Chapter 6
Classification

6.1 Overview

The classification of the candidate record pairs that were generated in the indexing step (Chap. 4) and compared in detail in the comparison step (Chap. 5) is primarily based on the similarity values in the comparison vectors of these record pairs, as illustrated in Fig. 6.1. The general idea is that the more similar two records are, the more likely they refer to the same real-world entity.

A classification approach can either be unsupervised or supervised. Unsupervised approaches classify pairs or groups of records based on similarities between them without having access to any information about the characteristics of true matching and true non-matching record pairs. Supervised approaches, on the other hand, require training data that are known true matches and true non-matches. Specifically, a set of comparison vectors is required, each of which has a match status (match or non-match) attached, to enable training of a supervised classifier. These comparison vectors need to be generated using the same comparison functions as the ones that will be used when pairs of records with unknown match status are compared. Obtaining or generating such training data, that need to be of high quality and cover a large variety of the possible similarity value combinations that can occur in comparison vectors, can be difficult, as will be discussed further in Sect. 7.1 in the context of evaluating the outcomes of data matching classification.

In certain matching situations, additional to the similarities between records, relational information between records might be available. Examples of such relational information include the lists of co-authors of scientific publications, or people who share the same land-line telephone number. Several recently developed classification techniques for data matching take such relational links or connections into account when building a clustering or graph-based classification model [31, 142, 155, 272], as will be discussed in Sects. 6.9 and 6.10. An alternative approach to model such relational information is to include it as exact similarities into the comparison vectors for candidate record pairs. As was discussed in Sect. 5.2, an exact comparison would for example, result in a normalised similarity value of 1.0 if two records have the

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	SimSum
a1	john	smith	18	miller st	dickson	12	11	1970	
b1	jonny	smyth	73	miller st	dixon	11	12	1970	
	0.6	0.8	0.0	1.0	0.6	0.5	0.5	1.0	5.0
a2	mary	harris	42	swamp rd	sydney	21	04	1918	
b2	mandy	garrett	42	smither pl	sydenham	27	04	1979	
	0.6	0.4	1.0	0.4	0.6	0.5	1.0	0.5	5.0

Fig. 6.1 Example candidate record pairs and their comparison vectors as calculated in the comparison step (Chap. 5), and their summed similarity values (SimSum)

same telephone number, or if two publications have been written by the same two co-authors.

When the classification into matches and non-matches is conducted independently for individual candidate record pairs rather than in a collective fashion [31], then several issues need to be considered. First, classifying each record pair independently can lead to sub-optimal match decisions. For example, assume two databases are matched, with the restriction that a record from the first database can only match to a maximum of one record in the second database (a topic that will be discussed further in Sect. 6.11). Now assume that record 'a1' (from the first database) is classified to match with record 'b4' (from the second database). Later on in the classification process, record 'a9' and 'b4' are also found to be a match, for example, if their similarity is higher than the similarity between 'a1' and 'b4'. The first classified match between 'a1' and 'b4' is therefore not the best match. Such sub-optimal decisions can occur if the classification is conducted in a greedy fashion where, once matched, a pair of records is not reconsidered for other matches.

A second issue is that independent match decisions can lead to contradictions, an issue known as transitive closure [190]. Assume in the deduplication of a database two record pairs, 'a1' and 'a2', and 'a1' and 'a3', have both been classified as matches, but the pair 'a2' and 'a3' has been classified as a non-match. This contradicts the assumption that when a group of three or more records are classified as being duplicates of each other, then none of the individual pairs in that group of records should be classified as a non-duplicate. How to handle this situation will be further discussed in Sect. 6.8.

After the classification step, depending upon the data matching or deduplication situation, records that were matched in certain applications need to be merged into compound new records [26]. This step itself is challenging, as it requires decisions to be made about how to merge individual attribute values that potentially can contradict with each other. A set of recently developed techniques to accomplish this will be presented in Sect. 6.12.

6.2 Threshold-Based Classification

The simplest way to classify candidate record pairs into the two classes of matches and non-matches (and possibly the third class of potential matches) is to sum the similarity values in their comparison vectors into a single total similarity value (called 'SimSum' in Fig. 6.1), and to then apply a similarity threshold (or two in the case where potential matches are considered) to decide into which class a candidate record pair belongs. Figure 6.2 shows an example histogram of the summed comparison vectors obtained from a deduplication of a real health data set that contained 175,211 records.

With two classes (matches and non-matches), a single classification threshold, t, is needed for classifying a record pair (r_i, r_j):

$$\begin{aligned} SimSum[r_i, r_j] \geq t &\Rightarrow [r_i, r_j] \rightarrow \text{Match}, \\ SimSum[r_i, r_j] < t &\Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}. \end{aligned} \tag{6.1}$$

With three classes (matches, non-matches and potential matches), two classification thresholds, t_l (lower) and t_u (upper), are needed, and a record pair (r_i, r_j) is classified according to:

$$\begin{aligned} SimSum[r_i, r_j] \geq t_u &\Rightarrow [r_i, r_j] \rightarrow \text{Match}, \\ t_l < SimSum[r_i, r_j] < t_u &\Rightarrow [r_i, r_j] \rightarrow \text{Potential Match}, \\ SimSum[r_i, r_j] \leq t_l &\Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}. \end{aligned} \tag{6.2}$$

Selecting a threshold (or thresholds) that results in high matching quality can either be done manually or (if training data are available) be learned in such a way that either the number of false matches or the number of false non-matches is minimised, or that the sum of both false matches and false non-matches is minimised. When candidate record pairs are also classified into potential matches, there is a trade-off between the quality of the classified matches and non-matches and the amount of manual clerical review of the potential matches that needs to be conducted. This topic will be discussed in more detail in Sect. 7.4.

If $t_u = t_l = t$ then the class of potential matches disappears and the classification follows Eq. 6.1. Increasing the upper threshold t_u and lowering the lower threshold t_l will result in less false matches and non-matches, but lead to a larger number of potential matches that need to be inspected and classified manually. On the other hand, lowering t_u and increasing t_l results in a smaller number of potential matches but likely also in an increased number of false matches and non-matches. This issue will be discussed further in Sect. 7.4.

The simple threshold-based classification has two major drawbacks. The first is that, assuming all similarity values are normalised between 0 and 1, all attribute similarities contribute in the same way towards the final summed similarity value. The importance of different attributes, as well as their discriminative power with

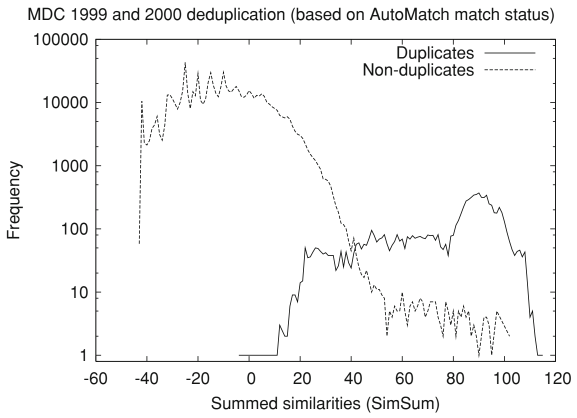

Fig. 6.2 Example histogram of the summed similarity values of a deduplication of a real health data set using comparisons on twelve attributes, with different weights assigned to the various comparisons. The true match status of this data set was determined earlier using the commercial data matching software AutoMatch [251], while the comparison vectors for this deduplication were generated using the FEBRL system [62]. A detailed description of the settings that were used for this deduplication exercise is provided in [71], from where this figure has been adapted

regard to distinguishing matches from non-matches, is not considered by such a simple summation approach. This drawback can be overcome by summing similarity values that are weighted, with different attributes given different weights according to their importance or discriminative power. The weighed sum is calculated by first multiplying the similarity value calculated on a certain attribute with the weight value for this attribute prior to the summation.

For example, assuming the following weights (empirically determined) are assigned to the attributes of Fig. 6.1: $w_{GivenName} = 2$, $w_{Surname} = 3$, $w_{StrNum} = 1$, $w_{StrName} = 3$, $w_{Suburb} = 2$, $w_{BDay} = 2$, $w_{BMonth} = 1$, and $w_{BYear} = 2$. The weighted summed similarities for the two given record pairs then become: $SimSum[a1, b1] = 2 \times 0.6 + 3 \times 0.8 + 3 \times 1.0 + 2 \times 0.6 + 2 \times 0.5 + 1 \times 0.5 + 2 \times 1.0 = 11.3$, and $SimSum[a2, b2] = 2 \times 0.6 + 3 \times 0.4 + 1 \times 1.0 + 3 \times 0.4 + 2 \times 0.6 + 2 \times 0.5 + 1 \times 1.0 + 2 \times 0.5 = 8.8$. The question of how to choose good weight values for the different attributes will be discussed in the next section.

The second drawback of summing similarities is that the detailed information contained in the individual similarity values is lost in the summation step (with both an unweighted or weighted approach). As the two example candidate record pairs in Fig. 6.1 show, despite having different similarity values both result in an unweighted sum of SimSum = 5.0. This is even though record pair (a2, b2) is unlikely to refer to the same entity given the differences in the attribute values of these two records. Giving different weights to attributes can to some degree overcome this drawback and lead to better classification results. More sophisticated classifiers, presented later in this chapter, are utilising the individual similarity values. This generally leads to improved matching quality compared to the simple threshold-based approach using summed similarities only.

6.3 Probabilistic Classification

This traditional classification approach to data matching, proposed in 1969 by Ivan Fellegi and Alan Sunter in their seminal paper [108], is commonly known as 'probabilistic record linkage'. Many data matching and deduplication systems that have been developed over the past four decades are based on the approach described in this paper.

The basic ideas of probabilistic record linkage were introduced by Newcombe et al. [198] in 1959 and detailed further by Newcombe and Kennedy in 1962 [197]. They recognised that in the absence of unique entity identifiers the attributes available in common in two databases (such as the names, addresses, or dates of birth of patients or customers) need to be used to match records. As the values in such attributes can be wrong, missing, or out of data, and because the number of values and their distributions can differ between attributes, different weights should be assigned to different attributes when they are used to calculate the similarities between records (as was discussed in the previous section). Newcombe and Kennedy also recognised that such weights should not only depend upon the general characteristics of attributes, but also on the actual attribute values in a certain candidate record pair. For example, if two records have a surname value 'Smith' then the weight given for this agreement of values should be smaller than the weight given to two records that both have a surname value 'Dijkstra', assuming the number of people with surname 'Dijkstra' is much smaller than the number of people with surname 'Smith' in the databases that are matched. This is because the likelihood that two randomly picked records have a surname value 'Smith' is much higher than the likelihood that they have a surname value 'Dijkstra'.

