Probabilistic Graphical Model for Entity Resolution

by

Ziyue He

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

The task of entity resolution, also known as record linkage, is to find out which records refer to the same entity across several datasets and link them together. In real applications, record linkage is widely used in multiple fields, such as business, healthcare, criminal investigation, among others. Up to now, many techniques have been developed to accomplish this task. Accuracy and efficiency are the two most important factors when quantifying the quality of a record linkage approach. However, finding the exactly matched records is still a big challenge since several types of noise/errors are always present in real-world data. These errors (noise) could be phonetical, typographical, and may be the result of optical character recognition (OCR). For years, a significant number of comparison methods have been proposed to describe the level of similarity (similarity score) between identity fields among pairs of records. Nevertheless, a proper selection of comparison methods, appropriate identity fields, suitable classifiers, and the determination of the thresholds' values remains a challenging problem. The objective of this dissertation is to design and analyze a probabilistic graphical model (PGM) to realize a proper record linkage task. In this study, a Bayesian network, which is an example of PGM, is used to calculate the probabilities of being matched among record pairs to decide if these can be linked or not. Furthermore, several comparison methods and fields are considered for this model. For each combination of a comparison method and a field, a similarity score is obtained. With these scores, along with two predefined thresholds, a decision can be made to determine whether a record pair is a match, not a match, or a probable match which would need a closer inspection (clerical review). Not every comparison method or field is equally relevant in practice. Therefore, to describe the roles of the selected comparison methods and fields, weights are added to the Bayesian network. These weights are previously optimized by a modified supervised gradient descent learning

scheme. Synthetic datasets with different levels of noise are used to perform the experiments. The experimental studies show that the proposed record linkage model can calculate the matching probabilities of records (that could hypothetically be matched) in an accurate and efficient manner. Furthermore, the proposed model can offer an insight on which comparison methods and fields are more significant for a correct record linkage.

Preface

This thesis was completed under the supervision of Dr. Witold Pedrycz. Chapter 3.1, Chapter 3.3, and Chapter 4.1 of this dissertation have been submitted to "*Data & Knowledge Engineering*" as O. F. Reyes-Galaviz, W. Pedrycz, Ziyue He, Nick J. Pizzi, "A Supervised Gradient-Based Learning Algorithm for Optimized Entity Resolution" and it is under review. I was responsible for literature review, data generation, some experimental studies, and results analysis. Dr. O. F. Reyes-Galaviz and Dr. W. Pedrycz were the idea developers and were involved with manuscript composition and concept formation. The other chapters were developed by myself.

Acknowledgements

First of all, I would like to thank my supervisor, Professor Witold Pedrycz, for the development of the ideas in this thesis and the comments on the thesis writing. His patience and support made my Master's study completed in a relaxed and enjoyable atmosphere. I am also very grateful for the constant encouragement and guidance from him.

Secondly, I would like to thank Dr. Orion Fausto Reyes Galaviz, a postdoctoral research fellow in our group, for his advice on how to develop the experiments and how to analyze the experimental results.

Furthermore, I would like to thank my parents for their concerns, support, and understanding. Their love helped me a lot during the process of my graduate study.

I also would like to thank my boyfriend, Mingjiang Xie, for his encouragement. Thank him for accompanying me in the past six years and supporting me all the time.

Last but not least, I am very thankful to the staff at the Department of Electrical and Computer Engineering, who are always friendly and enthusiastic, especially Pinder Bains and Wendy Barton. They are always willing to help and patiently answering questions.

Ziyue He
University of Alberta
September 2017

Table of Contents

List of Tables

List of Figures

List of Symbols

$\boldsymbol{u}$	Input datum (similarity scores between identity fields with selected comparison functions) of the Artificial Neural Network (ANN) and the Bayesian network
t	Target datum (1 for a matching record pair and 0 for a non-matching record pair) of the ANN
v	Output datum of the ANN
n	Number of candidate record pairs
c	Number of selected comparison methods
m	Number of selected data fields
$\boldsymbol{z}$	Vector representing the weights of the neurons in the input layer of the ANN
$\boldsymbol{w}$	Vector representing the weights of the neurons in the hidden layer of the ANN
$\boldsymbol{\mu}$	Vector representing the normalized weights of the variables in the first layer of the Bayesian network
$\boldsymbol{\lambda}$	Vector representing the normalized weights of the variables in the second layer of the Bayesian network
$\boldsymbol{\sigma}$	Vector representing the conditional probability distribution of the variable in the first layer of the Bayesian network
T	Matrix representing the condition probability distribution of the variable in the second or third layer of the Bayesian network
Q	Performance Index for the ANN
τ_1	*Clerical review* threshold
τ_2	*Autolink* threshold

1. Introduction

Entity Resolution (also known as Record Linkage or Object Matching) is to identify which records represent the same real-world entity across several data sources and merge the matching record pairs. The entities can be a set of publications, consumer goods, incidents, or people (customers, patients, employers, or convicts). Record linkage is of great use in various situations, for example, retailers who want to analyze and merge the shopping predispositions of customers, hospitals who want to gather the medical records of a patient across hospitals that he or she has visited, policemen who want to check a suspect's background from records related to him or her.

Up to now, a diversity of approaches have been developed to try to resolve the entity resolution problem where, in essence, records containing personal information are indexed, and then the identifiers are compared to obtain a similarity score which is used to classify the records pairs. The identifiers (also known as 'fields') can be date of birth, social security ID, given name, surname, address, suburb or phone number. The well-developed entity resolution approaches include pair-wise classification, clustering methods, rule-based approaches, probabilistic inference, and others. However, it is still a challenging task to find an exact match for a given entity as records may contain phonetical errors, typographical errors, missing fields, or fields that change over time (e.g. phone numbers, address, etc.). On the other hand, some individuals may have the same given name/surname (father/junior); some may be born on the same date (twins); some may live in the same suburb. Hence, even when the records referring to the same entity are compared, it is very likely that the result does not show a full agreement, or in contrast, the similarity score can be very high for two different identities. Thus, it is better to select as many fields as possible to have matching record pairs separated from the non-matching pairs. And if the model allows it, obtain significant information about the importance of each field.

Also, many comparison methods (functions) have been developed to determine a similarity score which indicates how significantly two fields agree with each other. Different selections of comparison functions may lead to different similarity scores (for the same record pairs) since some methods outperform others on specific types of strings. For example, some methods perform better on long strings; some are good at comparing the strings that include swapping of adjacent characters; some are working better when comparing alphanumerical strings; some are more efficient at comparing name strings. Besides, names that are same phonetically may have different spellings, and some certain comparison methods do a good job for this situation. Therefore, the

second challenge is to select proper comparison methods. Moreover, eliminating the comparison methods that contribute little to an accurate record linkage will perhaps help to augment the accuracy of the linkage of records.

After the similarity scores are obtained from the selected fields and comparison methods, a third challenge still remains, to classify record pairs in terms of the similarity scores. The objective is to classify the record pairs into three classes: match, non-match, and potential match. Therefore, two thresholds are defined to complete the classification task. These are commonly known as the *clerical review* and *autolink* thresholds. The record pairs whose similarity scores are falling above *autolink* are classified as matching pairs. If a similarity level is found below *clerical review*, the corresponding record pair will be classified as a non-matching pair. When a score falls between these two thresholds, the record pair is classified as a potential match. These records should be reviewed manually by a clerk (hence the name) to decide whether it is a true match or a non-match. Thus, it is critically important to adjust the values of these two thresholds not only to minimize the number of potential matches, but also to reduce the number of misclassified pairs.

The objective of this dissertation is to design and analyze a probabilistic graphical model (PGM) to accomplish the record linkage task. Many data fields and comparison methods are aggregated by the constructed PGM. Aside from the classification of record pairs, the other objective of this study is to select the most relevant data fields and the most significant comparison methods. An Artificial Neural Network (ANN) is constructed beforehand to determine the weights which are used to describe the significance of the selected fields and comparison methods. With the supervised gradient descent learning scheme, the ANN can adjust the weights to approximate the desired output. The structure of ANN is quite similar to the PGM. Therefore, the optimized weights coming from the ANN can be inserted into the PGM. In addition, the structure of the ANN is also flexible to aggregate the similarity scores from different fields and comparison methods. Among various types of PGMs, the Bayesian Network is selected to implement record matching because it contains directed edges between the nodes, which can clearly show causal relationships between the nodes (variables) [1][2]. In this study, we consider a matching score as the probability of being a match for each record pair. The Bayesian network is used to compute the probabilities, which indicate how likely the record pairs are a true match or not a match. Generally, in the proposed Bayesian network, the similarity scores coming from each field and each comparison method are multiplied by the optimized weight and aggregated to achieve the final matching score.

With the final matching score, along with two predefined thresholds, a decision can be made to determine whether a record pair is a match, not a match, or a potential match. Furthermore, these matching scores can be imported into Neo4j, a highly scalable graph database, to visualize the results as graphs with directed edges and nodes. Queries can be done quickly and accurately with the graph inside the Neo4j environment.

The motivation of this dissertation is to build an efficient model to complete the record linkage task. PGM is widely used in the area of knowledge representation, and it has excellent performance on inferences under uncertainty [1]. Given the similarity scores for all selected fields and comparison methods, an inference system (PGM) is constructed to estimate the probability of being a match for each record pair. In the proposed model, an exact inference, which is based on variable elimination algorithm, is performed to calculate the probabilities. With the optimized weights calculated by the learning scheme, along with the proposed inference system (PGM), the number of misclassified record pairs is able to be minimized significantly. Moreover, the optimized weights can give us an insight on which fields and comparison methods contribute more to the linkage of records. This will help us select the most significant comparison methods and the most relevant data fields. The structure of the constructed PGM is very simple, and it is flexible enough to allow the aggregation of the scores coming from different data fields and comparison methods. Also, the structure can be modified easily when some data fields or comparison methods need to be deleted/added.

This study is structured as follows. In Chapter 2, an overview of entity resolution methods is offered. In Chapter 3, we generally describe the background of the record linkage model to be constructed. We primarily focus on the ANN model and the PGM. Moreover, the records indexing method and the selected comparison methods are described in this chapter. In Chapter 4, the modeling of the record linkage is described, including the optimization of weights, the determination of the conditional probability distributions, and performance assessment. The experimental studies are shown in Chapter 5, where four synthetic datasets are used to perform the experiments. Furthermore, a revised record linkage model which eliminates some less significant comparison methods is tested with the four datasets. The experiment with given name indexing is also shown in this chapter. Conclusions and future studies are offered in Chapter 6.

2. Overview of Entity Resolution Methods

The entity resolution (as known as record linkage) problem was defined by Newcombe et al. [3] in 1959, and later, in 1969, it was formalized by Fellegi and Sunter [4]. Their work lay the mathematic foundation of most entity resolution approaches developed until now. In their work, two agreement probabilities were defined, one for being matching and one for being non-matching for record pairs. After adding a weight denoting the agreement or disagreement, the matching score could be determined based on the agreement probability. Here, they used a single threshold for classification. Furthermore, two indexes, *precision* and *recall*, were used to measure the accuracy of record linkage. However, there is a limitation of their work that the method was developed based on the assumption that each field for the data was independent of each other.

Later, naïve Bayes classifier [5][6] came up by some researchers as a probability record linkage method. It is similar to the approach that Fellegi and Sunter worked on, which is to compute the probability of being a match or a non-match. A decision was made according to the relationship between the probability of being a match and a single threshold determined by a trade-off between *precision* and *recall*. [7] compared the naïve Bayes classifier and a distance-based algorithm, and found that the naïve Bayes classifier outperformed the other one by measuring *precision*, *recall*, and *f-measure*. Up to now, these three indexes are still used for quantifying the quality of record linkage problems. *Precision* is also known as Positive Predictive Value (PPV). *Recall* is the True Positive Rate (TPR) which is also named as *Sensitivity* [8]. *F-measure* combines *precision* and *recall* together as it is the harmonic mean of these two indexes. Nevertheless, in [3] two thresholds were used to classify the record pairs in record linkage. The record pairs whose scores were above the higher threshold were labeled as *linked*. On the contrary, when the scores were below the lower threshold, the corresponding record pairs were labeled as *unlinked*. Then the record pairs whose scores were within these two thresholds were labeled as *possible links* and needed to be reviewed manually. In [9], these two thresholds were known as *clerical review* and *autolink*, and these two names still serve today. In record linkage process, as two thresholds are employed to classify the record pairs into three classes: match, non-match, and potential match, how to select the values of them plays a really important role. To obtain a high accuracy record linkage result, the optimized values of these two thresholds should be determined by minimizing the number of misclassified records. These misclassified record pairs always contain false negatives, false positives, and the number of record pairs that need to be reviewed.

Nowadays, the idea known as *three-way decision rules* [10][11] is widely used, which is an advanced version of the above ideas. In this idea, two thresholds generate three regions: positive (*P*), boundary (*B*), and negative (*N*). There are three decisions (rules, or actions) associated with them, respectively acceptance, abstaining, and rejection. The work in [10][11] indicates that this probabilistic three-way decision rules method performs better than the two-way decisions method in certain conditions. The costs of different decisions can be defined by a 3×2 matrix with two states, positive (*P*) and negative (*N*). The costs (risks) of taking actions *P*, *B*, and *N* in the *P* region are denoted by λ_{PP}, λ_{PB}, and λ_{PN}. Similarly, when taking actions *P*, *B*, and *N* in the *N* region, the costs (risks) are denoted by λ_{NP}, λ_{NB}, and λ_{NN}. The values of these risks are usually provided by experts under specific study. However, it is evident that λ_{PP} and λ_{NN} are both zeros since they refer to two correct actions, accepting successfully and rejecting successfully. Therefore, the target is to minimize the remaining costs, where λ_{NP} means false positives, λ_{PN} means false negatives, λ_{PB} and λ_{NB} mean abstaining.

Another optimal decision model, which depended on a cost-based Bayesian method, was found in [12]. This approach preferred to minimize the cost of making a decision rather than the probability of error like the other existing models. The efficiency of this cost-based approach was compared with that of an error-based approach on the mean cost and the error probability. In the end, they found that the cost-based Bayesian approach took less computation time and mean cost, which indicated that this approach was more efficient than other error-based approaches. Moreover, a general model came up as a Bayesian idealization of entity resolution in [13].

Except for the probabilistic record linkage method which was adopted by the above literature, the machine learning techniques, like Artificial Neural Networks (ANNs), are also commonly used in the area of record linkage. In [14], an ANN was used to discover the relationships between variables, and to translate the information from one file to another by a supervised back-propagation technique. Furthermore, different forms of ANNs were applied for linking records in [15][16]. The results showed that ANNs performed better than the naïve Bayesian classifier. In probabilistic record linkage methods, the probabilistic formulas can produce weights. However, the errors generated by assuming the independence cannot be overcame since this method fails to adjust the weights. On the contrary, ANNs are able to adjust the weights to make the defined performance index satisfy a specific value. This shows that it is a good choice to select an ANN to determine the optimized weights and match records. Particularly, a structured neural

network (SNN) was proposed in [16] to test multi-source sensing data. There were several separate multi-input single-output ANNs with one hidden layer in this structure, and the outputs of these separate ANNs were connected to another single output neuron. Since there was only one sensor in this structure, all individual ANNs' results were aggregated to one output. Similarly, in our study, an augmented structure is used to emphasize the impact of each data field and each comparison method. It is obvious that the ANN's architecture is determined by the numbers of selected data fields and comparison methods.

