Chapter 2
The Data Matching Process

2.1 Overview

An overview of the data matching process with its five major steps is shown in Fig. 2.1. The first step is the process of *data pre-processing*, which assures the data from both sources are in the same format. The second step, *indexing*, aims to reduce the quadratic complexity of the data matching process through the use of data structures that facilitate the efficient and effective generation of candidate record pairs that likely correspond to matches (i.e. refer to the same real-world entity).

In the third step, the actual *record pair comparison* occurs, where candidate record pairs are generated from the indexing data structures built in the previous step. These pairs are compared using a variety of field and record comparison functions. In the *classification* step, candidate record pairs are classified into *matches*, *non-matches*, and *potential matches* (depending upon the decision model used [129]). If record pairs are classified into potential matches, a manual *clerical review* process is needed to decide their final match status (match or non-match). In the *evaluation* step, the quality and completeness of the matched data, and the complexity of a data matching exercise, are evaluated.

For the deduplication of a single database, all steps of the data matching process are still applicable. Data pre-processing is important to assure the complete database is in a standardised format. This is especially important if records have been added to a database over time, potentially with changes in data entry techniques or methods that lead to different data formats and encodings over time. The indexing step is also of importance for deduplication, because comparing each record in a database with all others has a quadratic computation complexity.

2.1.1 A Small Data Matching Example

To illustrate the various challenges and tasks involved throughout the data matching process, an example consisting of two small database tables is used throughout this

P. Christen, *Data Matching*, Data-Centric Systems and Applications,
DOI: 10.1007/978-3-642-31164-2_2, © Springer-Verlag Berlin Heidelberg 2012

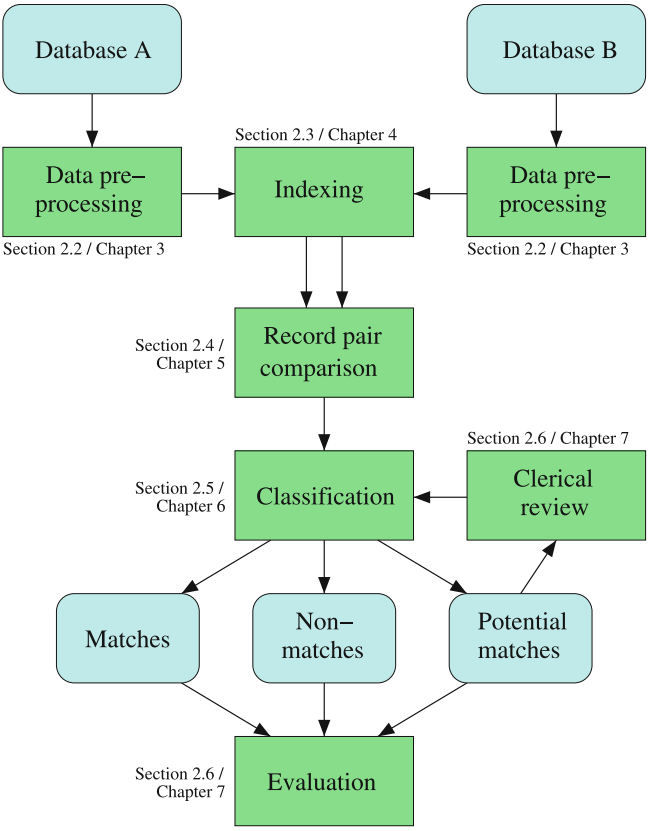

Fig. 2.1 The general process of matching two databases. The section and chapter numbers shown provide the road map through this chapter and Part II of this book

chapter. Figure 2.2 shows the two raw database tables that are to be matched. As can be seen, while they both contain name, address, and date of birth information, the structure of the two tables is different, as is the format of the values stored in the two tables. Each record is identified through a unique value in the 'RecID' attribute.

