International Journal of Data Science and Analytics (2018) 6:109–129
https://doi.org/10.1007/s41060-018-0129-7

REGULAR PAPER

Unsupervised record matching with noisy and incomplete data

Yves van Gennip[1] · Blake Hunter[2] · Anna Ma[3] · Daniel Moyer[4] · Ryan de Vera[5] · Andrea L. Bertozzi[6]

Received: 10 April 2017 / Accepted: 30 April 2018 / Published online: 23 May 2018

Abstract
We consider the problem of duplicate detection in noisy and incomplete data: Given a large data set in which each record has multiple entries (attributes), detect which distinct records refer to the same real-world entity. This task is complicated by noise (such as misspellings) and missing data, which can lead to records being different, despite referring to the same entity. Our method consists of three main steps: creating a similarity score between records, grouping records together into "unique entities", and refining the groups. We compare various methods for creating similarity scores between noisy records, 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 also discuss two methods to deal with missing data in computing similarity scores. 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 the methods that use words as the basic units are preferable to those that use 3-grams. Moreover, in some (but certainly not all) parameter ranges soft term frequency-inverse document frequency methods can outperform the standard term frequency-inverse document frequency method. The results 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.

Keywords Duplicate detection · Data cleaning · Data integration · Record linkage · Entity matching · Identity uncertainty · Transcription error

1 Introduction

Fast methods for matching records in databases that are similar or identical have growing importance as database sizes increase [2,21,42,68,70]. 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.

Ryan de Vera—formerly California State University, Long Beach, USA; ryan.devera.03@gmail.com.

✉ Yves van Gennip
y.vangennip@nottingham.ac.uk

Blake Hunter
bhunter@cmc.edu

Anna Ma
anna.ma@cgu.edu

Daniel Moyer
moyerd@usc.edu

Ryan de Vera
ryan.devera@shophush.com

Andrea L. Bertozzi
bertozzi@math.ucla.edu

1 University of Nottingham, Nottingham, UK
2 Claremont McKenna College, Claremont, USA
3 Claremont Graduate University, Claremont, USA
4 University of Southern California, Los Angeles, USA
5 Hush, Inc., Los Angeles, CA, USA
6 University of California, Los Angeles, USA

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 are no longer a true representation of the real world. After a detailed description of the problem in Sect. 2 and a review of related methods in Sect. 3, we present in Sect. 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 duplicates. In Sect. 5, we explore the performances of different variations of our method on four text databases that contain duplicates.

2 Terminology and problem statement

We define a data set D to be an $n \times a$ array where each element of the array is 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 $e_{i,k}$ (referring to the ith entry in the kth 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 which makes up entry $e_{i,k}$ into multiple features. In our implementations in this paper, 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 $e_{i,k}$ which is made up of 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 $e_{i,k}$ 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 $N=3$, $e_{i,k}$ contains 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 spaces.

When discussing our results 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" (as is common in practice) and to this reduced dictionary as the *set of features*, f^k, where:

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

if the kth field contains m features. This reduced dictionary represents an ordered set of unique features found in field c^k.

Note that m, the number of features in f^k, depends on k, since a separate set of features 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 Sect. 3.2. Note that an entry can be "empty" if it only contains stop words, since those are not included in the set of features f^k.

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., subsets containing those records that are duplicates of each other. *Duplicate records*, in this sense, are not necessarily identical records but can also be 'near identical' records. They are allowed to vary due to spelling errors or missing entries.

3 Related methods

Numerous algorithms for duplicate detection exist, including various probabilistic methods [32], string comparison metrics [31,67], feature frequency methods [54], and hybrid methods [14]. There are many other proposed methods for data matching, record linkage and various stages of data cleaning, that have a range of success in specific applications but also come with their own limitations and drawbacks. Surveys of various duplicate detection methods can be found in [1,4,21,28,53].

Probabilistic rule-based methods, such as Fellegi–Sunter-based models [67], are methods that attempt to learn features and rules for record matching using conditional probabilities; however, these are highly sensitive to the assumed model which is used to describe how record duplicates are distributed across the database and become completely infeasible at large scale when comparing all pairs. Other rule-based approaches such as [58] attempt to create a set of rules that is flexible enough to deal with different types of data sets.

Privacy-preserving record matching techniques [26,56], based on hash encoding, are fast and scalable, but can only handle exact matching (single character differences or small errors in input result in completely different hash codes); approximate matching-based methods are often possible but typically not scalable.

Collective record matching techniques [24,47] have been proposed that match records across multiple databases, using a graph based on similarity of groups of entities. These methods have shown promise in some applications where entity relationships are identifiable (such as sharing the same address or organization), but direct applications are limited and are currently not generalizable or scalable.

Unsupervised or supervised techniques [23] can also be used directly, using records as features, but in most applications labeled data does not exist for training or evaluation. Additionally, standard testing data sets, used for comparing methods, are extremely limited and weakly applicable to most applications. Some techniques are developed specifically to deal with hierarchical data, such as XML data [1,41]. We do not consider that situation here.

For larger data sets, a prefix filtering [71], blocking [17,18,49,50] or windowing [7,18,33] step can be used. Such methods can be seen as a preprocessing step which identifies records which are not likely to be duplicates, such that the pairwise feature similarity does only need to be computed for those features that co-appear in likely duplicates. A survey of various such indexing methods is given in [13]. We did not include an indexing step in our experiments in this paper, so that our experiments are run without excluding any record pairings a priori, but they can be incorporated into our method.

Pay-as-you-go [66] or progressive duplicate detection methods [33,51] have been developed for applications in which the duplicate detection has to happen in limited time on data which is acquired in small batches or in (almost) real-time [40]. In our paper, we consider the situation in which we have all data available from the start.

In [8], the authors suggest to use trainable similarity measures that can adapt to different domains from which the data originate. In this paper, we develop our method using given similarity measures, such that our method is applicable in the absence of training data.

In the remainder of this section, we present in more detail those methods which are related to the proposed method we introduce in Sect. 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 [31] was originally devised for duplicate detection in government census data and modified by Winkler [67] 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 [14]. 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 can be called 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*

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

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 $l - W_{i,j}^k \leq l' \leq l + W_{i,j}^k$. 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 $a_l \neq b_l$. Each such pair (a_l, b_l) is called a *transposition pair*. Now the *Jaro distance* [31] $J(f_i^k, f_j^k)$ 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.

The Jaro–Winkler metric, $\mathrm{JW}(f_i^k, f_j^k)$, 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

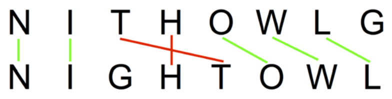


Fig. 1 Example of a comparison of two features in the computation of the Jaro distance, with character window size $W = 4$. The example has seven matching character pairs, two of which are transposition pairs, represented by the red lines. The green lines indicate matching pairs that are not transpositions. 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, $J = \frac{1}{3}(\frac{7}{8} + \frac{7}{8} + \frac{7-1}{7}) = 0.869$ (color figure online)

to features that start with the same characters. Given two features f_i^k and f_j^k, the *Jaro–Winkler metric* is

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

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 are the same as the corresponding prefix characters in f_j^k (i.e., the first $\ell_{i,j}$ characters in f_i^k are the same as the first $\ell_{i,j}$ characters in f_k^j and the $(\ell_{i,j}+1)$th characters in both features differ). When we want to stress that, for fixed k, $\mathrm{JW}(f_i^k, f_j^k)$ is an element of a matrix, we write $\mathrm{JW}_{i,j}^k := \mathrm{JW}(f_i^k, f_j^k)$, such that $\mathrm{JW}^k \in \mathbb{R}^{m\times m}$.

In Winkler's original work, he set $p = 0.1$ and restricted $\ell_{i,j} \le 4$ (even when prefixes of five or more characters were shared between features) [67]. We follow the same parameter choice and restriction in our applications in this paper. So long as $p\,\ell_{i,j} \le 1$ 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 Fig. 1, we have $\ell = 2$, as both features have identical first and second characters, but not a matching third character. This leads to $\mathrm{JW} = 0.869 + 0.1 \cdot 2 \cdot (1 - 0.869) = 0.895$.

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

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").

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 $f^k = (f_1^k, \dots, f_m^k)$
for each pair of features $(f_i^k, f_j^k)$ do
    Compute Jaro distance $J_{i,j} = J(f_i^k, f_j^k)$
    Compute Jaro–Winkler similarity
    $JW_{i,j}^k = \begin{cases} J_{i,j} + p\,\ell_{i,j}(1 - J_{i,j}), & \text{if neither feature } f_i^k \text{ or } f_j^k \text{ is a null feature,} \\ 0, & \text{else} \end{cases}$
end
```

This form of duplicate detection is closely related to vector space models of text corpora [55], 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 [54]. 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, $\mathrm{TF}^k \in \mathbb{R}^{n\times m}$, such that $\mathrm{TF}_{i,j}^k$ is the number of times the feature f_j^k 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* $\mathrm{IDF}^k \in \mathbb{R}^{m\times m}$ with diagonal elements[1]

$$\mathrm{IDF}_{i,i}^k := \log \frac{n}{|\{e \in c^k : f_i^k \in e\}|},$$

where $|\{e \in c^k : f_i^k \in e\}|$ is the number of entries[2] in field c^k containing feature f_i^k, and where n is the number of records in the data set. The matrix IDF^k uses this number of entries in the field which contain a given feature to give this feature 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. The latter often are empirically more informative than common features, since a feature that occurs frequently in many entries is unlikely to be a good discriminator.

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

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

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

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

where $\mathbf{1}$ is an $n \times m$ matrix of ones, the log operation acts on each element of $\mathrm{TF}^k + \mathbf{1}$ individually, and $N^k \in \mathbb{R}^{n\times n}$ is a diagonal normalization matrix such that each nonzero row of TFIDF^k has unit ℓ^1 norm.[4] The resulting matrix has dimension $n \times m$. Each element $\mathrm{TFIDF}^k_{i,j}$ represents the weight assigned to feature j in field k for record i. Note that each element 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 $f^k = (f^k_1, \ldots, f^k_m)$
for each pair of features $(f^k_i, f^k_j)$ do
    Compute term frequency $TF^k_{i,j}$
end
for each feature $f^k_i$ do
    Compute inverse document frequency $IDF^k_{i,i}$
end
Initialize $TFIDF^k = \log(TF^k + \mathbf{1})IDF^k$
Normalize rows of $TFIDF^k$ to have unit $\ell^1$ norm
```

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 [46]. In this section, we will discuss another such method, soft TF-IDF [14], which combines TF-IDF with a character similarity measure. In our method, we use the Jaro–Winkler metric, discussed in Sect. 3.1, as the character similarity measure in soft TF-IDF.