For the algorithm to be used in ANNs, several researchers' work contributes a lot to develop our understanding. [17] presented a novel unsupervised two-step approach to automate the record linkage process. In this paper, four methods, viz. iterative Support Vector Machine (SVM), fully supervised SVM, nearest-neighbor approach and hybrid approach, were compared with each other by measuring completeness, reduction ration, and *f*-measure. Finally, it was found that the fully supervised SVM method performed best on all experimental datasets. For the supervised learning algorithms applied in record linkage, the principal ones are the gradient decent algorithm, SVM, and neural network. In [18], a comparison between the gradient descent based heuristic algorithm and Choquet integral optimization algorithm was proposed. The Choquet integral optimization algorithm guaranteed the convergence to the optimized solution, but it required much time. For the heuristic algorithm, it needed low and stable time, even though the error rate might be a little higher. Since computational time and resources are always limited in practice, the gradient descent based heuristic algorithm is the better choice given that it is able to provide a good approximation of the optimized solution. Hence, the supervised gradient descent based algorithm is applicable to be used in ANNs for record linkage.

However, the phenomenon of overtraining is always seen in ANNs, which indicates that ANNs may become so trained towards the measurement methods employed during the training process. Therefore, the ANN is not applicable in general cases. The Probabilistic Graphical Model (PGM) is a generic model that represents the probability-based relationships among random variables, and it is widely used for knowledge representation and inferences under uncertainty [1]. This concept was first proposed by J. Whittarke in 1990 [19]. In the following period, PGMs were applied in objective segmentation [20][21], document image matching [22], video fusion technique [23], and native language identification [24]. Furthermore, it can also be used for classification and retrieval of multimedia documents as it can learn the statistical structure of

various kinds of media, such as video, audio, and text [25]. In addition, [26][27] presented that graphical models are usually employed to analyze sequence, which indicates that PGMs do a lot of contribution to the research in bioinformatics. In [28], a PGM used for breast cancer diagnosis and prognosis was proposed. From these researchers' work, it can be seen that the PGM is a powerful tool for making decisions under uncertainties across big data sources with poor quality. It is appropriate for building predictive models or inference models in multiple areas.

From [29][30][31], we can see that among various types of PGMs, the Bayesian network is one of the most widely used models for decision-making involving uncertainty and modeling causality. In [30], a distributed data fusion architecture, which was based on distributed relational Bayesian networks, was designed for data fusion. This paper indicates that an effective data fusion could be completed by variable-resolution Bayesian modeling. [31] proposed a new idea that Bayesian networks could be applied to intelligent environments (IEs) to complete information validation or information adaptation to the user. IEs are spaces that integrate information technologies, and they can combine the information technologies with user's requests to make the best decision [31]. Therefore, IEs need intelligent agents that are able to reason under uncertainties to complete the goals of the system. Opportunely, PGMs allow the intelligent agents. Moreover, they can make the decision-making process completed in an efficient and effective way. Hence, PGMs are applicable to intelligent environments. However, when dealing with massive hierarchical data across large datasets, Bayesian networks do not work very well. Thus, a scalable PGM was given in [29] as a solution for this kind of problem. Regardless, PGM still means a lot for mining and discovering insights of the data with various sizes, or even noisy data. Regarding record linkage, [32] combined the first-order logic with a PGM to solve the entity resolution problem. In this study, the Bayesian network (PGM) is used to construct an inference system for record linkage as it is able to automatically take the information available and make reasonable decisions under uncertainty. Furthermore, by using the Bayesian network, queries about variable distributions could be answered by several existing inference algorithms [33]. For entity resolution problem, the Bayesian network can look at all the information available, and decide which class does each record pair belong to, *match*, *potential match* or *non-match*. A typical Bayesian network consists of nodes and directed edges, where nodes denote variables and edges denote the relationships between variables. Like the ANN, the architecture of the Bayesian network is also decided by the numbers of selected data fields and comparison methods.

Typically, edges in Bayesian networks have no weights, so we are not able to tell the role of each selected field or each comparison method. In other words, it is not reasonable to assume that each field and each comparison method has the same weight or contributes to the final decision equally. Hence, we consider combining an ANN, where a gradient descent based algorithm is used, with a Bayesian network to construct a new record linkage model to solve the entity resolution problem. The ANN is used to determine the weights for the selected data fields and comparison methods since it is able to adjust the weights until a specific value of performance index is achieved. Next, the weights are imported into the Bayesian network to do inferences based on all the information available. In this study, the available information is the similarity scores for the selected data fields calculated by the selected comparison methods. We consider similarity scores/matching scores as probabilities of being matched for record pairs. Then the final matching scores are able to be determined by the Bayesian network in the form of probabilities. Afterward, the *three-way decision rules* are employed to quantify the quality of record linkage with the proposed model. To visualize the matching results directly, Neo4j [34], a powerful graph database, is used to construct a graph set to display the matching results. With Neo4j, queries about the matching results can be done quickly and easily.

During the literature review process, some similarities regarding the techniques prior to the classification step are found. From [12][35][36][37], the most common encoding method is Soundex, which is one of the phonetic encoding methods. With respect to indexing methods, the Standard Blocking is most widely used [17][36][38]. Moreover, Edit-distance, Jaro, and Winkler comparison methods are commonly used to link records [35][36][38][39][40].

3. Background Knowledge

In this chapter, we present the methods and models used to accomplish the record linkage task. We start by presenting the records indexing method. Also, we describe the comparison methods used to quantify the level of similarity among record pairs in the proposed model. Moreover, the ANN used to determine the weights is presented. In the end of this chapter, we describe the PGM used for calculating the matching probabilities among record pairs to decide if these can be linked or not.

The record linkage process mainly contains seven steps, as shown in Figure 3.1. The first step is to generate synthetic data with noise by a modified script in the Freely Extensible Biomedical Record Linkage (FEBRL) tool [41][42]. With the FEBRL tool, the noise can be added by selecting the maximum number of duplicates per record, the maximum number of modifications per field, and the maximum number of modifications per record. Furthermore, four types of errors are included in noise, typographical errors, phonetical errors, errors from Optical Character Recognition (OCR), and deletion of fields. After synthetic data is generated, the step of selecting identify fields starts. The identify fields can be date of birth (DOB), gender, social security ID (SSI), given name, surname, address, suburb, state, or phone number, and they will be compared in the following steps. The third step is records indexing. In this step, the number of record pairs that need to be compared is reduced. With records indexing, we can avoid comparing all the records from one database with all the records coming from another one. After indexing, the remaining record pairs (also known as 'candidate record pairs') may refer to the same entities.

In the fourth step, each identity field for the candidate record pairs is compared by the selected comparison methods with the FEBRL tool. In the fifth step, the comparison results (similarity scores) are imported into the ANN to optimize the weights of selected identify fields and comparison methods. In this step, the candidate record pairs are split into a training set and a testing set. The training set is used to train the ANN, and the testing set is used to evaluate the performance of the ANN, viz. the quality of the optimized weights.

Next, in the sixth step, a Bayesian network, which is an example of PGM [2][43][44], is used to determine the matching score for each candidate record pair. The weights obtained from the ANN are normalized beforehand. Then the normalized weights are imported into the Bayesian network to indicate which fields or comparison methods are more relevant. Besides, the similarity scores coming from the fourth step are regarded as the input for the Bayesian network to determine

the final matching scores based on the Bayesian probability theory [2][44]. As the synthetic data was generated with marking IDs, we have known which record pairs are matching or non-matching at the beginning of the record linkage process. After the comparisons of the candidate record pairs are completed, the performance of the proposed record linkage model can be well assessed in the seventh step. We classify the candidate record pairs into three classes: match, potential match, and non-match, and count the numbers of these three groups separately. The records in each matching record pair can be linked together. Moreover, the numbers of false positives, false negatives, and potential matches are also counted. At last, a deep analysis for the misclassified record pairs is provided.

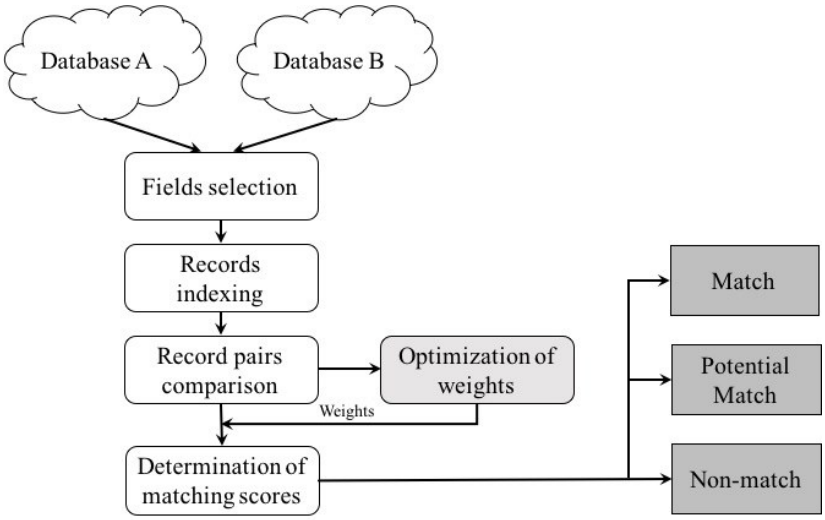

Figure 3.1. Process of Entity Resolution

Additionally, a powerful graph database named Neo4j [34] is employed to visualize the matching results. The information of records and the matching scores are imported into this tool. Nodes and directed edges can be added by the artificial control to represent the relationships among the records. Furthermore, queries regarding the constructed graph set can be done in a convenient and fast manner, and the query results can be shown in the form of tables or graphs.

3.1. Records Indexing

A detailed comparison of records, viz. comparing all records from one dataset with all records coming from another one, is a computationally expensive task. It also makes the comparison step

become the most expensive one during the process of record linkage. If two datasets contain *n* and *m* records, respectively, there will be $n \times m$ record pair comparisons at most. It is not practical to give so many comparisons, especially for large datasets. Hence, the records indexing step is an indispensable part of record linkage.

Indexing the records will reduce the number of record pairs to be compared in the following step by removing the record pairs that are not likely to refer to the same entity. In this way, the record pairs that are likely to refer to the same entity are retained for the comparison step. Two indexing techniques are commonly used in record linkage, viz. sorting data by gathering similar records and inserting each record into one or several blocks by some criteria [9]. The criteria are also known as *blocking keys*, whose values are determined by one field or several fields' content. For example, the blocking keys can be generated by sorting field *name* or by sorting the combination of fields *name* and *DOB*. The more specific a blocking key' definition is, the smaller number of blocks will be obtained. As a result, fewer record pair comparisons will be attained. Thus, concatenating two or more fields when generating blocking keys can reduce the number of candidate record pairs, viz. record pair comparisons. In this study, three indexes are defined for generating indexing keys (blocking keys): the first index key is generated by concatenating the values from fields *DOB*, *SSI*, and *phone numbe*r; the second index key is created by concatenating the values from fields *given name* and *surname*; the third index key is generated by the combining the values from fields *address* 1, *suburb* and *state*. *Given name* and *surname* are combined together since they both refer to personal names and have similar characteristics. For *address* 1, *suburb* and *state*, these three fields are all used to identify where a person lives. *DOB*, *SSI* and *phone number* are the fields that only contain numbers. Hence, they are combined to generate the third block key.

In this study, the encoding method used for defining blocking keys is known as Soundex [9], which is one of the most widely used phonetic encoding algorithms. This method aims to find out the level of phonetic similarity between two strings. It works as follows: the first letter of the string is kept, and the remaining letters are converted into three numbers in accordance with a transformation table. Based on the transformation table, 'h', 'w', 'y' and all vowels in the remaining letters of the string are converted into zeros. And the zeros should be deleted due to the encoding rule. When an encoded string contains repetitive numbers, the repetitive numbers should also be removed. In the end, if the number part of the encoded string contains more than three

digits, it is truncated to three digits only. If less than three digits exist in the number part of the encoded string, the number part will be extended with zeros to make sure its length is three. For example, the name '*Joseph*' will be encoded to '*J02010*' first; then zeros are removed from the code, and the new code turns to be '*J21*'. Since the number part only has two digits, a zero is added, and the final encoded string is '*J210*'.

Furthermore, Standard Blocking [9] technique is used to generate index blocks in this study. This approach aims to insert each record's blocking keys into a single block. For a single blocking key, only one blocking key value (BKV) will be created for each record. This BKV will determine into which block will the corresponding record be inserted. In this way, all records that have the same BKV and come from different datasets are considered as candidate record pairs, and these candidate record pairs will be compared in the next step. Moreover, for the records that have the same BKV but come from the same dataset, they are not candidate record pairs since there are no corresponding records in other datasets. In a word, the records indexing step reduces the number of record pair comparisons and leaves similar records to be compared in the next step. It accelerates the progress of record linkage.

3.2. Comparison Methods

Errors are commonly seen in real-world data due to phonetic or typographical variations. Besides, phone numbers and addresses always change over time. Surnames can also be changed when people get married. Additionally, nicknames, spelling variation, typing errors, Optical Character Recognition (OCR) errors, poor tone quality on the phone, and many other cases can also result in variations of names. In order to have a high-quality record linkage result, many researchers try to standardize the datasets to improve the data's quality. They focus on converting the attributes to the same form. However, this kind of technique cannot work very well to generate completely accurate results, and may leave several variations even in the standardized data. Hence, it is impossible to use a single function to find the exact matches among the data. Several comparison functions should be used to calculate the level of similarity between attributes. Comparison functions will generate similarity scores, which are always within [0, 1], to describe how similar two attributes are. A score close to 1 signifies that the two attributes are very similar, and a score close to 0 indicates that the two attributes are quite different.

In this study, ten comparison methods (functions) are selected to calculate the similarity scores among records. They are Syllable Alignment Distance, Bigram, Positional Bigram, Trigram, Longest Common Substring 2, Longest Common Substring 3, Smith-Waterman Edit Distance, Levenshtein Distance, Damerau-Levenshtein Distance, and Exact String comparison method.

The exact string comparison method is the simplest one among all comparison functions. It just compares two input strings directly, and computes an exact similarity score. If the two input strings are exactly the same, the similarity score will be 1. If two strings are totally different, the score will be 0. This computation method can be represented using the following expression, where s_1 and s_2 are two input strings.

$$\mathrm{sim}_{\mathrm{exact}}(s_1, s_2) = \begin{cases} 1 & \text{if } s_1 = s_2 \\ 0 & \text{if } s_1 \neq s_2 \end{cases} \tag{3.1}$$

The most widely used comparison methods in literature are those based on Edit Distance. Edit Distance method counts the smallest number of edit operations that are needed when converting a string to another one. For example, the edit distance of string '*Chris*' and string '*Kris*' is 2 as '*C*' should be replaced by '*K*' first and then '*h*' should be eliminated. Since two edits are needed at least here, the edit distance is 2. It is obvious that the more similar two strings are, the smaller will the edit distance be. As a result, the similarity score will be small. In this study, three edit distance based comparison methods are used, viz. Smith-Waterman Edit Distance, Levenshtein Distance, and Damerau-Levenshtein Distance [9].