Fellegi and Sunter formalised these ideas and they developed a theory for record linkage that allows the calculation of weights for agreeing and disagreeing pairs of attribute values, which leads to an optimal decision making when record pairs are classified [108]. Probabilistic record linkage considers two databases (or files), **A** and **B**, and record pairs in the product space $\mathbf{A} \times \mathbf{B}$ that are to be classified into three classes: Matches (links), non-matches (non-links) and potential matches (potential links) [108, 143]. Record pairs classified as potential matches need to be manually assessed and classified in a clerical review process, as will be described in Sect. 7.4. Each record pair in $\mathbf{A} \times \mathbf{B}$ is assumed to correspond to either a true match or a true non-match. The space $\mathbf{A} \times \mathbf{B}$ is therefore partitioned into the set M of true matches and the set U of true non-matches. Formally,

$$\mathbf{A} \times \mathbf{B} = \{(a, b);\; a \in \mathbf{A}, b \in \mathbf{B}\} \tag{6.3}$$

consists of the two disjoint sets

$$M = \{(a, b); a = b, a \in \mathbf{A}, b \in \mathbf{B}\} \tag{6.4}$$

of true matches (also called the matched set), where both records a and b refer to the same real-world entity, and

$$U = \{(a, b); a \neq b, a \in \mathbf{A}, b \in \mathbf{B}\} \tag{6.5}$$

of true non-matches (also called the unmatched set), where the two records a and b refer to two different real-world entities.

The assumption is that records in **A** and **B** were generated based on one process for each of the two databases. Each record is assumed to refer to an individual in a population, such as a patient, customer, citizen and so on. The records in the two databases are drawn from two populations that have some overlap. For each member of the two populations, it is assumed that a record was generated with certain characteristics (such as certain name values, a certain date of birth and so on). The record generation process also led to errors and missing values in the records with certain distributions. As a result, it is possible that unmatched entities in **A** and **B** can be represented by two records that both have the same values in all their attributes. On the other hand, two records in **A** and **B** that refer to the same entity can have different values in some of their attribute.

When record pairs are compared (as was discussed in Chap. 5), a comparison vector, γ, is generated for each record pair. In the basic formulation of probabilistic record linkage, only binary comparisons are considered (with similarity value 1 when two attribute values are the same and 0 otherwise) [108]. Therefore, each γ corresponds to an agreement pattern in a comparison space, Γ. If each record pair was compared using K comparison functions, then each γ consists of a vector of K agreement or disagreement values. In total, assuming binary comparisons (i.e. exact matching) only, there will be 2^K different possible patterns.

For a given candidate record pair, r, probabilistic record linkage classification considers ratios of conditional probabilities, $P(\cdot|\cdot)$, of the form

$$R = \frac{P(\gamma \in \Gamma | r \in M)}{P(\gamma \in \Gamma \mid r \in U)} \tag{6.6}$$

where γ is an arbitrary agreement pattern in a comparison space Γ. Fellegi and Sunter [108] then propose the following decision rule:

$$\begin{aligned} R \geq t_u &\Rightarrow r \rightarrow \text{Match}, \\ t_l < R < t_u &\Rightarrow r \rightarrow \text{Potential Match}, \\ R \leq t_l &\Rightarrow r \rightarrow \text{Non-Match}. \end{aligned} \tag{6.7}$$

The two cutoff thresholds t_l and t_u are determined by a priori error bounds on false matches and false non-matches [108, 143]. It is easy to see that the three rules in Eq. 6.7 make intuitive sense. If γ for a certain candidate record pair r mostly consists of agreements, then the ratio R in Eq. 6.6 would be large, because it is more likely that $r \in M$ rather than $r \in U$, and the pair is more likely designated as a match. On the other hand, for a γ that primarily consists of disagreements the ratio R would be small, because it is more likely that $r \in U$ rather than $r \in M$, and thus the pair will be designated as a non-match.

Fellegi and Sunter showed that with fixed bounds on the errors in the match and non-match regions of R the decision rule in Eq. 6.7 is optimal, in that the middle region of potential matches is minimised [108].

Calculating the conditional probabilities in Eq. 6.6 is a crucial aspect of the probabilistic record linkage approach. It is commonly assumed that these probabilities are conditionally independent for the different attributes that are used in the comparison step to calculate the agreement patterns γ. Under this assumption, an individual agreement weight, w_i, $1 \leq i \leq K$ can be calculated for each attribute (or field) i based on the m- and u-probabilities

$$m_i = P([a_i = b_i,\ a \in \mathbf{A},\ b \in \mathbf{B}] \mid r \in M), \tag{6.8}$$

and

$$u_i = P([a_i = b_i,\ a \in \mathbf{A},\ b \in \mathbf{B}] \mid r \in U), \tag{6.9}$$

where a_i and b_i are the values in attribute i that are being compared. Equation 6.8 is the probability that two records have the same value in attribute i given the pair is a true match (i.e. both records refer to the same entity). On the other hand, Eq. 6.9 is the probability that two records have the same value in attribute i given the pair is a true non-match (i.e. the two records refer to different entities).

The probabilities in Eqs. 6.8 and 6.9 are called the m- and u-probabilities, respectively, and they are also known as the matching parameters [143]. Based on these two probabilities, the individual weight w_i for attribute i is calculated as:

$$w_i = \begin{cases} log_2(\frac{m_i}{u_i}) & \text{if} \quad a_i = b_i, \\ log_2(\frac{(1-m_i)}{(1-u_i)}) & \text{if} \quad a_i \neq b_i. \end{cases} \tag{6.10}$$

To make a simple example, assume the two databases **A** and **B** contain an attribute ‘MonthOfBirth’ (MoB) with twelve possible values ‘January’ to ‘December’. Assume also that it is known that in both **A** and **B** this attribute contains 3 % errors, i.e. 3 % of all month of birth values have been recorded wrongly. The likelihood that two records, $a \in \mathbf{A}$ and $b \in \mathbf{B}$, that are known to refer to the same entity ($(a, b) \in M$) have the same month of birth value is 97 %. Therefore, $m_{MoB} = 0.97$. The likelihood that two records referring to the same entity have a different month of birth is $(1 - m_{\text{MoB}}) = 0.03$ (3 %). For two records $a \in \mathbf{A}$ and $b \in \mathbf{B}$ that are known to refer to two different entities ($(a, b) \in U$), the likelihood that their month of birth value is the same is $1/12 = 0.083$, because there is a $1/12$ (8.3 %) chance that two randomly picked individuals in a population have the same month of birth. Therefore, $u_{MoB} = 0.083$. Conversely, the likelihood that two randomly picked records that refer to two different entities have a different month of birth is $11/12 = 0.917 = (1-u_{MoB})$ (91.7 %). Using Eq. 6.10, if two records have the same value in the 'MonthOfBirth' attribute, then the corresponding weight (called match or agreement weight) is calculated as $w_{MoB} = log_2(0.97/0.083) = 3.54$, while if two records have different month of birth values then the weight (called non-match or disagreement weight) is calculated as $w_{MoB} = log_2(0.03/0.917) = -4.92$.

Assuming conditional independence, the overall weight for a record pair r can be calculated by summing the weights w_i over the K attribute match/non-match weights:

$$log_2(R) = \sum_{i}^{K} w_i. \tag{6.11}$$

Figure 6.2 shows an example histogram of such summed weights for the deduplication of a real health data set. As can be seen, the number of non-matches (non duplicates) is much larger than the number of matches (duplicates), as would be expected.

In real-world data, it is likely that there are some dependencies between attributes. For example, records that have the same post- or zipcode with a high likelihood will also have the same locality (suburb or town) name, because in many countries most post- or zipcodes are contained within a certain locality. Records that have the same post- or zipcode then potentially also more likely have the same street name. However, despite most real-world data violating the conditional independence assumption, practical data matching projects have shown that good matching quality can still be achieved under this assumption [143].

One of the difficulties with probabilistic record linkage is the accurate calculation or estimation of the error rates required in Eqs. 6.8 and 6.9. Sometimes these probabilities are known from the manual assessment of the quality of the databases to be matched, or from a manual evaluation of an earlier matching of the same databases. Alternatively, these estimates can be calculated based on population estimates, like in the month of birth example given above. Herzog et al. [143] discuss in detail how the m_i and u_i parameters can be estimated using either data from prior data matching projects or by employing the unsupervised expectation–maximisation (EM) algorithm.

Extensions to the basic Fellegi and Sunter approach to probabilistic record linkage include allowing for approximate comparisons of attribute values that result in similarity values in the agreement patterns γ rather than only agreement and disagreement values. Porter and Winkler [215, 279, 286] showed that modifying the m- and u-probabilities in Eqs. 6.8 and 6.9 using the normalised similarities (between 0.0 and 1.0) calculated by approximate string comparison algorithms can lead to significant improvements in matching quality.

The second extension is concerned by taking the frequency of attribute values into account when calculating the m- and u-probabilities [108, 286]. The intuition behind this idea is that the more frequent an attribute value is in a database, the less discriminative this value is for classifying a record pair as a match or non-match. The example using the surname values 'Smith' and 'Dijkstra' given at the beginning of this section has already illustrated this issue. Match and non-match weights should be adjusted according to the frequency of occurrence of individual attribute values, with lower m-probabilities for more frequent attribute values [108]. Herzog et al. [143] provide a detailed discussion of how frequency-based matching parameters can be calculated.

Winkler developed a method that combines the traditional Fellegi and Sunter approach to probabilistic record linkage with Bayesian networks [281]. Bayesian networks [138] can model selected dependencies between attributes. In general they, however, require training data (in the case of data matching in the form of record pairs with known true match status). Using both labelled and unlabelled training data, a modification of the EM algorithm was used by Winkler to estimate parameter settings. Viewing probabilistic record linkage from a Bayesian perspective has also been discussed by Fortini et al. [112] and Herzog et al. [143].

6.4 Cost-Based Classification

In the traditional probabilistic record linkage approach the two thresholds t_l and t_u are set such that the overall number of misclassified candidate record pairs is minimised. Two types of errors can occur (as will be further discussed in Chap. 7). First, a pair of records that refers to the same real world entity (and therefore is a true match) is classified as a non-match. Second, a pair of records that refers to two different entities (and thus is a true non-match) is classified as a match. Traditionally, it is assumed both types of errors have the same costs.

In many data matching and deduplication applications, however, these two types of errors have different costs [129, 263]. For example, imagine a health application where patient data from several databases (that contain information, for example, about prescriptions, hospital admissions and doctor consultations) are matched. Assuming these databases were matched such that each patient in the matched database is assumed to have a serious illness based on their medical history. These patients are invited by the hospital for a series of special medical tests to confirm if they do have this illness or not. Testing a patient for this illness will incur a certain amount of money, possibly in the hundreds or even thousands of dollars. Therefore, each patient that has been matched falsely will mean an increase in costs for an additional test that might have been unnecessary. On the other hand, each patient that was not classified as a match but who potentially has this serious illness might die because they are not given the medical test that could confirm if they have the illness or not. The costs for such a missed true match can therefore be the loss of life of an individual.