For $\theta \in [0, 1)$, let $S^k_{i,j}(\theta)$ be the set of all index pairs $(p, q) \in \mathbb{R}^{m\times m}$ such that $f^k_p \in e_{i,k}$, $f^k_q \in e_{j,k}$, and $\mathrm{JW}(f^k_p, f^k_q) > \theta$, where JW is the Jaro–Winkler similarity metric from (1). The *soft TF-IDF similarity score* between two entries $e_{i,k}$ and $e_{j,k}$ in field c^k is defined as

$$\mathrm{sTFIDF}^k_{i,j} := \begin{cases} \sum\limits_{(p,q)\in S^k_{i,j}(\theta)} \mathrm{TFIDF}^k_{i,p} \cdot \mathrm{TFIDF}^k_{j,q} \cdot \mathrm{JW}^k_{p,q}, & \text{if } i \neq j, \\ 1, & \text{if } i = j. \end{cases} \tag{3}$$

The parameter θ allows for control over the similarity of features, removing entirely pairs that do not have Jaro–Winkler similarity above a certain threshold. The results presented in this paper are all obtained with $\theta = 0.90$.

The soft TF-IDF similarity score between two entries is high if they share many similar features, where the similarity between features is measured by the Jaro–Winkler metric and the contribution of each feature is weighted by its TF-IDF score. If we contrast the soft TF-IDF score with the TF-IDF score described in Sect. 3.4, we see that the latter only uses those features which are exactly shared by both entries, whereas the former also incorporates contributions from features that are very similar (but not exactly the same). This means that the soft TF-IDF score allows for high similarity between entries in the presence of both misspellings and varying word (or feature) order more so than the TF-IDF score does.

Note from (3) that for all i, j, and k, we have $\mathrm{sTFIDF}^k_{i,j} \in [0, 1]$. The expression for the case $i \neq j$ does not necessarily evaluate to 1 in the case $i = j$. Therefore, we explicitly included $\mathrm{sTFIDF}^k_{i,i} = 1$ as part of the definition, since this is a reasonable property for a similarity measure to have. Luckily, these diagonal elements of sTFIDF^k will not be relevant in our method, so the $i = j$ part of the definition is more for definiteness and computational ease,[5] than out of strict necessity for our method.

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

$$M^{k,\theta}_{p,q} := \begin{cases} \mathrm{JW}(f^k_p, f^k_q), & \text{if } \mathrm{JW}(f^k_p, f^k_q) \geq \theta, \\ 0, & \text{if } \mathrm{JW}(f^k_p, f^k_q) < \theta. \end{cases}$$

[3] Note that, following [14], we use a slightly different logarithmic scaling, than the more commonly used $\mathrm{TFIDF}^k_{i,j} = \big(\log(\mathrm{TF}^k_{i,j}) + 1\big)\mathrm{IDF}^k_{i,i}$, if $\mathrm{TF}^k_{i,j} \neq 0$, and $\mathrm{TFIDF}^k_{i,j} = 0$, if $\mathrm{TF}^k_{i,j} = 0$. This avoids having to deal with the case $\mathrm{TF}^k_{i,j} = 0$ separately. The difference between $\log(\mathrm{TF}^k_{i,j}) + 1$ and $\log(\mathrm{TF}^k_{i,j} + 1)$ is bounded by 1 for $\mathrm{TF}^k_{i,j} \geq 1$.

[4] Here, we deviate from [14], in which the authors normalize by the ℓ^2 norm. We do this so that later in Eq. (3), we can guarantee that the soft TF-IDF values are upper bounded by 1.

[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 are not set to have value 1, care must be taken that this does not influence the numerical implementation.

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

$$\text{sTFIDF}^k_{i,j} = \sum_{p,q=1}^{m} \left[\left(\text{TFIDF}^{k\,\mathrm{T}}_i \text{TFIDF}^k_j \right) * M^{k,\theta} \right]_{p,q},$$

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

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

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

$$\text{sTFIDF}^k_{i,j} = (\text{TFIDF}^k_i \otimes \text{TFIDF}^k_j) * \overline{M}^{k,\theta},$$

if $i \neq j$. Here $\otimes$ is the Kronecker product. Finally, we set the diagonal elements $\text{sTFIDF}^k_{i,i} = 1$.

Algorithm 3: soft TF-IDF Algorithm

```
Data: $JW^k \in \mathbb{R}^{m\times m}$, $TFIDF^k \in \mathbb{R}^{n\times m}$, $\theta$
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 for $i \neq j$:
    $sTFIDF_{i,j}^k = (TFIDF_i^k \otimes TFIDF_j^k) * \overline{M}^{k,\theta}$
end
Set the diagonal elements $sTFIDF_{i,i}^k = 1$
```

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

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* $\text{ST}_{i,j}$ by adding their soft TF-IDF scores over all fields:

$$\text{ST}_{i,j} := \sum_{k=1}^{a} \text{sTFIDF}^k_{i,j}. \tag{4}$$

Hence, $\text{ST} \in \mathbb{R}^{n\times n}$ and $\text{ST}_{i,j}$ is the score between the ith and jth records. Remember that a is the number of fields in the data set, thus each composite similarity score $\text{ST}_{i,j}$ is a number in $[0, a]$.

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:

$$\text{ST}^w_{i,j} := \sum_{k=1}^{a} w_k\, \text{sTFIDF}^k_{i,j},$$

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 scores* $\text{ST}^w_{i,j}$ take values in $[0, a]$, like $\text{ST}_{i,j}$. In this paper, we use the unweighted composite similarity score matrix ST.

3.4 Using TF-IDF instead of soft TF-IDF

In our experiments in Sect. 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 Sect. 3.3, if we replace $\text{JW}^k_{p,q}$ in (3) by the Kronecker delta $\delta_{p,q} := \begin{cases} 1, & \text{if } p = q, \\ 0, & \text{otherwise.} \end{cases}$ The dependency on θ disappears and we get

$$\text{sTFIDF}^k_{i,j} := \begin{cases} \sum_{p=1}^{m} \left(\text{TFIDF}^k_{i,p} \right)^2, & \text{if } i \neq j, \\ 1, & \text{if } i = j. \end{cases} \tag{5}$$

Note that the values for $i \neq j$ correspond to the off-diagonal values in the matrix $\text{TFIDF}^k \left(\text{TFIDF}^k\right)^{\mathrm{T}} \in \mathbb{R}^{n\times n}$, where TFIDF^k is the TF-IDF matrix from (2) and the superscript T denotes the matrix transpose.[6]

We used the same notation for the matrices in (3) and (5), because all the other computations, in particular the computation of the composite similarity score in (4) which is used in the applications in Sect. 5, follow the same recipe when using either matrix. Where this is of importance in this paper, it will be clear from the context if ST has been constructed using the soft TF-IDF or TF-IDF similarity scores.

[6] Our choice to normalize the rows of TFIDF^k by their ℓ^1 norms instead of their ℓ^2 norms means that the diagonal elements of $\text{TFIDF}^k \left(\text{TFIDF}^k\right)^{\mathrm{T}}$ are not necessarily equal to 1.

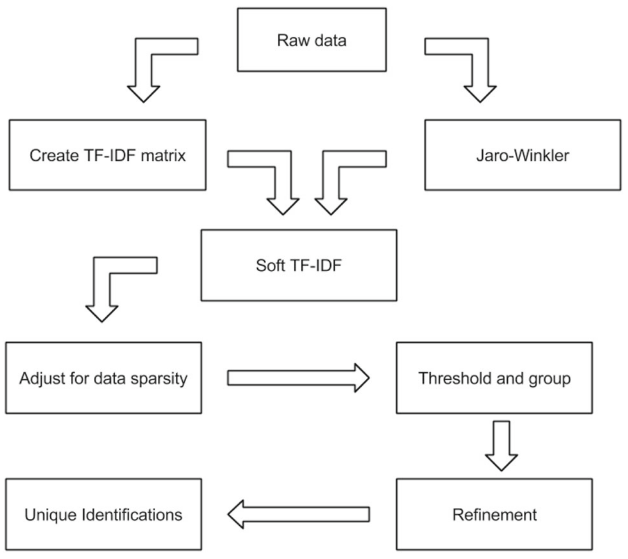


Fig. 2 An outline of our method for duplicate detection

4 The proposed methods

We extend the soft TF-IDF method to address two common situations in duplicate detection: sparsity due to 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 from the start or one that contains only null features and thus ends up being empty for our purposes. 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 be clustered together (i.e., labeled as probable duplicates). In [62], [67], and [3], 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), estimation by an EM scheme becomes computationally intractable [29,52,69]. 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 between 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 similarity score 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 (from a larger data set containing $n > 2$ 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 (1 out of a maximum of 3) 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. The issue described above for the soft TF-IDF method is also present for the TF-IDF method described in Sect. 3.4.