The Levenshtein Distance method is the most basic one among these three edit-distance based methods. In this method, three basic edit operations are applicable when converting a string to another, viz. substitutions, deletions, and intersections. After the edit distance is obtained, the similarity score can be computed using the following formula,

$$\mathrm{sim}_{\mathrm{levenshtein}}(s_1, s_2) = 1.0 - \frac{\mathrm{dist}_{\mathrm{levenshtein}}(s_1, s_2)}{\max(|s_1|, |s_2|)} \tag{3.2}$$

where s_1 and s_2 represent two input strings, and $|\cdot|$ denotes the length of the string. $\mathrm{dist}_{\mathrm{levenshtein}}(s_1, s_2)$ represents the edit distance between string s_1 and string s_2. In the previous example, if we set '*Chris*' as s_1, '*Kris*' as s_2, we have $\mathrm{dist}_{\mathrm{levenshtein}}(s_1, s_2) = 2$, $|s_1| = 5$, $|s_2| = 4$, and the similarity score is 0.6. Hence, we can say that these two strings are relatively similar.

For the Damerau-Levenshtein Distance method, a fourth edit operation is added for computing the edit distance, viz. transposition (swapping) of two adjacent characters. Similarity

scores are computed by the same way as the Levenshtein Distance method. Generally, the Damerau-Levenshtein distance is smaller or equal to Levenshtein distance for the same string pair. Therefore, for the same two input strings, the similarity score from Damerau-Levenshtein Distance method is larger or equal to the score from Levenshtein Distance method. For example, the similarity scores based on Damerau-Levenshtein Distance and Levenshtein Distance for strings '*Cris*' and '*Cirs*' are 0.75 and 0.5, respectively. Obviously, the first method gives a better description of the similarity level between these two strings. Hence, the Damerau-Levenshtein Distance method is more accurate than the Levenshtein Distance method when the input strings contain transposition errors.

The Smith-Waterman Edit Distance method allows for gaps and character specific match scores or costs. It has five basic operations, and each operation is assigned to a different match score *ms*. But there are four different match scores in total, viz. 5, 2, -1, -5, as two operations are assigned to the same match score. The similarity score is computed as follows,

$$\mathrm{sim}_{\mathrm{smith\_waterman}}(s_1, s_2) = \frac{bs_{\mathrm{smith\_waterman}}}{\mathrm{div}_{\mathrm{smith\_waterman}} \times ms_m} \tag{3.3}$$

where ms_m is the match score when two characters are exactly matching; $bs_{\mathrm{smith\_waterman}}$ is the highest value of needed operations; $\mathrm{div}_{\mathrm{smith\_waterman}}$ is a factor. $\mathrm{div}_{\mathrm{smith\_waterman}}$ can be calculated in the following three ways:

1) $\mathrm{div}_{\mathrm{smith\_waterman}} = \min(|s_1|, |s_2|)$
2) $\mathrm{div}_{\mathrm{smith\_waterman}} = \max(|s_1|, |s_2|)$
3) $\mathrm{div}_{\mathrm{smith\_waterman}} = \frac{|s_1| + |s_2|}{2}$

where the first and third factors correspond to the overlap coefficient and Dice coefficient, respectively.

These three edit-distance based comparison methods are suitable for both name and non-name comparisons. Nevertheless, this kind of method has a high computational complexity.

Bigram, Trigram, and Positional Bigram methods are all based on *q*-grams [9]. The *q*-grams technique works as follows, first splitting two input strings into short substrings with length *q* by using a sliding window approach, and then counting the number of the *q*-grams (length *q* characters) that exist in both two input strings. The most commonly used values for *q* in record

linkage are 2 and 3, which corresponds to Bigram method and Trigram method, respectively. The normalized similarity score can be determined by the following expressions,

$$\mathrm{sim}_{\mathrm{overlap}}(s_1, s_2) = \frac{c_{\mathrm{common}}}{\min(c_1, c_2)} \tag{3.4}$$

$$\mathrm{sim}_{\mathrm{jaccard}}(s_1, s_2) = \frac{c_{\mathrm{common}}}{c_1 + c_2 - c_{\mathrm{common}}} \tag{3.5}$$

$$\mathrm{sim}_{\mathrm{dice}}(s_1, s_2) = \frac{2 \times c_{\mathrm{common}}}{c_1 + c_2} \tag{3.6}$$

where c_{common} is the number of the q-grams that exist in both two input strings; c_1 and c_2 are respectively the numbers of q-grams in input strings s_1 and s_2. Obviously, these three expressions correspond to overlap coefficient, Jaccard coefficient and Dice coefficient, respectively. And the expression we use depends on which coefficient is selected. For example, for two input strings '*chris*' and '*kris*', the 2-grams of them are '*ch*', '*hr*', '*ri*', '*is*', and '*kr*', '*ri*', '*is*', respectively. Then we have $c_1 = 4$, $c_2 = 3$, and $c_{\mathrm{common}} = 2$. The similarity scores corresponding to overlap, Jaccard, and Dice coefficients are 0.67, 0.4, and 0.57. If 3-grams is used to calculate the similarity score of these two strings, their substrings will be '*chr*', '*hri*', '*ris*', and '*kri*', '*ris*'. We will have $c_1 = 3$, $c_2 = 2$, and $c_{\mathrm{common}} = 1$. The three similarity scores will be 0.5, 0.25, and 0.4.

Compared with Bigram method, the Positional Bigram method takes into account the positional information. In other words, each q-gram has a number indicating its position in the string. In this method, only the common q-grams whose position values are within a certain maximum distance can be used to determine c_{common}. Then (3.4), (3.5) and (3.6) can be used to calculate the similarity score. Compared with edit-distance based methods, q-grams based comparison methods have smaller complexity. Hence, this kind of method has a high efficiency and is quite suitable for comparing long strings. When comparing Trigram with Bigram, it can be found that Trigram is more sensitive to single character differences.

In Longest Common Substring comparison method, the objective is to iteratively remove the longest substrings s_c that two input strings have in common. This process is kept running until the length of the common substring is smaller than a certain number, $l_{\min}$. $l_{\min}$, which is also called the minimum length of characters, is always set as 2 or 3. Then the lengths of all found common substrings are added together, and the sum is used to compute the similarity score. The expressions are shown as follows, which are similar to the expressions of q-grams based methods, viz. (3.4), (3.5) and (3.6),

$$\text{sim}_{\text{lcs\_overlap}}(s_1, s_2) = \frac{l_c}{\min(|s_1|, |s_2|)} \tag{3.7}$$

$$\text{sim}_{\text{lcs\_jaccard}}(s_1, s_2) = \frac{l_c}{|s_1| + |s_2| - |l_c|} \tag{3.8}$$

$$\text{sim}_{\text{lcs\_dice}}(s_1, s_2) = \frac{2 \times l_c}{|s_1| + |s_2|} \tag{3.9}$$

where $l_c = \sum_{i=1}^{n} |s_{c_i}|$ denotes the sum of the lengths for all common substrings, and n is the number of the common substrings. Similarly, the above three expressions correspond to three conditions when overlap coefficient, Jaccard coefficient, or Dice coefficient is applied. This comparison function performs well on the strings that contain some characters not in the same order. For example, strings '*cris johnson*' and '*johnson cris*' have common substrings $s_{c_1} = '\textit{johnson}'$ and $s_{c_2} = '\textit{cris}'$. Thus, we have $l_c = 11$, $|s_1| = |s_2| = 12$, and the similarity scores corresponding to the above three coefficients are 0.917, 0.846 and 0.917.

The Syllable Alignment Distance method uses the Syllable Alignment Pattern Searching technique to compute similarity scores. This technique combines phonetic information with edit-distance based computation [9]. The idea behind this technique is to convert two input strings into sequences of syllables, and then count the number of edits that are needed when converting one sequence of syllables to the other one. Seven edit operations are included in this method, and they are assigned to different edit scores, which is similar to Smith-Waterman Edit Distance method. The edit scores in this method range between -4 to 6. After the distance is obtained, the similarity score can be computed in three ways. And the expressions are identical to those in the Smith-Waterman Edit Distance method. As phonetic information is contained in this method, it is usually used to compare names, where phonetic variations often occur.

3.3. Artificial Neural Network Model

Now, let us make a deep insight on how to construct the ANN model in the fifth step. The structure of the ANN is shown in Figure 3.2. The experimental data used for constructing this model is a set of input-output pairs $(u_{ij}(k), t(k))$, where u_{ij} represents the matching result for a candidate record pair, namely the similarity score for the ith field with the jth comparison method, and t is the target output for the record pair. The value of t is determined when generating the data since labels have been added to show whether the record pair is matched or not. Specifically speaking, if the

record pair is matched in practice, its target output will be 1. On the contrary, the target output of a non-matching record pair is 0. Furthermore, k=1, 2, …, n; j=1, 2, …, c; i=1, 2, …, m, where n is the number of record pairs that could hypothetically be matched (candidate record pairs); c is the number of selected comparison methods (functions), and m is the number of selected data fields. To be specific, $\boldsymbol{u}(k)\in[0, 1]^{mc}$, $t(k)\in[0, 1]$. Therefore, mc is the size of each vector $\boldsymbol{u}$. In the structure shown in Figure 3.2., we assume that field 1 is *given name*; field 2 is *surname*; …; field m is *date of birth* (*DOB*). Each field is compared by c comparison methods (functions).

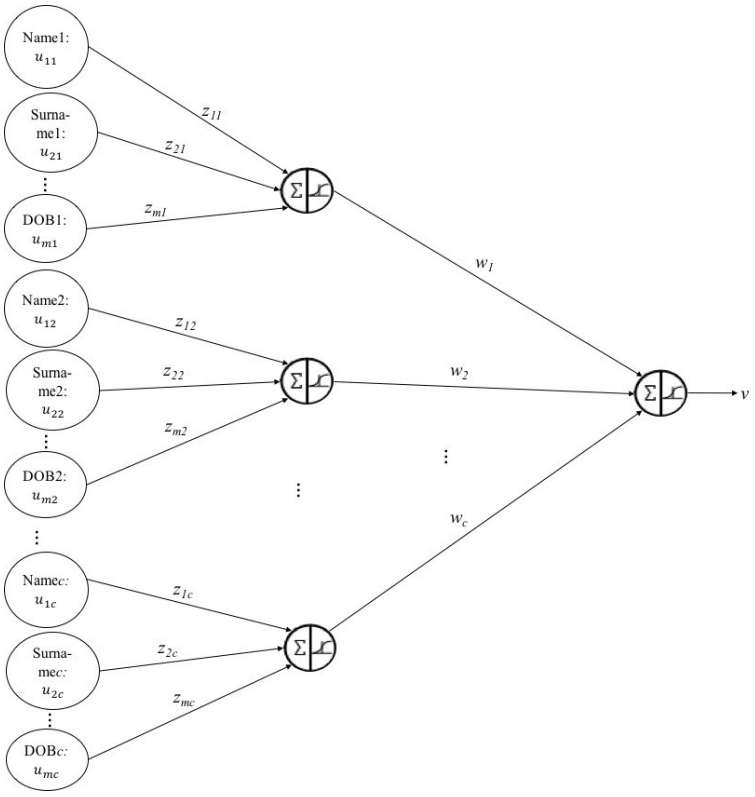

Figure 3.2. Structure of the Artificial Neural Network

The records after indexed are used to calculate the similarity score for each data field with each comparison method. The values of similarity scores are all within [0, 1], where 0 indicates that the field is totally non-matched, and 1 corresponds to a total matching field. Then the scores are imported into the ANN as input data. The input data is split into a training set and a testing set first. In this model, the jth hidden neuron is connected to the fields where the jth comparison method is used. In other words, it is not connected to (j+1)th, (j+2)th, …, (j+c-1)th comparison methods. The output of the neuron in the hidden layer, y_j, is expressed as follows,

$$p_j = \sum_{i=1}^{m} u_{ij} z_{ij}, \tag{3.10}$$

$$y_j = f(p_j) \tag{3.11}$$

where f is a monotonically increasing function, whose value is within (0,1). $\boldsymbol{z} = [z_{11}, z_{21}, \ldots, z_{m1}, z_{12}, z_{22}, \ldots, z_{m2}, \ldots, z_{1c}, z_{2c}, \ldots, z_{mc}]$ is a vector of weights, and the initial values of these weights are all within [-1, 1]. z_{ij} represents the relevance of the ith field with the jth comparison function. The output of the model v, namely the output of the single neuron in the output layer, is,

$$l = \sum_{j=1}^{c} y_j w_j, \tag{3.12}$$

$$v = f(l) \tag{3.13}$$

where $\boldsymbol{w} = [w_1, w_2, \ldots, w_c]$ is also a vector of weights with initial values within [-1, 1]. w_j links the jth hidden neuron with the single output neuron, and it denotes the significance of the jth comparison method. In this study, we select the sigmoidal function as the activation function that can be used in (3.11) and (3.13). This function is expressed as follows,

$$f(x) = \frac{1}{1 + e^{(-x)}} \tag{3.14}$$

With the ANN, several similarity scores are aggregated into a single output. This makes a lot of superiority in the research. First, the weights are able to be adjusted to make matching record pairs apart from those non-matching ones as far as possible. Moreover, by looking at the values of the optimized weights, we could know which comparison methods and which data fields contribute more to a correct record linkage. After the optimized weights are obtained, normalization is employed to make these weights within [0, 1]. When a weight's value is 1, it signifies that the specific data field or comparison method contributes the most to record linkage. A weight with the value 0 means the corresponding data field or comparison method has no impact on record linkage. Then the data fields and comparison methods that have small weights can be eliminated, as they make little contribution to the record linkage.