2.2 Data Pre-Processing

As the database tables in Fig. 2.2 show, data that are used for data matching can vary in format, structure and content. Because data matching commonly relies on personal information, such as names, addresses, and dates of birth, it is important to make sure that data sourced from different databases have been appropriately cleaned and standardised. The aim of this process is to ensure that the attributes used for the

Database A

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a1	Smith	John	42 Miller St	O'Connor	2602	A.C.T.	12-11-1970
a2	Neighan	Joanne	Brown Pl	Dickson	2604	ACT	8 Jan 1968
a3	Meyer	Marie	3/12-14 Hope Cnr	SYDNEY	2050	NSW	01-01-1921
a4	Smithers	Lyn	Browne St	DIXON	2012	N.S.W.	13/07/1970
a5	Nguyen	Ling	1 Milli Rd	Nrth Sydeny	2022	NSW	10/08/1968
a6	Faulkner	Christine	13 John St	Glebe	2037	NSW	02/23/1981
a7	Sandy	Robert	RMB 55/326 West St	Stuart Park	2713	NSW	7/10/1970

Database B

RecID	Name	Address	BYear	BMonth	BDay
b1	Meier, Mary	14 (App 3) Hope Corner, Sydney 2000	1927	4	29
b2	Janice Meyer	Bryan St, O'Connor ACT 2604	1968	11	20
b3	Jonny Smith	47 Miller Street, 2619 Canberra ACT	1970	12	11
b4	Lyng Nguyen	1 Millie Road, 2002 North Sydney, NSW	1968	8	10
b5	Kristina Fawkner	13 St John Street, 2031 Glebe	1981	2	23
b6	Bob Santi	55 East St; Stuart's Point; NSW 2113	1970	12	11
b7	Lynette Cain	6 / 12 Hope Corner, 2020 Sydney N.S.W.	1970	7	13

Fig. 2.2 Two small example database tables that are to be matched

matching have the same structure, and their content follows the same formats. It has been recognised that data cleaning and standardisation are crucial steps to successful data matching [78, 143]. The raw input data need to be converted into well-defined and consistent formats, and inconsistencies in the way information is represented and encoded need to be resolved [76, 224].

There are various factors that influence data quality, including different types of data entry errors, and the design of databases, such as the format and structure of their attributes. Some data quality factors are specific to personal information such as names and addresses. Name and address values are frequently entered either from handwritten forms using optical character recognition (OCR) software, read and typed, or typed as somebody speaks their personal information (possibly over the telephone). These different data entry modes can lead to typing, scanning, or phonetic errors [72]. How to deal with these challenges will be discussed in more detail in Chap. 3.

There are three (for certain types of data possibly four) major steps involved in data pre-preprocessing.

1. *Remove unwanted characters and words.* This step corresponds to an initial cleaning, where characters such as commas, colons, semicolons, periods, hashes, and quotes are removed. In certain applications, some words can also be removed if it is known that they do not contain any information that is of relevance to the data matching process. These words are also known as *stop words* [288].
2. *Expand abbreviations and correct misspellings.* This second step of data pre-processing is crucial to improve the quality of the data to be matched. Commonly

this step is based on look-up tables that contain name variations, nicknames, and common misspellings, and their correct or expanded versions. The standardisation of values conducted in this step will result in much reduced variations in attributes that contain name values.

3. *Segment attributes into well-defined and consistent output attributes.* This step deals with the common situation of database attributes that contain several pieces of information, such as the 'Address' attribute of the second database in Fig. 2.2. Finding a match between the content of this attribute and the content of the corresponding set of attributes in the first database ('Street', 'Suburb', 'Postcode' and 'State') is challenging. It is of advantage for data matching to split the content of attributes that contain several pieces of information into a set of new attributes that each contain one well-defined piece of information. The process of segmenting attribute values is also called *parsing* [143]. It is of high importance for both names and addresses, but also for dates. Various techniques have been developed to achieve such segmentation, either using rule-based systems or employing probabilistic techniques such as hidden Markov models [76]. These techniques will be covered in detail in Chap. 3.
4. *Verify the correctness of attribute values.* This last step can, for example, be employed for addresses if an external database is available that contains all known and valid addresses in a country or region. The detailed information in such an external database should include the range of street numbers, and the street name and type combinations that occur in towns and suburbs. Such a database will allow the verification of addresses and potentially even their correction, if for example it is known that there is no 'Miller Corner' in a certain town but only a 'Millers Court'. Applying such verification and correction might even be possible for name attributes, if, for example, a database of known residents is available that contains their full name and address details. However, because people can move, change their names, or might not even be registered (for example in a telephone directory), such name verification and correction might not help much to improve data quality. Rather, it might lead to wrong 'corrections' being introduced.