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 the $n \times a$ binary matrix defined by

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

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 $BB^{\mathrm{T}} \in \mathbb{R}^{n \times n}$, each $(BB^{\mathrm{T}})_{i,j}$ 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 *adjusted (soft) TF-IDF similarity score* is given by

$$\mathrm{adjST}_{i,j} := \begin{cases} \frac{\mathrm{ST}_{i,j}}{(BB^{\mathrm{T}})_{i,j}}, & \text{if } i \neq j \text{ and } (BB^{\mathrm{T}})_{i,j} \neq 0, \\ 0, & \text{if } i \neq j \text{ and } (BB^{\mathrm{T}})_{i,j} = 0, \\ 1, & \text{if } i = j. \end{cases} \tag{6}$$

Remembering that $\mathrm{JW}(f_p^k, f_q^k) = 0$ if f_p^k is a null feature or f_q^k is a null feature, we see that, if $e_{i,k}$ is a missing entry or $e_{j,k}$ is a missing entry, then the set $S_{i,j}^k(\theta)$ used in (3) is empty (independent of the choice of θ) and thus $\mathrm{sTFIDF}_{i,j}^k = 0$. The same conclusion is true in (5) since the ith or jth row of TFIDF^k consists of zeros in that case. Hence, we have that, for all i, j $(i \neq j)$, $(\mathrm{ST})_{i,j} \in [0, (BB^{\mathrm{T}})_{i,j}]$ (which refines our earlier result that $(\mathrm{ST})_{i,j} \in [0, a]$) and thus $(\mathrm{adjST})_{i,j} \in [0, 1]$.

In the event that there are records r_i and r_j such that $(BB^{\mathrm{T}})_{i,j} = 0$, it follows that $\mathrm{ST}_{i,j} = 0$. Hence, it makes sense to define $\mathrm{adjST}_{i,j}$ to be zero in this case. In the data sets we will discuss in Sect. 5, no pair of records was without shared fields. Hence, we can use the shorthand expression $\mathrm{adjST} = \mathrm{ST} \oslash BB^{\mathrm{T}}$ for our purposes in this paper,[7] where $\oslash$ denotes element-wise division.

Algorithm 4: Adjusting for Sparsity

```
Data: $sTFIDF^k \in \mathbb{R}^{n\times n}$ for $k \in \{1, \dots, a\}$, $D$ an $n \times a$ 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
    Compute $B_{i,k}$
end
Initialize $ST = \sum_k sTFIDF^k$
Adjust ST for sparsity: $adjST = ST \oslash BB^T$
```

Instead of the method proposed above to deal with missing data, we can also perform data imputation to replace the missing data with a “likely candidate” [4,30,34,35,39,65]. To be precise, before computing the matrix B, we replace each missing entry $e_{i,k}$ by the entry which appears most often in the kth field.[8] In case of a tie, we choose an entry at random among all the entries with the most appearances (we choose this entry once per field, such that each missing entry in a given field is replaced by the same entry). For a clean comparison, we still compute the matrix B (which has now no 0 entries) and use it for the normalization in (6). The rest of our method is then implemented as usual. We report the results of this comparison in Sect. 5.4.

4.2 Thresholding and grouping

The similarity score $\mathrm{adjST}_{i,j}$ gives us an indication of how similar the records r_i and r_j are. If $\mathrm{adjST}_{i,j}$ is 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, [45] 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 $\mathrm{SIM} \in \mathbb{R}^{n\times n}$ be a matrix of similarity scores, i.e., for all i, j, the entry $\mathrm{SIM}_{i,j}$ is a similarity score between the records r_i and r_j. We assume that, for all $i \neq j$, $\mathrm{SIM}_{i,j} = \mathrm{SIM}_{j,i} \in [0, a]$.[9] If we use our adjusted (soft) TF-IDF method, SIM is given by adjST from (6). In Sect. 4.1 we saw that in that case we even have $\mathrm{SIM}_{i,j} \in [0, 1]$.

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

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

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 the same cluster, or with each cluster containing only one record) the threshold value τ must be chosen in the half-open interval

$$\Big(\min_{i,j:j\neq i} \mathrm{SIM}_{i,j}, \max_{i,j:j\neq i} \mathrm{SIM}_{i,j}\Big].$$

If $S_{i,j} = 1$, then the records r_i and r_j are clustered together. Note that this is a sufficient, but not necessary condition for two records to be clustered together. For example, if $S_{i,j} = 0$, but $S_{i,k} = 1$ and $S_{j,k} = 1$, then r_i and r_k are clustered together, as are r_j and r_k, and thus so are r_i and r_j. 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

[7] Since we defined the inconsequential diagonal entries to be $\mathrm{sTFIDF}_{i,i}^k = 1$ in (3) and (5), it could be that $(\mathrm{ST})_{i,i} > (BB^{\mathrm{T}})_{i,i}$ for some i, which is why we explicitly defined $(adjST)_{i,i} = 1$ in (6) for consistency with the other values. Since the diagonal values will play no role in the eventual clustering this potential discrepancy between (6) and $\mathrm{adjST} = \mathrm{ST} \oslash BB^{\mathrm{T}}$ is irrelevant for our purposes.

[8] We use the mode, rather than the mean, because all our data is either textual or, when numeric, it is ordinal, rather than cardinal, such as in the case of social security numbers.

[9] 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, $\mathrm{SIM}_{i,i} = a$.

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

$$H_i := \max_{\substack{1 \le j \le n \\ j \ne i}} \mathrm{SIM}_{i,j}.$$

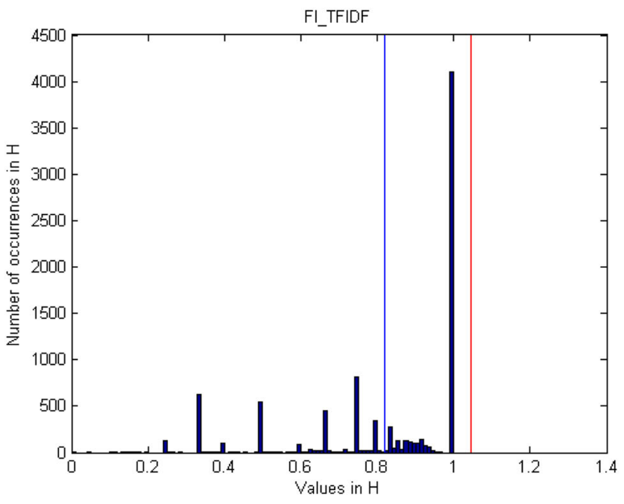
In other words, the ith element of H is the maximum similarity score $\mathrm{SIM}_{i,j}$ between the ith record and every other record. Now define

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

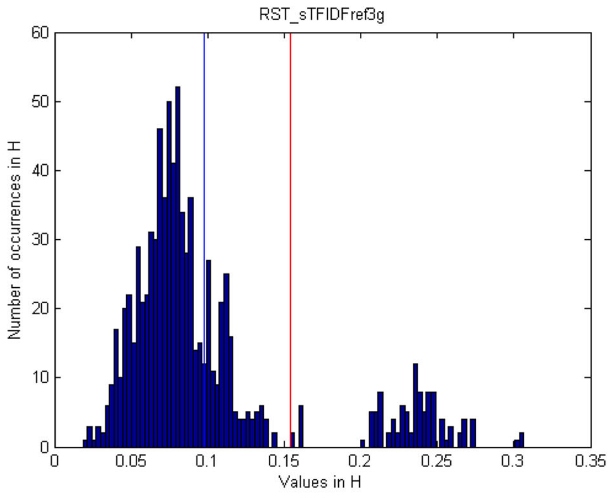
where $\mu(H)$ is the mean value of H and $\sigma(H)$ is its corrected sample standard deviation.[10]

We choose τ_H in this fashion, because it is easily implementable, has shown to work well in practice (see Sect. 5) even if it is not always the optimal choice, and is based on some underlying heuristic ideas and empirical observations of the statistics of H in our data sets (which we suspect to be more generally applicable to other data sets) that we will explain below. It provides a good alternative to trial-and-error attempts at finding the optimal τ, which can be quite time-intensive.

For a given record r_i, the top candidates to be duplicates of r_i are those records r_j for which $\mathrm{SIM}_{i,j} = H_i$. A typical data set, however, will have many records that do not have duplicates at all. To reflect this, we do not want to set the threshold τ_H lower than $\mu(H)$. If H is normally distributed, this will guarantee that at least approximately half of the records in the data set will not be clustered together with any other record. In fact, in many of our runs (Fig. 3a is a representative example), there is a large peak of H values around the mean value $\mu(H)$. Choosing τ_H equal to $\mu(H)$ in this case will lead to many of the records corresponding to this peak being clustered together, which is typically not preferred. Choosing $\tau_H = \mu(H) + \sigma(H)$ will place the threshold far enough to the right of this peak to avoid overclustering, yet also far enough removed from the maximum value of H so that not only the top matches get identified as duplicates. In some cases, however, the distribution of H values has a peak near the maximum value instead of near the mean value (as, for example, in Fig. 3b) and the value $\mu(H) + \sigma(H)$ will



(a)



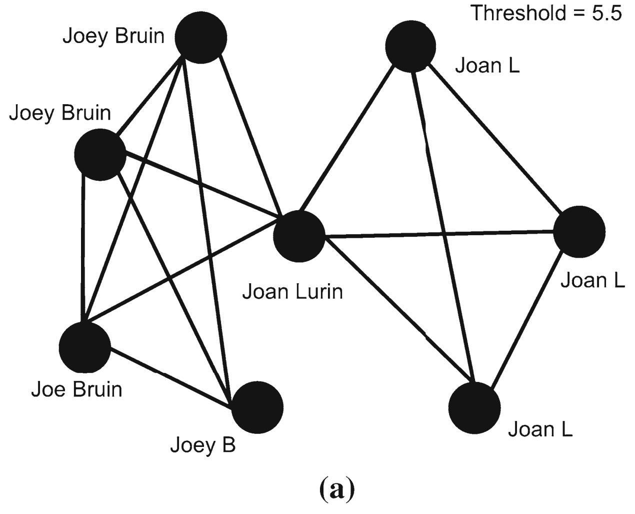
(b)

Fig. 3 Histograms of H for different methods applied to the FI and RST data sets (see Sect. 5.1). **a** H corresponding to the TF-IDF method (with word feature, without refinement step, see Sect. 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 Sect. 4.3) applied to the RST data set. The blue line indicates the chosen value $\tau_H = \mu(H)$; the red line indicates $\mu(H) + \sigma(H)$ (color figure online)

be larger than the maximum H value. In those cases, we can choose $\tau_H = \mu(H)$ without risking overclustering.