3.4. Probabilistic Graphical Model

Among various types of PGMs, the Bayesian network is selected to calculate matching scores among candidate record pairs. The structure of the Bayesian network is shown in Figure 3.3, where μ_{ij} is the weight that expresses the significance of the *i*th field with the *j*th comparison method, and λ_j is the weight representing the relevance of the *j*th comparison method. Moreover, i=1, 2, …, m, j=1, 2, …, c, where m is still the number of data fields, and c is the number of the comparison methods. μ_{ij} is the normalized form of z_{ij}, where z_{ij} is the weight obtained from the ANN. Similarly, λ_j is also computed by normalizing w_j. The initial values of weights $\boldsymbol{z}$ and $\boldsymbol{w}$ are within [-1,1], where 1 represents a largest positive impact, and -1 represents a largest negative impact. After training the ANN, the values of the optimized weights are still within [-1, 1]. However, these optimized weights cannot be applied to the Bayesian network directly since they may make some probabilities (matching scores of fields or comparison methods) larger than 1 or smaller than 0. For example, assume that the similarity scores of a record pair on fields *name*, *surname* and *DOB* are 0.7, 0.8 and 0.6, and the corresponding weights obtained from the ANN are 0.8, 0.7 and -0.1. Then the probability of being a match under the condition of matching name, matching surname and matching DOB is 0.7×0.8+0.8×0.7+0.6×(-0.1)=1.06, which is beyond the range [0, 1] and does not meet the requirements of the probability theory. Therefore, a normalization of the weights should be done as follows,

$$\delta_{ij} = \frac{z_{ij} - z_{min}}{z_{max} - z_{min}} \tag{3.15}$$

$$\mu_{ij} = \frac{\delta_{ij}}{\sum_{i=1}^{m} \delta_{ij}} \tag{3.16}$$

$$\varepsilon_j = \frac{w_j - w_{min}}{w_{max} - w_{min}} \tag{3.17}$$

$$\lambda_j = \frac{\varepsilon_j}{\sum_{j=1}^{c} \varepsilon_j} \tag{3.18}$$

where z_{max} and z_{min} represent the maximum and minimum values of weights z_{ij}, and w_{max} and w_{min} are the maximum and minimum values of weights w_j. First, z_{ij} and w_j are respectively scaled to [0, 1], and the scaled weights are denoted by δ_{ij} and ε_j, where i=1, 2, …, m, j=1, 2, …, c. Then the normalization is done by dividing scaled weight δ_{ij} by the sum of the scaled weights that

are related to the *j*th comparison method. In this way, the sum of the normalized weights that correspond to the *j*th comparison method will be 1, namely $\sum_{i=1}^{m}\mu_{ij}=1$, and $\mu_{ij}\in[0, 1]$. Similarly, normalized weight λ_j is obtained by dividing scaled weight ε_j by the sum of all comparison methods' scaled weights. Then the sum of all comparison methods' normalized weights is 1, namely $\sum_{j=1}^{c}\lambda_j=1$, and λ_j is within [0, 1]. Considering the previous example again, after the normalization, the weights of fields *name*, *surname* and *DOB* are 0.53, 0.47 and 0. The probability of being a match under the same condition is 0.7×0.53+0.8×0.47+0.6×0=0.747, which is within [0, 1] and meets the requirements of the probability theory.

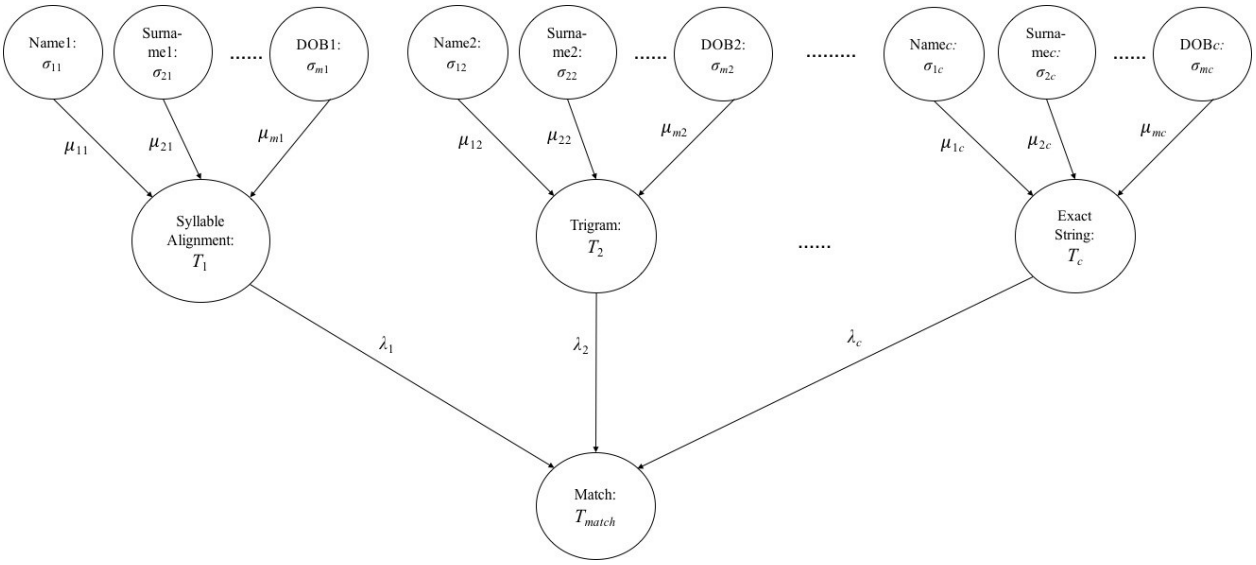

Figure 3.3. Structure of the Bayesian Network

Bayesian networks mainly consist of nodes and directed edges. In this proposed model, there are *mc* nodes in the first layer, *c* nodes in the second layer, and a single node in the third layer. Nodes are always labeled with variables in Bayesian networks. For example, *name*, *surname*, *DOB* could be variables' names, and they can be showed inside the nodes. Here, we name the node (variable) in the last layer '*Match*'. The variable's name of each node in the second layer is actually the name of the corresponding comparison method. The nodes located at the end points of arrows are called children nodes. Then the parent nodes refer to the nodes that are located at the start points of arrows. Bayesian networks can be parametrized with conditional probabilistic distributions (CPDs). For each variable (node), there is a CPD associated with it. Given parent

nodes' CPDs, children nodes' probability distributions can be calculated based on the probability theory. The determination of each variable's CPD is an important task. The details on how to determine the CPDs will be discussed in Chapter 4. Here, we describe how to use the Bayesian network to calculate matching scores.

Probability theory is a tool to study uncertainty (randomness) which is a native nature of the real world [44]. The probability theory is constructed with random variables. Random variables can be continuous or discrete, and they may have a set of possible outcomes. For each possible outcome for random variable x, a value is assigned to it to represent the probability of occurrence, which is always denoted by $P(x)$. In fact, we often consider several variables at the same time, and the probability distribution over multiple variables is known as the joint probability distribution [44]. Some variables are independent of each other, and some variables are dependent on others. For the independent variables, the joint probability distribution can be expressed as,

$$P(x_1, x_2, \ldots, x_e) = P(x_1)\, P(x_2) \cdots P(x_e) \tag{3.19}$$

where $x_1, x_2, ..., x_e$ are e independent random variables. We can also say that this distribution P satisfies $(x_i \perp x_j)$ for any i, j that $i \neq j$, where '$\perp$' means the corresponding two variables are independent of each other. For example, in our study, we have $P(name_1, surname_1, \ldots, DOB_1) = P(name_1)\, P(surname_1) \cdots P(DOB_1)$ since these variables are independent of each other. Sometimes the probability distribution of a variable depends on other variables. For this kind of situation, CPDs are suitable to describe the probabilistic relation among the variables. The CPD for two variables can be expressed as $P(x_1 \mid x_2)$, which represents the probability distribution of x_1 with a given outcome of x_2. Likewise, $P(x_2 \mid x_1)$ is the probability distribution of x_2 given the status of x_1.

Similarly, in the Bayesian network, some variables (nodes) are independent, and some are dependent. The probability distributions of children nodes usually depend on their parent nodes' probabilities. The parent nodes for variable x_i are represented as $Pa_{x_i}^{\mathcal{G}}$, where $\mathcal{G}$ denotes the Bayesian network structure [2]. For the nodes that are not the descendants of x_i, they are expressed as $\text{NonDescendants}_{x_i}$. Then, for variable x_i, the relation among these three sets of nodes are expressed as follows,

$$(x_i \perp \text{NonDescendants}_{x_i} \mid Pa_{x_i}^{\mathcal{G}}) \tag{3.20}$$

This expression is also known as local independencies [2]. Assuming that $\mathcal{G}$ contains e variables $x_1, x_2, ..., x_e$, then the probability distribution over these variables can be expressed as,

$$P(x_1, x_2, \dots, x_e) = \prod_{i=1}^{e} P(x_i \mid Pa_{x_i}^{\mathcal{G}}) \tag{3.21}$$

where $P(x_i \mid Pa_{x_i}^{\mathcal{G}})$, i=1, 2, …, e, are the CPDs. This formula is also known as the chain rule for Bayesian networks [2].

After the determination of the CPDs for all variables (nodes), the objective is to perform an exact inference to calculate the final matching score for each candidate record pair. In other words, the aim is to obtain the individual probability distribution of variable *Match*. To complete an exact inference over a specific variable, an algorithm named variable elimination [2] is used in the proposed model. The fundamental idea of this algorithm is to keep summing the joint probabilities over one variable at a time until all variables' calculations are finished. However, the joint probabilities are usually unknown. As conditional probabilities are always available, they are used to compute the joint probabilities beforehand. For example, assume that there is a simple Bayesian network: $a \rightarrow b \rightarrow d$, where a, b and d are three variables, and our goal is to acquire the probability distribution of variable d, viz. $P(d)$, given the CPDs of all variables. By using basic probabilistic reasoning, we have the following expressions,

$$P(b) = \sum_{a} P(a)\, P(b \mid a) \tag{3.22}$$

$$P(d) = \sum_{b} \mathrm{P}(b)\, P(d \mid b) \tag{3.23}$$

where $P(a)$, $P(b \mid a)$, and $P(d \mid b)$ are already known. Using the chain rule in Bayesian networks, the joint probabilities are given as,

$$P(a, b, d) = P(a)\, P(b \mid a)\, P(d \mid b) \tag{3.24}$$

After inserting (3.22) and (3.24) into (3.23), we get,

$$
\begin{aligned}
P(d) &= \sum_{b} P(d \mid b) \sum_{a} P(a)\, P(b \mid a) \\
&= \sum_{b} \sum_{a} P(a)\, P(b \mid a)\; P(d \mid b) \\
&= \sum_{b} \sum_{a} P(a, b, d)
\end{aligned}
\tag{3.25}
$$

From (3.25), we can see that this algorithm is a procedure of sum-product variable elimination [2]. Next, a general expression for computing variables' probability distributions is shown as follows, which is also well discussed in [2]. Let $\boldsymbol{x}$ be a set of variables, namely $\boldsymbol{x}$ contains variables x_1, x_2, ..., x_e. $q \notin \boldsymbol{x}$ is a query variable, and $\boldsymbol{\rho}$ is a set of variables that $\boldsymbol{\rho} = \boldsymbol{x}+q$. $\phi(\boldsymbol{x}, q)$ is a factor, and Φ is a set of factors where $\phi \in \Phi$. Then the probability distribution $P(q)$ given by the sum-product variable elimination algorithm is,

$$
P(q) = \sum_{\boldsymbol{\rho}} \prod_{\phi \in \Phi} \phi \tag{3.26}
$$

where $\Phi = \left\{ \phi_{x_i} \right\}_{i=1}^{e}$, $\phi_{x_i} = P(x_i \mid Pa_{x_i}^{\mathcal{G}})$, and $\mathcal{G}$ is the Bayesian network.

Combining the background of the proposed Bayesian model, the probability distributions for the nodes are determined as follows. In this structure, as shown in Figure 3.3., there are c variables (nodes) in the second layer (corresponding to c comparison methods), and each one of these nodes is connected to m parent nodes (corresponding to m data fields). As the m parent nodes are independent of each other, the probability of being matched or being a non-match for a '*comparison method*' node (variable) is expressed as follows,

$$
P(method\ j) = \sum_{field\ 1j,\, field\ 2j, \ldots,\, field\ mj} P(field\ 1j, field\ 2j, \ldots, field\ mj, method\ j) \tag{3.27}
$$

$$
\begin{aligned}
&P(field\ 1j, field\ 2j, \ldots, field\ mj, method\ j) = \\
&P(method\ j \mid field\ 1j, field\ 2j, \ldots, field\ mj) \cdot P(field\ 1j) \cdot P(field\ 2j) \cdot \cdots \cdot P(field\ mj)
\end{aligned}
\tag{3.28}
$$

where $P(method\ j)$ denotes the probability distribution of the jth node in the second layer, namely the variable named the jth comparison method, and $P(field\ ij)$ is the probability distribution of the ith data field that is connected to the jth node in the second layer. $P(method\ j \mid field\ 1j, field\ 2j, \ldots, field\ mj)$ is the CPD of the jth node in the second layer (the

variable named the *j*th comparison method) given the probabilities of *field* 1*j*, *field* 2*j*, ..., and *field mj* . $P(field\ 1j, field\ 2j, \ldots, field\ mj, method\ j)$ is the joint probability distribution. The probability distribution of the *j*th node (variable) in the second layer is obtained by summing over the joint probability distribution. Since each '*comparison method*' node (the node in the second layer of the Bayesian network) only has two statuses, being matched and not being matched, the probability distribution of each '*comparison method*' node contains two probabilities, namely probabilities of being matched and not being matched.

The probability distribution of the '*Match*' node is determined by the same way. Using variable elimination, its probability distribution is defined as follows,

$$P(match) = \sum_{method\ 1,\ method\ 2, \ldots,\ method\ c} P(method\ 1, method\ 2, \ldots, method\ c, match) \tag{3.29}$$

$$P(method\ 1, method\ 2, \ldots, method\ c, match) = P(match \mid method\ 1, method\ 2, \ldots, method\ c) \cdot P(method\ 1) \cdot P(method\ 2) \cdot \ldots \cdot P(method\ c) \tag{3.30}$$

where $P(match)$ is the probability distribution of the '*Match*' variable (node) for a candidate record pair, and $P(method\ j)$ denotes the probability distribution of the variable named the *j*th comparison method. $P(match \mid method\ 1, method\ 2, \ldots, method\ c)$ is the conditional probability distribution of the *Match* variable given the probabilities of *method* 1, *method* 2, ..., and *method c* . $P(method\ 1, method\ 2, \ldots, method\ c, match)$ is the joint probability distribution. Similarly, the probability distribution of the *Match* variable for a record pair is also obtained by summing over the joint distribution. The *Match* variable still only has two statuses. In other words, its probability distribution consists of two probabilities, being a match and being a non-match, and the sum of these two probabilities always equals to 1. The probability of being a match is certainly within [0, 1]. If the probability of being a match is close to 1, the corresponding record pair is most likely to be matched. In other words, the two records in this record pair refer to the same entity. When the probability of being a match is 0, it means that the two input records are totally not matched. Thus, they do not refer to the same entity in the real world, and we can also say that they refer to two different entities.

4. The Modeling of Record Linkage

In Chapter 3, the operating principles of the ANN model and the Bayesian network model are fully discussed. To illustrate the construction of the record linkage model further, several other modules are described in this chapter, including the optimization of weights, the determination of CPDs, and performance assessment.