It is also possible, as illustrated in the pre-processed database tables in Fig. 2.3, to add attributes that are derived from existing attributes. For example, the gender of a person can often be correctly established from their given name (if a given name is distinctively used for males or females only). Similarly, if a postcode (or zipcode) value is missing in a record, its value could be extracted from the corresponding suburb or town name in case there is a unique postcode and suburb name combination.

It is important to note that the data pre-processing process must not overwrite the original input data. Once original values are overwritten (and if no backup has been made), then there is often no way to retrieve the original values in case a mistake was made during data pre-processing. Rather, new attributes should be created that contain the cleaned and standardised data. Ideally, data pre-processing is done in such a way that new database tables (or files) are generated that contain the cleaned and standardised data in such a format and structure that it can be easily used for the next step of the data matching process.

Database A – Cleaned and standardised

RecID	GivenName	Surname	Gender	StrPrefix	StrNum	StrName	StrType	Suburb	Postcode	State	BDay	BMonth	BYear
a1	john	smith	m		42	miller	street	oconnor	2602	act	12	11	1970
a2	joanne	neighan	f			brown	place	dickson	2604	act	8	1	1968
a3	mary	meier	f	3	12-14	hope	corner	sydney	2050	nsw	1	1	1921
a4	lynette	smithers	f			browne	street	dixon	2012	nsw	13	7	1970
a5	ling	nguyen	?		1	milli	road	north sydney	2022	nsw	10	8	1968
a6	christine	faulkner	f		13	john	street	glebe	2037	nsw	23	2	1981
a7	robert	sandy	m	rmb 55	326	west	street	stuart park	2713	nsw	7	10	1970

Database B – Cleaned and standardised

RecID	GivenName	Surname	Gender	StrPrefix	StrNum	StrName	StrType	Suburb	Postcode	State	BDay	BMonth	BYear
b1	mary	meier	f	apt 3	14	hope	corner	sydney	2000	nsw	29	4	1927
b2	janice	meier	f			bryan	street	oconnor	2604	act	20	11	1968
b3	john	smith	m		47	miller	street	canberra	2619	act	11	12	1970
b4	lyng	nguyen	?		1	millie	road	north sydney	2002	nsw	10	8	1968
b5	kristina	fawkner	f		13	saint john	street	glebe	2037	nsw	23	2	1981
b6	robert	santi	m		55	east	street	stuarts point	2113	nsw	11	12	1970
b7	lynette	cain	f	6	12	hope	corner	sydney	2020	nsw	13	7	1970

Fig. 2.3 The pre-processed (cleaned and standardised) versions of the two database tables from Fig. 2.2. Both databases now consist of the same attributes. The format and content of these attributes have been standardised in that various punctuations were removed, all letters were converted into lower case, nicknames replaced by the corresponding proper names, typographical errors corrected, dates and addresses were split into several well-defined fields, and contradicting data corrected (such as the postcode for suburb 'Glebe' which has a correct value of '2037' and not '2031', as was recorded in the original record 'b5'). Additionally, the attribute 'Gender' was added. Its values are based on the given name values of the corresponding records only for given names that are known to be distinctively male or female

2.3 Indexing

The cleaned and standardised database tables (or files) are now ready to be matched. Potentially, each record from one database needs to be compared with all records in the other database to allow the calculation of the detailed similarities between two records. This leads to a total number of record pair comparisons that is quadratic in the size of the databases to be matched. Matching the example databases from Fig. 2.3 leads to a total of $7 \times 7 = 49$ comparisons (between one record from database **A** and one record from database **B**).

