by vectorization. Let $M^{k,\theta} \in \mathbb{R}^{m \times m}$ be the Jaro-Winkler similarity matrix defined by

$$M_{p,q}^{k,\theta} := \begin{cases} JW(f_p^k, f_q^k), & \text{if } JW(f_p^k, f_q^k) \geq \theta, \\ 0, & \text{if } JW(f_p^k, f_q^k) < \theta. \end{cases}$$

The soft TF-IDF similarity for each (i,j) pairing can then be computed as

$$sTFIDF_{i,j}^k = \sum_{p,q=1}^m \left[\left(TFIDF_i^{k^T} TFIDF_j^k \right) * M^{k,\theta} \right]_{p,q},$$

where $TFIDF_i^k$ denotes the ith row of the TF-IDF matrix of field c^k and $*$ denotes the Hadamard product, or element-wise product. We can further simplify this using tensor products. Let $\overline{M}^{k,\theta}$ denote the vertical concatenation of the rows of $M^{k,\theta}$.

$$\overline{M}^{k,\theta} = \begin{bmatrix} M_1^{k,\theta T} \\ M_2^{k,\theta T} \\ \vdots \\ M_m^{k,\theta T} \end{bmatrix}$$

where $M_i^{k,\theta}$ is the ith row of $M^{k,\theta}$. We then have:

$$sTFIDF_{i,j}^k = (TFIDF_i^k \otimes TFIDF_j^k) * \overline{M}^{k,\theta},$$

where $\otimes$ is the Kronecker product.

Finally we set (redefine) the diagonal elements $sTFIDF_{i,i}^k = 1$.

The similarity matrices produced by the TF-IDF and Jaro-Winkler are typically sparse. This sparsity can be leveraged to reduce the computational cost of the soft TF-IDF method as well.

---

[5] The values of the diagonal elements are not relevant theoretically, because any record is always a 'duplicate' of itself and trivially will be classified as such, i.e. each record will be clustered in the same cluster as itself. However, if the diagonal elements do not have value 1, care must be taken that this does not influence the numerical implementation.

Algorithm 3: soft TF-IDF Algorithm

Data: $JW^k \in \mathbb{R}^{m \times m}$, $TFIDF^k \in \mathbb{R}^{n \times m}$, θ
Result: $sTFIDF^k \in \mathbb{R}^{n \times n}$
Create the set of features $f^k = (f_1^k, \ldots, f_m^k)$
for *each pair of features* (f_i^k, f_j^k) **do**
 | *Compute the thresholded Jaro-Winkler matrix*
 | $M_{i,j}^{k,\theta}$
end
Vertically concatenate rows of $M^{k,\theta}$:
$\overline{M}^{k,\theta} = [M_1^{k,\theta\,T}; M_2^{k,\theta\,T}; \ldots; M_m^{k,\theta\,T}]$
for *each pair of entries* $(e_{i,k}, e_{j,k})$ *in field* c^k **do**
 | *Compute soft TF-IDF:*
 | $sTFIDF_{i,j}^k = (TFIDF_i^k \otimes TFIDF_j^k) * \overline{M}^{k,\theta}$
end
Set the diagonal elements $sTFIDF_{i,i}^k = 1$

The soft TF-IDF scores above are defined between entries for a single field. For each pair of records we produce a *composite similarity score* $ST_{i,j}$ by adding their soft TF-IDF scores over all fields:

$$ST := \sum_{k=1}^{a} sTFIDF^k. \tag{4}$$

Hence, $ST \in \mathbb{R}^{n \times n}$ with $ST_{i,j}$ the score between the ith and j^{th} records. Remember that a is the number of fields in the data set. Each composite similarity score $ST_{i,j}$ is a number between 0 and a.

For some applications it may be desirable to let some fields have a greater influence on the composite similarity score than others. In the above formulation this can easily be achieved by replacing the sum in (4) by a weighted sum:

$$ST_w := \sum_{k=1}^{a} w_k \, sTFIDF^k,$$

for positive weights $w_k \in \mathbb{R}$, $k \in \{1, \ldots, a\}$. If the weights are chosen such that $\sum_{k=1}^{a} w_k \leq a$, then the *weighted composite similarity score* ST_w takes values in $[0, a]$, like ST. In this paper we use the unweighted composite similarity score ST.

3.4 Using TF-IDF instead of soft TF-IDF

In our experiments in Section 5 we will also show results in which we use TF-IDF, not soft TF-IDF, to compute similarity scores. This can be achieved in a completely analogous way to the one described in Section 3.3, if we replace (3) by

$$sTFIDF^k := TFIDF^k \big(TFIDF^k\big)^T \in \mathbb{R}^{n \times n},$$

where $TFIDF^k$ is the TF-IDF matrix from (2) and the superscript T denotes the matrix transpose. Note that this is equivalent to setting

$$JW(f_p^k, f_q^k) = \delta_{p,q}$$

in (3), with

$$\delta_{p,q} := \begin{cases} 1, & \text{if } p = q, \\ 0, & \text{otherwise,} \end{cases}$$

the Kronecker delta.

All the other computations from Section 5, in particular the computation of the composite similarity score in (4), then continue as before.

4 New method

We extend the soft TF-IDF method to address two common situations in duplicate detection: missing entries and large numbers of duplicates. For data sets with only one field, handling a missing field is a non-issue; a missing field is irreconcilable, as no other information is gathered. In a multi-field setting, however, we are faced with the problem of comparing partially complete records. Another issue is that a record may have more than one duplicate. If all entries are pairwise similar we can easily justify linking them all, but in cases where one record is similar to two different records which are dissimilar to each other the solution is not so clear cut.

Figure 2 shows an outline of our method. First we use TF-IDF to assign weights to features that indicate the importance of that feature in an entry. Next, we use soft TF-IDF with the Jaro-Winkler metric to address spelling inconsistencies in our data sets. After this, we adjust for sparsity by taking into consideration whether or not a record has missing entries. Using the similarity matrix produced from the previous steps, we threshold and group records into clusters. Lastly, we refine these groups by evaluating how clusters break up under different conditions.

4.1 Adjusting for sparsity

A *missing entry* is an entry that is either entirely empty or one that contains only null features. Here, we assume that missing entries do not provide any information about the record and therefore cannot aid us in determining whether two records should
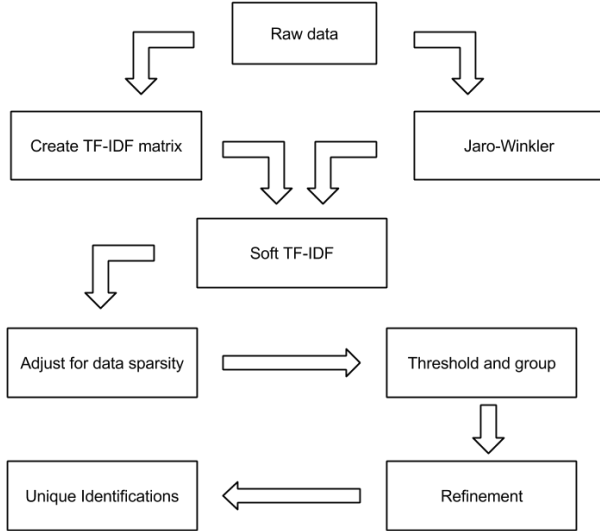
Fig. 2: An outline of our method for duplicate detection

be clustered together (i.e. labeled as probable duplicates). In [35], [36], and [2], records with missing entries are discarded, filled in by human fieldwork, and filled in by an expectation-maximization (EM) imputation algorithm, respectively. For cases in which a large number of entries are missing, or in data sets with a large number of fields such that records have a high probability of missing at least one entry, these first two methods are impractical. Furthermore, the estimation of missing fields is equivalent to unordered categorical estimation. In fields where a large number of features are present (i.e. the set of features is large), the type of estimation by EM scheme becomes computationally intractable [26] [38] [16]. Thus, a better method is required.