4.1. Optimization of Weights

As described before, the weights in the Bayesian network model are determined by the ANN model. In the ANN, a supervised gradient descent algorithm [45] is employed to optimize the weights. This algorithm is to adjust the weights by using the back-propagation learning rule until the objective function Q reaches the minimum value. The objective function (performance index) of the ANN is,

$$Q = \sum_{k=1:t_k<\tau_1}^{n} v_k^2 + \sum_{k=1:t_k>\tau_2}^{n} (1 - v_k)^2 \tag{4.1}$$

where τ_1 and τ_2 ($\tau_1 < \tau_2$) are two thresholds; v_k, k=1, 2, …, n, is the output of the ANN and corresponds to the input $u_{ij}(k)$; n is the number of candidate record pairs. Given the target output $t(k)$, this algorithm will adjust the weights ($\boldsymbol{z}$ and $\boldsymbol{w}$) until the minimum value of Q is achieved. To be specific, if $t(k)$ is below τ_1, the error that will be back-propagated is defined as v_k^2. If $t(k)$ is above τ_2, the error will be defined as $(1- v_k)^2$. The target output $t(k)$ has two statuses, non-match, namely $t(k)$=0, and match, namely $t(k)$=1. The performance index Q will optimize the weights until the final output, v_k, is below τ_1 or above τ_2. Specifically, if $t(k)$=0, the corresponding v_k should be smaller than τ_1 in the end. On the contrary, the final value of v_k should be larger than τ_2 when the corresponding target output $t(k)$=1.

After the values of τ_1 and τ_2 are selected, minimizing Q is done by iteratively employing the gradient descent algorithm. The updates of the weights in each iteration is expressed as follows,

$$\boldsymbol{z}(iter+1) = \boldsymbol{z}(iter) - \alpha \nabla_{\boldsymbol{z}} Q \tag{4.2}$$

$$\boldsymbol{w}(iter+1) = \boldsymbol{w}(iter) - \alpha \nabla_{\boldsymbol{w}} Q \tag{4.3}$$

where α is the learning rate of the ANN and its value is within (0, 1). ∇Q is the gradient of the objective function Q, and *iter* denotes the number of iterations. The initial values of $\boldsymbol{z}$ and $\boldsymbol{w}$ are

within [-1, 1] and follow a normal distribution. Afterward, these weights are adjusted according to (4.2) and (4.3) until the minimum value of Q is achieved. To describe the process of calculating $\nabla_z Q$ and $\nabla_w Q$, we take a single input-output pair ($\boldsymbol{u}$, t) as an example. The calculation is based on the chain rule, and the partial derivatives of the performance index with respect to the weights are shown in the following expressions,

$$\frac{\partial Q}{\partial w_j} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial w_j} \tag{4.4}$$

$$\frac{\partial Q}{\partial z_{ij}} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial y_j}\frac{\partial y_j}{\partial p_j}\frac{\partial p_j}{\partial z_{ij}} \tag{4.5}$$

Next, each part of the above equations should be solved. The performance index Q has two parts, as shown in (4.1), so the derivative of Q with respect to the output v also contains two parts,

$$\frac{\partial Q}{\partial v} = 2v, \text{ if } t_k < \tau_1 \tag{4.6}$$

$$\frac{\partial Q}{\partial v} = 2(v\text{-}1), \text{ if } t_k > \tau_2 \tag{4.7}$$

Then the partial derivative of v with respect to the aggregation l is expressed as follows, which in essence is the derivate of the activated function shown in (3.14),

$$\frac{\partial v}{\partial l} = v(1 - v) \tag{4.8}$$

The aggregation l is defined as follows,

$$l = w_1 y_1 + w_2 y_2 + \ldots + w_j y_j + \ldots + w_c y_c \tag{4.9}$$

where y_j is the output of the jth hidden neuron, and w_j is the weight of the jth hidden neuron (the jth comparison method). For the partial derivatives of l with respect to w_j and y_j, the expressions are,

$$\frac{\partial l}{\partial w_j} = y_j \tag{4.10}$$

$$\frac{\partial l}{\partial y_j} = w_j \tag{4.11}$$

The partial derivative of y_j with respect to the aggregation p_j is similar to (4.8). In essence, it is also the derivative of the activated function shown in (3.14), and its expression is,

$$\frac{\partial y_j}{\partial p_j} = y_j(1 - y_j) \tag{4.12}$$

Finally, the partial derivative of the aggregation p_j with respect to the weight z_{ij} is calculated by the following two expressions,

$$p_j = z_{1j}u_{1j} + z_{2j}u_{2j} + \ldots + z_{ij}u_{ij} + \ldots z_{mj}u_{mj} \tag{4.13}$$

$$\frac{\partial p_j}{\partial z_{ij}} = u_{ij} \tag{4.14}$$

After combing (4.6), (4.7), (4.8), (4.10) and (4.6), (4.7), (4.8), (4.11), (4.12), (4.13), (4.14), respectively, $\nabla_z Q$ and $\nabla_w Q$ are expressed as follows,

$$\frac{\partial Q}{\partial w_j} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial w_j} = 2v^2(1 - v)\, y_j, \text{ if } (t_k < \tau_1) \tag{4.15}$$

$$\frac{\partial Q}{\partial w_j} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial w_j} = -2v(1 - v)^2\, y_j, \text{ if } (t_k > \tau_2) \tag{4.16}$$

$$\frac{\partial Q}{\partial z_{ij}} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial y_j}\frac{\partial y_j}{\partial p_j}\frac{\partial p_j}{\partial z_{ij}} = 2v^2(1 - v)w_j\, y_j(1 - y_j)u_{ij}, \text{ if } (t_k < \tau_1) \tag{4.17}$$

$$\frac{\partial Q}{\partial z_{ij}} = \frac{\partial Q}{\partial v}\frac{\partial v}{\partial l}\frac{\partial l}{\partial y_j}\frac{\partial y_j}{\partial p_j}\frac{\partial p_j}{\partial z_{ij}} = -2v(1 - v)^2 w_j\, y_j(1 - y_j)u_{ij}, \text{ if } (t_k > \tau_2) \tag{4.18}$$

Therefore, (4.2), (4.3), (4.15), (4.16), (4.17) and (4.18) are used to update the weights ($\boldsymbol{w}$ and $\boldsymbol{z}$). The above calculation process is iteratively applied until the performance index Q reaches its minimum value. At this point, $\boldsymbol{w}$ and $\boldsymbol{z}$ reach their optimized values. These optimized weights can be used to classify the candidate record pairs into *match* group or *non-match* group. Meanwhile, these optimized weights can minimize the numbers of false positives, false negatives, and clerical review records. Then we normalize these optimized weights to [0, 1], where a weight close to 1 signifies that the corresponding data field or comparison method has a huge impact on the record linkage. On the contrary, when the value of a weight is close to 0, the corresponding field or comparison method's contribution to the record linkage is minimal. The fields and comparison methods whose weights are close to 0 can be eliminated as they can hardly play their roles during the process of record linkage.

4.2. Determination of Conditional Probability Distributions

In Chapter 3, we described how to calculate the matching scores (probabilities of being matched) among record pairs based on the Bayesian network. Nevertheless, the prerequisite of the computation is that the CPDs for all variables are already known. Hence, in this chapter, the process of determining CPDs is fully discussed.

From the Bayesian network model shown in Figure 3.3., we can know that to obtain the probability distributions of the variables (nodes) in the second layer, the CPDs of the variables (nodes) in the first layer should be determined first. In the first layer, there are $m \times c$ nodes, where m is the number of fields and c is the number of comparison methods (functions). The second layer has c nodes, and each one of these nodes has m parent nodes. If we know the CPDs of the m parent nodes, the probability distribution of the children node can be determined by the Bayesian network. After a comparison method is selected, the similarity scores for all fields can be calculated. The CPDs of the variables (nodes) in the first layer are considered as follows,

$$\boldsymbol{\sigma}_{ij} = [u_{ij},\ 1\text{-}\ u_{ij}] \tag{4.19}$$

where u_{ij} is the similarity score for the ith field with the jth comparison method, and its value is within [0, 1]. When the similarity score is 0, the attributes of the corresponding field are totally non-matched. If the attributes are totally matched, the score will be 1. σ_{ij} is the CPD of the ith node (field) which is connected to the jth node in the second layer (also known as the jth comparison method), where i=1, 2, …, m, j=1, 2, …, c. m is the number of selected data fields, and c is the number of comparison methods. u_{ij} and 1- u_{ij} are also known as the probability of being matched and the probability of not being matched for the combination of the ith field and the jth comparison method, respectively.

As the CPDs of all the nodes (variables) in the first layer have two probabilities, the CPD of each node (variable) in the second layer has 2^{m+1} probabilities. Let f_{ij}^{0} and f_{ij}^{1} denote the probability of being matched and the probability of not being matched for the ith field which is connected to the jth comparison method. Then we have $f_{ij}^{0}=u_{ij}$, $f_{ij}^{1}=1\text{-}\ u_{ij}$. $\boldsymbol{g_j^0}$ and $\boldsymbol{g_j^1}$ are the column vectors that represent the conditional probabilities of being matched and being non-matched for the jth method (jth node in the second layer). Each of them has 2^m probabilities (elements). For each selected comparison method, the CPD can be determined by the following process,

for a in $[f_{1j}^0, f_{1j}^1]$*:*

  for b in $[f_{2j}^0, f_{2j}^1]$*:*

    .....

      for m in $[f_{mj}^0, f_{mj}^1]$*:*

$$g = a \cdot \mu_{1j} + b \cdot \mu_{2j} + \ldots + m \cdot \mu_{mj}$$

where μ_{ij} is the normalized weight for the *i*th field with the *j*th comparison method. The value of *g* obtained from each iteration comprises $\boldsymbol{g_j^0}$. Then $\boldsymbol{g_j^1}$ is considered as,

$$g_j^1(k) = 1 - g_j^0(k) \tag{4.20}$$

where k=1, 2, ..., 2^m; $g_j^0(k)$ represents the kth element of vector $\boldsymbol{g_j^0}$; $g_j^1(k)$ represents the kth element of vector $\boldsymbol{g_j^1}$. $g_j^0(k)$ is the weighted sum of f_{ij}^0 and/or f_{ij}^1 with weights μ_{ij}. The process of determining μ_{ij} has been fully discussed in Chapter 3.4 and Chapter 4.1. An example of determining CPD is shown in Table 4.1., where three fields are selected (m=3). The formulas for calculating each conditional probability are also shown in the following expressions,

Table 4.1. An Example of Determining CPD

Name		f_{1j}^0				f_{1j}^1			
Surname		f_{2j}^0		f_{2j}^1		f_{2j}^0		f_{2j}^1	
DOB		f_{3j}^0	f_{3j}^1	f_{3j}^0	f_{3j}^1	f_{3j}^0	f_{3j}^1	f_{3j}^0	f_{3j}^1
*j*th Comparison Method	$\boldsymbol{g_j^0}$	$g_j^0(1)$	$g_j^0(2)$	$g_j^0(3)$	$g_j^0(4)$	$g_j^0(5)$	$g_j^0(6)$	$g_j^0(7)$	$g_j^0(8)$
	$\boldsymbol{g_j^1}$	$g_j^1(1)$	$g_j^1(2)$	$g_j^1(3)$	$g_j^1(4)$	$g_j^1(5)$	$g_j^1(6)$	$g_j^1(7)$	$g_j^1(8)$

$$g_j^0(1) = \mu_{1j} f_{1j}^0 + \mu_{2j} f_{2j}^0 + \mu_{3j} f_{3j}^0,\ g_j^1(1) = 1 - g_j^0(1) \tag{4.21}$$

$$g_j^0(2) = \mu_{1j} f_{1j}^0 + \mu_{2j} f_{2j}^0 + \mu_{3j} f_{3j}^1,\ g_j^1(2) = 1 - g_j^0(2) \tag{4.22}$$

$$g_j^0(3) = \mu_{1j} f_{1j}^0 + \mu_{2j} f_{2j}^1 + \mu_{3j} f_{3j}^0,\ g_j^1(3) = 1 - g_j^0(3) \tag{4.23}$$

$$g_j^0(4) = \mu_{1j} f_{1j}^0 + \mu_{2j} f_{2j}^1 + \mu_{3j} f_{3j}^1,\ g_j^1(4) = 1 - g_j^0(4) \tag{4.24}$$

$$g_j^0(5) = \mu_{1j} f_{1j}^1 + \mu_{2j} f_{2j}^0 + \mu_{3j} f_{3j}^0,\ g_j^1(5) = 1 - g_j^0(5) \tag{4.25}$$

$$g_j^0(6) = \mu_{1j} f_{1j}^1 + \mu_{2j} f_{2j}^0 + \mu_{3j} f_{3j}^1,\ g_j^1(6) = 1 - g_j^0(6) \tag{4.26}$$

$$g_j^0(7) = \mu_{1j} f_{1j}^1 + \mu_{2j} f_{2j}^1 + \mu_{3j} f_{3j}^0,\ g_j^1(7) = 1 - g_j^0(7) \tag{4.27}$$

$$g_j^0(8) = \mu_{1j} f_{1j}^1 + \mu_{2j} f_{2j}^1 + \mu_{3j} f_{3j}^1,\ g_j^1(8) = 1 - g_j^0(8) \tag{4.28}$$

To explain the meaning of $g_j^0(*)$, we take $g_j^0(3)$ as an example. $g_j^0(3)$ denotes the probability of being matched with the jth comparison method under the condition of matching name, non-matching surname, and matching DOB. Then the CPDs of the nodes (variables) in the second layer can be expressed as,

$$T_j = [\boldsymbol{g_j^0} \quad \boldsymbol{g_j^1}]^T \tag{4.29}$$

After the CPDs of the nodes in the second layer are obtained, variable elimination [44] is used to compute the probability distributions of these nodes, and these probability distributions will be used for computing the CPD of their children node. The variable elimination algorithm has already been described in Chapter 3. Now the problem is how to determine the CPD of the '*Match*' node. The computation method is similar to the one for computing the CPDs of the nodes in the second layer. Assume that $\boldsymbol{r^0}$ and $\boldsymbol{r^1}$ are two column vectors representing the conditional possibilities of being matched and being non-matched for '*Match*' node, then the CPD of the '*Match*' node can be determined by the following *for* loops,

for a in $[h_1^0, h_1^1]$*:*
  for b in $[h_2^0, h_2^1]$*:*
    .....
      for m in $[h_c^0, h_c^1]$*:*
        $x = a \cdot \lambda_1 + b \cdot \lambda_2 + \ldots + m \cdot \lambda_c$

where $h_j^0 = P(method\ j)$ and $h_j^1 = 1\text{-}P(method\ j)$ are obtained from the probability distribution of the jth comparison method (the jth node in the second layer). h_j^0 represents the probability of being matched for the jth comparison method. h_j^1 is the probability of being non-matched for the jth comparison method. λ_j is the normalized weight for the jth comparison method. The value of r obtained from each iteration comprises $\boldsymbol{r^0}$. $\boldsymbol{r^1}$ is considered as,

$$r^1(k) = 1 - r^0(k) \tag{4.30}$$

where $k = 1, 2, ..., 2^c$; $r^0(k)$ represents the kth element of vector $\boldsymbol{r^0}$; $r^1(k)$ represents the kth element of vector $\boldsymbol{r^1}$. $r^0(k)$ is the weighted sum of h_j^0 and/or h_j^1 with weights λ_j. Then the CPD of '*Match*' node is,

$$T_{match} = [\boldsymbol{r^0} \quad \boldsymbol{r^1}]^T \tag{4.31}$$

Up to now, the CPDs of all variables in the Bayesian network are determined.