Another, less dramatic, example can be found in marketing, where often databases are matched to generate mailing lists of potential customers who are interested in certain topics, based on their shopping history (like sporting, gardening, music or reading). The cost of sending an advertisement flyer about a certain topic to somebody who is not interested in this topic is very small, compared to not sending the flyer to somebody who will probably respond to the advertisement [263]. Missing such a customer can potentially result in a significant loss in profit.

Table 6.1 Costs associated with various matching decisions as proposed by Verykios et al. [263]

Cost	Classification	True match status
$c_{U,M}$	Non-Match	True match (M)
$c_{U,U}$	Non-Match	True non-match (U)
$c_{P,M}$	Potential Match	True match (M)
$c_{P,U}$	Potential Match	True non-match (U)
$c_{M,M}$	Match	True match (M)
$c_{M,U}$	Match	True non-match (U)

As shown in these two examples, clearly there can be different costs associated with false matches and false non-matches. A cost-optimal decision model based on a Bayesian approach has been developed by Verykios et al. [263]. In this approach, the decision rule for an agreement pattern γ in Eq. 6.7 is formulated in a Bayesian setting:

$$\begin{aligned} P(\gamma \in \Gamma | r \in M) &\geq P(\gamma \in \Gamma | r \in U) \Rightarrow r \rightarrow \text{Match}, \\ P(\gamma \in \Gamma | r \in M) &< P(\gamma \in \Gamma | r \in U) \Rightarrow r \rightarrow \text{Non-match}. \end{aligned} \tag{6.12}$$

As shown in Table 6.1, different costs can be assigned to each of the six decision outcomes in the traditional Fellegi and Sunter model, where record pairs are classified into matches, non-matches and potential matches. The objective of a cost optimal decision rule is then to minimise the overall cost c:

$$\begin{aligned} c = & c_{U,M} \cdot P(r \in \text{Non-Match}, r \in M) + c_{U,U} \cdot P(r \in \text{Non-Match}, r \in U) + \\ & c_{P,M} \cdot P(r \in \text{Potential Match}, r \in M) + c_{P,U} \cdot P(r \in \text{Potential Match}, r \in U) + \\ & c_{M,M} \cdot P(r \in \text{Match}, r \in M) + c_{M,U} \cdot P(r \in \text{Match}, r \in U), \end{aligned} \tag{6.13}$$

where $P(x, y)$ is the joint probability that a record pair r has been classified into class x (with $x \in$ {Non-Match, Potential Match, Match}) while the true match status of r is y (with $y \in \{M, U\}$). Bayes theorem can then be applied to replace these six probabilities with the probabilities of a certain match decision, given the true match status and the a priori probabilities of $P(M)$ and $P(U)$:

$$P(r = x, r = y) = P(r = x \mid r = y) \cdot P(r = y), \tag{6.14}$$

with x and y being a value of the corresponding two sets given above. The probabilities $P(r = x \mid r = y)$ and $P(r = y)$ can both be estimated using training data that are available in the form of record pairs with known true match status [263]. An optimal decision rule, similar to the one given in Eq. 6.7, can then be developed, which for different values of the different costs provides an overall cost-optimal decision [263].

Cost-based classification is not just possible for the probabilistic record linkage approach as was presented in this section, but for other classification techniques for data matching as well. In rule-based classifiers (discussed next), rules can for example, be reordered such that the rules that classify candidate record pairs into matches are evaluated before rules that classify them into non-matches, while for many supervised machine learning classifiers different costs for different classes can be incorporated into the learning process.

6.5 Rule-Based Classification

A rule-based classification approach is different to the probabilistic approaches presented in the previous two sections. It employs rules that classify candidate record pairs into matches and non-matches (and maybe potential matches that are passed on for manual clerical review) [82, 141, 195]

A rule-based classifier can be applied on the similarity values of the comparison vectors generated in the comparison step. Rules are made of individual tests on certain similarity values that are combined with conjunctions (logical and), disjunctions (logical or) and negations (logical not). Figure 6.3 shows an example set of such rules.

The form of a rule is $P \Rightarrow C$, where P is a predicate that is applied on the similarity values (as available in a comparison vector) for a record pair (r_i, r_j), and C is the classification outcome of the pair (r_i, r_j). The predicate P is a boolean expression of the general form:

$$P = (term_{1,1} \vee term_{1,2} \vee \ldots) \wedge \ldots \wedge (term_{n,1} \vee term_{n,2} \vee \ldots). \qquad (6.15)$$

P is written in conjunctive normal form as a conjunction of disjunctions of terms [195]. Each term is a test applied on the similarity value of a single element in a comparison vector of the record pair (r_i, r_j). For example, a term can be a test such as $s(\text{GivenName})[r_i, r_j] \geq 0.7$), i.e. if the similarity value for the record pair r_i and r_j for the given name attribute is equal to or greater than 0.7. In Fig. 6.3, each disjunction only contains one term.

The classification outcome C of a rule assigns a candidate record pair into the class given in C when a rule is triggered (i.e. when the predicate P is true). A rule system can either consist of rules that classify record pairs into matches only, or of rules that classify pairs into matches, non-matches and even potential matches. In the first case, all record pairs that are not covered by any rule will implicitly be classified as non-matches. For the second case, a rule set needs to cover all possible values in the similarity values of the comparison vectors, as there is no default class, or a default class needs to be set explicitly (for example in the form of a rule with an empty predicate P and where C classifies a record pair as a non-match).

If a rule set only contains rules that classify candidate record pairs as matches, then the ordering of these rules is irrelevant. On the other hand, if a rule set consists of rules that classify record pairs into more than one class, then the ordering of rules is crucial. For a given record pair, the first rule where the predicate P becomes true (the first rule that is triggered or 'fired') is the rule that classifies the pair.

$$\begin{aligned}&(s(\text{GivenName})[r_i,r_j] \geq 0.9) \ \wedge \ (s(\text{Surname})[r_i,r_j] = 1.0)\\ &\quad \wedge \ (s(\text{BMonth})[r_i,r_j] = 1.0) \ \wedge \ (s(\text{BYear})[r_i,r_j] = 1.0) \ \Rightarrow \ [r_i,r_j] \rightarrow \ \text{Match}\end{aligned}$$

$$\begin{aligned}&(s(\text{GivenName})[r_i,r_j] \geq 0.7) \ \wedge \ (s(\text{Surname})[r_i,r_j] \geq 0.8)\\ &\quad \wedge \ (s(\text{BDay})[r_i,r_j] = 1.0) \ \wedge \ s(\text{BMonth})[r_i,r_j] = 1.0)\\ &\quad\quad \wedge \ (s(\text{BYear})[r_i,r_j] = 1.0) \ \Rightarrow \ [r_i,r_j] \rightarrow \ \text{Match}\end{aligned}$$

$$\begin{aligned}&(s(\text{GivenName})[r_i,r_j] \geq 0.7) \ \wedge \ (s(\text{Surname})[r_i,r_j] \geq 0.8)\\ &\quad \wedge \ (s(\text{StrName})[r_i,r_j] \geq 0.8) \ \wedge \ (s(\text{Suburb})[r_i,r_j] \geq 0.8) \ \Rightarrow \ [r_i,r_j] \rightarrow \ \text{Match}\end{aligned}$$

$$\begin{aligned}&(s(\text{GivenName})[r_i,r_j] \geq 0.7) \ \wedge \ (s(\text{Surname})[r_i,r_j] \geq 0.8)\\ &\quad \wedge \ (s(\text{BDay})[r_i,r_j] \leq 0.5) \ \wedge \ (s(\text{BMonth})[r_i,r_j] \leq 0.5)\\ &\quad\quad \wedge \ (s(\text{BYear})[r_i,r_j] \leq 0.5) \ \Rightarrow \ [r_i,r_j] \rightarrow \ \text{Non-Match}\end{aligned}$$

$$\begin{aligned}&(s(\text{GivenName})[r_i,r_j] \geq 0.7) \ \wedge \ (s(\text{Surname})[r_i,r_j] \geq 0.8)\\ &\quad \wedge \ (s(\text{StrName})[r_i,r_j] \leq 0.6) \ \wedge \ (s(\text{Suburb})[r_i,r_j] \leq 0.6) \ \Rightarrow \ [r_i,r_j] \rightarrow \ \text{Non-Match}\end{aligned}$$

Fig. 6.3 An example set of classification rules that could be applied on the comparison vectors from Fig. 2.6 shown on page 31. Conjunctions (logical and) are shown as $\wedge$ and disjunctions (logical or) as $\vee$. '$s(\cdot)$' refers to a similarity value taken from the comparison vector for a given record pair. The first three rules classify a pair of records r_i and r_j as a match if their name values are similar, and either their dates of birth or their addresses are similar as well. On the other hand, the last two rules classify a pair as a non-match if their name is similar but they have either a different date of birth or a different address

Ideally, each rule in a set of rules should be of high accuracy and high coverage [135]. A high accuracy means that a rule that classifies record pairs into a certain class should mostly cover pairs that do belong to this class but not pairs that belong into another class. In order to be able to assess the accuracy of rules, candidate record pairs and their true match status (match or non-match) must be available. Without the true match status it is not possible to assess the accuracy of rules. A high coverage means that the predicate P of a rule covers a large portion of all candidate record pairs. A rule which has a coverage of 10 % is triggered (i.e. its predicate P is true) for 10 % of all candidate record pairs. A rule with an empty predicate P (i.e. no test on any similarity value) would have a coverage of 100 %. The more specific a rule is (i.e. the more conditions are tested in the predicate P) the lower the coverage of a rule generally becomes. More specific rules are usually more accurate, while less specific rules often have lower accuracy because they cover a larger number of candidate record pairs that are in both the match and non-match classes.

The quality of a rule set can be measured by its overall accuracy and its coverage [135]. Additionally, a smaller rule set is generally preferable over a larger rule set, because a smaller number of rules is easier to maintain. Because rules are depending upon the characteristics and the quality of the data that are matched or deduplicated, either a new set of rules needs to be developed for each new database, or an existing set of rules needs to be adjusted when data with different characteristics are matched or deduplicated.

This raises the question of how a set of rules can be generated to achieve a high classification accuracy of the compared candidate record pairs. The two basic approaches are to either develop a rule set manually or to learn a set of rules from training data.