Clearly, this naïve comparison of all record pairs does not scale to very large databases. Matching two databases with one million records each (as are common in many public and private sector organisations today) will result in $1{,}000{,}000 \times 1{,}000{,}000 = 1{,}000{,}000{,}000{,}000$, i.e. one trillion, record pair comparisons. Even if 100,000 comparisons can be performed in one second (10 µs or 0.01 ms per comparison), it would take 2,777.78 h, or nearly 116 days, to compare these two databases.

The majority of the comparisons will be between two records that are clearly not matches. As can be seen from Fig. 2.3, most record pairs have no or only a small number of attribute values that are equal or highly similar with each other. For

example, record 'a1' in database **A** has the same year of birth (1970) as records 'b3', 'b6', and 'b7' from database **B**, but it only has two other attribute values in common with record 'b6' (gender 'm' and street type 'street'), and no other attribute value in common with record 'b7'.

It is generally the case that when matching two databases, the potential number of comparisons grows quadratically with the number of records in the databases to be matched, while the number of possible true matches only increases linearly. This is because it is likely that one record from database **A** only matches to a small number of records from database **B**. In the case where both databases **A** and **B** do not contain duplicate records (i.e. several records that refer to the same entity), then the maximum number of true matches that are possible is always smaller than or equal to the number of records in the smaller of the two databases.

To reduce the possibly very large number of pairs of records that need to be compared, *indexing* techniques are commonly applied [64]. These techniques filter out record pairs that are very unlikely to correspond to matches. They generate *candidate record pairs* that will be compared in more detail in the comparison step of the data matching process to calculate the detailed similarities between two records, as will be described in the following section.

Various indexing techniques for data matching and deduplication have been developed [64]. The traditional approach to indexing is called *blocking* [20]. It splits each database into smaller blocks according to some *blocking criteria* (generally known as a *blocking key*). Only records from the two databases that have been inserted into the same block, i.e. who share the same value for a blocking criteria (have the same blocking key value), are compared with each other. An example blocking criteria could be that records that have the same postcode value are inserted into the same block, while another blocking criteria could be that records that have the same phonetically encoded surname value are inserted into the same block. Such phonetic encoding algorithms, like for example *Soundex* [57], are commonly used in the indexing step to ensure that records are inserted into the same blocks even if they have some typographical variations in the value of their blocking criteria. Chapter 4 will discuss traditional blocking and several other indexing techniques in more detail, and also provide an experimental evaluation of these techniques to illustrate their performance on different types of data.

When the traditional blocking technique is applied to the cleaned and standardised databases from Fig. 2.3 using the two blocking criteria (1) Soundex of surname values ('Sndx-SN') and (2) taking the first three digits of postcode values ('F3D-PC'), then the blocks and candidate record pairs shown in Figs. 2.4 and 2.5 are generated. As can be seen, from the full number of 49 record pairs (without indexing), only 12 candidate record pairs are generated. These candidate pairs will be compared in detail, as will be described in the following section.

Looking at the candidate record pairs generated and comparing the corresponding record pairs in Fig. 2.3, one can see that this specific blocking approach selects most of the record pairs that likely refer to a match, such as (a1, b3) ('John Smith'), (a3, b1) ('Mary Meier'), and (a5, b4) ('Ling Nguyen'). This blocking approach does, however, miss the pair (a4, b7) ('Lynette Smithers' / 'Lynette Cain') which is possibly

Database A – Blocking information

RecID	Surname	Sndx-SN	Postcode	F3D-PC
a1	smith	s530	2602	260
a2	neighan	n250	2604	260
a3	meier	m600	2050	205
a4	smithers	s536	2012	201
a5	nguyen	n250	2022	202
a6	faulkner	f425	2037	203
a7	sandy	s530	2713	271

Database B – Blocking information

RecID	Surname	Sndx-SN	Postcode	F3D-PC
b1	meier	m600	2000	200
b2	meier	m600	2604	260
b3	smith	s530	2619	261
b4	nguyen	n250	2002	200
b5	fawkner	f256	2037	203
b6	santi	s530	2113	211
b7	cain	c500	2020	202