4.3. Performance Assessment

In this study, confusion matrices are employed to evaluate the performance of the proposed record linkage model. The confusion matrix for record linkage contains six numbers, viz. the number of record pairs that are correctly classified as matches, the number of record pairs that are correctly classified as non-matches, the number of false negatives (matching record pairs classified as non-matches), the number of false positives (non-matching record pairs classified as matches), and the number of record pairs that are classified as potential matches. The confusion matrix indicating the misclassification costs is shown in Table 4.2,

Table 4.2. Confusion Matrix for Record Linkage

No. of record pairs/Unit cost	Non-match	Potential Match	Match
Non-match	$\eta_{no,\,no}$ /0	$\eta_{no,\,?}$ /β	$\eta_{no,\,yes}$ /γ
Match	$\eta_{yes,\,no}$ /γ	$\eta_{yes,\,?}$ /β	$\eta_{yes,\,yes}$ /0

where $\eta_{no,\,no}$, $\eta_{no,\,?}$, and $\eta_{no,\,yes}$ respectively represent the numbers of the non-matching record pairs that are classified as non-matches, potential matches, and matches; $\eta_{yes,\,no}$, $\eta_{yes,\,?}$, and $\eta_{yes,\,yes}$ are the numbers of the matching record pairs that are classified as non-matches, potential matches, and matches, respectively; γ is the unit cost for the misclassified record pairs, viz. false positives and false negatives; β is the unit cost for the record pairs that need clerical review. Of course, for the record pairs that are classified correctly, the cost is 0. Then the performance is expressed as follows,

$$Performance = \gamma\,(\eta_{no,\,yes} + \eta_{yes,\,no}) + \beta\,(\eta_{no,\,?} + \eta_{yes,\,?}) \tag{4.32}$$

Furthermore, the classification accuracy can be expressed as,

$$Accuracy = \frac{\eta_{no,\,no} + \eta_{yes,\,yes}}{\eta_{no,\,no} + \eta_{no,\,?} + \eta_{no,\,yes} + \eta_{yes,\,no} + \eta_{yes,\,?} + \eta_{yes,\,yes}} \times 100\% \tag{4.33}$$

Obviously, we should keep the performance as low as possible, and meanwhile, keep the accuracy rate as high as possible. $\eta_{no,\,no} + \eta_{no,\,?} + \eta_{no,\,yes} + \eta_{yes,\,no} + \eta_{yes,\,?} + \eta_{yes,\,yes}$ is the number of candidate record pairs, and it is usually fixed after indexing two selected datasets. Hence, the objective is to minimize the numbers of false positives, false negatives, and the record pairs that need clerical review. The numbers of false positives and false negatives are desired to be zeros since their unit costs are higher than the unit cost for the potential matches. Furthermore, the potential matches will be checked by a clerk to determine their categories, but the false positives and the false negatives have no opportunity to be checked again. Therefore, the primary objective is to make the numbers of false positives and false negatives as close to zeros.

Furthermore, *precision*, *recall*, and *f-measure* are also used to quantify the quality of the proposed record linkage model. Since the decision model we used contains three regions, the traditional formulas, which are based on the binary classification [8] to compute these three indexes, cannot be used directly. The traditional formulas are expressed as follows, where *tp* is the number of true positives; *fp* and *fn* respectively represent the number of false positives and the number of false negatives. Here, *f-measure* is defined as the harmonic mean of *precision* and *recall*.

$$Precision = \frac{tp}{tp + fp} \tag{4.34}$$

$$Recall = \frac{tp}{tp + fn} \tag{4.35}$$

$$F\text{-}measure = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{4.36}$$

In this study, the matching record pairs classified as potential matches or non-matches are considered as false negatives. Similarly, the non-matching record pairs that are classified as potential matches or matches are regarded as false positives. Therefore, the following expressions are used for computing *precision* and *recall*,

$$Precision = \frac{\eta_{yes,\,yes}}{\eta_{no,\,?} + \eta_{no,\,yes} + \eta_{yes,\,yes}} \tag{4.37}$$

$$Recall = \frac{\eta_{yes,\,yes}}{\eta_{yes,\,no} + \eta_{yes,\,?} + \eta_{yes,\,yes}} \tag{4.38}$$

5. Experimental Studies

To test the performance of the proposed record linkage model, we generated four synthetic datasets with different levels of noise (errors) to do the experiments. All these datasets were generated by the modified FEBRL (Freely Extensible Biomedical Record Linkage) tool. In this study, eight fields (m=8) were selected to identify records, namely *date of birth* (*DOB*), *social security ID* (*SSI*), *given nam*e, *surname*, *address* 1, *suburb*, *state*, and *phone number*. The ten comparison methods (functions) we selected were already described in Chapter 3.2, and they were used to compare all data fields. Then we have c=10. These ten comparison methods are Syllable Alignment Distance, Trigram, Bigram, Positional Bigram, Longest Common Substring 2 and 3, Levenshtein Edit Distance, Smith-Waterman Edit Distance, Damerau-Levenshtein Edit Distance, and Exact String Comparison. Therefore, the ANN has eight input nodes, ten hidden nodes, and one output node. In the Bayesian network, there are $mc = 80$ nodes (variables) in the first layer, ten nodes (variables) in the second layer, and one node (variable) in the third layer.

Next, in the records indexing step, three index keys were generated for blocking. *Given name* and *surname* were combined to generate the first block key. The second block key was generated by sorting *DOB*, *SSI*, and *phone number*. For the third block key, the combination of *address* 1, *suburb*, and *state* is used for sorting the data. Afterward, the candidate record pairs were obtained, and the comparison step began. The comparison result (similarity score) for a specific field with a selected comparison method is within [0, 1]. The higher the similarity score is, the more similar the two attributes are.

5.1. Synthetic Data

When generating the data, there are two remarkable aspects. First, the synthetic data need to be reasonable. In other words, it should have the characteristics of the real-world data. To be specific, the attributes of the synthetic data should be similar to those of the real-world data, and these attributes should follow similar frequency distributions that the real-world data follows. For example, the common names in the real-world need to appear frequently in this synthetic data. Sometimes, nicknames are included in the *given name*, so the synthetic data should also cover this case. Second, the synthetic data should contain some noise/errors, which is commonly seen in the real-world data. Typically, the errors occurred in the real world obey certain frequency distributions. Thus, in synthetic data, the errors should be as close as possible to the real-world data.

Furthermore, various types of errors should be included, such as missing fields, typing errors, and OCR errors. Phonetical variations are very common, and they usually occur when an entry does not affect the phonetical sound of names. For example, '*Chris*', '*Cris*', and '*Kris*' have the same phonetical sound. All these kinds of errors follow a certain frequency distribution in the real world, so the errors in the generated synthetic data should also obey the distribution. As the data is generated manually, we can have a deeper insight on it, and this will help us make a better qualitative assessment of the proposed record linkage model.

In this study, an open source record linkage system named FEBRL [41][42] helps us generate synthetic datasets. This tool has a graphical user interface (GUI), which is very convenient and easy to operate. When using this tool to complete record linkage tasks, Python codes are generated automatically in the background. Therefore, we are able to modify the Python codes to change the settings of the record linkage task. With this tool, we can not only generate synthetic datasets, but also obtain similarity scores that will be fed to the ANN and the Bayesian network. The generator in this tool can create a desired number of original records and a desired number of duplicates at the same time. Errors can be added by choosing a maximum number of duplicates per record, a maximum number of modifications per field, and a maximum number of modifications per record. Obviously, the maximum number of modifications per record should be larger or equal to the maximum number of modifications per field. By using the tables that contain field values and the corresponding frequency distributions, records and errors can be generated by the tool. Originally, the data generated by this tool contains 16 fields, viz. *record ID*, *culture*, *title*, *age*, *date of birth* (*DOB*), *sex*, *given name*, *surname*, *state*, *suburb*, *address* 1, *address* 2, *street number*, *postal code*, *social security ID* (*SSI*), and *phone number*. Furthermore, a single CSV file that contains all generated original records and duplicates is created by the generator. However, two documents are needed for the record linkage task. One contains some original records, and the other one contains all duplicates and the other original records. Thus, a module is added to the original Python codes by us to generate two CSV files. In addition, the attribute in field *age* is always changing over time, and it can be computed by the corresponding DOB, so we also modify the Python codes to eliminate the field *age*. With the modified Python codes, four synthetic datasets are generated as follows,

1) 4,500 original records and 1,500 duplicates with a maximum of 1 duplicate/record, a maximum of 1 modifications/field, and a maximum of 2 modifications/record;
2) 4,500 original records and 1,500 duplicates with a maximum of 2 duplicates/record, a maximum of 2 modifications/field, and a maximum of 3 modifications/record;
3) 4,500 original records and 1,500 duplicates with a maximum of 2 duplicates/record, a maximum of 3 modifications/field, and a maximum of 4 modifications/record.
4) 4,500 original records and 1,500 duplicates with a maximum of 3 duplicates/record, a maximum of 3 modifications/field, and a maximum of 5 modifications/record.

For each dataset, two CSV files are generated by the modified generator. File A contains 3,000 records in total, viz. 1,500 original records that do not have duplicates, and 1,500 original records that have at least one duplicate. For File B, it also has 3,000 records totally, including 1,500 original records that do not have duplicate records and 1,500 duplicate records. An example is shown in Table 5.1. to illustrate how the records are generated and modified. We can see that there were two duplicate records for 'record-404-org'. For 'rec-404-dup-0', three fields were modified, including *state*, *phone number*, and *SSI*. A phonetical error and a typing error (replacing '0' with an adjacent number '9') occurred in field *state* and field *SSI*, respectively. Furthermore, the attribute in field *phone number* changed a lot. The possible reason is that this person changed his/her phone number. There were also three modified fields in 'rec-404-dup-1'. In fields *surname* and *street number*, typing errors occurred again. Moreover, an OCR error (replacing number '1' by a special character '|') existed in field *phone number*. Overall, three types of errors were contained in this example, viz. typing errors, phonetical errors, and OCR errors.

Table 5.1. An Original Record and Its Duplicates

Rec_id|Culture|Sex|DOB|Title|Name|Surname|State|Suburb|Postcode|street #|address1|address 2|phone #|SSI
rec-404-org|usa| |23|19940927|mr|jo**seph**|bro**ck**|nsw|pyrmont|7000|**44**|the verge| |**08 65000115**|2043**0**47
rec-404-dup-0|usa| |23|19940927|mr|joseph|bro**nc**|nsw|pyrmont|7000|44|the verge| | **07 77579416**|2043**9**47
rec-404-dup-1|usa| |23|19940927|mr|jo**sej**|brock|nsw|pyrmont|7000|**42**|the verge| |**08 65000**|**15**|2043047

Among these 15 fields, we select eight fields to be compared in the ANN and the Bayesian network. They are *DOB*, *given name*, *surname*, *state*, *suburb*, *address* 1, *SSI*, and *phone number*. Field *address* 2 is discarded since it is blank on most records. In other words, most address information is concentrated in field *address* 1. Since these eight fields are the most common ones in the real-world data and they can provide enough information to represent a person's identity, they are selected to do the experiments for the record linkage.

5.1.1. Low Noise

In the first experiment, dataset 1 was used to perform the record linkage task. It has 4,500 original records and 1,500 duplicates with a maximum of one duplicate per record, a maximum of one modification per field, and a maximum of two modifications per record. Furthermore, four types of errors were included the data, viz. typographical errors, phonetical errors, OCR errors, and deletion of fields. Then this dataset was split into two files, one containing 3,000 original records, and the other one containing 1,500 original records and 1,500 duplicates. After records indexing with the Standard Blocking method, n=1641 candidate record pairs were generated. To obtain the weights for all fields and comparison methods, the ANN was used first. In the ANN training process, the candidate record pairs were separated into a training set and a testing set with the ratio 7:3. Hence, the training set has 1149 record pairs, and the testing set has 492 record pairs.

As described before, the objective is to classify the record pairs as a match or not a match, so the target vector $\boldsymbol{t}$ is represented in binary form. In other words, the target vector consists of 0 and 1. The final decision is determined according to which region does the matching score belong to, below τ_1 (non-match), above τ_2 (match), or within τ_1 and τ_2 (potential match), where τ_1 and τ_2 are two predefined thresholds. Obviously, the aim is to make record pairs classified as match/non-match as much as possible. To determine τ_1 and τ_2, for each candidate record pair, the similarity scores for all fields with all comparison methods were summed to get a global score. The histogram of the global scores for all candidate record pairs is shown in Figure 5.1. The values of τ_1 and τ_2 can be estimated from the histogram. Here, we set $\tau_1 = 0.4$, $\tau_2 = 0.8$. From Figure 5.1, we can also see that the training set was perfectly separated into two classes after the training process.

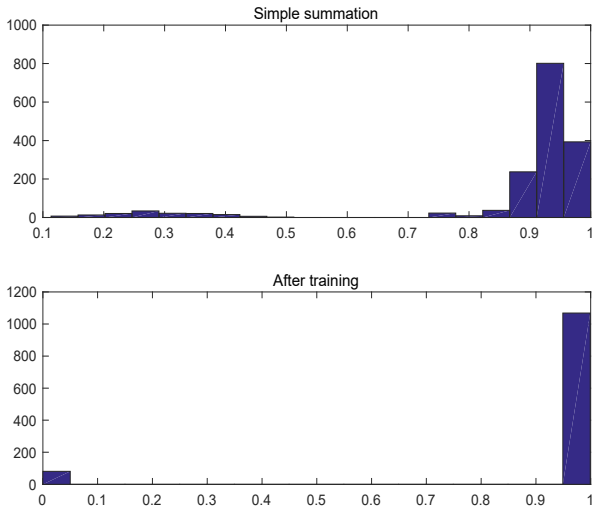

Figure 5.1. Histograms of Summation Scores for All Candidate Record Pairs and the Training Set in ANN with Dataset 1 (Low Noise)

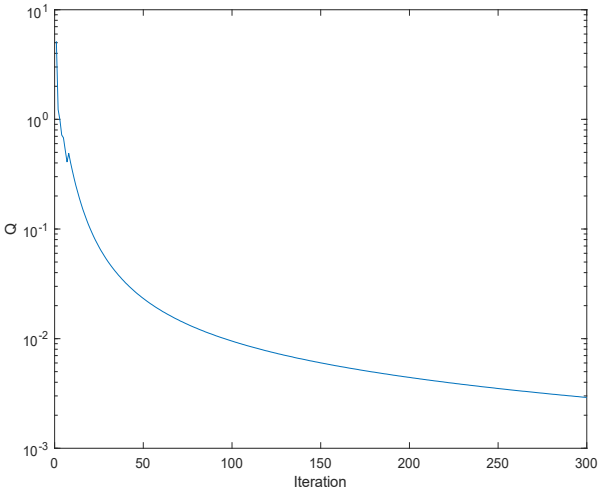

Figure 5.2. Q-plot for Candidate Record Pairs with Dataset 1 (Low Noise)

Furthermore, in the ANN, the number of iterations was set to 300, and the learning rate α was set to 0.4. The objective function (performance index) of the learning algorithm, viz. Q, should be as small as possible, and its trend graph is shown in Figure 5.2. It converged to 0.003 after 300 iterations.