- The traditional approach to generating rules is to manually develop them based on domain knowledge of the databases to be matched or deduplicated. Developing such rules is usually done hand in hand with selecting appropriate indexing approaches and comparison functions, because both of these will affect the candidate record pairs that are generated and the similarity values in their corresponding comparison vectors.
  Manually generating rules is a labour-intensive process that is generally iterated over many variations of potential rules. These rules need to be tested and manually evaluated using some form of training data that contain the true match status of candidate record pairs. If such training data are not available (as is the case in many real world data matching situations, as will be discussed further in Chap. 7), then the evaluation of each rule requires manual inspection of all candidate record pairs that are covered by a rule, and for each covered pair it needs to be manually decided if the classification is correct or not. This is a tedious and labour-intensive process.
- An alternative approach to generating a set of rules is to learn them from training data that consist of candidate record pairs and their true match status. Similar to the learning of blocking keys discussed in Sect. 4.12, the learning of rules can be accomplished by employing a sequential covering algorithm [135], where a set of rules that cover one class (usually the candidate record pairs that correspond to matches) is learned first, followed by rules that cover the other class (the non-matches).
  One rule is learned after another, by starting with an empty predicate P for a rule and evaluating its accuracy and coverage. Candidate rules are then generated by adding a term to P based on the similarity values in the different elements of comparison vectors. Such candidate rules could for example be (similar to the example given in Fig. 6.3):

$$\begin{aligned}(s(\text{GivenName})[r_i, r_j] = 1.0) &\Rightarrow [r_i, r_j] \rightarrow \text{Match}\\ (s(\text{Surname})[r_i, r_j] = 1.0) &\Rightarrow [r_i, r_j] \rightarrow \text{Match}\\ (s(\text{StrName})[r_i, r_j] = 1.0) &\Rightarrow [r_i, r_j] \rightarrow \text{Match}\\ (s(\text{Suburb})[r_i, r_j] = 1.0) &\Rightarrow [r_i, r_j] \rightarrow \text{Match}\end{aligned}$$

  The best candidate rule (according to some criteria that takes accuracy and coverage into account [135]) is selected, and this becomes the new base rule which will be expanded with new candidate terms in the next step [135]. Assuming for example, that the first of the four above rules was the best candidate, the next set of expanded candidate rules could consist of:

$$(s(\text{GivenName})[r_i, r_j] = 1.0) \wedge (s(\text{Surname})[r_i, r_j] \geq 0.8) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$
$$(s(\text{GivenName})[r_i, r_j] = 1.0) \wedge (s(\text{StrName})[r_i, r_j] \geq 0.8) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$
$$(s(\text{GivenName})[r_i, r_j] = 1.0) \wedge (s(\text{Suburb})[r_i, r_j] \geq 0.8) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$

This process of testing candidate rules and expanding the best candidate with another term is repeated until a stopping criteria is fulfilled. All candidate record pairs that are covered by the latest generated rule are then removed from the training set, and if candidate record pairs are left in the training set then a new rule is learned.

Two data matching research prototypes were developed in the late 1990s that were employing rule-based classification approaches. A system based on an extension of SQL that allows rule-based matching operators to be defined was proposed by Galhardas et al. [117]. These matching operators included similarity predicates, the setting of thresholds, as well as normal SQL statements. Complex matching statements were therefore written using SQL statements. A related approach was the WHIRL system developed by Cohen [81], which combined similarity calculations based on cosine similarity (described in Sect. 5.8) with conjunctive rules applied on record attributes.

More recently, Schewe and Wang [236] proposed a reasoning approach to acquire knowledge about entities stored in different databases by identifying objects through knowledge patterns. Such patterns can capture details such as abbreviations and variations in title, name and address values. An advantage of knowledge patterns is that they can capture knowledge at different levels of abstractions (i.e. not just at the level of individual entities but also at the level of attributes), and by using the contexts of where patterns occur (i.e. taking the relations between different patterns into account). Knowledge patterns allow a user to identify, for example, the types of name and address variations that commonly occur in two databases that are to be matched, which in turn can facilitate the development of rule-based classifiers that determine if two records correspond to the same entity or not depending upon the variations in their attribute values.

6.6 Supervised Classification Methods

When the compared candidate record pairs are only classified into matches and non-matches (but not potential matches), then this classification is known as a *binary classification problem*. Further, if training data in the form of record pairs with their true match status (match or non-match) are available, then a supervised classification approach can be employed to train a classification model using these training data. The trained model is then used to classify record pairs with an unknown match status into matches and non-matches. Many binary classification techniques have been developed by the AI, machine learning and data mining communities over the past few decades [135, 189], and several of these techniques have been applied in the area of data matching and deduplication. This section provides an overview of this work and highlights important issues that need to be considered when a supervised classification technique is used to classify record pairs.

Most classification techniques (including the probabilistic record linkage, cost-based and rule-based approaches discussed earlier in this chapter), classify each compared record pair individually and independently from all other record pairs (Sect. 6.10 below covers techniques that are aimed at classifying all compared record pairs in a collective approach). From the classification point of view, each compared record pair is represented by its comparison vector that contains the individual similarity values that were calculated in the comparison step (as was discussed in Sect. 5.16). These comparison vectors correspond to the *feature vectors* (the notation used in machine learning or data mining) that are employed to train a classification model, and to classify record pairs with unknown match status. Figure 6.4 shows such a set of comparison vectors and their true match status. A supervised classification approach consists of three steps [135].

1. A supervised classification technique is selected and a classification model is built by training the classifier using available training data which include the known true match status of candidate record pairs. Overviews of supervised classification techniques are provided in text books on machine learning and data mining [135, 189]. Most classification techniques require a user to tune a variety of parameters to achieve high classification accuracy. Selecting appropriate parameter values can be conducted either via a guided search through the parameter space or via manual tuning.
2. The accuracy of the built classification model is evaluated using a set of testing data that must be in the same format and structure as the training data (i.e. these data must be comparison vectors that were generated using the same comparison functions as the comparison vectors in the training data). These testing data must also contain the known true match status of record pairs, so that the match or non-match decision of the trained classifier can be compared with the true match status (this topic is covered in more detail in Sect. 7.2).
   It is important that the testing data are different from the training data, because otherwise *over-fitting* can occur [135]. Over-fitting refers to the issue that the accuracy of a classification model as measured on the training data is very high, because the model will learn the intrinsic characteristics of the training data. Testing a model on data sets that are different from the training data set is more meaningful and more realistic, because in practice the data upon which a classifier is applied on will be different from the data the classifier was trained on. The accuracy reported using a testing data set is therefore closer to the accuracy that can be expected when the classification model is applied on new, unseen data where the match status of candidate record pairs is unknown.
   If the accuracy reported on the testing data is not good enough according to some criteria set for a certain data matching exercise, then one needs to go back to step 1 and either change some of the parameter settings used when the classifier was trained, or alternatively employ a different classification technique altogether.

RecPairID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	Class
(a1,b1)	0.6	0.8	0.0	1.0	0.6	0.5	0.5	1.0	*M*
(a1,b2)	0.0	0.15	0.0	0.5	0.0	0.5	0.0	0.75	*U*
(a2,b1)	0.2	0.0	0.0	0.1	0.15	0.0	0.0	0.75	*U*
(a2,b2)	0.0	0.25	1.0	0.4	0.6	1.0	1.0	0.75	*M*

Fig. 6.4 Example comparison vectors based on records shown in Fig. 6.1 and their true match status (the column 'Class', where *M* corresponds to matches and *U* corresponds to non-matches), which can be used to train a supervised classifier

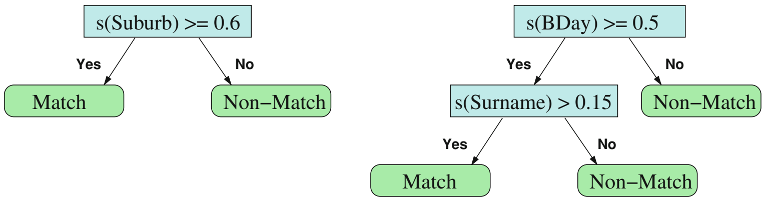

Fig. 6.5 Two example decision trees resulting from the four training comparison vectors from Fig. 6.4. The tests are conducted on the similarity values (indicated by 's(·)' in the tree nodes) for certain attributes. The leaf nodes correspond to the two classes into which record pairs are classified. Clearly, the tree on the left side is better as it is not only smaller (less tests) and thus faster when new candidate record pairs with unknown match status are to be classified, but also more robust. The test 's(Surname) $>$ 0.15', which is a very low threshold, is unlikely to lead to accurate matching results. Rather, the tree on the right-hand side is over-fitting the training data shown in Fig. 6.4

It is important to note that the selection of the best classification technique is dependent on the data that are to be classified [80]. For different types of data different techniques might perform best.

3. Once a satisfactory accuracy has been achieved with a trained classification model, in the third step the model is applied to classify new unseen data, i.e. comparison vectors that correspond to candidate record pairs where the match status is not known.

In the following, two popular supervised classification techniques that have been employed in the area of data matching and deduplication are described in more detail.

- *Decision tree induction*: Decision tree induction is one of the most popular supervised classification techniques used in data mining and machine learning [135]. Decision trees, as the example shown in Fig. 6.5 illustrates, are favoured by many researchers and practitioners over other techniques because they can be visualised easily and are thus understandable even by people who are not data mining or machine learning experts. Additionally, decision trees can be directly converted into a set of rules. The two trees from Fig. 6.5 for example, can be converted into the following two sets of rules:

$$(s(\text{Suburb})[r_i, r_j] \geq 0.6) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$
$$(s(\text{Suburb})[r_i, r_j] < 0.6) \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}$$

$$(s(\text{BDay})[r_i, r_j] \geq 0.5) \wedge (s(\text{Surname})[r_i, r_j] > 0.15) \Rightarrow [r_i, r_j] \rightarrow \text{Match}$$
$$(s(\text{BDay})[r_i, r_j] \geq 0.5) \wedge (s(\text{Surname})[r_i, r_j] \leq 0.15) \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}$$
$$(s(\text{BDay})[r_i, r_j] < 0.5) \Rightarrow [r_i, r_j] \rightarrow \text{Non-Match}$$

Like with the rule-based classification approach described in the previous section, each internal node of a decision tree corresponds to a test on a similarity value in a comparison vector for a certain attribute, as illustrated in Fig. 6.5. Each internal node therefore corresponds to a test in the predicate of a rule, where the leaf nodes in a tree correspond to the possible classification outcome of a rule. In the case of data matching, the two possible outcomes are the match and non-match classes.

In the learning phase, a tree is built recursively, starting with an empty tree. At each step in the tree generation process, an attribute that results in the purest split of the training data set is selected (such that matches are moved into one branch of the tree and non-matches into the other branch). Different decision tree algorithms and splitting criteria have been developed. The interested reader is referred to text books in machine learning or data mining, such as the ones by Mitchell [189] or Han and Kamber [135].