Fig. 2.4 The blocking key values (BKVs) generated from the two database attributes 'Surname' and 'Postcode'. For surnames, BKVs are generated by applying Soundex encoding [57] on surname values (labelled 'Sndx-SN'), while the BKVs of postcodes are generated by taking their first three digits only (labelled 'F3D-PC')

Candidate record pairs generated from Surname blocking

BKVs	Candidate record pairs
m600	(a3, b1), (a3, b2)
n250	(a2, b4), (a5, b4)
s530	(a1, b3), (a1, b6), (a7, b3), (a7, b6)

Candidate record pairs generated from Postcode blocking

BKVs	Candidate record pairs
202	(a5, b7)
203	(a6, b5)
260	(a1, b2), (a2, b2)

(a1, b2)
(a1, b3)
(a1, b6)
(a2, b2)
(a2, b4)
(a3, b1)
(a3, b2)
(a5, b4)
(a5, b7)
(a6, b5)
(a7, b3)
(a7, b6)

Fig. 2.5 The candidate record pairs generated from the BKVs that occur in both database **A** and **B**. The table on the right-hand side shows the union of all generated candidate record pairs

the same woman because both records have the same given name and the same date of birth. This woman might have married and changed her surname and address, and is therefore missed by the two blocking criteria used. This example highlights the careful need for domain and data matching knowledge when defining blocking criteria. Both the quality and completeness, as well as the frequency distribution of the values in an attribute need to be considered when attributes are selected to be used as blocking keys. These issues will be further discussed in Chap. 4.

2.4 Record Pair Comparison

The candidate record pairs that were generated in the indexing step require detailed comparisons to determine their overall similarity. Generally, the similarity between two records is calculated by comparing several record attributes. Ideally, not just the attributes used in the indexing step are used for this, but also other attributes that

are available in the databases that are matched. In the running example used in this chapter, while the blocking was based on the 'Surname' and 'Postcode' attributes, the comparison should for example also include the attributes 'GivenName', 'StreetNum', 'StreetName', 'Suburb', and the three date of birth attributes. The more similar values two records have in common across these attributes, the more likely it will be that they correspond to the same individual.

Even after records have been cleaned and standardised, it is possible that there are different attribute values in the records that correspond to true matches (i.e. that refer to the same entity). In the example, the records 'a6' and 'b5' very likely correspond to the same individual. However, the given name, surname and street name values of these two records are all slightly different. Rather than only conducting exact matching between attribute values, it is therefore essential to conduct some form of approximate comparison that for a compared pair of attribute values returns a measure of their similarity.

Generally, similarity values are normalised numerical values, with a similarity of 1.0 corresponding to an exact match between two attribute values, a similarity of 0.0 corresponding to a total dissimilarity between two values, and similarities in-between 0.0 and 1.0 corresponding to some degree of similarity between two attribute values. Figure 2.6 shows the similarities calculated between attribute values for the 12 candidate record pairs from Fig. 2.5.

Given different attributes contain various types of data, different approximate similarity comparison functions are required [61]. For attributes that contain string values, such as names and addresses, a large number of approximate string comparison functions is available [57]. Specific comparison functions for dates, ages, times, locations and numerical values are used for attributes that contain such data [61]. For certain sets of attributes, such as given names, surnames, or dates (consisting of a day, month and year value), it is also advisable to compare attributes as a group rather than only individually. For example, for names from several Asian cultures, certain name values can interchangeably be used as given name and surname (such as 'Qing Yang' and 'Yang Qing'). Therefore, comparing the given name value from one record with the surname value from another record, and the other way round, will help to detect pairs of records where these two name components have been swapped. Similarly, dates can have their day and month values swapped as they are recorded either following the American date format (MM/DD/YYYY) or the format used in many other countries (DD/MM/YYYY). Chapter 5 covers a large number of different comparison functions for different types of data, and highlights various issues that need to be considered when using for example names and addresses for data matching.

For each candidate record pair several attributes are generally compared, resulting in a vector of numerical similarity values for each pair. These vectors are called *comparison vectors*. They will be used in the classification step to decide if a record pair is classified as a match or a non-match.