Next, the trained ANN was tested with the data not used during the training process, namely the testing set. 20 groups of weights with random initial values were used to test the trained ANN. From the experimental results, we can see whether the weights coming from different groups have the same influence on the final decision or not. Among the 20 experiments, the most common confusion matrixes are shown in Table 5.2. The results show a 100% accuracy for the training set and a 99.80% accuracy for the testing set, where only one non-matching record pair was classified as a potential match. Therefore, the weights obtained from the ANN are quite suitable for describing the role of each field and each comparison method. In other words, these weights can be used in the Bayesian network to match the records. Besides, since the confusion matrixes among the 20 experiments had a small variation, about 0.13%, we can conclude that the final weights derived from different random initials had the same influence on the final decision. Hence, we just need to extract one set of weights to be used in the next step, viz. the determination of matching scores.

Table 5.2. Confusion Matrixes for Training and Testing Sets in ANN with Dataset 1 (Low Noise)

	Training set		
	Non-match	**Review**	**Match**
Non-match	81	0	0
Match	0	0	1068
	Testing set		
Non-match	59	1	0
Match	0	0	432

We extracted the final hidden-output weights $\boldsymbol{w}$ to find out which comparison methods contributed most during the training process. As described in Chapter 4, in the ANN, the final weights after normalized are all within [0, 1], and the weights whose values are close to 1 contribute more than the weights whose values are close to 0. As 20 sets of weights were obtained from the experiments, we calculated the average values and standard deviations of these weights to have a clear insight into these weights. The mean values and standard deviations of these weights are shown in Table 5.3. We can see that the Trigram comparison method had the highest overall weight, so it was the biggest influence factor for record linkage. Besides Trigram, Positional Bigram and Levenshtein Distance methods also had higher mean weights than the others.

Table 5.3. Hidden-output Weights $\boldsymbol{w}$ with Dataset 1(Low Noise)

SyD	**Trigram**	Bigram	LCS3	PBigram	LCS2	SW-D	LE-D	DLE	Exact
0.4672 ± 0.464	**0.6793 ± 0.401**	0.3994 ± 0.394	0.4618 ± 0.302	0.5821 ± 0.345	0.4447 ± 0.300	0.5073 ± 0.280	0.5610 ± 0.333	0.4317 ± 0.325	0.4387 ± 0.406

The final input-hidden weights $\boldsymbol{z}$ corresponding to the Trigram comparison method are shown in Table 5.4. Clearly, *DOB* field was the most relevant one since its weights had least variability, and the following fields were *address* and *surname*. Moreover, *SSI* and *suburb* also played an important role in separating the matching record pairs from the non-matching ones as their mean weights were higher than the others.

Table 5.4. Input-hidden Weights $\boldsymbol{z}$ Corresponding to Trigram Comparison Method with Dataset 1 (Low Noise)

DOB	SSI	Name	Surname	Address	Suburb	State	Tel.
0.5473 ± 0.121	0.6851 ± 0.405	0.4883 ± 0.183	0.5243 ± 0.165	0.5731 ± 0.150	0.6467 ± 0.242	0.5477 ± 0.187	0.6136 ± 0.306

However, the mean weights cannot be used in the Bayesian network since they have no actual meaning. In other words, the mean values cannot represent the actual distribution of these weights. As the confusion matrixes for the training and testing sets among the 20 experiments had a minor variation, we just extracted one experiment's final weights and used them to classify the candidate record pairs. The input-hidden weights and hidden-output weights we selected are shown in Table 5.5. The Positional Bigram comparison function contributed most for record linkage, and it was followed by Syllable Alignment Distance function and Levenshtein Distance function. For the input-hidden weights corresponding to Positional Bigram method, field *suburb* played an important role during the process of record linkage. Besides *suburb*, *address* and *SSI* also contributed a lot to the record linkage. Recalling that these weights cannot be directly employed to the Bayesian network, normalization should be done beforehand, and the details were already fully discussed in Chapter 3.4. After normalization, not only the sum of the hidden-output weights but also the sum of the input-hidden weights for each comparison method was 1. The details of the normalized weights are also shown in Table 5.5. Then the normalized weights were imported into the Bayesian network to complete the record linkage task.

Table 5.5. Selected Hidden-output Weights ***w*** and Input-hidden weights ***z*** Corresponding to Positional Bigram Comparison Function with Dataset 1 (Low Noise)

(a) Hidden-output Weights

Normalization	SyD	Trigram	Bigram	LCS3	**PBigram**	LCS2	SW-D	LE-D	DLE	Exact
Before	0.9639	0.1171	0.1089	0.2976	**1**	0.4375	0.4081	0.8819	0.5510	0
After	0.2022	0.0246	0.0228	0.0624	**0.2098**	0.0918	0.0856	0.1850	0.1156	0

(b) Input-hidden Weights

Normalization	DOB	SSI	Name	Surname	Address	**Suburb**	State	Tel.
Before	0.5638	0.8408	0.2413	0.25	0.8679	**0.9811**	0.6302	0.6483
After	0.1122	0.1674	0.0480	0.0498	0.1728	**0.1953**	0.1255	0.1122

After 1641 candidate record pairs and the normalized weights were imported into the Bayesian network, the probabilities of being a match (matching scores) were computed by the variable elimination algorithm. Since we have known which record pairs are matched or not matched when generating the synthetic data, a confusion matrix can be constructed to quantify the quality of the proposed record linkage model. The thresholds used for constructing the confusion

matrix were determined by the histogram in Figure 5.3, which shows the distribution of the candidate record pairs' matching scores. Most matching record pairs' scores were within [0.7, 1]. Furthermore, the matching scores for most non-matching record pairs were lower than 0.6. And there was a gap between [0.6, 0.7], which could be the region of potential matches. Therefore, we set the thresholds as 0.6 and 0.7.

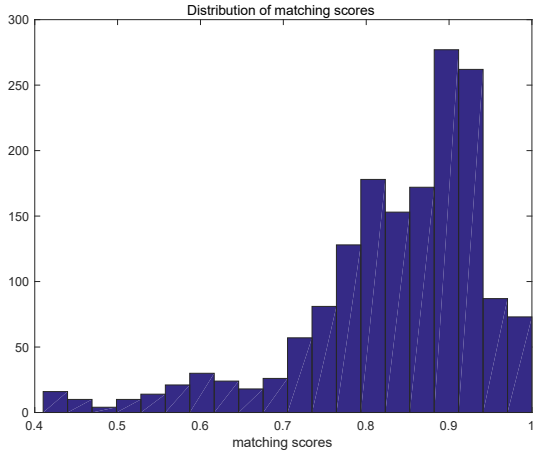

Figure 5.3. Histogram of Matching Scores for Candidate Record Pairs with Dataset 1 (Low Noise)

The confusion matrix for the record linkage results coming from the Bayesian network is shown in Table 5.6. 62 record pairs were misclassified and 81 record pairs required clerical review. In particular, 33 false positives and 29 false negatives were included in the misclassified records. In this experiment, the accuracy rate was 91.29%. Furthermore, the *precision*, *recall*, and *f-measure* were 94.49%, 96.07%, and 95.27%, respectively. Here, we can see that the proposed model accomplished the record linkage task with a great result.

Table 5.6. Confusion Matrix for the Final Record Linkage Results with Dataset 1 (Low Noise)

	Non-match	**Review**	**Match**
Non-match	57	51	33
Match	29	30	1441

5.1.2. Medium-low Noise

Next, dataset 2, which has 4,500 original records and 1,500 duplicates with two maximum duplicates per record, two maximum modifications per field, and three maximum modifications

per record, was used to perform the experiment. After records indexing by using the Standard Blocking method, 1650 record pairs were compared by the proposed record linkage model. Similarly, the data was split into a training set and a testing set, which consist of 1155 candidate record pairs and 495 candidate record pairs, respectively.

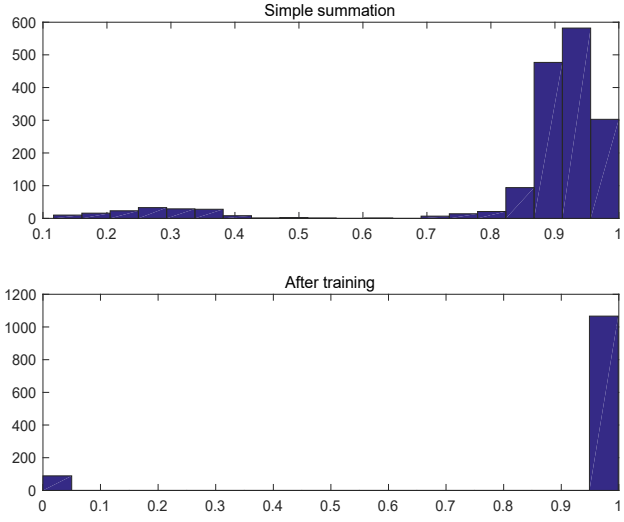

Figure 5.4. Histograms of Summation Scores for All Candidate Record Pairs and the Training Set in ANN with Dataset 2 (Medium-low Noise)

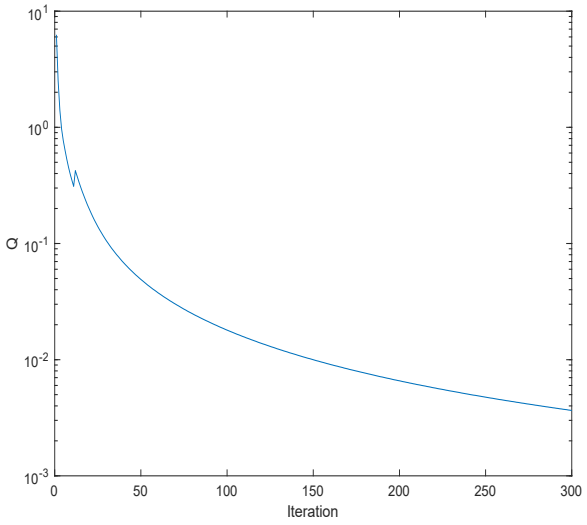

Figure 5.5. *Q*-plot for Candidate Record Pairs with Dataset 2 (Medium-low Noise)

After observing the histogram of simple scores summation for all candidate record pairs, as shown in Figure 5.4, we set the thresholds as 0.4 and 0.8. In addition, all the other parameters in the ANN learning structure remained the same values as before, including the number of iterations, learning rate, etc. The objective function (Figure 5.5) converged to 0.0035 when 300 iterations were accomplished.

Table 5.7. Confusion Matrixes for Training and Testing Sets in ANN with Dataset 2 (Medium-low Noise)

	Training set		
	Non-match	**Review**	**Match**
Non-match	89	0	0
Match	0	0	1066
	Testing set		
Non-match	60	1	1
Match	0	0	433

After training and testing the ANN learning structure, the confusion matrices which frequently appeared in the 20 experiments are shown in Table 5.7. The accuracy rates for the training set and testing set were 100% and 99.6%, respectively. In the testing set, one non-matching record pair was classified as a match, and another non-matching record pair was classified as a potential match.

The overall hidden-output weights $\boldsymbol{w}$ after training are shown in Table 5.8. The Trigram method had the highest mean weight among the ten comparison methods, so it influenced most on the record linkage results. Furthermore, the Longest Common Substring 3 method also had a high mean weight. However, the standard deviation of the Trigram function's weights was also higher, which means Trigram's influence on the record linkage changed a lot when performing the 20 experiments. The input-hidden weights $\boldsymbol{z}$ corresponding to the Trigram function are also shown in Table 5.8, where field *surname* had the least variation. Hence, the *surname* was the most important influence factor in the process of record linkage. Besides *surname*, *state*, *address* and *name* (*given name*) also had fewer variations than the other fields. Overall, there were little differences in the mean weights of these eight fields. Comparing the final weights for dataset 1 and dataset 2, as shown in Table 5.3, Table 5.4 and Table 5.8, we found that in both two cases the Trigram function was the most significant one among the ten comparison functions. Regarding the field weights, the weights for *name*, *surname*, *address*, and *state* had fewer variations than the others.

Table 5.8. Hidden-output Weights $\boldsymbol{w}$ and Input-hidden weights $\boldsymbol{z}$ Corresponding to Trigram Comparison Function with Dataset 2 (Medium-low Noise)

(a) Hidden-output Weights

SyD	**Trigram**	Bigram	LCS3	PBigram	LCS2	SW-D	LE-D	DLE	Exact
0.4690 ± 0.453	**0.6354 ± 0.406**	0.4533 ± 0.326	0.5781 ± 0.216	0.4527 ± 0.389	0.5305 ± 0.246	0.5106 ± 0.221	0.5479 ± 0.325	0.4861 ± 0.332	0.4821 ± 0.443

(b) Input-hidden Weights

DOB	SSI	Name	**Surname**	Address	Suburb	State	Tel.
0.5499 ± 0.201	0.6449 ± 0.427	0.5231 ± 0.182	**0.5499 ± 0.141**	0.5884 ± 0.176	0.6276 ± 0.420	0.5647 ± 0.154	0.5895 ± 0.186

The input-hidden and hidden-output weights that were selected to import into the Bayesian network are shown in Table 5.9, where the Trigram comparison method made the largest contribution for record matching, and it was followed by Damerau-Levenshtein Edit Distance

method, Bigram method, and Longest Common Substring 3 method. Regarding the data fields, *suburb*, *SSI*, and *phone number* had bigger impacts on the record linkage results. The normalized results of these weights are also shown in Table 5.9. As before, the sum of the normalized hidden-output weights and the sum of the normalized input-hidden weights corresponding to each comparison method were all 1.

Table 5.9. Selected Hidden-output Weights $\boldsymbol{w}$ and Input-hidden weights $\boldsymbol{z}$ Corresponding to Trigram Comparison Function with Dataset 2 (Medium-low Noise)

(a) Hidden-output Weights

Normalization	SyD	**Trigram**	Bigram	LCS3	Pbigram	LCS2	SW-D	LE-D	DLE	Exact
Before	0.2896	**1**	0.7656	0.7189	0.1667	0.6769	0.4109	0.2859	0.7898	0
After	0.0567	**0.1959**	0.1500	0.1408	0.0327	0.1326	0.0805	0.0560	0.1547	0

(b) Input-hidden Weights

Normalization	DOB	SSI	Name	Surname	Address	**Suburb**	State	Tel.
Before	0.3984	0.9951	0.3470	0.4110	0.4667	**1**	0.3860	0.7586
After	0.0837	0.2089	0.0729	0.0863	0.0980	**0.2100**	0.0810	0.1593

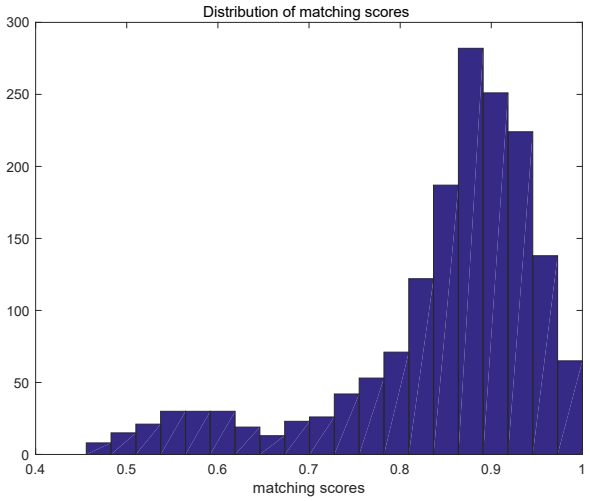

Figure 5.6. Histogram of Matching Scores for Candidate Record Pairs with Dataset 2 (Medium-low Noise)

Next, the selected weights were imported into the Bayesian network to illustrate the role of each comparison function and field. After the Bayesian network was constructed and variable elimination was completed, the matching scores for 1650 candidate record pairs were attained.