An early work that used a decision tree classifier for data matching was presented by Cochinwala et al. [80]. Their work aimed at matching two databases with customer records. They manually generated training data in the form of sampled pairs of records that were then used to train a Classification and Regression Tree (CART) classifier [42]. Once a tree was generated, they applied tree pruning in order to reduce the complexity of the rules that were extracted from the tree, and to make these rules more robust. The reduced tree not only generated less complex rules (i.e. rules made of a smaller number of tests), it also lead to rules that were more robust when applied to matching the full customer record databases [80].

Elfeky et al. implemented the ID3 decision tree algorithm into their TAILOR data matching tool box [102]. They provided two approaches to generate training data. In the first approach, selected candidate record pairs are manually classified as matches and non-matches by a domain expert, and the comparison vectors of these record pairs are then used to train a decision tree. The second approach aims to overcome the manual step by first applying a clustering technique to group all candidate record pairs into three clusters based on their comparison vectors. The first cluster corresponds to the class of matches, the second cluster to the class of non-matches, and the third cluster to the class of potential matches. The comparison vectors in the match and non-match clusters are then used to train the decision tree classifier. In their experimental study, the authors found that both the decision tree based on manual training data generation and the one based on the cluster pre-processing (called the 'hybrid classification approach') achieved better matching quality than the threshold-based probabilistic record linkage classifier described in Sect. 6.3.

- *Support vector machine (SVM)*: This relatively recent classification technique, developed in the 1990s [259], is based on the idea of mapping the training data set, which consists of comparison vectors and their class labels (match or non-match), into a multi-dimensional vector space in such a way that the training records from the two classes are separated and the gap between the two classes is made as wide as possible.
  This idea is illustrated in Fig. 6.6. A decision boundary corresponds to a hyperplane in the high-dimensional space (a line in two dimensions or a plane in three dimensions), and the optimal decision boundary is the one which has the widest margins to training records in both classes. The mapping from the original input space (i.e. the comparison vectors containing similarity values) into a high-dimensional space is conducted using a *kernel function*, which allows the efficient calculation of the dot product required in the training process of a SVM. This training process corresponds to solving a quadratic optimisation problem [259], for which efficient techniques are available.
  Bilenko et al. [35] employed a SVM classifier to learn the costs for edit operations (such as character inserts, deletes or substitutions) within the Levenshtein edit distance approximate string comparison function (which was presented in Sect. 5.3). Learning these costs allows a better separation of the string pairs that correspond to matches from those that correspond to non-matches. The training data required for this approach consist of pairs of strings and their match status.
  Christen [59, 60] developed an automatic classification approach for data matching based on a SVM, which is similar to the clustering-based hybrid approach developed by Elfeky et al. [102] described above. In a first step, training examples that clearly correspond to matches and non-matches are selected from the set of all comparison vectors. Clear match examples are comparison vectors where all similarity values are equal to or very close to the exact similarity of 1, while clear non-match examples are comparison vectors where all similarity values are equal to or close to 0. Based on this initial training set, a first SVM is trained. All comparison vectors that are not in one of the two training sets are classified using this initial SVM. In the second step, the comparison vectors that were classified to be furthest away from the SVM decision boundary are added into one of the two training sets (depending upon if they are located on the side of matches or on the side of non-matches), and a second SVM is trained on these enlarged training sets. This process of adding more comparison vectors into the training sets followed by training a new SVM is repeated until a stopping criteria is fulfilled. In an experimental study, this automatic classification approach outperformed a basic clustering approach as well as the hybrid approach by Elfeky et al. [102] in experiments on several data sets.

Employing supervised classification techniques for data matching has several challenges. First, classifying candidate record pairs is often an imbalanced problem, in that there are many more record pairs that correspond to true non-matches com-

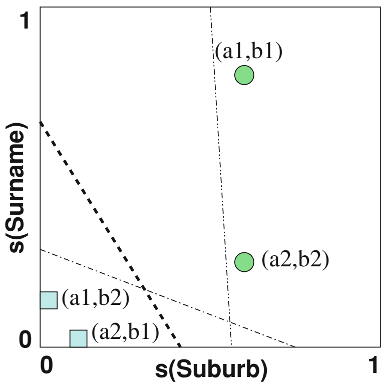

Fig. 6.6 A simplified illustration of a 2D vector space (made of the similarities of the attributes 'Surname' and 'Suburb'), containing the similarity values of the four comparison vectors from Fig. 6.4, and three decision boundaries (*dotted lines*) that correspond to three trained support vector machine (SVM) classifiers. The *thick dotted line* is the SVM which has the widest margins to both the class of matches (*circles*) and non-matches (*squares*)

pared to the number of record pairs that correspond to true matches. This holds even after some form of indexing has been applied. As a result, a classification technique, as well as the measure(s) used to evaluate how good a trained classification model is, must be able to handle imbalanced classes. The way training data are generated can help to overcome this problem, for example by sampling the same number of training examples from both the match and non-match classes. This issue will be discussed further in Sect. 7.1.

The second issue is the difficulty to generate, obtain, select or sample training data that are representative of the actual data that are to be matched. Good training data will more likely result in a robust and accurate classification model. Acquiring or manually generating training data can be quite costly and time-consuming. As a result, training data sets are often small compared to the databases that are to be matched or deduplicated. Training data should, however, represent the detailed characteristics of the full database(s) as much as possible. An alternative to train a supervised classifier using a large training data set is to create training data interactively using an active learning approach, as will be discussed next.

6.7 Active Learning Approaches

A major drawback of supervised classification techniques is their need for training data sets, made of comparison vectors that correspond to matches and non-matches, that represent the characteristics of the full database(s) to be matched or deduplicated. An alternative to generating or obtaining such comprehensive training data sets is to use a classification approach that only requires a small amount of training data in order to achieve high classification accuracy. Based on an initial small training data set, a classification model is built interactively by asking an experienced user for further training examples that help to improve the classification model. Such interactive approaches are known as *active learning* [11, 231, 252].

An active learning classifier starts by building a first classification model using a small set of *seed* training examples. These can, for example, be comparison vectors that correspond to clear matches and clear non-matches. This initial classification model will likely have a low classification accuracy. Specifically, it will have difficulties to classify comparison vectors with certain characteristics, such as comparison vectors that do not correspond to clear matches or non-matches. If a SVM classifier is used, for example, then the comparison vectors that are located closest to the decision boundary (see Fig. 6.6) correspond to matches or non-matches with almost equal likelihood. A manual classification of the candidate record pairs that correspond to these comparison vectors can be highly beneficial to improve the accuracy of the classification model.

An active learning classifier works iteratively by (1) training a classification model, (2) classifying all comparison vectors not in the training set as matches or non-matches, (3) asking a user to provide manual classification of the candidate record pairs that were most difficult to classify, (4) adding these manually classified comparison vectors to the corresponding training data set (of either matches or non-matches) and (5) training the next, improved, classification model. This process is repeated until a certain stopping criteria is met. This stopping criteria either terminates this process after a maximum number of iterations, or more commonly when the last trained classifier achieves a certain matching quality on the testing data set.

The following three classification approaches using active learning have been proposed in the area of data matching and deduplication.

- Sarawagi et al. [231] presented the ALIAS system, which is an interactive deduplication system that (similar to the traditional probabilistic record linkage classification approach discussed in Sect. 6.3) works with the three classes of matches, non-matches and potential matches. Rather than building only one classification model, a set of several models is trained on the training data set, each of them with a randomised choice of parameter setting. Three decision trees were used in ALIAS. For those comparison vectors where different decisions were made by the three trained decision trees (for example, two classify a comparison vector as a match and one as a non-match), a manual decision is required by the user. According to this manual classification, comparison vectors are added into either the training set of matches or the set of non-matches, and the next set of classifiers is trained on this enhanced data set.
- A similar approach was presented by Tejada et al. [252] aimed at integrating data objects from different Web sources. Their system, called Active Atlas,[1] learns *mapping rules* using an active learning approach. These mapping rules include tests for string equality, string prefix or string suffix equality, or if two strings

[1] It is interesting to note that both the ALIAS and Active Atlas systems were presented in the same year (2002) and at the same conference.

contain the same abbreviations or acronyms (like 'IBM' vs. 'International Business Machines'). Similar to the ALIAS system, a committee of three decision tree classifiers was used to learn the rules that best distinguish matches from non-matches, with a manual classification required for pairs of strings where the three decision trees returned different classification outcomes.

- More recently, Arasu et al. [11] presented a novel approach to active learning specifically designed for data matching. Their technique integrates indexing with active learning. Either a decision tree or SVM classification model can be employed, and a user can specify the minimum precision (to be discussed in Sect. 7.2) the final classification model must achieve. The active learning process then aims to achieve a high recall for the classification model while reducing the number of examples to be classified manually as much as possible. Experiments on two large databases showed that this proposed new technique outperformed both ALIAS [231] and Active Atlas [252].

6.8 Managing Transitive Closure

The result of the classification of individual candidate record pairs into matches and non-matches is often not the final outcome of a data matching or deduplication exercise. If candidate record pairs are classified individually, each record can be part of a match with several other records, as illustrated in Fig. 6.7. In certain situations, however, a one-to-one match restriction has to be applied, as will be further discussed in Sect. 6.11. If multiple matches are allowed, then the issue of *transitive closure* needs to be addressed.

Transitive closure refers to the situation where two record pairs, (r_i,r_j) and (r_i,r_k), have been classified as matches but the pair (r_j,r_k) has been classified as a non-match. This contradicts the intuition that if record r_i is considered to be a match with record r_j (i.e. referring to the same entity) and record r_j is considered to be a match with r_k, then record r_i must also be considered a match with r_k. Applying the transitive closure refers to changing the match status of record pairs such that no contradictions of the match status within groups of records occurs [195].

The transitivity of matches can also lead to problems in that 'chains' of records, where individual pairs are classified as matches, are formed. The records at the two ends of a chain can, however, be quite different from each other, and they would not be considered to correspond to a match. For example, consider the four records 'a1' to 'a4' in Fig. 6.7, where the three individual pairs (a1,a2), (a2,a3) and (a3,a4) have been classified as matches, but the summed similarities between other pairs is below the match classification threshold $t = 5$. Pair (a1,a4), for example, only has a summed similarity of $\text{SimSum}(a1, a4) = 1.15$, and it is unlikely that these two records refer to the same individual. The clustering approaches discussed in the following section aim to overcome this problem of chains of matching records.