It may not always be possible to choose a threshold in such a way that all the clusters generated by our TGS correspond to sets of actual duplicates, as the following example, illustrated in Fig. 4, shows. We 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 ($i \ne j$) by an edge if and only if their similarity score $\mathrm{SIM}_{i,j}$ equals or exceeds

[10] We used MATLAB's `std` function.



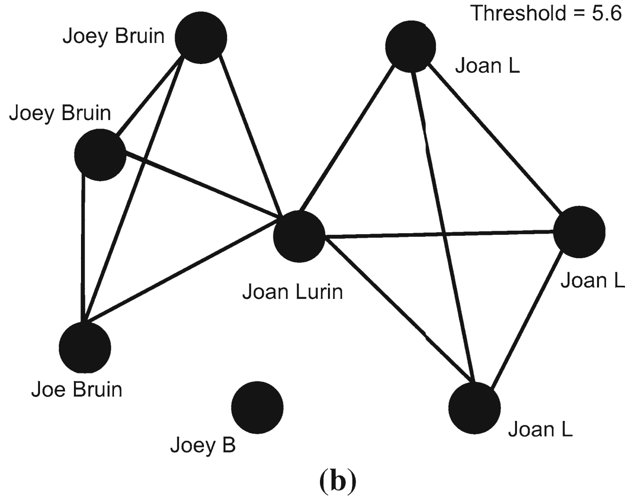


Fig. 4 Two examples of clusters created by the TGS applied to an artificial data set, with different threshold values τ. **a** Result of the TGS with $\tau = 5.5$, **b** Result of the TGS with $\tau = 5.6$

the chosen threshold value τ. The connected components of the resulting graph then correspond to the clusters the TGS outputs.

For simplicity, Fig. 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 (Fig. 4a). Using $\tau = 5.6$, we do obtain two clusters (Fig. 4b), but it is not true that one cluster represents “Joey Bruin” and the other “Joan Lurin”, as desired. Instead, one cluster 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 step* 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 $\tau$ (manual choice or
      automatic $\tau = \tau_H$)
Result: a collection of $c$ clusters $\mathcal{C} = \{R_1 \dots R_c\}$
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 $S_{i,j} = 1$, assign $r_i$ and $r_j$ to the same cluster
end
```

4.3 Refinement

As the discussion of the TGS and the example in Fig. 4 have 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 $S_{i,j} = 0$. 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 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) $R = \{r_{t_1}, \dots, r_{t_p}\}$ 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

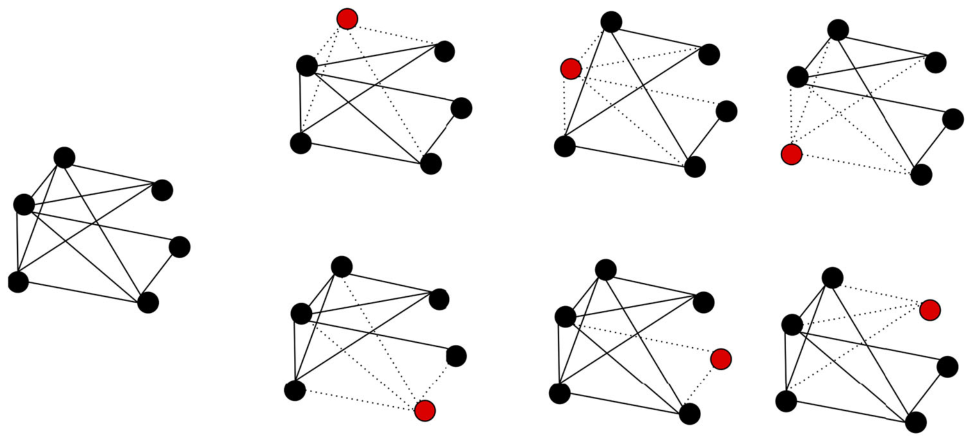


Fig. 5 An example of a cluster R that does not require refinement. Each node represents a record. In each test, we remove one and only one node from the cluster and apply TGS again. The red node represents the removed record r_{t_i}, the remaining black nodes make up the set $R(t_i)$. Notice that every 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 (color figure online)

$R(r_{t_i}) := \{r_{t_1}, \dots, r_{t_{i-1}}, r_{t_{i+1}}, \dots, r_{t_p}\}$. We define the *subclusters* $R_1, \dots R_q$ 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 Fig. 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 $\tilde{R}$:

$$s(\tilde{R}) := \begin{cases} \dfrac{\sum\limits_{\substack{i,j=1\\ i\neq j}}^{p} (S|_{\tilde{R}})_{i,j}}{\binom{p}{2}}, & \text{if } p \geq 2, \\ 0, & \text{if } p = 1. \end{cases} \tag{7}$$

Note that $s(\tilde{R}) = 1$ if $S|_{\tilde{R}} = \mathbf{1}^{p\times p}$ (it suffices if the off-diagonal elements satisfy this equality). In other words, a cluster has a strength of 1 if every pair of distinct 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:

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

Here, the sum is over all j such that R_j is a subcluster of $R(r_{t_i})$, and $q(i)$ is the (i-dependent) number of subclusters of $R(r_{t_i})$. In the unlikely event that the maximizer is not unique, we arbitrarily choose one of the maximizers as k^*. Since the strength of a subcluster measures the average similarity between the records in that subcluster, we want to keep the strength of the remaining subclusters as high as possible after removing r^* and optimizing the average strength is a good strategy to achieve that.

Step 3. After finding the optimal r^* to remove, we now must determine the subcluster to which to add it. We again use the strength of the resulting subclusters as a measure to decide this. 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_{l^*} to form $R^* := R_{l^*} \cup \{r^*\}$, where

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

In the rare event that the maximizer is not unique, we arbitrarily choose one of the maximizers as l^*. Choosing l^* in this way ensures that r^* is similar to the records in R_{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 (7), $s(\{r^*\}) = 0 < s(R^*)$, because r^* was connected to at least one other record in the original cluster R.

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

Figure 6 shows an example of how the refinement helps us to 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.

We only applied the refinement step process once (i.e., we executed Step 1 once and for each cluster identified in that step we applied Steps 2 and 3 once each). It is possible to iterate this three-step process until no more 'unstable' clusters are found in Step 1.

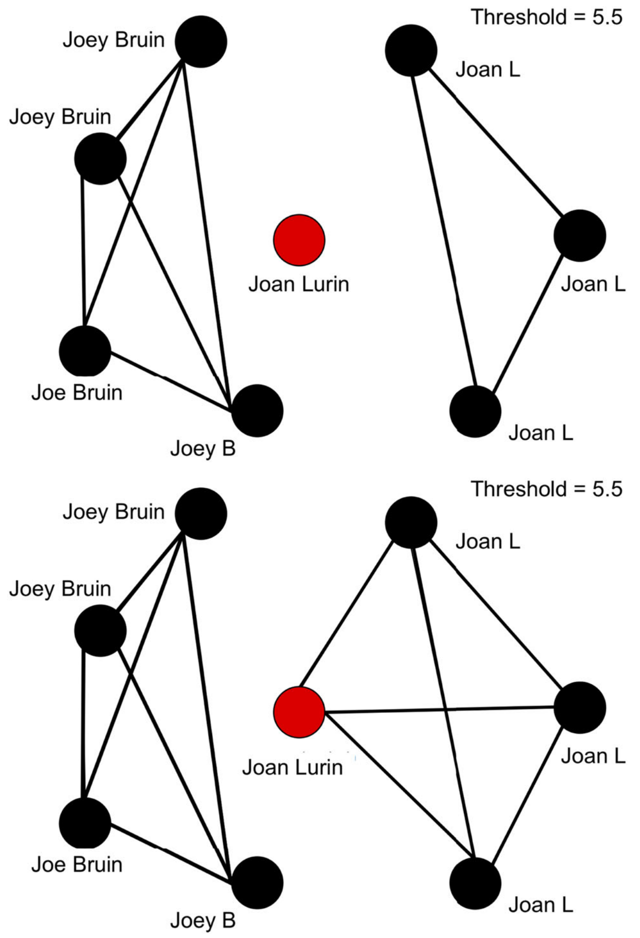


Fig. 6 An example of how refinement is used to improve our clusters. The left figure shows that by removing the record "Joan Lurin", we obtain the two desired subsets. The right figure shows that "Joan Lurin" is inserted back into the appropriate cluster. Note that we have not changed the threshold value τ during this process

Algorithm 6: Refinement