Leaving the records with missing entries in our data set, both TF-IDF and Jaro-Winkler remain well defined, allowing soft TF-IDF schemes to proceed. However, because the Jaro-Winkler metric for a null feature and any other feature is 0, the soft TF-IDF score between a missing entry and any other entry is 0. This punishes sparse records in the composite soft TF-IDF matrix ST. Even if two records have the exact same entries in fields where both records do not have missing entries, their missing entries deflate their composite soft TF-IDF similarity. Consider the following example using two records and three fields: ["Joe Bruin", " ", "male"] and ["Joe Bruin', "CA", " "]. The two records are likely to represent a unique entity "Joe Bruin", but the composite soft TF-IDF

score between the two records is on the lower end of the similarity score range (0.33) due to the missing entry in the second field for the first record and the missing entry in the third field for the second record. To correct for this, we take into consideration the number of mutually present (not missing) entries in the same field for two records.

This can be done in a vectorized manner to accelerate computation. Let B be an $n \times a$ binary matrix, where a is the number of fields in the data set, such that

$$B_{i,k} := \begin{cases} 0, & \text{if } e_{i,k} \text{ is a missing entry,} \\ 1, & \text{otherwise.} \end{cases}$$

This is a binary mask of the data set, where 1 denotes a non-missing entry (with or without error), and 0 denotes a missing entry. In the product $BB^T \in \mathbb{R}^{n \times n}$, each $(BB^T)_{i,j}$ is the number of "shared fields" between records r_i and r_j, i.e. the number of fields c^k such that both $e_{i,k}$ and $e_{j,k}$ are non-missing entries. Our "sparsity adjusted soft TF-IDF similarity" is given by

$$adjST := [ST] \oslash (BB^T), \tag{5}$$

where $\oslash$ denotes element-wise division.

Remembering that $JW(f_p^k, f_q^k) = 0$ if f_p^k or f_q^k is a null feature, we see that, if $e_{i,k}$ or $e_{j,k}$ is a missing entry, then the set $S(\theta, e_{i,k}, e_{j,k})$ used in (3) is empty (independent of the choice of θ) and thus $sTFIDF_{i,j}^k = 0$. Hence, we have that, for all i, j ($i \neq j$), $(ST)_{i,j} \in [0, (BB^T)_{i,j}]$ (which refines our earlier result that $(ST)_{i,j} \in [0, a]$) and thus $(adjST)_{i,j} \in [0, 1]^6$.

In particular, in the event that there are records r_i and r_j such that $(BB^T)_{i,j} = 0$, we have $ST_{i,j} = 0$. Hence, if $(BB^T)_{i,j} = 0$ and thus the expression in (5) is not defined for $adjST_{i,j}$, we set $adjST_{i,j} = 0$ instead. In the data sets we will discuss in Section 4, no pair of records was without shared fields, and so (5) suffices for our purposes in this paper.

4.2 Thresholding and grouping

The similarity score $adjST_{i,j}$ gives us an indication of how similar the records r_i and r_j are. If $adjST_{i,j}$ is

---

[6] Note that, since we artificially redefined the inconsequential diagonal entries to be $sTFIDF_{i,i}^k := 1$ in Section 3.3, it could be that $(ST)_{i,i} > (BB^T)_{i,i}$ for some i, in which case we just redefine $(adjST)_{i,i} := 1$ for consistency with the other values. Remember that the diagonal values will play no role in the eventual clustering.

Algorithm 4: Adjusting for Sparsity

Data: $sTFIDF^k \in \mathbb{R}^{n \times n}$ for $k \in \{1, \ldots, a\}$, D an
$\quad\quad\quad n \times a$ array of text
Result: $adjST \in \mathbb{R}^{n \times n}$
for *each entry $e_{i,k}$ in each field c^k of D* **do**
$\quad |\quad$ *Compute $B_{i,k}$*
end
Initialize $ST = \sum_k sTFIDF^k$
Adjust ST for sparsity: $adjST = ST \oslash BB^T$

---

close to 1, then the records are more likely to represent the same entity. Now, we present our method of determining whether a set of records are duplicates of each other based on $adjST$. There exist many clustering methods that could be used to accomplish this goal. For example, [24] considers this question in the context of duplicate detection. For simplicity, in this paper we restrict ourselves to a relatively straightforward thresholding procedure, but other methods could be substituted in future implementations. We call this the *thresholding and grouping step* (TGS).

The method we will present below is also applicable to clustering based on other similarity scores. Therefore it is useful to present it in a more general format. Let $SIM \in \mathbb{R}^{n \times n}$ be a matrix of similarity scores, i.e., for all i, j, the entry $SIM_{i,j}$ is a similarity score between the records r_i and r_j. We assume that, for all $i \neq j$, $SIM_{i,j} = SIM_{j,i} \in [0, a]$[7]. If we use our adjusted soft TF-IDF method, SIM is given by $adjST$ from (5). In Section 4.1 we saw that in that case we even have $SIM_{i,j} \in [0, 1]$.

Let $\tau \in [0, a]$ be a threshold and let S be the *thresholded similarity score matrix* defined for $i \neq j$ as

$$S_{i,j} := \begin{cases} 1, & \text{if } SIM_{i,j} \geq \tau, \\ 0, & \text{if } SIM_{i,j} < \tau. \end{cases}$$

The outcome of our method does not depend on the diagonal values, but for definiteness (and to simplify some computations) we set $S_{i,i} := 1$, for all i. If we want to avoid trivial clusterings (i.e. with all records in one cluster, or with each cluster containing only one record) the threshold value τ must be chosen in the half-open interval

$$\Big(\min_{i,j: j \neq i} SIM_{i,j}, \max_{i,j: j \neq i} SIM_{i,j} \Big].$$

---

[7] We will not be concerned with the diagonal values of SIM, because trivially any record is a 'duplicate' of itself, but for definiteness we may assume that, for all i, $SIM_{i,i} = a$.

Records r_i and r_j $(i \neq j)$ are clustered together (as probable duplicates) if at least one of the following two conditions is satisfied:

1. $S_{i,j} = 1$,
2. there exists a record r_k $(i \neq k \neq j)$ such that $S_{i,k} = 1$ and $S_{j,k} = 1$.

Note that (if we have set $S_{i,i} = 1$ as above) we can combine both conditions into one condition: $(S^2)_{i,j} \geq 1$. The output of the TGS is a clustering of all the records in the data set, i.e. a collection of clusters, each containing one or more records, such that each record belongs to exactly one cluster.

The choice of τ is crucial in the formation of clusters. Choosing a threshold that is too low leads to large clusters of records that represent more than one unique entity. Choosing a threshold that is too high breaks the data set into a large number of clusters, where a single entity may be represented by more than one cluster. Here, we propose a method of choosing τ.

Let $H \in \mathbb{R}^n$ be the $n \times 1$ vector defined by

$$H_i := \max_{j: j \neq i} SIM_{i,j},$$

where the maximum is taken over $j = 1, \ldots, n$. In other words, each element of H is the maximum similarity score $SIM_{i,j}$ between a fixed record and every other record. Now define

$$\tau_H := \begin{cases} \mu(H) + \sigma(H), & \text{if } \mu(H) + \sigma(H) < \max_i H_i, \\ \mu(H), & \text{else,} \end{cases}$$

where $\mu(H)$ is the mean value of H and $\sigma(H)$ is its standard deviation.