In real-world databases, the problem of record chains being generated by a pair-wise classification technique seems to occur only rarely because the space of all

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear
a1	john	smith	18	miller st	dickson	12	11	1970
a2	jonny	smith	73	miller st	dixon	11	10	1970
a3	joan	smith	73	dawson cr	lyneham	11	12	1979
a4	max	miller	73	dawson cr	lyneham	11	2	1969
a5	sal	bass	67	milles rd	ainslie	28	5	1981
a6	sally	bass	64	miles rd	ainsile	23	5	1981

Candidate pair	SimSum	Classification
(a1, a2)	5.20	Match
(a1, a3)	3.30	Non-match
(a1, a4)	1.15	Non-match
(a2, a3)	5.05	Match
(a2, a4)	2.70	Non-match
(a3, a4)	5.25	Match
(a5, a6)	6.20	Match

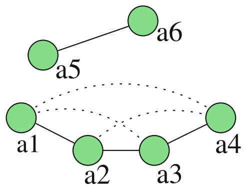

Fig. 6.7 An example of the transitive closure problem for a set of six records (top row table, 'a1' to 'a6'). It is assumed the summed nonzero similarities SimSum for the candidate record pairs in the lower left table have been calculated, and a simple classification threshold-based classifier with threshold $t = 5.0$ has been applied to classify each record pair individually. The result of this classification are two groups of records (possibly corresponding to two entities). The transitive closure would require that the record pairs (a1,a3), (a1,a4) and (a2,a4) are also considered to be matching (illustrated with dotted lines), even though their summed similarities are below the classification threshold

possible values in the different record attributes is very large. The likelihood that records that are not matching have a high similarity with each other is therefore very small [140, 190, 191]. Clustering and collective classification algorithms, which will be discussed in the following two sections, deal with the problem of transitivity by default, by classifying groups of records as matches rather than individual pairs of records only.

6.9 Clustering-Based Approaches

So far all techniques presented have viewed the problem of deciding which candidate record pairs correspond to matches and which to non-matches as a traditional classification problem. An alternative approach is to view this classification as a clustering (grouping) approach, where each cluster consists of records that refer to one entity. Clustering is the process of grouping data objects that are similar to each other according to some criteria into the same cluster [135]. The aim of clustering is to generate clusters that have high intra-cluster similarity and low inter-cluster similarity. This means all data objects within a cluster should be similar with each other, while data objects in different clusters should be dissimilar to each other.

Clustering is generally conducted in an unsupervised fashion, and therefore no training data in the form of record pairs with known true match status are required. Many different clustering techniques have been developed by the statistics, machine learning and data mining communities [135]. Different clustering techniques employ different heuristics to guide the clustering process [135]. They either partition data objects into a fixed number of clusters or into a hierarchy of clusters, or they generate graphs that correspond to clusters (to be discussed in more detail below), or they generate clusters that correspond to dense areas where many data objects are located close to each other. In data matching, the data objects to be clustered correspond to the records that represent entities.

A clustering-based approach is clearly suited for deduplication, where all records to be matched are stored in one database. For the matching of two or more databases, however, all records first need to be inserted into one common set. This can be accomplished by assigning each record a unique identifier which not only identifies the record but also the source database from where the record has originated.

Because each entity in a database will ideally be represented by one cluster, the number of clusters is not only unknown at the beginning of a clustering process, it will also be very large when the databases that are deduplicated or matched contain many entities. Partitioning based clustering algorithms [135], which require the number of clusters to be specified at the beginning, are therefore not applicable for clustering records in data matching or deduplication applications.

The clusters generated in data matching are generally very small, containing only a few records. Some, potentially many, clusters will only consist of a single record, if there is only one record in the database(s) that corresponds to this entity.

Different clustering approaches for data matching and deduplication have been investigated. In the following, five different approaches are discussed in more detail.

- In an early clustering approach, Monge [190] proposed an adaptive deduplication system where records are clustered according to some similarity measure, and a priority queue is kept in memory consisting of the most recently formed clusters. Each cluster corresponds to an entity, and is made of one or several records that represent this entity. To save memory, however, not necessarily all records that refer to an entity are kept in memory for a given cluster. Initially all records to be deduplicated or matched are sorted according to a sorting key (as was discussed in Sects. 4.2 and 4.5). One record in the sorted database is then processed after another. Each record is compared with the records stored in the priority queue. If a match is found the current record is attached to the matching cluster, and this cluster is put at the top of the priority queue. If no match is found then a new cluster is formed made of the current record only. To make sure that only a certain amount of memory is used, the oldest cluster is removed from the end of the priority queue if a new cluster is generated and the queue exceeds a maximum length limit.

  The experimental results of this combined sorted-neighbourhood and clustering technique presented by Monge showed that the approach can achieve matching accuracies similar to the basic sorted-neighbourhood approach [140, 141]. It can, however, reduce the number of record pair comparisons that are conducted by upto

75 %, because each record is only compared to a small number of representative records in a cluster.

- Clustering can also be applied as a post-processing step after the pair-wise classification of record pairs has been conducted, and a graph of all matching records, as for example shown in Fig. 6.7, has been generated. The aim of clustering using such a graph is to decide for each sub-graph (consisting of connected records) which record subsets correspond to the actual entities that are to be matched or deduplicated [140, 195]. For example, the sub-graph made of the four records 'a1' to 'a4' from Fig. 6.7 is unlikely to refer to one but rather to three entities (only 'a1' and 'a2' seem to be duplicate records of the same individual).

  One approach to reducing the size of sub-graphs (and thus the number of records that can correspond to the same entity) is to iteratively remove edges between two nodes (corresponding to a record pair classified as a match) starting from the edge that has the lowest similarity in a sub-graph. For the sub-graph made of records 'a1' to 'a4' in Fig. 6.7, the edge from 'a2' to 'a3' has the lowest similarity (5.05), therefore this edge would be removed first, leaving two new smaller sub-graphs. This process can be repeated until either each sub-graph only contains edges with a certain minimum intra-cluster similarity value t_c, until the transitive closure property has been fulfilled, or alternatively until each sub-graph contains no more than a maximum number n_c of records [195]. Which of these stopping criteria is best suited depends upon the requirements of the data matching or deduplication application.

  Continuing on with the example from Fig. 6.7, if the minimum intra-cluster threshold is set to $t_c = 5.25$, then the link between records 'a2' and 'a3' is removed first. The link between records 'a1' and 'a2' will also be removed, resulting in three separate entities (which possibly corresponds to a missed true match), while the link between records 'a3' and 'a4' is kept as a match (possibly a wrong match). On the other hand, if the maximum size of a sub-graph is set to $n_c = 2$, then the record pair 'a1' and 'a2' is considered as one entity and the pair 'a3' and 'a4' as another entity, and only the link between 'a2' and 'a3' is removed.

- Another approach to clustering a graph of matching record pairs is to find centres within each sub-graph and to then assign nodes (records) to their closest centre, i.e. the centre record they are most similar to. This approach, named CENTER [137], first sorts the edges of a sub-graph in descending order of their similarities. The first time a record r_i, appears in an edge of the sub-graph it is assigned as the centre of a cluster. All records r_j that appear in edges (r_i, r_j) later on in the sorted list are then assigned to this cluster, but not to any other clusters [195].

  When this clustering technique is applied on the sub-graph of records 'a1' to 'a4' from Fig. 6.7, the sorted list of edges is: (a3,a4), (a1,a2) and (a2,a3), with similarities 5.25, 5.20 and 5.05, respectively. If node 'a3' is marked as the centre then 'a4' is obviously considered to be part of this cluster. In the next e.g., (a1,a2), neither 'a1' nor 'a2' have been marked as centres or as being part of a cluster, and so 'a1' is marked as a new centre. In the third e.g., (a2,a3), both nodes have already been assigned to clusters and so this edge is not considered. As a result, the clustering of this sub-graph leads to two sub-graphs that correspond to two entities, one consisting of records ‘a1’ and ‘a2’ and the other of records ‘a3’ and ‘a4’. The selection of which node in a pair becomes the centre of a new cluster obviously affects the final clustering outcome. One approach to overcoming this problem, called MERGE-CENTER [137], is to merge two clusters if their centres are very similar to each other.

- The two previously described approaches to clustering are based on a graph of matching candidate record pairs which was built using a pair-wise comparison and classification technique. A drawback of these approaches is that the minimum similarity threshold that has been used to classify record pairs into matches and non-matches is determining the structure of the cluster graph. This threshold is a global parameter applied to all compared record pairs. An alternative approach is to cluster the set of records based on all similarity values calculated between pairs of records (not just the ones classified as matches), and to guide the clustering based on records that are similar to each other relative to the number of records that are located in the neighbourhood around them [195]. This is an approach similar to density based clustering [135].
  Chaudhuri et al. [52] proposed such an approach based on the concepts of compact sets and sparse neighbourhoods. A compact set, CS, is a group of records that are all more similar with each other (i.e. have small distances $dist(\cdot)$ with each other) than they are similar to any other records. Specifically, for all pairs of records $r_i, r_j \in CS : dist(r_i, r_j) < dist(r_i, r_k) \; \forall \; r_k \notin CS$. The neighbourhood set of a record r_i is defined as $N(r_i) = p \cdot nn(r_i)$, where $nn(r_i)$ is the distance of record r_i to its closest neighbour and p determines the size of the radius around r_i that is considered. The neighbourhood of r_i is defined to be sparse if the number of records in the set $N(r_i)$ is below a certain constant threshold [52]. The advantage of this clustering approach is that clusters of records are generated depending upon the number and density of their neighbouring records, rather than based on a global threshold.
- A different clustering approach was proposed by Verykios et al. [261] and Elfeky et al. [102]. Rather than clustering the actual records based on the similarities calculated between them, clustering was applied on the comparison vectors that are generated in the comparison step. Specifically, comparison vectors were inserted into three clusters, one each corresponding to matches, non-matches and potential matches, similar to the traditional probabilistic record linkage approach presented in Sect. 6.3. Identifying the clusters that correspond to matches and non-matches is easy because they will either have a centroid vector that is close to an exact match (with comparison vector [1.0, . . . , 1.0]) or a centroid vector that is close to a total non-match (with comparison vector [0.0, . . . , 0.0]), respectively. In the second step of this approach, the comparison vectors in the match and non-match clusters were used as training data for a decision tree classifier, as was previously discussed in Sect. 6.6.

6.10 Collective Classification

Pair-wise classification techniques make a match or non-match decision independently for each compared candidate record pair, and clustering techniques further refine the classification of groups of records that likely correspond to the same entity. With both these approaches, decisions about the match status of a pair or a group of records are made independently from all other records or groups in the database(s) that are matched or deduplicated. These techniques therefore make local decisions without taking the characteristics of all records in the full database(s) into account.