```
Data: $R = \{r_{t_1}, \dots, r_{t_n}\}$ 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 $R_1, \dots R_q$ 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 $l^* = \arg\max_j s(R_j \cup \{r^*\})$
    $\mathcal{R} = \{R_1, \dots, R_{l^*-1}, R_{l^*}, R_{l^*+1}, \dots, R_{q(k^*)}\}$
end
else
    Do not refine R: $\mathcal{R} = \{R\}$
end
```

5 Results

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. The other data sets currently can be found at [20]. Cora can also be accessed at [16]. RST and Cora are also used in [8] 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 of which we use seven: last name, first name, middle name, alias/moniker, operator license number (driver's license), social security number, and date of birth. A subset of this data set is used and described in more detail in [63]. 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 data set. The FI data set has many misspellings as well as different names that correspond to the same individual. 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 [59], 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 in the RST data set. 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.

Table 1 Summary of methods used

Name	Similarity matrix	Features	Ref.
TFIDF	ST using (5)	Words	No
TFIDF 3g	ST using (5)	3-grams	No
sTFIDF	ST using (3)	Words	No
sTFIDF 3g	ST using (3)	3-grams	No
sTFIDF ref	ST using (3)	Words	Yes
sTFIDF 3g ref	ST using (3)	3-grams	Yes

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. Equation (4) is always used to compute the similarity score, but the important difference is whether the soft TF-IDF matrix from (3) or the TF-IDF matrix from (5) is used in (4)

Cora The records in the *Cora Citation Matching data set*[11] are citations to research papers [43]. 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 in which the paper is published). 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 methods outputs a similarity matrix, which we then use in the TGS to create clusters.

To evaluate the methods, we use *purity* [27], *inverse purity*, their *harmonic mean* [25], the *relative error in the number of clusters*, *precision*, *recall* [9,15], the *F-measure* (or F_1 *score*) [6,64], *z-Rand score* [44,60], and *normalized mutual information* (NMI) [57], 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

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

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 favor many small clusters over a few large ones. In particular, if each record forms its own cluster, $\mathrm{Pur} = 1$. To counteract this bias, we also consider *inverse purity*,

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

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 $\mathrm{Inv} = 1$.

We combine purity and inverse purity in their *harmonic mean*,[12]

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

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,

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

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

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

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

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

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

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

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

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

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

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

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.

Another evaluation metric based on pair counting 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 hypergeometric distribution of equally likely assignments with the same number and sizes of clusters. For further details about the z-Rand score, see [44,60,63]. The *relative z-Rand score* of $\mathcal{C}$ is the z-Rand score of that clustering divided by the z-Rand score of $\mathcal{C}'$, so that the ground truth $\mathcal{C}'$ has a relative z-Rand score of 1.[13]

A final evaluation metric we consider, is *normalized mutual information* (NMI). To define this, we first need to introduce mutual information and entropy. We define the *entropy* of the collection of clusters $\mathcal{C}$ as

$$\mathrm{Ent}(\mathcal{C}) := -\sum_{i=1}^{c} \frac{|R_i|}{n} \log\left(\frac{|R_i|}{n}\right), \tag{8}$$

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

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

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

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

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 $\mathrm{Ent}(\mathcal{C})$ and $\mathrm{Ent}(\mathcal{C}')$ to give the *normalized mutual information*

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

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 practice this is avoided by adding a small number (e.g., the floating-point relative accuracy `eps` in MATLAB) to the argument of the logarithm in (8) for $\mathrm{Ent}(\mathcal{C})$ and $\mathrm{Ent}(\mathcal{C}')$.

Because we are testing our methods on data sets for which we have ground truth available, the metrics we use all compare our output with the ground truth. This would not be an option in a typical application situation in which the ground truth is not available. If the methods give good results in test cases in which comparison with the ground truth is possible, it increases confidence in the methods in situations with an unknown ground truth. Which of the metrics is the most appropriate in any given situation depends on the needs of the application. For example, in certain situations (for example when gathering anonymous statistics from a data set) the most important aspect to get right might be the number of clusters and thus the relative error in the number of clusters metric would be well suited for use, whereas in other situations missing out on true positives or including false negatives might carry a high cost, in which case precision or recall, respectively, or the F_1 score are relevant metrics. For more information on many of these evaluation metrics, see also [5].

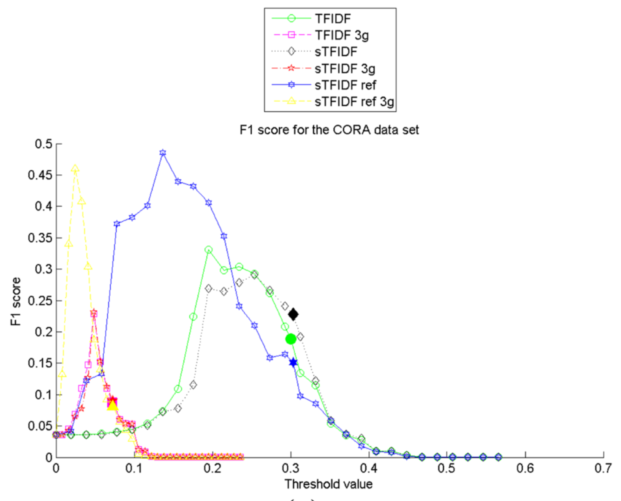
5.3 Results

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 metrics: 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 metric versus the threshold values, for the six different methods in one plot and (b) by plotting the evaluation scores obtained with a particular method versus the threshold values, for all five evaluation metrics in one plot. Since this paper does not offer space to present all figures, we show some illustrative plots and describe the main results in the text. In Sect. 6 we will discuss conclusions based on these results.

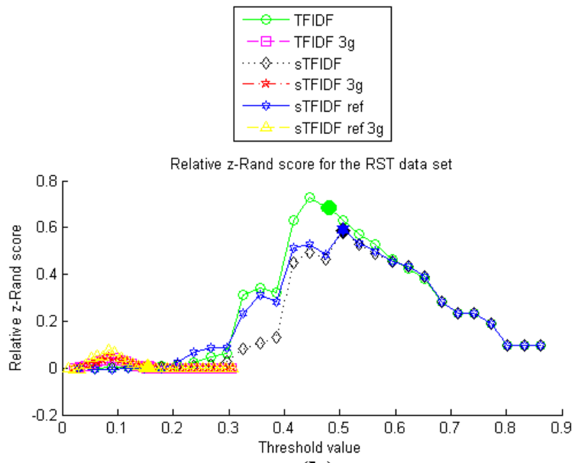
5.3.1 The methods

When we compare the different methods by plotting the scores for a particular evaluation metric versus the threshold value τ for all the methods in one plot (as can be seen, for example, in Fig. 7a), one notable attribute is that the behav-

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



(a)



(b)

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 corresponds to one of the six methods used. The filled markers indicate the metric's value at the automatically chosen threshold value τ_H 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. **a** The F_1 score for the Cora data set, **b** the relative z-Rand score for the RST data set

ior of the methods that use word features typically is quite distinct from that of the methods that use 3-gram features. This is not very surprising, 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. First, we compare the word-based methods with the 3-gram-based methods. On the FI data set the word feature-based methods outperform the 3-gram-based methods (judged on the basis of best-case performance, i.e., the optimal score attained over the full threshold range) for every evaluation metric by quite a margin, except for the NMI for which the margin is minimal (but still extant).

On both the RST and RST30 data sets, the word feature-based methods outperform the 3-gram feature-based methods on the pair counting-based metrics, i.e., F_1 score and relative z-Rand score (Fig. 7b), but both groups of methods perform equally well for the other metrics.

An interesting difference between the Cora data set and the other data sets is that while sTFIDF ref (see Table 1) does outperform sTFIDF 3g ref on the pair counting-based metrics for the Cora data set, the difference is much less pronounced than for the other data sets. The difference in the relative error in the number of clusters is more pronounced, however, in favor of the former method. Only on the relative error in the number of clusters does it perform somewhat worse than sTIDF ref. In fact, on all other metrics sTFIDF 3g ref outperforms the other two word-based methods (TFIDF and sTFIDF). The other 3-gram-based methods perform worse than their word-based counterparts on the pair counting metrics and on par with them on the other metrics.

Next, we compare the TF-IDF methods with the soft TF-IDF methods (without refinement step in all cases). There are very few observable differences between TFIDF 3g and sTFIDF 3g in any of the metrics or data sets, and where there are, the differences are minor.