In many of our runs (Figure 3a is a representative example), there is a large peak of H values around the mean value $\mu(H)$. Choosing $\tau_H = \mu(H) + \sigma(H)$ will typically place the threshold far enough to the right of this peak so that the records corresponding to this peak do not get clustered together, yet also far enough removed from the maximum value so that more than only the top matches get identified as duplicates. In some cases however, where the distribution of H values has a peak near the maximum value (as, for example, in Figure 3b), the value $\mu(H) + \sigma(H)$ will be larger than the maximum and we chose $\tau_H = \mu(H)$ instead.

It may not always be possible to choose a threshold in such a way that all the clusters generated by our TGS correspond well to sets of duplicates, as the following example, illustrated in Figure 4, shows. We
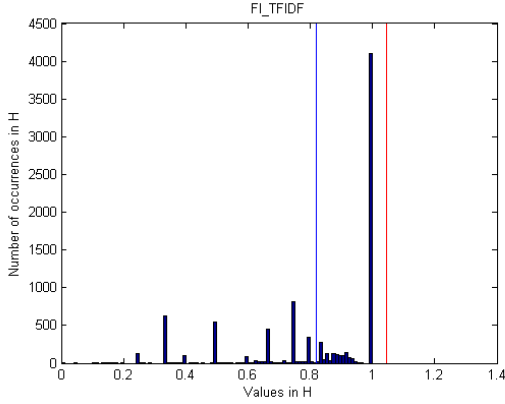
(a) H corresponding to the TF-IDF method (with word feature, without refinement step, see Section 4.3) applied to the FI data set. The red line is the chosen value $\tau_H = \mu(H) + \sigma(H)$; the blue line indicates $\mu(H)$.
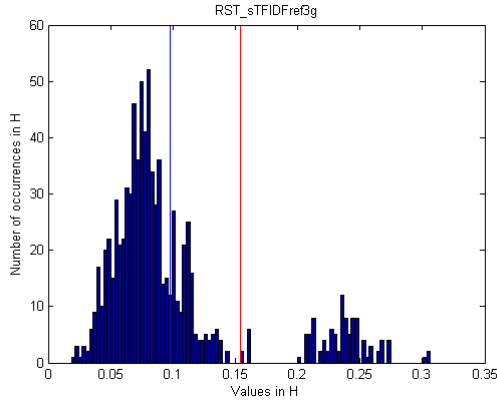
(b) H corresponding to the soft TF-IDF method (with 3-gram features, with refinement, see Section 4.3) applied to the RST data set. The blue line indicates the chosen value $\tau_H = \mu(H)$; the red line indicates $\mu(H) + \sigma(H)$.

Fig. 3: Histograms of H for different methods applied to the FI and RST data sets (see Section 5.1)

consider an artificial toy data set for which we computed the adjusted soft TF-IDF similarity, based on seven fields. We represent the result of the TGS as a graph in which each node represents a record in the data set. We connect nodes i and j ($i \neq j$) by an edge if and only if their similarity score $SIM_{i,j}$ equals or exceeds the chosen threshold value τ. The connected components of the resulting graph then correspond to the clusters the TGS outputs.

For simplicity, the Figure 4 only shows the features of each entry from the first two fields (first name and last name). Based on manual inspection, we declare the ground truth for this example to contain two unique entities: "Joey Bruin" and "Joan

Lurin". The goal of our TGS is to detect two clusters, one for each unique entity. Using $\tau = 5.5$, we find one cluster (Figure 4a). Using $\tau = 5.6$, we do obtain two clusters (Figure 4b), but it is not true that one cluster represents "Joe Bruin" and the other "Joan Lurin", as desired. Instead, one clusters consists of only the "Joey B" record, while the other cluster contains all other records. Increasing τ further until the clusters change, would only result in more clusters, therefore we cannot obtain the desired result this way. This happens because the adjusted soft TF-IDF similarity between "Joey B" and "Joey Bruin" (respectively "Joe Bruin") is less than the adjusted soft TF-IDF similarity between "Joey Bruin" (respectively "Joe Bruin") and "Joan Lurin". To address this issue, we apply a *refinement* to each set of clustered records created by the TGS, as explained in the next section.

The graph representation of the TGS output turns out to be a very useful tool and we will use its language in what follows interchangeably with the 'cluster' language.

Algorithm 5: Thresholding and grouping

Data: $SIM = ST \in \mathbb{R}^{n \times n}$, threshold value τ (manual choice or automatic $\tau = \tau_H$)
Result: a collection of c clusters $\mathcal{C} = \{R_1 \ldots R_c\}$
for *each i* **do**
  | *Initialize $S_{i,i} = 1$*
end
for *each pair of distinct records r_i and r_j* **do**
  | *Compute $S_{i,j}$*
end
for *each pair of distinct records r_i and r_j* **do**
  | If $(S^2)_{i,j} \geq 1$, assign r_i and r_j to the same cluster
end

4.3 Refinement

As the example in Figure 4 has shown, the clusters created by the TGS are not necessarily complete subgraphs: it is possible for a cluster to contain records r_i, r_j for which $S_{i,j} = 0$, while also containing a record r_k such that $S_{i,k} = 1$ and $S_{j,k} = 1$ ($i \neq j \neq k \neq i$). In such cases it is *a priori* unclear if the best clustering is indeed achieved by grouping r_i and r_j together or not. We introduce a way to refine clusters created in the TGS, to deal with situations
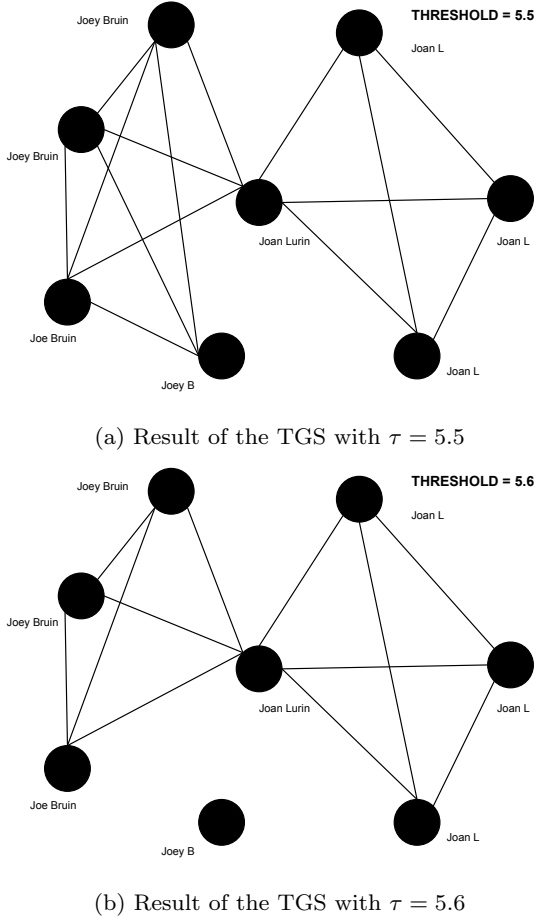
(a) Result of the TGS with $\tau = 5.5$

(b) Result of the TGS with $\tau = 5.6$

Fig. 4: Two examples of clusters created by the TGS applied to an artificial data set, with different threshold values τ

like these. We take the following steps to refine a cluster R:

1. determine whether R needs to be refined by determining the cluster stability with respect to single record removal;
2. if R needs be to refined, remove one record at a time from R to determine the "optimal record" r^* to remove;
3. if r^* is removed from R, find the subcluster that r^* does belong to.