The comparison vectors resulting from the comparison of the 12 candidate records pairs of the running example are shown in Fig. 2.6. Different approximate comparison functions were used. The sum of all similarity values for each comparison vector is

RecID	GivenName	Surname	StrNum	StrName	Suburb	BDay	BMonth	BYear	SimSum
a1	john	smith	42	miller	oconnor	12	11	1970	
b2	janice	meier		bryan	oconnor	20	11	1968	
	0.61	0.6	0.0	0.0	1.0	0.0	1.0	0.5	3.71
a1	john	smith	42	miller	oconnor	12	11	1970	
b3	john	smith	47	miller	canberra	11	12	1970	
	1.0	1.0	0.5	1.0	0.6	0.5	0.5	1.0	6.10
a1	john	smith	42	miller	oconnor	12	11	1970	
b6	robert	santi	55	east	stuarts point	11	12	1970	
	0.47	0.6	0.0	0.0	0.31	0.5	0.5	1.0	3.39
a2	joanne	neighan		brown	dickson	8	1	1968	
b2	janice	meier		bryan	oconnor	20	11	1968	
	0.78	0.56	0.0	0.73	0.51	0.0	0.5	1.0	4.08
a2	joanne	neighan		brown	dickson	8	1	1968	
b4	lyng	nguyen	1	millie	north sydney	10	8	1968	
	0.47	0.64	0.0	0.0	0.45	0.0	0.0	1.0	2.56
a3	mary	meier	12-14	hope	sydney	1	1	1921	
b1	mary	meier	14	hope	sydney	29	4	1927	
	1.0	1.0	0.4	1.0	1.0	0.0	0.0	0.75	5.15
a3	mary	meier	12-14	hope	sydney	1	1	1921	
b2	janice	meier		bryan	oconnor	20	11	1968	
	0.47	1.0	0.0	0.0	0.44	0.0	0.5	0.5	2.91
a5	ling	nguyen	1	milli	north sydney	10	8	1968	
b4	lyng	nguyen	1	millie	north sydney	10	8	1968	
	0.83	1.0	1.0	0.94	1.0	1.0	1.0	1.0	7.78
a5	ling	nguyen	1	milli	north sydney	10	8	1968	
b7	lynette	cain	12	hope	sydney	13	7	1970	
	0.6	0.47	0.5	0.0	0.5	0.5	0.0	0.5	3.07
a6	christine	faulkner	13	john	glebe	23	2	1981	
b5	kristina	fawkner	13	saint john	glebe	23	2	1981	
	0.81	0.87	1.0	0.45	1.0	1.0	1.0	1.0	7.12
a7	robert	sandy	326	west	stuart park	7	10	1970	
b3	john	smith	47	miller	canberra	11	12	1970	
	0.47	0.47	0.0	0.0	0.54	0.0	0.5	1.0	2.98
a7	robert	sandy	326	west	stuart park	7	10	1970	
b6	robert	santi	55	east	stuarts point	11	12	1970	
	1.0	0.73	0.0	0.83	0.78	0.0	0.5	1.0	4.85

Fig. 2.6 Similarity values (comparison vectors) calculated using different approximate similarity comparison functions for the 12 candidate record pairs from Fig. 2.5. For attributes containing names, the Jaro–Winkler [215] approximate string comparison function was used, while for the attributes that contain numbers the edit distance [89] function was employed

shown on the right-hand end of each vector (SimSum). These sums can be used for a simple threshold-based classification approach as will be described in the following section.

2.5 Record Pair Classification

Classifying the compared record pairs based on their comparison vectors or their summed similarities is a two-class (binary) or three-class classification task. In the two-class case, each compared record pair is classified to be either a *match* or a *non-match*. The first class contains the pairs of records that are assumed to refer to the same real-world entity, while for the second class it is assumed that the two records in a pair do not refer to the same entity. All record pairs that were removed by the indexing step and that were not compared in the comparison step are implicitly classified as non-matches.