The comparison between TFIDF and sTFIDF shows more variable behavior. The most common behavior among all metrics and data sets is that both methods perform equally well in the regions with very small or very large values of τ, although in some cases these regions themselves can be very small indeed. In the intermediate region, TFIDF usually performs better at small τ values, whereas sTFIDF performs better at larger τ values. The size of the these different regions, as well as the size of the difference in outcome can differ quite substantially per case. For example, in the case of NMI for the Cora data set, NMI and the harmonic mean of purity and inverse purity for the RST data set, and all metrics except the relative error in the number of clusters for the RST30 data set, TFIDF outperforms sTFIDF quite consistently in the regions where there is a difference.

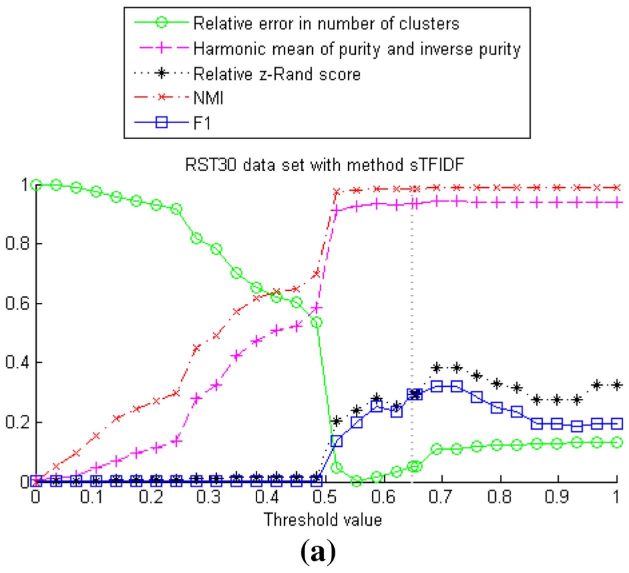
When it comes to the benefits of including the refinement step, the situation is again somewhat different depending on the data set. First, we compare sTFIDF 3g with sTFIDF 3g ref. For small threshold values including the refinement step is beneficial (except in a few cases when there is little difference for very small τ values). This is to be expected, since the refinement will either increase the number of clusters formed or keep it the same, so its effect is similar to (but not the same as) raising the threshold value. For larger τ values typically one of two situations occurs: either sTFIDF 3g outperforms sTFIDF 3g ref for intermediate τ values and there is little difference for higher τ values, or there is little difference on the whole range of intermediate and large τ values. The former occurs to a smaller or larger degree for all metrics except NMI for the Cora data set, for the harmonic mean of purity and inverse purity and the relative error in the number of clusters for the FI data set, and also for the relative error in the number of clusters for the RST30 data set. The other cases display the second type of behavior.

If we compare sTFIDF with sTFIDF ref there are three approximate types of behavior that occur. In the region with very small τ values the performance is usually similar for both methods, but this region can be very small. Next to this region, there is a region of small τ values in which sTFIDF ref outperforms sTFIDF. For the same reason as explained above, this is not surprising. This region can be followed by a region of the remaining intermediate and large τ values in which sTFIDF outperforms sTFIDF ref (the F_1 score and harmonic mean of purity and inverse purity for the FI data set), or by a region of the remaining intermediate and large τ values in which both methods are on par (NMI for the Cora data set, the F_1 score, the harmonic mean of purity and inverse purity, and NMI for the RST30 data set, and all metrics for the RST data set), or by first a region of intermediate τ values on which sTFIDF outperforms sTFIDF ref, followed by a region on which there is little difference between the methods (all other metric/data set combinations).

It is also noteworthy that all methods do significantly worse on RST30 than on RST, when measured according to the pair counting-based methods (the F_1 and relative z-Rand scores), while there is no great difference, if any, measured according to the other metrics. In this context, it is interesting to remember that RST30 is created by removing 30% of the entries from all but one of the fields of RST.

5.3.2 The metrics

When plotting the different evaluation metrics per method, we notice that the two pair counting-based metrics, i.e., the F_1 score and relative z-Rand score, behave similarly to each other, as do the harmonic mean of purity and inverse purity and the NMI. The relative error in the number of clusters is correlated with 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 (this is much less clear for the Cora data set as it is for the others). Those are also usually the lowest threshold values for which the harmonic mean and NMI attain their high(est) values. 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.



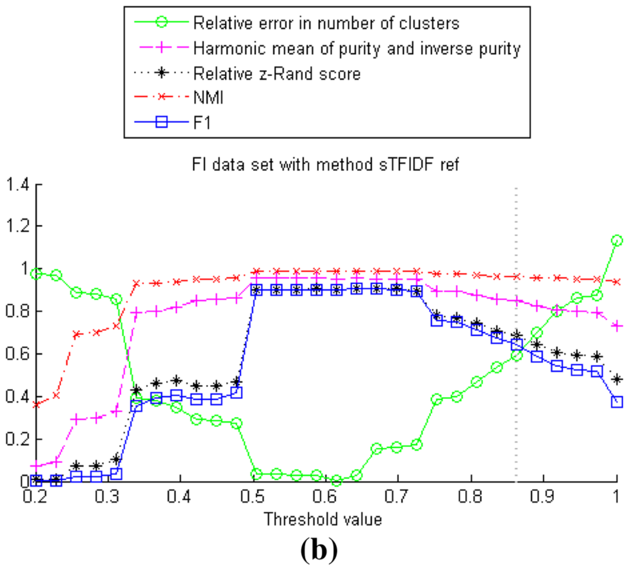


Fig. 8 Different evaluation metrics as a function of the threshold value τ, computed on two different data sets. Each of the five graphs in a plot corresponds to one of five evaluation metrics. The vertical dotted line indicates the automatically chosen threshold value τ_H for the method used. **a** Soft TF-IDF (on word-based features) without the refinement step applied to the RST30 data set, **b** Soft TF-IDF (on word-based features) with the refinement step applied to the FI data set

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 subtle 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., Fig. 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.

Another meaningful observation is that for τ values lower than the value at which the relative error in the number of clusters is minimal, TFIDF performs better for this metric than does sTFIDF. This situation is reversed for τ values higher than the optimal value. This can be understood from the difference between (3) and (5). Soft TF-IDF incorporates contributions into the similarity score not only from features that are exactly the same in two entries, but also from features that are very similar. Hence the soft TF-IDF similarity score between two entries will be higher than the TF-IDF score between the same entries and thus clusters are less likely to break up at the same τ value in the soft TF-IDF method than in the TF-IDF method. For τ values less than the optimal value the breaking up of clusters is beneficial, as the optimal cluster number has not yet been reached and thus TFIDF will outperform sTFIDF on the relative error in the number of clusters metric in this region. For τ larger than the optimal value, the situation is reversed.

5.3.3 The choice of threshold

On the RST and RST30 data sets, our automatically chosen threshold performs well (e.g., see Figs. 7b, 8a, 9a). It usually is close to (or sometimes even equal to) the threshold value at which some or all 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 than 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., Figs. 7a, 8b, and 9b. This can be explained by the left-skewedness of the H-value distribution, as illustrated in Fig. 3a. A good proxy for the volume of the tail is the ratio of number of records referring to unique entities to the total number of entries in the data set. For RST and RST30 this ratio is a high 0.87, whereas for FI it is 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.

We also compared whether TFIDF, sTFIDF, or sTFIDF ref performed better at the value $\tau = \tau_H$. Interestingly, sTFIDF ref never outperformed all the other methods. At best it tied with other methods: for the F_1 and relative z-Rand scores for

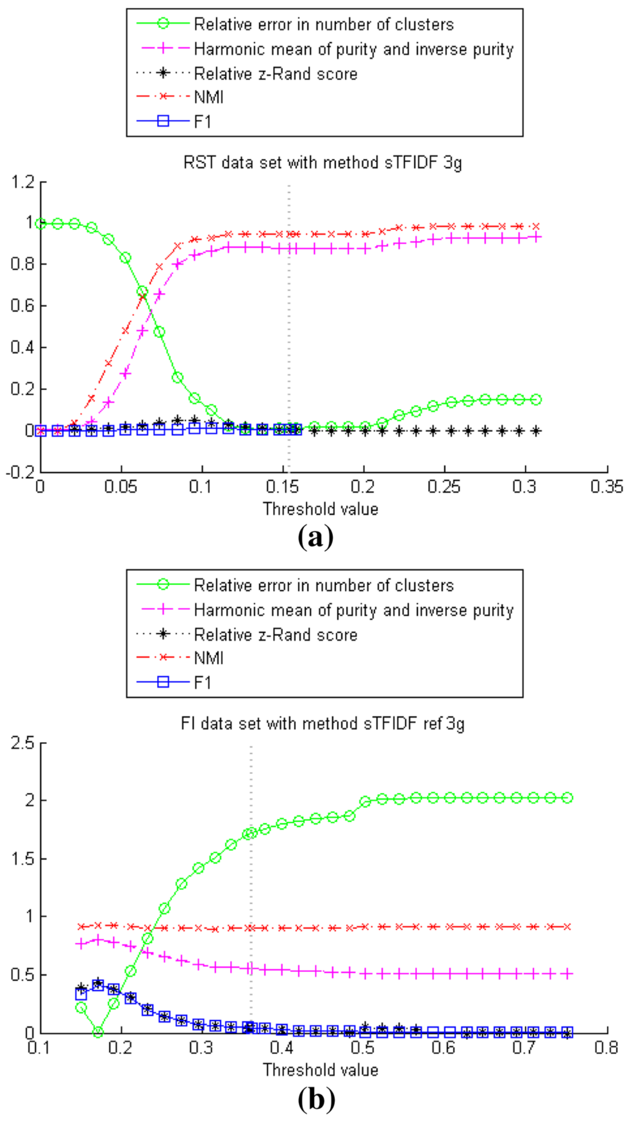


Fig. 9 Different evaluation metrics as a function of the threshold value τ, computed on two different data sets. Each of the five graphs in a plot corresponds to one of five evaluation metrics. The vertical dotted line indicates the automatically chosen threshold value for the method used. **a** Soft TF-IDF (on 3-gram-based features) without the refinement step applied to the RST data set. **b** Soft TF-IDF (on 3-gram-based features) with the refinement step applied to the FI data set

the RST30 data set it performed equally well as TFIDF; all three methods performed equally well for the NMI for the Cora data set, for the NMI and relative error in the number of clusters for the RST data set, and for NMI and the harmonic mean of the purity and inverse purity for the RST30 data set. TFIDF and sTFIDF tied for the F_1 and relative z-Rand scores for the FI data set. TFIDF outperformed the other methods on the RST data set for the F_1 and relative z-Rand scores, as well as the harmonic mean of purity and inverse purity. Finally, sTFIDF outperformed the other methods across the board for the FI data set, as well as for all metrics but the NMI for the Cora data set and for the relative error in the number of clusters for the RST30 data set. To recap, at $\tau = \tau_H$, the soft TF-IDF method seems to be a good choice for the Cora and FI data set, while for most metrics for the RST and RST30 data sets the TF-IDF method is preferred at $\tau = \tau_H$. (Remember that the value τ_H depends on the data set and the method).

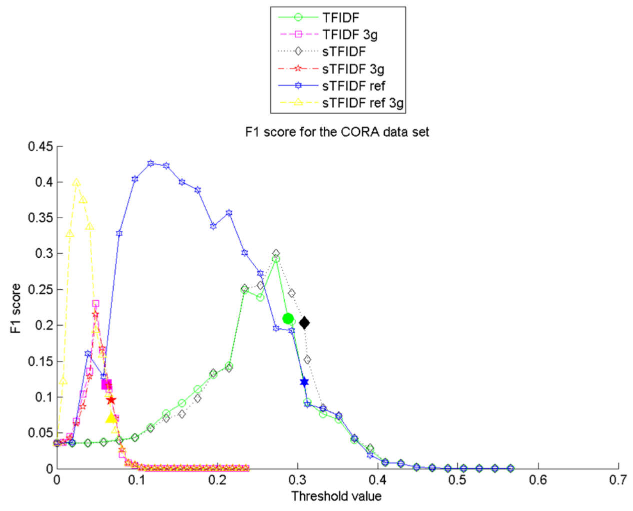
5.4 Results for alternative sparsity adjustment

At the end of Sect. 4.1, we described an alternative sparsity adjustment step, which replaces missing entries by the mode in each field. All the results reported so far use the sparsity adjustment step described in the first part of Sect. 4.1 (which we will call here the "original" step); in this section we describe the results obtained using the alternative sparsity adjustment step.

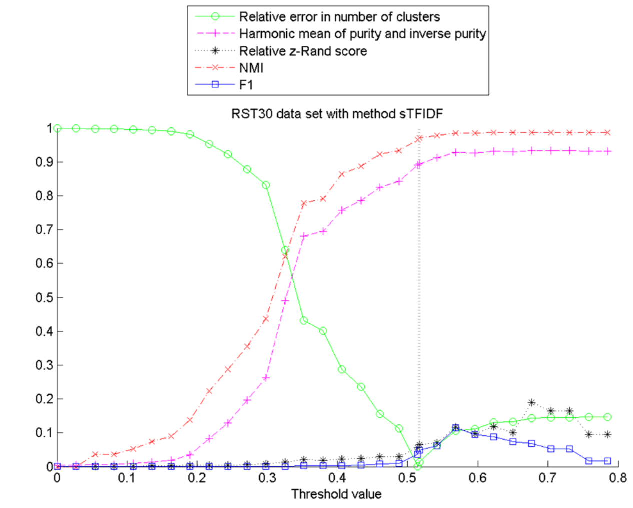