Before we describe these steps in more detail, we introduce more notation. Given a cluster (as determined by the TGS) $R = \{r_{t_1}, \ldots, r_{t_p}\}$ containing p records, the thresholded similarity score matrix of the cluster R is given by the restricted matrix $S|_R \in \mathbb{R}^{p \times p}$ with elements $(S|_R)_{i,j} := S_{t_i, t_j}$. Remember we represent R by a graph, where each node corresponds to a record r_{t_i} and two distinct nodes

are connected by an edge if and only if their corresponding thresholded similarity score $(S|_R)_{i,j}$ is 1. If a record r_{t_i} is removed from R, the remaining set of records is
$R(r_{t_i}) := \{r_{t_1}, \ldots, r_{t_{i-1}}, r_{t_{i+1}}, \ldots, r_{t_p}\}$. We define the *subclusters* $R_1, \ldots R_q$ of $R(r_{t_i})$ as the subsets of nodes corresponding to the connected components of the subgraph induced by $R(r(t_i))$.

Step 1. Starting with a cluster R from the TGS, we first determine if R needs to be refined, by investigating, for each $r_{t_i} \in R$, the subclusters of $R(r_{t_i})$. If, for every $r_{t_i} \in R$, $R(r_{t_i})$ has a single subcluster, then R need not be refined. An example of this is shown in Figure 5. If there is an $r_{t_i} \in R$, such that $R(r_{t_i})$ has two or more subclusters, then we refine R.

Step 2. For any set $\tilde{R}$ consisting of p records, we define its *strength* as the average similarity between the records in $\tilde{R}$:

$$s(\tilde{R}) := \begin{cases} \dfrac{\sum\limits_{\substack{i,j=1 \\ i \neq j}}^{p} (S|_{\tilde{R}})_{i,j}}{\binom{p}{2}}, & \text{if } p \geq 2, \\ 0, & \text{if } p = 1. \end{cases} \quad (6)$$

Note that $s(\tilde{R}) = 1$ if $S|_{\tilde{R}} = \mathbf{1}^{p \times p}$ [8]. In other words, a cluster has a strength of 1 if every pair of records in that cluster satisfy condition 1 of the TGS.

If in Step 1 we have determined that the cluster R requires refinement, we find the optimal record $r^* = r_{t_{k^*}}$ such that the average strength of subclusters of $R(r^*)$ is maximized:

$$k^* = \arg\max_i \frac{1}{q(i)} \sum_{j=1}^{q(i)} s(R_j).$$

Here the sum is over all j such that R_j is a subcluster of $R(t_i)$, and $q(i)$ is the (i-dependent) number of subclusters of $R(t_i)$. In the unlikely event that the maximizer is not unique, we arbitrarily choose one of the maximizers as k^*.

Step 3. After finding the optimal r^* to remove, we now must determine the subcluster to which to add it. To do so, we evaluate the strength of the set $R_j \cup \{r^*\} \subset R$, for each subcluster $R_j \subset R(r^*)$. We then add r^* to subcluster $R^* = R_{l^*}$, where

$$l^* := \arg\max_{\substack{j:\, R_j \text{ is a subcluster} \\ \text{of } R(r^*)}} s(R_j \cup \{r^*\}).$$

---

[8] It suffices if the off-diagonal elements satisfy this equality.
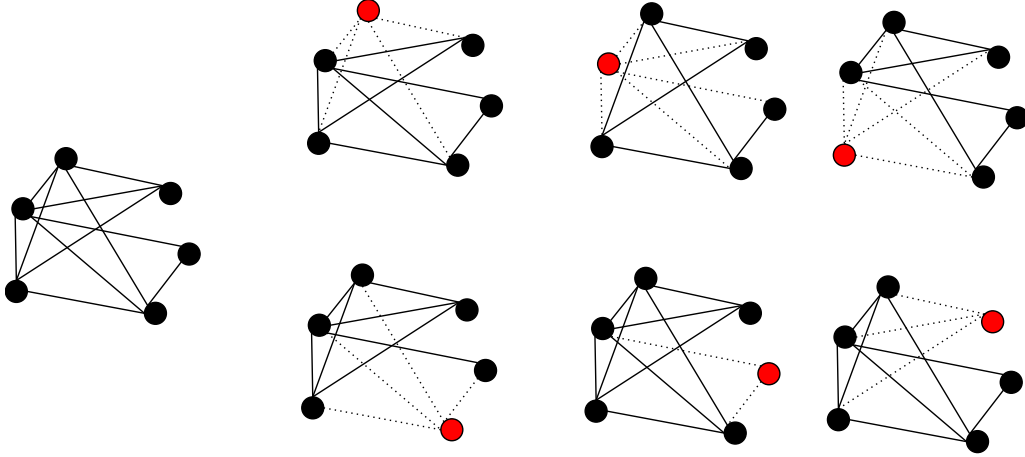
Fig. 5: An example of a cluster R that does not require refinement. Each node represents a record. In each test we remove one and only one node from the cluster and apply TGS again. The red node represents the removed record r_{t_i}, the remaining black nodes make up the set $R(t_i)$. Notice that every time we remove a record, all other records are still connected to each other by solid lines, hence R does not need to be refined.

In the rare event that the maximizer is not unique, we arbitrarily choose one of the maximizers as l^*.

We always add r^* to one of the other subclusters and do not consider the possibility of letting $\{r^*\}$ be its own cluster. Note that this is justified, since from our definition of strength in (6), $s(\{r^*\}) = 0 < s(R_{l^*} \cap \{r^*\})$, because r^* was connected to at least two other records in the original cluster R.

Finally, the original cluster R is removed from the output clustering, and the new clusters $R_1, \ldots, R_{l^*-1}, R^*, R_{l^*+1}, \ldots, R_{q(k^*)}$ are added to the clustering.

Figure 6 shows an example of how the refinement helps us find desired clusters.

In our implementation, we computed the optimal values k^* and l^* are via an exhaustive search over all parameters. This can be computationally expensive when the initial threshold τ is small, leading to large initial clusters.

Algorithm 6: Refinement

Data: $R = \{r_{t_1}, \ldots, r_{t_n}\}$ a cluster resulting from the TGS

Result: $\mathcal{R}$ set of refined clusters

if *there exists r_{t_i} such that $R(r_{t_i})$ has more than 1 subcluster* **then**

    for *each $r_{t_i} \in R$* **do**

        Find the subclusters $R_1, \ldots R_q$ of $R(r_{t_i})$

        Compute $\frac{1}{q}\sum_{j=1}^{q} s(R_j)$

    end

    Assign $r^* = r_{t_{k^*}}$ where

      $k^* = \arg\max_i \frac{1}{q}\sum_{j=1}^{q} s(R_j)$

    for *each subcluster $R_i \subset R(r^*)$* **do**

        Compute $s(R_i \cup \{r^*\})$

    end

    Assign $R^* = (R_{l^*} \cup \{r^*\})$ where

      $l^* = \arg\max_j s(R_j \cup \{r^*\})$

    $\mathcal{R} = \{R_1, \ldots, R_{l^*-1}, R_{l^*}, R_{l^*+1}, \ldots, R_{q(k^*)}\}$

end

else

    Do not refine R: $\mathcal{R} = \{R\}$

end

---

5 Applications

5.1 The data sets

The results presented in this section are based on four data sets: the *Field Interview Card data set* (FI), the *Restaurant data set* (RST), the *Restaurant data set with entries removed to induce sparsity* (RST30), and the *Cora Citation Matching data set* (Cora). FI is not publicly available at the mo-

ment. The other data sets currently can be found at [27]. Cora can also be accessed at [4]. RST and Cora are also used in [6] to compare several approaches to evaluate duplicate detection.