In traditional data matching approaches, for example those based on probabilistic record linkage [108, 143], record pairs are classified into one of three classes, rather than only matches and non-matches. The third class are the *potential matches*. These are the record pairs where the classification outcome is not clear, and where a manual *clerical review* [143] is required to decide the final match status.

Most research in data matching in the past decade has concentrated on improving the classification accuracy of record pairs. Various machine learning techniques have been investigated, both unsupervised and supervised [31, 59, 85, 102]. So called *active learning* techniques have also been investigated [231, 252]. With these techniques, a subset of (difficult to classify) record pairs is given for manual assessment and classification (into matches and non-matches), and the resulting classified record pairs are used to re-train a new and improved classifier. After several iterations, this process can achieve an improved matching accuracy with much reduced manual efforts compared to the traditional approach of full manual clerical review of all potential matches.

The classification of each compared record pair can be based on either the full comparison vectors or on only the summed similarities. Figure 2.6 shows the 12 compared record pairs, their comparison vectors and their summed similarities (SimSum). The maximum possible summed similarity (of two records that are matching exactly on all compared attributes) would be 8.0, because eight attributes are compared each returning a similarity between 0 and 1. Figure 2.7 shows the outcomes of a simple threshold-based classifier where all compared record pairs with a SimSum value equal to or above 6 are classified as matches, all pairs with a SimSum value between 4 and 6 as potential matches, and all other pairs as non-matches. As a result, the three pairs (a1, b3), (a5, b4) and (a6, b5) will (presumably) be correctly classified as matches. Of the three potential match pairs (a2, b2), (a3, b1) and (a7, b6) given for manual clerical review, the second pair (a3, b1) will likely be classified as a match, while the other two pairs might be classified as non-matches. Figure 2.8 shows the actual records of the three pairs that were classified as matches.

Candidate pair	SimSum	Classification
(a1, b2)	3.71	Non-match
(a1, b3)	6.10	Match
(a1, b6)	3.39	Non-match
(a2, b2)	4.08	Potential match
(a2, b4)	2.56	Non-match
(a3, b1)	5.15	Potential match
(a3, b2)	2.91	Non-match
(a5, b4)	7.78	Match
(a5, b7)	3.07	Non-match
(a6, b5)	7.12	Match
(a7, b3)	2.98	Non-match
(a7, b6)	4.85	Potential match

Fig. 2.7 Three-class classification of the compared record pairs from Fig. 2.6 into matches (SimSum $\geq$ 6.0), non-matches (SimSum $\leq$ 4.0) and potential matches (6.0 > SimSum > 4.0)

Database A

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a1	Smith	John	42 Miller St	O'Connor	2602	A.C.T.	12-11-1970

Database B

RecID	Name	Address	BYear	BMonth	BDay
b3	Jonny Smith	47 Miller Street, 2619 Canberra ACT	1970	12	11

Database A

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a5	Nguyen	Ling	1 Milli Rd	Nrth Sydeny	2022	NSW	10/08/1968

Database B

RecID	Name	Address	BYear	BMonth	BDay
b4	Lyng Nguyen	1 Millie Road, 2002 North Sydney, NSW	1968	8	10

Database A

RecID	Surname	GivenName	Street	Suburb	Postcode	State	DateOfBirth
a6	Faulkner	Christine	13 John St	Glebe	2037	NSW	02/23/1981

Database B

RecID	Name	Address	BYear	BMonth	BDay
b5	Kristina Fawkner	13 St John Street, 2031 Glebe	1981	2	23

Fig. 2.8 The three record pairs that were classified as matches

The traditional approaches to record pair classification have the problem that each record pair is classified independently of all others pairs based only on its comparison vector (or its summed similarity). As a result, a single record from one database can be matched with several records from the other database. In certain applications this might not be permitted, for example if it is known that the two databases that are matched each only contain one record per entity (i.e. no duplicate records). Recent

research into *collective* classification techniques for data matching has aimed to overcome this drawback by classifying record pairs not only based on their pair-wise similarities, but also using information on how records are related or linked to other records. These approaches apply relational clustering or graph-based techniques [31, 155, 272] to generate a global decision model. Much improved matching results have been achieved with these collective classification techniques. Their computational complexities, however, make scaling these techniques to the matching of very large databases challenging [142]. These techniques, as well as the more traditional pair-wise classification techniques, will be presented in detail in Chap. 6.