We chose to test this alternative sparsity adjustment step on the Cora and RST30 data sets. The former has a very small percentage of missing data (approximately 3%), while the latter has a high percentage (30% in all but one of the fields). We use the alternative sparsity adjustment step as part of each of the six methods discussed in this paper. We judge the output again using the same five metrics used above.

In all our tests on the Cora data set, there is very little if any difference in the performance of all the methods, with two notable exceptions: the two methods that include the refinement step perform considerably worse according to the two pair counting-based metrics (the F_1 and relative z-Rand scores) when incorporating the alternative sparsity adjustment step (and one minor, yet noticeable exception: TFIDF also performs worse with the alternative adjustment step when measured according to the F_1 score). Figure 10a shows the results corresponding to Fig. 7a, with as sole difference that in the former the alternative sparsity adjustment step is used, while in the latter the original step is incorporated into the methods.

In all our tests on the RST30 data set the 3-gram-based methods which use the alternative sparsity adjustment step perform very similarly to those that use the original adjustment step (with the difference that those similar results are obtained at lower threshold values when using the alternative step instead of the original adjustment step). The word-based methods also perform similarly using either sparsity adjustment step, when measured according to the relative error in the number of clusters, the harmonic mean of purity and inverse purity, and NMI. However, word-based methods perform worse with the alternative adjustment step on the pair counting metrics. Figure 10 shows the results corresponding to the same method as was used in Fig. 8a, with as sole difference the incorporation of the alternative sparsity adjustment step. The worsened performance of the alternative method



(a)



(b)

Fig. 10 Results obtained using the alternative sparsity adjustment step. **a** The F_1 score for the Cora data set; each listed method has the alternative sparsity adjustment step incorporated, **b** Soft TF-IDF (on word-based features) without the refinement step applied to the RST30 data set, incorporating the alternative sparsity adjustment step

with respect to the two pair counting metrics can be seen at the high end of the τ-range.

If any general conclusion can be drawn based on these tests, it is that there does not seem to be an advantage in using the alternative sparsity adjustment step instead of the original step; in some cases the resulting output is even worse, when measured according to the pair counting metrics.

A sparsity adjustment method that was not tested in this paper is to replace each missing entry by the same placeholder, e.g., "[]" or "void" [39]. This in effect will encourage records with missing entries to be clustered together but carries less risk of them being clustered together with other non-duplicate documents. This could be slightly beneficial in data sets with few missing entries, even though it is effectively a soft version of removing records with missing entries from the data set altogether.

6 Conclusions and suggestions for future work

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 conclusion from our tests is that there is no clear benefit to constructing the features the methods work on using 3-grams as opposed to white space separated ‘words’. Keeping the other choices (TF-IDF or soft TFIDF, refinement step or not) the same, using words for the features either outperforms the corresponding 3-gram-based method or performs equally well at worst (in terms of the optimal values that are achieved for the evaluation metrics). See, for example, the graphs in Fig. 7 or compare Figs. 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, without the refinement step). For low to intermediate threshold values TF-IDF performs better, for higher threshold values either soft TF-IDF performs better, or the difference between the two methods is so small as to be negligible. This behavior is not always very pronounced and, as described in Sect. 5.3.1, there are even cases in which TF-IDF outperforms soft TF-IDF for almost every threshold value.

The question whether or not to include the refinement step into a (word based) soft TF-IDF method also requires some care. At low τ values inclusion of the refinement step is beneficial, but at higher values the behavior can vary substantially per data set and metric, as described in Sect. 5.3.1. As a rule of thumb (but not a hard and fast rule) we can say that for the Cora and FI data sets there is a region of intermediate and/or high τ values at which including the refinement step is detrimental, whereas for the RST and RST30 data sets soft TF-IDF with refinement at worst performs similar to soft TF-IDF without refinement, but it performs better for certain τ values as well. This might partly be explained by the observation made in Sect. 5.3.3: the FI and Cora data sets have a much lower ratio of unique entities to total number of entries than the RST and RST30 data sets have. Since the refinement step creates extra clusters, including it can be detrimental for data sets that are expected to contain relatively few unique entries. This suspicion is strengthened by the fact that we see in our experiments that the growth in the relative error of the number of clusters when τ is increased past its optimal value (for that metric) is much larger for the FI and Cora data sets than for the RST and RST30 data sets.

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.

Since our automated threshold value τ_H is usually a value in the intermediate or higher end of the τ range, the discussion above suggests that at $\tau = \tau_H$ it is typically beneficial to use either TF-IDF or soft TF-IDF, in either case without the refinement step. The former is preferred for data sets with a high ratio of unique entities to number of entries, whereas the latter is preferred when this ratio is low. This is consistent with the observations at the end of Sect. 5.3.2 (since τ_H is close to the optimal τ value where the number of clusters is concerned for the RST and RST30 data sets and overshoots the optimal value for the Cora and FI data sets) and 5.3.3. This should only be treated as guidance and not as a hard and fast rule.

Future work could explore the possibilities of using methods that first project the data into a lower dimensional latent variable space to allow for duplicate detection in very high dimensional and large data sets, e.g., topic modeling techniques such as latent Dirichlet allocation [10] and non-negative matrix factorization [36], or the CenKNN method from [48]. An overview of other such methods is given in [19]. Where possible, new scalable hashing methods that allow for approximate matching might also be considered to reduce computational complexity in such settings [12]. These methods could reduce the number of comparisons made by quickly identifying specific subsets of pairs (e.g., those that must have similarity zero), but the construction of efficient hash functions is non-trivial and usually domain dependent. Further, the hash functions themselves incur a computational cost, so there is no guarantee of an overall speed up. Finding the right hash function for a given application and exploring the potential benefits of its use in a preprocessing step can be a topic for future research.

Acknowledgements We would very much like to thank George E. Tita and Matthew A. Valasik for their involvement in the collection of the FI data set and the construction of a ground truth clustering. We are grateful to them, as well as to P. Jeffrey Brantingham, for many fruitful discussions. We would also like to thank Brendan Schneiderman, Cristina Garcia-Cardona, and Huiyi Hu for their participation in the 2012 summer Research Experience for Undergraduates (REU) project from which this paper grew. We would also like to thank the two anonymous reviewers for their very insightful feedback which has lead to significant improvements in the published version of this paper. In particular, we thank one of the reviewers of the first draft of our paper for the suggestion to compare our method with the method of replacing missing entries in a field with the mode of that field. This research is made possible via ONR Grant N00014-16-1-2119, City of Los Angeles, Gang Reduction Youth Development (GRYD) Analysis Program, AFOSR MURI Grant FA9550-10-1-0569, and NSF Grants DMS-1045536, DMS-1417674, and DMS-1737770. Additionally, the second author received research grants from Claremont McKenna College, the Claremont University Consortium, Alfred P. Sloan Foundation and the NSF, which were not directly related to this research. We would like to acknowledge the following sources which were incorporated into or adapted for use in our code: [11,22,37,38] and [60,61].

References

1. Abraham, A.A., Kanmani, S.D.: A survey on various methods used for detecting duplicates in XML data. Int. J. Eng. Res. Technol. **3**(1), 354–357 (2014)
2. Ahmed, I., Aziz, A.: Dynamic approach for data scrubbing process. Int. J. Comput. Sci. Eng. **2**(02), 416–423 (2010)
3. Allison, P.D.: Imputation of categorical variables with PROC MI. In: SUGI 30 Proceedings. SAS Institute Inc. Paper 113-30 (2005)