FI This data set consists of digitized *Field Interview cards* from the LAPD. Such cards are created at the officer's discretion whenever an interaction occurs with a civilian. They are not restricted to criminal events. Each card contains 61 fields, including
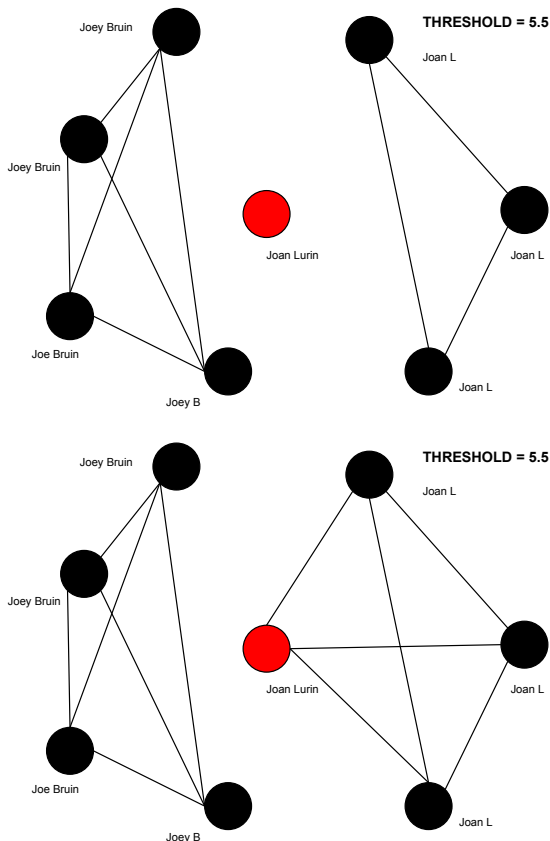
Fig. 6: An example of how refinement is used to improve our clusters. The left figure shows that by removing the record "Joan Lurin", we obtain the two desired subsets. The right figure shows that "Joan Lurin" is inserted back into the appropriate cluster. Note that we have not changed the threshold value τ during this process.

first name, last name, suffix, date of event, location of event, social security number, residential address, gang affiliation, and gang moniker. The latter two are based on expert knowledge. A subset of this data set is used and described in more detail in [13]. The FI data set has 8,834 records, collected during the years 2001-2011. A ground truth of unique individuals is available, based on expert opinion. There are 2,920 unique people represented in the FI Card data set. The FI card data set has many misspellings as well as different names that correspond to the same individual. Another issue is variation over time: a given person is not guaranteed to have the same real world home address in two separate observations, and thus we would not necessarily expect to have matching address fields in our data, regardless of

human error. Approximately 30% of the entries are missing, but the "last name" field is without missing entries.

RST This data set is a collection of restaurant information based on reviews from *Fodor* and *Zagat*, collected by Dr. Sheila Tejada [32], who also manually generated the ground truth. It contains five fields: restaurant name, address, location, phone number, and type of food. There are 864 records containing 752 unique entities/restaurants. There are no missing entries in this data set. The types of errors that are present include word and letter transpositions, varying standards for word abbreviation (e.g. "deli" and "delicatessen"), typographical errors, and conflicting information (such as different phone numbers for the same restaurant).

RST30 To be able to study the influence of sparsity of the data set on our results, we remove approximately 30% of the entries from the address, city, phone number and type of cuisine fields. The resulting data set we call RST30. We choose the percentage of removed entries to correspond to the percentage of missing entries in the FI data set. Because the FI data set has a field that has no missing entries, we do not remove entries from the "name" field.

Cora The records in the *Cora Citation Matching data set*[9] are citations to research papers [22]. Each of Cora's 1,295 records is a distinct citation to any one of the 122 unique papers to which the data set contains references. We use three fields: author(s), name of publication, and venue (name of the journal of publication). This data set contains misspellings and a small amount of missing entries (approximately 3%).

5.2 Evaluation metrics

We compare the performances of the methods summarized in Table 1. Each of these method outputs a similarity matrix, which we then use in the TGS to create clusters.

To evaluate the methods, we use *purity* [15], *inverse purity*, their *harmonic mean* [14], the *relative error in the number of clusters*, *precision*, *recall* [10,

---

[9] The Cora data set should not be confused with the *Coriolis Ocean database ReAnalysis* (CORA) data set.

Name	Similarity matrix	Features	Ref.
TFIDF	Section 3.4	words	no
TFIDF 3g	Section 3.4	3-grams	no
sTFIDF	ST from (4)	words	no
sTFIDF 3g	ST from (4)	3-grams	no
sTFIDF ref	ST from (4)	words	yes
sTFIDF 3g ref	ST from (4)	3-grams	yes

Table 1: Summary of methods used. The second, third, and fourth columns list for each method which similarity score matrix is used in the TGS, if words or 3-grams are used as features, and if the refinement step is applied after TGS or not, respectively. The similarity score matrix refers to either the matrix from equation (4) or the alternative as explained in Section 3.4.

7], the *F-measure* (or F_1 *score*) [28,5], *z-Rand score* [23,33], and *normalized mutual information* (NMI) [31], which are all metrics that compare the output clusterings of the methods with the ground truth.

Purity and inverse purity compare the clusters of records which the algorithm at hand gives with the ground truth clusters. Let $\mathcal{C} := \{R_1, \ldots, R_c\}$ be the collection of c clusters obtained from a clustering algorithm and let $\mathcal{C}' := \{R'_1, \ldots, R'_{c'}\}$ be the collection of c' clusters in the ground truth. Remember that n is the number of records in the data set. Then we define *purity* as

$$Pur(\mathcal{C}, \mathcal{C}') := \frac{1}{n} \sum_{i=1}^{c} \max_{1 \leq j \leq c'} |R_i \cap R'_j|,$$

where we use the notation $|A|$ to denote the cardinality of a set A. In other words, we identify each cluster R_i with (one of the) ground truth cluster(s) R'_j which shares the most records with it, and compute purity as the total fraction of records that is correctly classified in this way. Note that this measure is biased to favour many small clusters over a few large ones. In particular, if each record forms its own cluster, $Pur = 1$. To counteract this bias, we also consider *inverse purity*,

$$Inv(\mathcal{C}, \mathcal{C}') := Pur(\mathcal{C}', \mathcal{C}) = \frac{1}{n} \sum_{i=1}^{c'} \max_{1 \leq j \leq c} |R'_i \cap R_j|.$$

Note that inverse purity has a bias that is opposite to purity's bias: if the algorithm outputs only one cluster containing all the records, then $Inv = 1$.