New techniques have been proposed in the past few years that aim to make a decision about which records are matching in an overall collective fashion over all pairs or groups of records in the database(s) that are matched. These techniques are known as 'collective entity resolution' techniques, and they employ either iterative or hierarchical clustering [31, 181], or graph-based approaches [93, 155, 195]. All of these collective classification approaches have been developed for, and evaluated on, databases that contain different types of entities, where certain relationships between entities are known. These relationships can be represented in a *relationship graph*, as illustrated in Fig. 6.8. The most popular type of such data are bibliographic databases where the entity types include *authors*, *institutions* (or affiliations), *venues* (journals, conferences and workshops), and the actual *papers* (or articles) [31, 155].

The basic idea of collective classification approaches is to calculate the similarities of all connections (links) in the relationship graph that are ambiguous (such as the dotted links in Fig. 6.8) using information from the known relationships (the 'hard' connections between different entities). Because different types of entities are available, the known relationships between one type of entities can help to disambiguate (i.e. decide the match status) of other types of entities.

The first step in collective classification techniques is to generate the relationship graph, which can consist of relations between different types of entities. These relations can either be 'hard' connections (where a relationship is known without doubt from the data), or connections that have a probability or weight attached to them if it is not clear if a relationship exists or not. These probabilities or weights can, for example, be based on similarities calculated when pairs of records are compared, as was discussed in Chap. 5. The collective classification task is then to decide if these possible relationships correspond to matches or non-matches based on other connections in the relationship graph. This is generally accomplished through an iterative approach that updates the weights (or probabilities) on the connections that determine the matching outcomes. Note that while in Fig. 6.8 only one type of connection needs to be classified, in the most general case not just connections between different types of entities, but also different types of connections, are available in a relationship graph.

The main differences between the various collective classification techniques are (1) how the relationship graph is generated from the underlying database(s) and (2) how the iterative update of the probabilities or weights in the graph, and their classification into matches (i.e. a connection exists between two nodes) or non-

AuthorID	Author name	Affiliation
a1	Dave Smith	Purdue
a2	Don Smith	Patras
a3	Susan Miles	Stanford
a4	John Black	Stanford
a5	Joe Green	?
a6	Liz Redman	?

PaperID	Co-author names
p1	John Black, Don Smith
p2	Susan Miles, **D Smith**
p3	Dave Smith
p4	Don Smith, Joe Green
p5	Joe Green, Liz Redman
p6	Liz Redman, **D Smith**

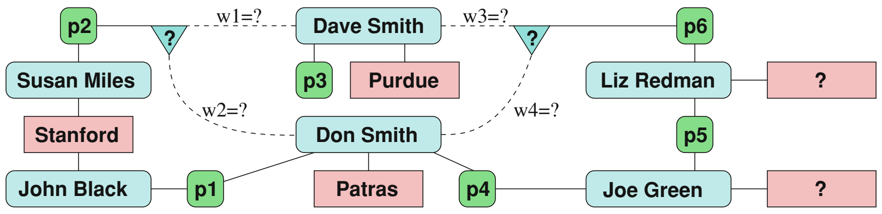

Fig. 6.8 Example of a graph-based collective matching approach of bibliographic records, adapted from [155]. The task is to identify (disambiguate) if the author ‘D Smith’ in papers ‘p2’ and ‘p6’ refers to either ‘Don Smith’ or ‘Dave Smith’. Given Don Smith has co-authored paper ‘p1’ with ‘John Black’, who is affiliated with ‘Stanford’, and ‘Susan Miles’ is also affiliated with ‘Stanford’, there is a higher likelihood that ‘Don Smith’ rather than ‘Dave Smith’ is a co-author of paper ‘p2’, because ‘Dave Smith’ does not have any other connection with ‘Stanford’. Similarly, given ‘Don Smith’ has written paper ‘p4’ with ‘Joe Green’, who has co-authored paper ‘p5’ with ‘Liz Redman’, there is a higher likelihood that ‘Don Smith’ is also the second co-author of paper ‘p6’ rather than ‘Dave Smith’ who has no connection to ‘Joe Green’

matches (no connection exists between two nodes) is conducted. In the following, the major approaches to collective classification techniques are described in more detail.

- Kalashnikov and Mehrotra [155] build a relationship graph between different types of entities and with different relations, as the example graph in Fig. 6.8 shows. The disambiguation of connections between entities, i.e. their classification as being matches or non-matches, is conducted in an iterative approach where the unknown weights (such as ‘w1’ to ‘w4’ in Fig. 6.8) are updated based on the number of connections in the path that needs to be covered to get from one end of the connection under question to the other. For example, in Fig. 6.8, the path from ‘Don Smith’ via connection ‘w4’ continues onto ‘p6’, ‘Liz Redman’, ‘p5’, ‘Joe Green’, ‘p4’ and then back to ‘Don Smith’. On the other hand, the only path starting from ‘Dave Smith’ would go via all other authors and even another path with unknown weight (‘w1’), which is much less likely than the first path because it is a much longer path. Therefore, the weight for ‘w4’ can be set to a higher value than the weight for ‘w3’. Kalashnikov and Mehrotra formalise this principle as the *context attraction principle* [155], and using this principle the unknown connection weights in the relationship graph are updated in an iterative fashion. An experimental evaluation on bibliographic data confirmed that this approach can lead to more accurate matching results compared to a pair-wise classification approach [155].

- Dong et al. [93] tackle the problem of collective classification of entities from multiple classes (types) by generating a dependency graph rather than a relationship graph. A node in the graph represents the similarity between a pair of entities of the same type, and a connection between nodes occurs when this similarity depends upon the similarity of another pair of entities. For example, the similarity between two papers (articles) depends upon the similarity between the titles, years of publication, page numbers, the authors listed with the two papers, as well as the similarity of the venues where the two papers have been published. A change in the similarity of authors or venues, for example, will affect the similarity calculated for the pair of papers.
  The collective classification task is conducted iteratively by initially marking all nodes as *active*. An active node is then selected, and depending upon the similarity in that node, it is either marked as *merged* (if its similarity is above a certain similarity threshold) or as *inactive* (otherwise). All neighbours of this just processed node that have a similarity below 1.0 (i.e. which do not have exact similarity) are then set as *active*. A queue of active nodes is maintained throughout the process, and in each iteration the similarity of the node at the top of the queue is recalculated. This process continues until no active node is left in the queue and all nodes are either marked as merged or inactive. This approach outperformed pair-wise classification techniques in experiments using several data sets [93].
- A machine learning based technique to collective classification has been proposed by Bhattacharya and Getoor [31]. In this approach, a relationship graph is built where the records (viewed as references to entities) are the nodes, and edges connect nodes if there is a relationship between them. For example, similar as shown in Fig. 6.8, the names of authors will be nodes in a reference graph, and all co-authors of a paper will be connected through an edge. These edges can be between more than two nodes, in which case they are called *hyper-edges*. If, for example, a paper was written by three co-authors, then one edge connects the three nodes that correspond to these co-authors. The similarity between two nodes is calculated as the weighted sum between the attribute value similarity and the relational similarity, where the latter considers the connectivity of two nodes through their hyper-edge as well as the connectivity of the neighbouring nodes they are connected to. Different relational similarity measures have been investigated [31].
  The collective classification is conducted using a priority queue that contains tuples made of two cluster identifiers and the similarity between the two clusters, sorted according to highest similarities first. An iterative algorithm merges clusters and updates the similarities between newly formed clusters as long as there are pairs of clusters in the queue that have a certain minimum similarity. When two nodes or clusters are merged, the similarities between the newly formed cluster and all its neighbours in the relational graph are updated, and the similarities between older clusters and the new cluster are added into the priority queue. The algorithm stops when no more clusters can be merged because the similarity between

them is below the minimum threshold set by the user. Experiments on three different bibliographic databases showed that this approach is superior to pair-wise classification, however, at the cost of longer run times [31].
A variation of this relational clustering approach has been developed by Bhattacharya and Getoor to allow query-time collective classification [32]. A single query record is matched to a database that contains entity records and that can include duplicates. Using a collective classification approach, the query record is matched with the full database. While the reported matching accuracy of this approach is again very high, the matching time for a single query record was reported as being around 30 s, making this approach not suitable for real-time data matching (a topic that will be covered in detail in Sect. 9.3).

While collective classification techniques have shown to result in improved matching quality compared to pair-wise classification techniques, these improvements come at the cost of a higher computation complexity and thus reduced scalability to large databases. Recent work has aimed to improve the scalability of collective classification techniques by running a collective matching process many times on small subsets of records that are in the same neighbourhood of the data [225]. These independent collective matching instances exchange messages about the local matches found, and the results of all matching instances are combined into a final overall solution.

Thus far collective classification techniques for data matching have mostly been applied on databases that contain bibliographic data, or other data that contain several types of entities. It is not clear if and how collective classification techniques can be applied on data that only contain one type of entities, such as databases containing records about individuals.

6.11 Matching Restrictions and Group Linking

The classification of pairs or groups of records into the class of matches and non-matches discussed so far has not taken into account that in certain data matching applications there are restrictions with regard to the number of matches a single record can be involved in. The three possible scenarios when matching two databases, **A** and **B**, are:

- **One-to-one**: A record from **A** can match at most one record from **B**.
- **One-to-many**: A record from **A** can match at most one record from **B**, while a record from **B** can be involved in none, one or several matches with records from **A**. The one-to-many scenario is symmetric by swapping the databases **A** and **B**.
- **Many-to-many**: A record from **A** can match none, one, or several records from **B**, and a record from **B** can match none, one, or several records from **A**.

A one-to-one matching restriction is, for example, required when records from (historical) census databases are matched across time, and each record corresponds

Candidate pair	SimSum
(a1,b2)	4.5
(a1,b4)	5.5
(a2,b3)	5.9
(a2,b5)	4.9
(a3,b2)	5.8
(a3,b4)	4.7
(a4,b1)	5.3
(a4,b5)	6.0
(a5,b5)	5.1

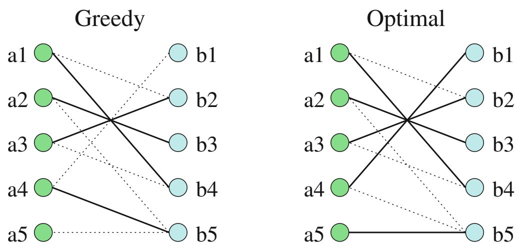

Fig. 6.9 Examples of two approaches to enforcing a one-to-one assignment of matched candidate record pairs. The thicker lines between records illustrate the matched (assigned) records. The 'Optimal' approach aims to maximise the overall sum of the similarities (SimSum) over all matched record pairs, while the 'Greedy' approach matches candidate record pairs starting from the pair that has the highest similarity value until no more un-assigned records can be matched. In this example, the sum of similarity values for the matched (assigned) record pairs with the optimal approach is 27.6 while for the greedy approach the sum is only 23.2

to one individual [115, 116]. Because it is assumed that each census database only contains a single record per individual, one record in one census database (for example from 1900) can only match at most one record from another census database (for example from 1910).