4. Allison, P.D.: Missing data. In: Millsa, R.E., Maydeu-Olivares, A. (eds.) The SAGE Handbook of Quantitative Methods in Psychology, Chap. 4, pp. 73–90. SAGE Publications Ltd., London (2009). https://doi.org/10.4135/9780857020994.n4
5. Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Inf. Retr. **12**, 461–486 (2009)
6. Baeza-Yates, R., Ribeiro-Neto, B., et al.: Modern Information Retrieval. ACM Press, New York (1999)
7. Bano, H., Azam, F.: Innovative windows for duplicate detection. Int. J. Softw. Eng. Appl. **9**(1), 95–104 (2015)
8. 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)
9. 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)
10. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
11. Chen, M.: Normalized mutual information. http://www.mathworks.com/matlabcentral/fileexchange/29047-normalized-mutual-information (2010). Contact: mochen@ie.cuhk.edu.hk. Accessed 26 Feb 2015
12. Chi, L., Zhu, X.: Hashing techniques: a survey and taxonomy. ACM Comput. Surv. **50**(1), 11:1–11:36 (2017)
13. Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. Knowl. Data Eng. **24**(9), 1537–1555 (2012)
14. Cohen, W.W., Ravikumar, P.D., Fienberg, S.E., et al.: A comparison of string distance metrics for name-matching tasks. In: IIWEB'03 Proceedings of the 2003 International Conference on Information Integration on the Web, pp. 73–78 (2003)
15. 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)
16. Cora citation matching data set. http://people.cs.umass.edu/~mccallum/data.html. Accessed 24 Mar 2014
17. De Vries, T., Ke, H., Chawla, S., Christen, P.: Robust record linkage blocking using suffix arrays and bloom filters. ACM Trans. Knowl. Discov. Data (TKDD) **5**(2), 9 (2011)
18. Draisbach, U., Naumann, F.: A generalization of blocking and windowing algorithms for duplicate detection. In: 2011 International Conference on Data and Knowledge Engineering (ICDKE), pp. 18–24. IEEE (2011)
19. Dumais, S.T.: Latent semantic analysis. Annu. Rev. Inf. Sci. Technol. **38**(1), 188–230 (2004)
20. Duplicate detection, record linkage, and identity uncertainty: datasets. http://www.cs.utexas.edu/users/ml/riddle/data.html. Accessed 24 March 2014
21. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. **19**(1), 1–16 (2007)
22. 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. Accessed 4 Sept 2014
23. Friedman, J., Hastie, T., Tibshirani, R.: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York (2001)
24. Fu, Z., Zhou, J., Christen, P., Boot, M.: Multiple instance learning for group record linkage. In: Advances in Knowledge Discovery and Data Mining, pp. 171–182. Springer, Berlin, Heidelberg (2012)
25. González, E., Turmo, J.: Non-parametric document clustering by ensemble methods. Procesamiento del Lenguaje Natural **40**, 91–98 (2008)
26. Hall, R., Fienberg, S.E.: Privacy-preserving record linkage. In: Domingo-Ferrer, J., Magkos, E. (eds.) Privacy in Statistical Databases. Lecture Notes in Computer Science, vol. 6344. Springer, Berlin (2010)
27. 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)
28. Hassanzadeh, O., Chiang, F., Lee, H.C., Miller, R.J.: Framework for evaluating clustering algorithms in duplicate detection. Proc. VLDB Endow. **2**(1), 1282–1293 (2009)
29. Horton, N.J., Kleinman, K.P.: Much ado about nothing. Am. Stat. **61**(1), 79–90 (2007)
30. Huisman, M.: Imputation of missing network data: some simple procedures. J. Soc. Struct. **10**(1), 1–29 (2009)
31. Jaro, M.A.: Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. J. Am. Stat. Assoc. **84**(406), 414–420 (1989)

32. Jaro, M.A.: Probabilistic linkage of large public health data file. Stat. Med. **14**, 491–498 (1995)
33. Kannan, R.R., Abarna, D., Aswini, G., Hemavathy, P.: Effective progressive algorithm for duplicate detection on large dataset. Int. J. Sci. Res. Sci. Technol. **2**(2), 105–110 (2016)
34. Kim, H., Golub, G.H., Park, H.: Missing value estimation for dna microarray gene expression data: local least squares imputation. Bioinformatics **21**(2), 187–198 (2005). https://doi.org/10.1093/bioinformatics/bth499
35. Kim, H., Golub, G.H., Park, H.: Missing value estimation for DNA microarray gene expression data: local least squares imputation. Bioinformatics **22**(11), 1410–1411 (2006). https://doi.org/10.1093/bioinformatics/btk053
36. Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. **30**(2), 713–730 (2008)
37. Koehler, M.: matrix2latex, updated. http://www.mathworks.com/matlabcentral/fileexchange/4894-matrix2latex (2004). Accessed 24 March 2014
38. 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). Accessed 24 March 2014
39. Larose, D.T., Larose, C.D.: Discovering Knowledge in Data: An Introduction to Data Mining, 2nd edn. Wiley, Hoboken, NJ (2014)
40. Layek, A.K., Gupta, A., Ghosh, S., Mandal, S.: Fast near-duplicate detection from image streams on online social media during disaster events. In: India Conference (INDICON), 2016 IEEE Annual, pp. 1–6. IEEE (2016)
41. Leitao, L., Calado, P., Herschel, M.: Efficient and effective duplicate detection in hierarchical data. IEEE Trans. Knowl. Data Eng. **25**(5), 1028–1041 (2013)
42. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval, vol. 1. Cambridge University Press, Cambridge (2008)
43. McCallum, A.K., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. J. Inf. Retr. **3**(2), 127–163 (2000)
44. Meilă, M.: Comparing clusterings—an information based distance. J. Multivar. Anal. **98**, 873–895 (2007)
45. Monge, A.E., Elkan, C.P.: Efficient domain-independent detection of approximately duplicate database records. In: Proceedings of the ACM-SIGMOD Workshop on Research Issues in on Knowledge Discovery and Data Mining (1997)
46. Naumann, F., Herschel, M.: An introduction to duplicate detection. Synth. Lect. Data Manag. **2**(1), 1–87 (2010)
47. Nuray-Turan, R., Kalashnikov, D.V., Mehrotra, S.: Adaptive connection strength models for relationship-based entity resolution. J. Data Inf. Qual. (JDIQ) **4**(2), 8 (2013)
48. Pang, G., Jin, H., Jiang, S.: CenKNN: a scalable and effective text classifier. Data Min. Knowl. Discov. **29**(3), 593–625 (2015)
49. Papadakis, G., Ioannou, E., Palpanas, T., Niederee, C., Nejdl, W.: A blocking framework for entity resolution in highly heterogeneous information spaces. IEEE Trans. Knowl. Data Eng. **25**(12), 2665–2682 (2013)
50. Papadakis, G., Nejdl, W.: Efficient entity resolution methods for heterogeneous information spaces. In: 2011 IEEE 27th International Conference on Data Engineering Workshops (ICDEW), pp. 304–307. IEEE (2011)
51. Papenbrock, T., Heise, A., Naumann, F.: Progressive duplicate detection. IEEE Trans. Knowl. Data Eng. **27**(5), 1316–1329 (2015)
52. Pigott, T.D.: A review of methods for missing data. Educ. Res. Evaluat. **7**(4), 353–383 (2001)
53. Ramya, R., Venugopal, K., Iyengar, S., Patnaik, L.: Feature extraction and duplicate detection for text mining: a survey. Glob. J. Comput. Sci. Technol. **16**(5), 1–20 (2017)
54. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
55. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
56. Scannapieco, M., Figotin, I., Bertino, E., Elmagarmid, A.K.: Privacy preserving schema and data matching. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 653–664. ACM (2007)
57. Strehl, A., Ghosh, J.: Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J. Mach. Learn. Res. **3**, 583–617 (2002)
58. Tamilselvi, J.J., Gifta, C.B.: Handling duplicate data in data warehouse for data mining. Int. J. Comput. Appl. (0975–8887) **15**(4), 1–9 (2011)
59. Tejada, S., Knoblock, C.A., Minton, S.: Learning object identification rules for information integration. Inf. Syst. **26**(8), 607–633 (2001)
60. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: Comparing community structure to characteristics in online collegiate social networks. SIAM Rev. **53**(3), 526–543 (2011)
61. Traud, A.L., Kelsic, E.D., Mucha, P.J., Porter, M.A.: zrand. http://netwiki.amath.unc.edu/GenLouvain/GenLouvain (2011). Accessed 24 March 2014
62. 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, p. 779. American Medical Informatics Association (2006)
63. 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)
64. van Rijsbergen, C.J.: Information Retrieval, 2nd edn. Butterworth-Heinemann, Newton, MA (1979)
65. Watada, J., Shi, C., Yabuuchi, Y., Yusof, R., Sahri, Z.: A rough set approach to data imputation and its application to a dissolved gas analysis dataset. In: 2016 Third International Conference on Computing Measurement Control and Sensor Network (CMCSN), pp. 24–27. IEEE (2016)
66. Whang, S.E., Marmaros, D., Garcia-Molina, H.: Pay-as-you-go entity resolution. IEEE Trans. Knowl. Data Eng. **25**(5), 1111–1124 (2013)
67. 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)
68. Winkler, W.E.: The state of record linkage and current research problems. In: Statistical Research Division, US Census Bureau (1999)
69. Winkler, W.E.: Methods for record linkage and Bayesian networks. Technical Report, Series RRS2002/05, U.S. Bureau of the Census (2002)
70. Winkler, W.E.: Overview of record linkage and current research directions. Technical Report, Bureau of the Census (2006)
71. Xiao, C., Wang, W., Lin, X., Yu, J.X., Wang, G.: Efficient similarity joins for near-duplicate detection. ACM Trans. Database Syst. (TODS) **36**(3), 15 (2011)