We combine purity and inverse purity in their *harmonic mean*[10],

$$HM(\mathcal{C}, \mathcal{C}') := \frac{2Pur \times Inv}{Pur + Inv}.$$

The relative error in the number of clusters in $\mathcal{C}$ is defined as

$$\frac{\big||\mathcal{C}| - |\mathcal{C}'|\big|}{|\mathcal{C}'|} = \frac{|c - c'|}{c'}.$$

We define precision, recall, and the F-measure (or F_1 score) by considering *pairs* of clusters that have correctly been identified as duplicates. This differs from purity and inverse purity as defined above, which consider individual records. To define these metrics the following notation is useful. Let G be the set of (unordered) pairs of records that are duplicates, according to the ground truth of the particular data set under consideration,

$$G := \big\{\{r,s\} : r \neq s \text{ and } \exists R' \in \mathcal{C}' \text{ s. t. } r, s \in R'\big\},$$

and let C be the set of (unordered) record pairs that have been clustered together by the duplicate detection method of choice,

$$C := \big\{\{r,s\} : r \neq s \text{ and } \exists R \in \mathcal{C} \text{ s. t. } r, s \in R\big\}.$$

Precision is the fraction of the record pairs that have been clustered together that are indeed duplicates in the ground truth,

$$Pre(\mathcal{C}, \mathcal{C}') := \frac{|C \cap G|}{|C|},$$

and *recall* is the fraction of record pairs that are duplicates in the ground truth that have been correctly identified as such by the method

$$Rec(\mathcal{C}, \mathcal{C}') := \frac{|C \cap G|}{|G|}.$$

The *F-measure* or F_1 *score* is the harmonic mean of precision and recall,

$$F(\mathcal{C}, \mathcal{C}') := 2 \frac{Pre(\mathcal{C}, \mathcal{C}') \times Rec(\mathcal{C}, \mathcal{C}')}{Pre(\mathcal{C}, \mathcal{C}') + Rec(\mathcal{C}, \mathcal{C}')} = 2 \frac{|C \cap G|}{|G| + |C|}.$$

Note that in the extreme case in which $|\mathcal{C}| = n$, i.e. the case in which each cluster contains only one record, precision, and thus also the F-measure, are undefined.

---

[10] The harmonic mean of purity and inverse purity is sometimes also called the F-score or F_1-score, but we will refrain from this terminology to not create confusion with the harmonic mean of precision and recall.

Another evaluation metric based on pair count-
ing, is the z-Rand score. The z-Rand score z_R is the
number of standard deviations by which $|C \cap G|$ is
removed from its mean value under a hypergeomet-
ric distribution of equally likely assignments with the
same number and sizes of clusters. For further details
about the z-Rand score, see [23,33,13]. The *relative
z-Rand score* of $\mathcal{C}$ is the z-Rand score of that clus-
tering divided by the z-Rand score of $\mathcal{C}'$, so that the
ground truth $\mathcal{C}'$ has a relative z-Rand score of 1^{11}.

A final evaluation metric we consider, is *normal-
ized mutual information* (NMI). To define this, we
first need to introduce mutual information and en-
tropy. We define the *entropy* of the collection of clus-
ters $\mathcal{C}$ as

$$Ent(\mathcal{C}) := -\sum_{i=1}^{c} \frac{|R_i|}{n} \log\left(\frac{|R_i|}{n}\right),$$

and similarly for $Ent(\mathcal{C}')$. The joined entropy of $\mathcal{C}$
and $\mathcal{C}'$ is

$$Ent(\mathcal{C}, \mathcal{C}') := -\sum_{i=1}^{c}\sum_{j=1}^{c'} \frac{|R_i \cap R_j'|}{n} \log\left(\frac{|R_i \cap R_j'|}{n}\right).$$

The *mutual information* of $\mathcal{C}$ and $\mathcal{C}'$ is then defined
as

$$I(\mathcal{C}, \mathcal{C}') := Ent(\mathcal{C}) + Ent(\mathcal{C}') - Ent(\mathcal{C}, \mathcal{C}')$$
$$= \sum_{i=1}^{c}\sum_{j=1}^{c'} \frac{|R_i \cap R_j'|}{n} \log\left(\frac{n|R_i \cap R_j'|}{|R_i||R_j|}\right),$$

where the right hand side follows from the equalities
$\sum_{i=1}^{c} |R_i \cap R_j'| = |R_j'|$ and $\sum_{j=1}^{c'} |R_i \cap R_j'| = |R_i|$.
There are various ways in which mutual information
can be normalized. We choose to normalize by the
geometric mean of $Ent(\mathcal{C})$ and $Ent(\mathcal{C}')$ to give the
normalized mutual information

$$NMI(\mathcal{C}, \mathcal{C}') := \frac{I(\mathcal{C}, \mathcal{C}')}{\sqrt{Ent(\mathcal{C})Ent(\mathcal{C}')}}.$$

Note that the entropy of $\mathcal{C}$ is zero, and hence the
normalized mutual information is undefined, when
$|\mathcal{C}| = 1$, i.e. when one cluster contains all the records.
In the practice this is avoided by adding a small
number (e.g. the floating-point relative accuracy `eps`
in MATLAB).

For more information on many of these evalua-
tion metrics, see also [3].

---

[11] We conjecture that the relative z-Rand score is
bounded from above by 1, but to the best of our knowledge
this remains unproven at the moment.

5.3 Results

In this section we consider six methods: TF-IDF,
soft TF-IDF without the refinement step, and soft
TF-IDF with the refinement step, with each of these
three methods applied to both word features and 3-
gram features. We also consider five evaluation met-
rics: the harmonic mean of purity and inverse purity,
the relative error in the number of clusters, the F_1
score, the relative z-Rand score, and the NMI. We
investigate the results in two different ways: (a) by
plotting the scores for a particular evaluation met-
ric versus the threshold values, for the six different
methods in one plot and (b) by plotting the evalua-
tion scores obtained with a particular method versus
the threshold values, for all five evaluation metrics
in one plot.

5.3.1 The methods

When we compare the different methods by plotting
the scores for a particular evaluation metric versus
the threshold value τ_H for all the methods in one
plot (as can be seen for example in Figure 7a), one
notable attribute is that the methods that use word
features typically all show similar behavior and so do
the methods using 3-gram features. There are some
useful distinctions to make, however, between the
methods that do and do not include the refinement
step. A further discussion of this will follow in Sec-
tion 6. This difference though is smaller than the
difference between the word feature and 3-gram fea-
ture based methods. Unsurprisingly, between those
two groups the behavior of the evaluation metrics is
quite distinct, since the similarity scores produced by
those methods, and hence their response to different
threshold values, are significantly different.

It is also interesting to note which methods give
better evaluation metric outcomes on which data
sets. On the FI data set the word feature based meth-
ods outperform the 3-gram based methods (judged
on the basis of best case performance, i.e. the opti-
mal score attained over the full threshold range) for
every evaluation metric, except the NMI for which
they perform similarly.

On both the RST and RST30 data sets, the word
feature based methods outperform the 3-gram fea-
ture based methods on the F_1 score and relative z-
Rand score (Figure 7b), but both groups of methods
perform equally well for the other metrics. It is note-
worthy that all methods also do significantly worse
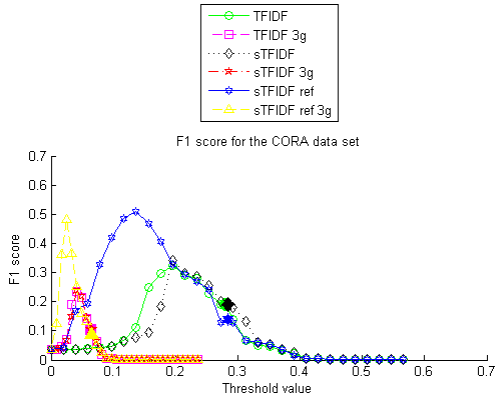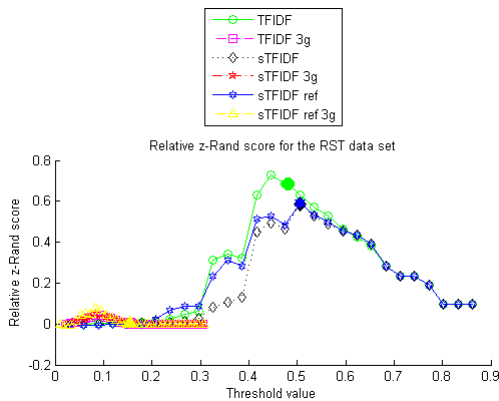on RST30 than on RST, when measured according

(a) The F_1 score for the Cora data set

(b) The relative z-Rand score for the RST data set

Fig. 7: Two evaluation metrics as a function of the threshold value τ, computed on two different data sets. Each of the six graphs in a plot correspond to one of the six methods used. The filled markers indicate the metric's value at the automatically chosen threshold value for each method. In the legend, "(s)TF-IDF" stands for (soft) TF-IDF, "3g" indicates the use of 3-gram based features instead of word based ones, and "ref" indicates the presence of the refinement step.

to the F_1 and relative z-Rand scores, while there is no great difference, if any, measured according to the other metrics.