A one-to-many matching restriction could be appropriate in a scenario where a client database of a government agency (that only contains one record per client) is updated with a set of new records that refer to individuals who in the recent past have been in contact with this agency. This new set of records can potentially contain several records for an individual because there might be several contact points for this government agency (online, telephone and face-to-face), and because this agency provides several programs (like a social security agency that provides housing, disability, unemployment and childcare support programs). Therefore, one client record in the cleaned and deduplicated client database maintained by this government agency can potentially match with several records in the set of new records.

A many-to-many matching is, for example, appropriate when two bibliographic databases are matched with the aim to identify and match all publications that refer to the same author, and there can be several records in each database that correspond to publications by one author. Returning to the example of matching census data, when the objective is to match households or families across census databases, rather than individuals, then a many-to-many matching scenario needs to be followed [114, 115].

While the clustering-based and collective classification techniques discussed in the previous two sections are mostly aimed at the many-to-many matching scenarios, the classification techniques presented in Sects. 6.2–6.7 classify individual record pairs independently from all others. Any one-to-one or one-to-many matching restriction can then be applied as a post-classification step on the set of candidate record pairs that were classified as matches.

A one-to-one matching restriction corresponds to finding an optimal solution to the problem of assigning individual records from the two databases into pairs (with one record originating in each database) based on the classified matched record pairs, such that the number of confirmed matched pairs and the sum of their similarities are maximised. As Fig. 6.9 illustrates, solving this problem corresponds to finding a solution to the *maximum weighted bipartite graph matching problem* [273].

A simple if not optimal approach to one-to-one matching is to sort the matched candidate record pairs according to their similarity values, and to assign pairs into the set of confirmed matches in a greedy fashion, as shown in the left graph in Fig. 6.9. The record pair with the highest similarity value is confirmed as a match first, and the two records of that pair are marked as being assigned matches (thick line). They can therefore not be part of any other matching pair. Then the next record pair (where both records are unassigned) with the highest similarity is confirmed as a match, and its two records are assigned as matches. This process is repeated as long as there are unassigned records that can be assigned to a record pair. For example, the pair (a4, b5) in Fig. 6.9 has the highest similarity value, SimSum = 6.0, and is therefore assigned as a confirmed match first. This, however, means that neither record 'a4' nor record 'b5' can be part of any other assigned pair. While this is a simple and fast approach (only requiring sorting the matching record pairs according to their similarities followed by a linear scan through that sorted list), this greedy approach is unlikely to produce a good solution because it is likely that not all records can be assigned into matching pairs. In Fig. 6.9, for example, the greedy approach cannot assign records 'a5' and 'b1' into a matching record pair.

Finding an optimal solution to the problem of assigning records into matching pairs is known as solving the *assignment problem*. Various algorithms have been developed to solve this problem [273]. One early approach is the so-called *Hungarian algorithm*, while another class of algorithms can solve this problem by viewing it as an auction problem [30]. The objective of an auction algorithm is to assign a group of people who all bid for several objects such that overall the highest profit can be obtained. People have maximum prices they are willing to pay for certain objects. When such an auction problem is mapped to the one-to-one matching restriction problem, people correspond to the records from one database, objects to the records from the second database, and the maximum prices to the similarities between pairs of records. Assignment algorithms are computationally more costly than the simply greedy approach presented before. Specifically, an auction algorithm has a computation complexity of the order $O(m \times n)$, where m is the number of links between records and n is the number of records involved [204, 205]. When a one-to-one matching restriction is required in data matching, then each subset of connected record pairs can be solved independently from all other subsets using an assignment algorithm applied on this subset only.

In some applications where many-to-many matchings are permissible, the main aim of a matching exercise is to identify groups of records that match across two databases rather than individual records [204, 205]. Groups can be defined according to some criteria, such as the value of a group identifier attribute. Example applications where such group linkage techniques are useful include the matching of families and households between census databases collected at different points in time [114, 115, 116], or the matching of bibliographic databases where sets of records correspond to the publications of one author [205]. The objective of group linkage is to identify an optimal matching of groups of records across two databases based on similarities calculated between individual pairs of records as well as a similarity measure that can be calculated for groups of individual record pairs. Both the Jaccard coefficient and a weighted bipartite graph matching approach have been successfully employed for the group linkage problem [204, 205].

6.12 Merging Matches

Thus far, it was assumed that the data matching process is completed once pairs or groups of records have been classified into matches and non-matches (with an acceptable quality as will be discussed in Chap. 7). In certain data matching and deduplication situations, however, matched records also need to be merged (in some way) before the matched data can be used further, either for data analysis or data mining, or for further data processing such as generating mailing lists. In this last example, the objective of a data matching exercise is to create a database that contains complete, accurate and up-to-date address and name details for all records in a mailing list. Achieving this goal means that the values in certain attributes for the matched records need to be merged.

While traditionally the merging of matched records has not been considered by most research in data matching, a recent research project has investigated how this merging step can be best incorporated into the overall data matching process. The Stanford Entity Resolution Framework (SERF) project [25, 26, 186] has developed generic data matching techniques that assume the actual matching of records as a black-box approach, represented as a function $match(r_i, r_j)$, which returns true if two records are matching and false otherwise. An additional black-box function, $merge(r_i, r_j)$, is defined on matching record pairs. It returns a new record that is generated by (somehow) merging the content of records r_i and r_j. While the actual merge function is domain and application specific, a *merge domination* is defined as the situation when for two records r_i and r_j it holds $merge(r_i, r_j) = r_j$. When the merge function corresponds to combining attribute values from r_i and r_j, r_j dominating r_i means that r_i does not contribute any new attribute value(s) to the merged record beyond what r_j already contains.

The generic entity resolution process on a database consists of an iterative matching and merging approach which results in a set of merged records that cannot be further matched or merged with each other, and no merged record is dominated by another merged record [26]. Based on these assumptions, a set of entity resolution algorithms (named G-Swoosh, R-Swoosh, F-Swoosh, D-Swoosh, and P-Swoosh) were developed by the SERF project. The G-Swoosh algorithm has no particular requirements on the $match$ and $merge$ functions. It helps to illustrate the process of entity resolution. In the R-Swoosh algorithm, if two matched records r_i and r_j

are merged into $r_{i,j}$, i.e. $r_{i,j} = merge(r_i, r_j)$, then the new record $r_{i,j}$ is added into the set of all records and the two original records r_i and r_j are removed from this set. This approach also means that dominated records do not need to be explicitly removed from the set of all records as they are eliminated in the merge and removal step.

The F-Swoosh algorithm improves performance by taking feature (attribute value) comparisons into account such that each pair of features is only compared once. D-Swoosh [25] and P-Swoosh [160] are algorithms aimed at distributed and parallel computing environments, respectively. Both these algorithms are described further in Sect. 9.5.

A more recently proposed approach is to employ locality sensitive hashing (LSH) for quick iterative blocking of the records in a databases [164]. All records hashed into the same bucket (block) by the hash-algorithm are matched and merged, and the merged records are re-hashed. This process is repeated until either no more matches and merges are found, the reduced number of record pairs reaches a certain minimum number, or a specified maximum number of iterations has been reached. The authors proposed several variations of their approach depending upon if the databases to be matched contain duplicates or not. Experimental results on a bibliographic database showed that this hash-based approach is able to achieve better scalability to large databases compared to the R-Swoosh algorithm [164].

6.13 Practical Considerations and Research Issues

The choice of what type of classification technique to employ for a certain data matching or deduplication exercise depends upon various factors, including the classification techniques available in the matching software that is used (or the techniques that can be implemented), and the type of data that are to be matched or deduplicated. If a supervised classification technique is to be used, training data in the form of record pairs with their known match status are needed.

A suggested approach is to evaluate different classification techniques, and in the case where no training data are available, to manually generate a set of record pairs (together with their match status) that represent the characteristics of the data (such as the distribution of values, and the types and distribution of errors and variations in the data that are to be matched). While time-consuming and labour-intensive, such an approach will enable an evaluation and comparison of the classification accuracies of different data matching algorithms.

Unfortunately, no comprehensive survey of classification techniques for data matching and deduplication has so far been published. What is needed is an experimental evaluation of different techniques on a variety of test data sets from different domains and of different sizes. These data sets should contain the true match status of record pairs so that the resulting matching quality can be evaluated. Data sets of different sizes are required so that the scalability with regard to training time, classification time and memory usage can be evaluated.

Future research in the area of classification for data matching and deduplication should be aimed at investigating if and how collective classification techniques can be applied to data that do not contain different types of entities (for example, data containing personal details such as names and addresses), and how classification techniques can be employed on very large databases that contain many millions of records. Given the difficulties of obtaining or generating training data (as will be discussed further in Chap. 7), a major focus of research should be on unsupervised and automatic classification techniques that do not require manual preparation of training data.

Another area of future research is the development of adaptive classification techniques, given that in many application areas data matching is no longer employed in batch mode and on static databases. Rather, in many modern information systems data matching and deduplication functionalities are integrated into larger systems where new records that contain the details of entities are being added into databases or data warehouses in an ongoing basis. Matching in real time and matching dynamic databases will be discussed further in Sects. 9.3 and 9.4.

6.14 Further Reading

The book by Herzog, Scheuren and Winkler [143] contains arguably the most accessible and detailed description of the probabilistic record linkage approach. Issues such as the conditional independence assumption and parameter estimation are discussed in detail and illustrated via examples. Further examples of probabilistic record linkage applications are also provided. Talburt nicely explains the Swoosh-based entity resolution approaches using several small example databases [249]. He also describes an algebraic model for data matching. For general introductions to classification techniques, the reader is referred to textbooks in the areas of machine learning or data mining [135, 189].

Naumann and Herschel [195] cover graph-based and collective classification techniques, as well as clustering and rule-based approaches (even though in their book rules-based approaches are discussed under the topic of comparison functions). Batini and Scannapieco [19] also provide an overview of different techniques, including a brief comparison with regard to the requirements (such as expected input, generated output and classification objectives with regard to a quality metric) of different classification techniques for data matching.

The best coverage of the topic of how to merge pairs or groups of records that have been classified as matches is provided by Benjelloun et al. [25] in their description of the techniques developed in the SERF project. Data fusion more generally is covered in the recent survey by Bleiholder and Naumann [38]. For a tutorial on the assignment problem that can be employed to finding a solution to the one-to-one matching problem the reader is referred to the excellent tutorial provided by Bertsekas [30].