2.6 Evaluation of Matching Quality and Complexity

Once the compared record pairs are classified into matches and non-matches, the quality of the identified matches needs to be assessed. Matching quality refers to how many of the classified matches correspond to true real-world entities, while matching completeness is concerned with how many of the real-world entities that appear in both databases were correctly matched [71]. As will be discussed in detail in Chap. 7, accuracy measures such as precision and recall, that are also used in fields such as data mining, machine learning, and information retrieval, are commonly used to assess matching quality

Both matching accuracy and completeness are affected by all steps of the data matching process, with data pre-processing helping to make values that are different to each other more similar, indexing filtering out pairs that likely are not matches, and the detailed comparison of attribute values providing evidence of the similarity between two records. While the accuracy of data matching is mostly influenced by the comparison and classification steps, the indexing step will impact on the completeness of a data matching exercise because record pairs filtered out in the indexing step will be classified as non-matches without being compared.

The complexity of a data matching or deduplication project is generally measured as the number of candidate record pairs that are generated by an indexing technique compared to the number of all possible pairs that would be generated in the naive. matching where no indexing is applied. For the running example shown in this chapter, the naive full pair-wise comparison of all records from database **A** with all records from database **B** would result in $7 \times 7 = 49$ record pair comparisons. The indexing (blocking) applied in this example has reduced this number to the 12 candidate pairs shown in Fig. 2.5. This corresponds to a reduction of over 75 %.

To evaluate the completeness and accuracy of a data matching project, some form of ground-truth data, also known as *gold standard*, are required. Such ground-truth data must contain the true match status of all known matches (the true non-matches can be inferred from them). However, obtaining such ground-truth data is difficult in many application areas. For example, when matching a large tax payers database with a social security database it is usually not known which record pair classified as a match refers to a real, existing individual who has a record in both databases.

Only further investigations, such as checking extra data about the individual under consideration, or even contacting them, can help determine the truth about such a classified match.

A related problematic issue is the manual classification of potential matches through clerical review. It is often difficult to make a manual match or non-match decision with high confidence if the two records in a potential match pair contain several attribute values that differ from each other. Without further external information, a decision that was made manually might be wrong. Additionally, the manual classification of a large number of potential matches is a time-consuming, cumbersome and error-prone process. Assuming that the manually classified potential matches can be used as training data for supervised classification or even as gold standard for another data matching project is dangerous. The issues relevant to evaluating data matching will be discussed in detail in Chap. 7.

As the classification results in Fig. 2.7 show, even the matching of two small example databases results in a quite imbalanced distribution of matches to non-matches (four matches to eight non-matches after clerical review in this example). This class-imbalance gets much worse as larger databases are being matched. The number of matches generally grows linear (or even sublinear), while the number of non-matches (even after indexing) grows subquadratic [71], as will be discussed further in Chap. 6. When evaluating the results of a data matching or deduplication project, even when ground-truth data are available, care must be taken. The normal accuracy measure that is generally used for many classification tasks is not recommended. Various measures that are suitable for assessing the quality and complexity of data matching and deduplication will be presented in detail in Chap. 7.

2.7 Further Reading

The data matching process (with some variations to the steps described in this chapter) is discussed in most books that cover data matching [19, 143, 195, 249], as well as in several reports and overview articles [103, 119]. A recent survey of indexing techniques is provided by Christen [64], while many surveys have been written over the past decades on approximate string comparisons techniques [57, 84, 133, 152, 175, 196]. On the other hand, while many different classification techniques have been explored within the domain of data matching, only a few publications have comparatively evaluated several techniques [59, 102, 168]. The issues involved in evaluating data matching results are being discussed in two recent publications [71, 187].

http://www.springer.com/978-3-642-31163-5

Data Matching
Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection
Christen, P.
2012, XX, 272 p., Hardcover
ISBN: 978-3-642-31163-5