On the Cora data set all the methods perform equally well according to all evaluation metrics we considered. An interesting characteristic of the results on this data set, that is not observably present in the results for the other data sets, is that the methods that include the refinement step clearly outperform the ones that do not, according the harmonic mean, F_1 score, relative z-Rand score and

NMI. Only the relative error in the number of clusters does not show a noticeable difference.

5.3.2 The metrics

When plotting the different evaluation metrics per method, we notice that the F_1 score and relative z-Rand score behave similarly, as do the harmonic mean of purity and inverse purity and the NMI. The relative error in the number of clusters is correlated to those other metrics in an interesting way. For the word feature based methods, the lowest relative error in the number of clusters is typically attained at or near the threshold values at which the F_1 and relative z-Rand scores are highest. Those are also usually the lowest threshold values for which the harmonic mean and NMI attain their high(est) values. The the harmonic mean and NMI, however, usually remain quite high when the threshold values are increased, whereas the F_1 and relative z-Rand scores typically drop (sometimes rapidly) at increased threshold values, as the relative error in number of clusters rises. Figure 8a shows an example of this behavior.

The relationship between the harmonic mean of purity and inverse purity and the NMI has some interesting subtleties. As mentioned before they mostly show similar behavior, but the picture is slightly more subtly in certain situations. On the Cora data set, the harmonic mean drops noticeably for higher threshold values, before settling eventually at a near constant value. This is a drop that is not present in the NMI. This behavior is also present in the plots for the 3-gram feature based methods on the FI data set and very slightly in the word feature based methods on the RST data set (but not the RST30 data set). For word feature based methods on the FI data set the behavior is even more pronounced, with little to no 'settling down at a constant value' happening for high threshold values (e.g Figure 8b).

Interestingly, both the harmonic mean and NMI show very slight (but consistent over both data sets) improvements at the highest threshold values for the 3-gram based methods applied to the RST and RST30 data sets.

5.3.3 The choice of threshold

On the RST and RST30 data sets our automatically chosen threshold performs well (e.g. see Figures 7b, 8a, and 9a). It usually is close to (or sometimes even equal to) the threshold value at which some or all
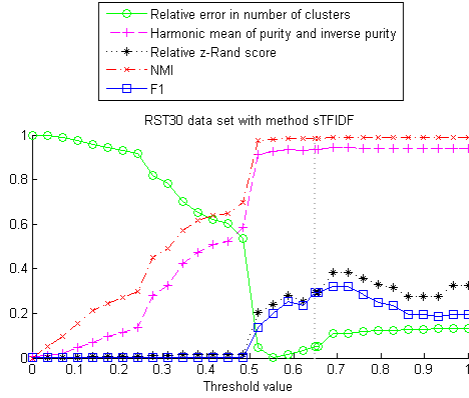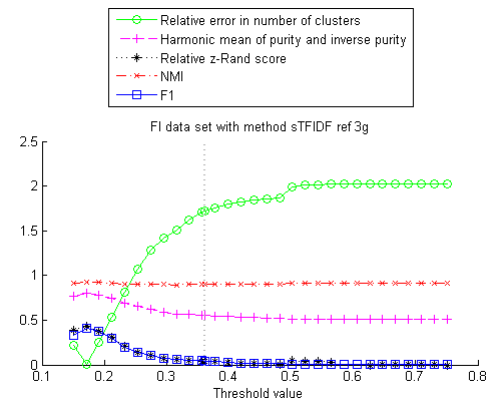
(a) Soft TF-IDF (on word based features) without the refinement step applied to the RST30 data set
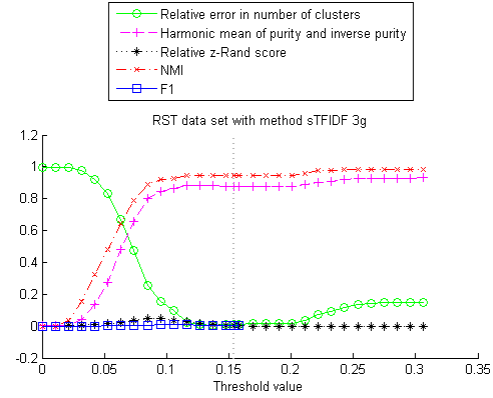
(a) Soft TF-IDF (on 3-gram based features) without the refinement step applied to the RST data set
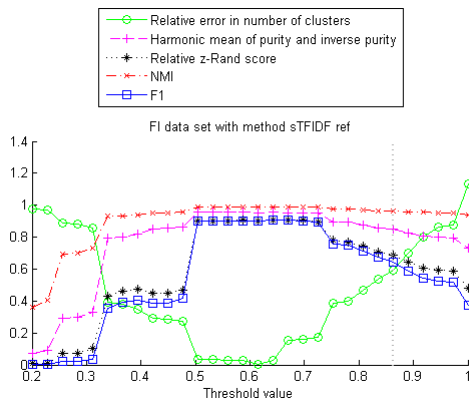
(b) Soft TF-IDF (on word based features) with the refinement step applied to the FI data set

(b) Soft TF-IDF (on 3-gram based features) with the refinement step applied to the FI data set

Fig. 8: Different evaluation metrics as a function of the threshold value τ, computed on two different data sets. Each of the six graphs in a plot correspond to one of five evaluation metrics. The vertical dotted line indicates the automatically chosen threshold value for the method used.

Fig. 9: Different evaluation metrics as a function of the threshold value τ, computed on two different data sets. Each of the six graphs in a plot correspond to one of five evaluation metrics. The vertical dotted line indicates the automatically chosen threshold value for the method used.

evaluation metrics attain their optimal value (remember this threshold value is not the same for all the metrics). The performance on RST is slightly better then on RST30, as can be expected, but in both cases the results are good.

On the FI and Cora data sets our automatically chosen threshold is consistently larger than the optimal value, as can be seen in e.g. Figures 7a, 8b, and 9b. This can be explained by the left-skewedness of the H-value distribution, as illustrated in Figure 3a. A good proxy for the volume of the tail is the ratio of number of records referring to unique entities to total number of entries in the data set. For RST and RST30 this ratio is a high 0.87, whereas

for FI it is only 0.33 and for Cora only 0.09. This means that the relative error in the number of clusters grows rapidly with increasing threshold value and the values of the other evaluation metrics will deteriorate correspondingly.

6 Conclusions

In this paper we have investigated six methods which are based on term frequency-inverse document frequency counts for duplicate detection in a record data set. We have tested them on four different data sets and evaluated the outcomes using five different metrics.

One clear conclusion from our tests is that there is no benefit to constructing the features the methods work on using 3-grams as opposed to white space separated 'words'. The latter choice leads to methods that either outperform the former or perform equally well at worst (in terms of the optimal values they achieve for the evaluation metrics). Compare, for example, Figures 8b and 9b.

Somewhat surprisingly, our tests lead to a less clear picture regarding the choice between TF-IDF and soft TF-IDF (with word based features). For low threshold values TF-IDF performs better, for higher threshold values either soft TF-IDF performs slightly better, or the difference between the two methods is so small as to be negligible. Interesting exceptions are the F_1 score and relative z-Rand score for the RST30 data set. Here TF-IDF outperforms soft TF-IDF for almost every threshold value. At the highest threshold values both methods perform the same, as expected.

When it comes to the benefits of including the refinement step, the situation is again somewhat different depending on the data set. For the RST and RST30 data sets, for small threshold values including the refinement step is beneficial, which is to be expected, since the refinement will either increase the number clusters formed or keep it the same, so its effect is similar to (but not the same) raising the threshold value. On these data sets, for large threshold values there is little difference between including and excluding the refinement step. For the Cora data set an intermediate region is present between the lower and very high threshold values, in which the algorithms perform somewhat better without the refinement step (this is most noticeable in the word feature based algorithms; in the 3-gram feature based algorithms this effect is either absent or minor). This effect is least pronounced (to the point of becoming unnoticeable or absent) for the NMI evaluation metric. For the FI data set this 'intermediate' region (at least for the word feature based algorithms; the qualitative behavior for the 3-gram feature based methods is similar here as in the Cora case) and we are left with a low threshold region in which the refinement step is an improvement and a high threshold region in which excluding that step gives better results. The effect is again least pronounced for the NMI metric.

Our tests with our automatically chosen threshold show that $\tau_H = \mu(H) + \sigma(H)$ is a good choice on data sets which have H-distributions that are approximately normal or right-skewed. If, however, the H-distribution is left-skewed, this choice seems to be consistently larger than the optimal threshold. It should be noted though that for most of the evaluation metrics and most of the data sets, the behavior of the metrics with respect to variations in the threshold value is not symmetric around the optimal value. Typically the decline from optimality is less steep and/or smaller for higher threshold values, than for lower ones. This effect is even stronger if we consider methods without refinement step. Combined with the fact that at low threshold values the refinement step requires a lot more computational time than at high threshold values, especially for larger data sets, we conclude that, in the absence of a priori knowledge of the optimal threshold value, it is better to overestimate than underestimate this value. Hence, our suggestion to choose $\tau_H = \mu(U) + \sigma(H)$ is a good rule of thumb at worst and a very good choice for certain data sets.

References

1. Ahmed, I., Aziz, A.: Dynamic approach for data scrubbing process. International Journal on Computer Science and Engineering **2**(02), 416–423 (2010)
2. Allison, P.D.: Imputation of categorical variables with PROC MI. In: SUGI 30 Proceedings. SAS Institute Inc. (2005). Paper 113-30
3. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf Retrieval **12**, 461–486 (2009)
4. Cora citation matching data set. `http://people.cs.umass.edu/~mccallum/data.html`. Last accessed: 24 March 2014
5. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern information retrieval, vol. 463. ACM press New York (1999)
6. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 39–48. ACM (2003)
7. Bilenko, M., Mooney, R.J.: On evaluation and training-set construction for duplicate detection. In: Proceedings of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, pp. 7–12 (2003)
8. Chen, M.: Normalized mutual information. `http://www.mathworks.com/matlabcentral/fileexchange/29047-normalized-mutual-information` (2010). Contact: mochen@ie.cuhk.edu.hk; Last accessed: 26 February 2015
9. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E., et al.: A comparison of string distance metrics for name-matching tasks. In: IIWeb, vol. 2003, pp. 73–78 (2003)
10. Cohen, W.W., Richman, J.: Learning to match and cluster large high-dimensional data sets for data integration. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 475–480. ACM (2002)
11. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. Knowledge and Data Engineering, IEEE Transactions on **19**(1), 1–16 (2007)
12. Fiedler, S.: Cell array to CSV-file [cell2csv.m], updated. `http://uk.mathworks.com/matlabcentral/fileexchange/4400-cell-array-to-csv-file--cell2csv-m-` (2010). Modified by Rob Kohr; last accessed: 4 September 2014
13. van Gennip, Y., Hunter, B., Ahn, R., Elliott, P., Luh, K., Halvorson, M., Reid, S., Valasik, M., Wo, J., Tita, G.E., Bertozzi, A.L., Brantingham, P.J.: Community detection using spectral clustering on sparse geosocial data. SIAM J. Appl. Math. **73**(1), 67–83 (2013)
14. González, E., Turmo, J.: Non-parametric document clustering by ensemble methods. Procesamiento del Lenguaje Natural **40**, 91–98 (2008)
15. Harris, M., Aubert, X., Haeb-Umbach, R., Beyerlein, P.: A study of broadcast news audio stream segmentation and segment clustering. In: Proceedings of EUROSPEECH99, pp. 1027–1030 (1999)
16. Horton, N.J., Kleinman, K.P.: Much ado about nothing. The American Statistician **61**(1) (2007)
17. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. Journal of the American Statistical Association **84**(406), 414–420 (1989)
18. Jaro, M.A.: Probabilistic linkage of large public health data file. In: Statistics in Medicine, vol. 14, pp. 491–498 (1995)
19. Koehler, M.: matrix2latex, updated. `http://www.mathworks.com/matlabcentral/fileexchange/4894-matrix2latex` (2004). Last accessed: 24 March 2014
20. Komarov, O.: Set functions with multiple inputs, updated. `http://uk.mathworks.com/matlabcentral/fileexchange/28341-set-functions-with-multiple-inputs/content/SetMI/unionm.m` (2010). Last accessed: 24 March 2014
21. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval, vol. 1. Cambridge University Press Cambridge (2008)
22. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Journal of Information Retrieval **3**(2) (2000)
23. Meilă, M.: Comparing clusterings — an information based distance. J. Multivariate Anal. **98**, 873–895 (2007)
24. Monge, A.E., Elkan, C.P.: Efficient domain-independent detection of approximately duplicate database records. In: Proc. of the ACM-SIGMOD Workshop on Research Issues in on Knowledge Discovery and Data Mining (1997)
25. Naumann, F., Herschel, M.: An introduction to duplicate detection. Synthesis Lectures on Data Management **2**(1), 1–87 (2010)
26. Pigott, T.D.: A review of methods for missing data. Educational research and evaluation **7**(4), 353–383 (2001)
27. Duplicate detection, record linkage, and identity uncertainty: Datasets. `http://www.cs.utexas.edu/users/ml/riddle/data.html`. Last accessed: 24 March 2014
28. van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Butterworth-Heinemann, Newton, MA, USA (1979)
29. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Information processing & management **24**(5), 513–523 (1988)
30. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
31. Strehl, A., Ghosh, J.: Cluster ensembles — a knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research **3**, 583–617 (2002)
32. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Information Systems **26**(8), 607–633 (2001)
33. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: Comparing community structure to characteristics in online collegiate social networks. SIAM review **53**(3), 526–543 (2011)
34. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: zrand. `http://netwiki.amath.unc.edu/GenLouvain/GenLouvain` (2011). Last accessed: 24 March 2014
35. Tromp, M., Reitsma, J., Ravelli, A., Méray, N., Bonsel, G.: Record linkage: making the most out of errors in linking variables. In: AMIA Annual Symposium

Proceedings, vol. 2006, p. 779. American Medical Informatics Association (2006)

36. Winkler, W.: String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In: Proceedings of the Section on Survey Research Methods, pp. 354–359. (American Statistical Association) (1990)

37. Winkler, W.E.: The state of record linkage and current research problems. In: Statistical Research Division, US Census Bureau (1999)

38. Winkler, W.E.: Methods for record linkage and Bayesian networks. Tech. rep., Series RRS2002/05, U.S. Bureau of the Census (2002)

39. Winkler, W.E.: Overview of record linkage and current research directions. Tech. rep., Bureau of the Census (2006)