From the histogram of these matching scores, as shown in Figure 5.6, we can see that most matching record pairs had scores between 0.7 and 1, which is similar to the results got from dataset 1. And most non-matching record pairs were concentrated within [0.5, 0.6]. Two bell-shaped distributions existed in this histogram: one is below 0.6 and the other one is above 0.7. Thus, 0.6 and 0.7 were selected as the thresholds for the record linkage task. If the matching scores were within [0.6, 0.7], the corresponding record pairs would fall in the region of potential matches.

With the above two thresholds, the confusion matrix for the final record linkage results is shown in Table 5.10. Here, 33 candidate record pairs were misclassified, including 11 false positives and 22 false negatives. Furthermore, 81 record pairs were classified as clerical reviews. Therefore, the accuracy rate was 93.09% for dataset 2. The *precision* and *recall* were 95.71% and 96.73%, respectively. As a result, the *f-measure* was 96.22%.

Table 5.10. Confusion Matrix for the Final Record Linkage Results with Dataset 2 (Medium-low Noise)

	Non-match	**Review**	**Match**
Non-match	86	54	11
Match	22	27	1450

5.1.3. Medium-high Noise

In the next experiment, we used dataset 3 to accomplish the record linkage task. This dataset contains 4,500 original records and 1,500 duplicate records. Moreover, several errors were added into the data by selecting a maximum of two duplicates per record, a maximum of three modifications per field, and a maximum of four modifications per record. 1629 candidate record pairs were obtained after records indexing. After splitting the candidate record pairs, 1140 record pairs comprise the training set, and the other 489 record pairs comprise the testing set. The training set was used to train the ANN to obtain the optimized weights. The histograms of summation scores before and after training are shown in Figure 5.7. The thresholds we selected were still the same as before, viz. 0.4 and 0.8. From the histograms, we can see that the matching record pairs were separated from the non-matching pairs after training. Figure 5.8 shows the plot of the objective function Q. It also converged to 0.003 after 300 iterations.

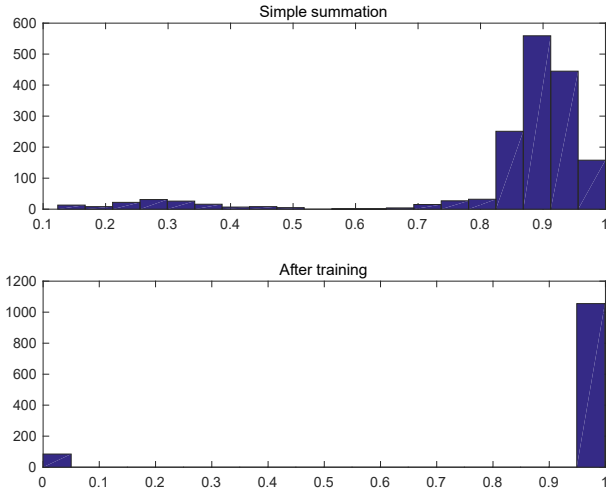

Figure 5.7. Histograms of Summation Scores for All Candidate Record Pairs and the Training Set in ANN with Dataset 3 (Medium-high Noise)

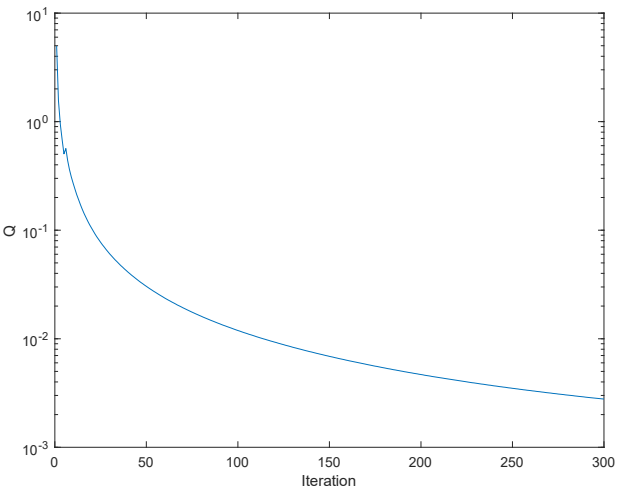

Figure 5.8. *Q*-plot for Candidate Record Pairs with Dataset 3 (Medium-high Noise)

Table 5.11. Confusion Matrixes for Training and Testing Sets in ANN with Dataset 3 (Medium-high Noise)

	Training set		
	Non-match	**Review**	**Match**
Non-match	85	0	0
Match	0	0	1055
	Testing set		
Non-match	51	0	0
Match	0	1	437

The confusion matrices that were most commonly seen during the 20 experiments are shown in Table 5.11. The training set was classified perfectly, but the testing set had one record pair to be reviewed. The accuracy rates for the training and testing sets were 100% and 99.8%, respectively. Therefore, the optimized weights obtained from the ANN were applicable to the Bayesian network to complete the record linkage task.

Table 5.12. Hidden-output Weights $\boldsymbol{w}$ and Input-hidden weights $\boldsymbol{z}$ Corresponding to Syllable Alignment Distance Comparison Function with Dataset 3 (Medium-high Noise)

(a) Hidden-output Weights

SyD	Trigram	Bigram	LCS3	PBigram	LCS2	SW-D	LE-D	DLE	Exact
0.6892 ± 0.403	0.4305 ± 0.400	0.4216 ± 0.326	0.5778 ± 0.254	0.5401 ± 0.351	0.5552 ± 0.279	0.5876 ± 0.218	0.5258 ± 0.236	0.5747 ± 0.237	0.2303 ± 0.396

(b) Input-hidden Weights

DOB	SSI	Name	**Surname**	Address	Suburb	State	Tel.
0.6767 ± 0.338	0.6781 ± 0.361	0.5278 ± 0.256	**0.5883 ± 0.092**	0.6303 ± 0.192	0.4645 ± 0.341	0.5527 ± 0.313	0.3985 ± 0.336

Table 5.12 shows the final hidden-output weights $\boldsymbol{w}$ in the form of mean value plus/minus standard deviation. The Syllable Alignment Distance method influenced most on the final record linkage results since it had the highest mean weight among the ten comparison methods. Besides Syllable Alignment Distance method, Smith-Waterman Edit Distance function, Longest Common Substring 3 function and Damerau-Levenshtein Edit Distance function also played important roles for the record linkage. These four methods had similar effects on separating the matching record pairs from the non-matching ones. However, the standard deviation of Syllable Alignment Distance function's weights was the highest among these four methods, which means this function had a high variation on the contribution to the record linkage. The overall input-hidden weights $\boldsymbol{z}$ corresponding to the Syllable Alignment Distance function are also included in Table 5.12. Field *surname* was the most important influence factor for record linkage, given that it had the least variation among the eight fields. *Address* and *name* (*given name*) were the following two fields that had less variations. After comparing the final weights for dataset 3 with dataset 1 and dataset 2, it can be found that *name*, *surname*, *address* and *state* were always the first four fields that had fewer variations. Moreover, for dataset 3, the weights for *DOB* and *SSI* were the top two, which indicates that these two fields were more useful for record linkage under the Syllable Alignment Distance method.

Then we selected one set of input-hidden and hidden-output weights among the 20 experiments' results, as shown in Table 5.13. Similarly, Syllable Alignment Distance comparison method still made the largest contribution to record matching, and it was followed by the Positional Bigram method and Bigram method. Regarding data fields, *DOB*, *SSI* and *phone number* were the most relevant fields, and their weights were all round 0.9. It seemed that the Syllable Alignment Distance method worked better on number fields than text fields. The normalized values of the selected weights are also described in Table 5.13. Clearly, the normalized hidden-output weights were summed up to 1, and so were the normalized input-hidden weights corresponding to each comparison method.

Table 5.13. Selected Hidden-output Weights w and Input-hidden weights z Corresponding to Syllable Alignment Distance Comparison Function with Dataset 3 (Medium-high Noise)

(a) Hidden-output Weights

Normalization	**SyD**	Trigram	Bigram	LCS3	PBigram	LCS2	SW-D	LE-D	DLE	Exact
Before	**1**	0.1416	0.8944	0.7964	0.9707	0.2772	0.3010	0.8062	0.7754	0
After	**0.1677**	0.0237	0.1500	0.1336	0.1628	0.0465	0.0505	0.1352	0.1300	0

(b) Input-hidden Weights

Normalization	**DOB**	SSI	Name	Surname	Address	Suburb	State	Tel.
Before	**0.9220**	0.9114	0.4455	0.7084	0.6577	0.7531	0.3846	0.8916
After	**0.1625**	0.1606	0.0785	0.1248	0.1159	0.1327	0.0678	0.1571

Afterward, the above selected weights were added to the Bayesian network to complete the record linkage task. The matching scores for the 1629 candidate record pairs were computed by the variable elimination algorithm, and the distribution of these scores is shown in Figure 5.9. Similar to the previous two datasets, the scores for most matching record pairs were larger than 0.7, and most non-matches had scores lower than 0.6. In this histogram, a gap existed within [0.6, 0.65]. Hence, the two boundary values were selected as the thresholds for the final record linkage. When the matching score was above 0.65, the corresponding record pair was classified as a match. On the contrary, a record pair that had a matching score lower than 0.6 was classified as a non-match. When a record pair had a score within [0.6, 0.65], it was identified as a potential match and required to be checked manually.

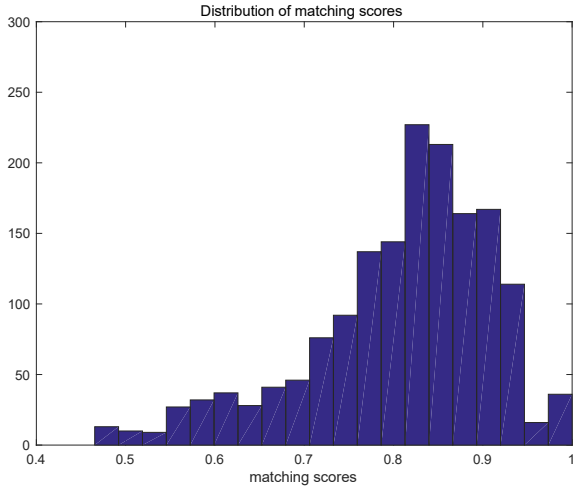

Figure 5.9. Histogram of Matching Scores for Candidate Record Pairs with Dataset 3 (Medium-high Noise)

Afterward, the confusion matrix based on the above two thresholds was obtained, as shown in Table 5.14. In this case, 86 candidate record pairs were misclassified, including 48 false positives and 38 false negatives. Furthermore, 64 candidate record pairs were classified as potential matches that needed clerical review. Therefore, the accuracy rate of record linkage for dataset 3 was 90.79%. The *precision* and *recall* were 94.50% and 95.51%, respectively. Then the *f-measure* for dataset 3 was 95.00%.

Table 5.14. Confusion Matrix for the Final Record Linkage Results with Dataset 3 (Medium-high Noise)

	Non-match	**Review**	**Match**
Non-match	53	35	48
Match	38	29	1426

5.1.4. High Noise

In the last experiment, dataset 4 was used to perform the record linkage task. It has 4,500 original records and 1,500 duplicate records with three maximum duplicates per record, three maximum modifications per field, and five maximum modifications per record. We still used the Standard Blocking method to index the records. When records indexing was finished, 1645 candidate record pairs were generated and they would be compared in the following steps. These 1645 candidate record pairs were split into a training set and a testing set first. Then the training set contained 1152 candidate record pairs, and the testing set contained the other 493 record pairs. Next, the ANN was trained with the training set to obtain the optimized weights. And the fully trained ANN was tested with the testing set to see if the optimized weights were qualified to link the records. The histograms of simply summed scores before and after training are shown in Figure 5.10. The histogram before training helped us decide the thresholds that should be used to train the ANN. Here, we set the thresholds to be 0.4 and 0.7. After training, all matching record pairs were separated from the non-matches. The plot of the objective function Q was shown in Figure 5.11. We can see that Q converged to 0.003 when 300 iterations were accomplished, which is similar to the results obtained from the previous three datasets.

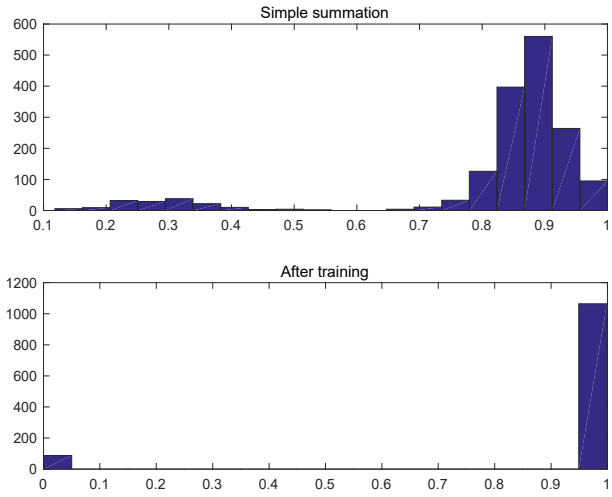

Figure 5.10. Histograms of Summation Scores for All Candidate Record Pairs and the Training Set in ANN with Dataset 4 (High Noise)

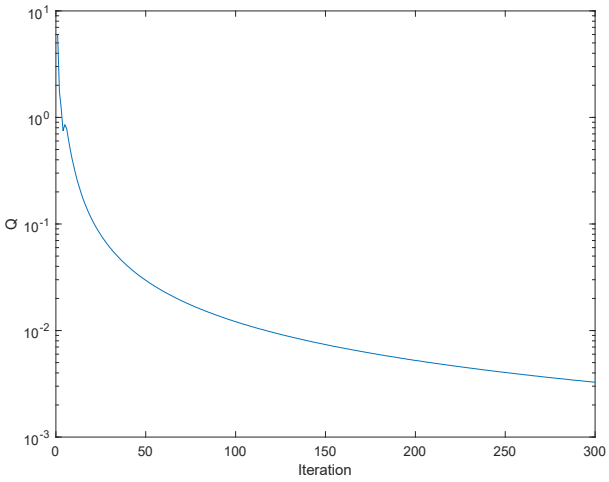

Figure 5.11. *Q*-plot for Candidate Record Pairs with Dataset 4 (High Noise)

Table 5.15. Confusion Matrixes for Training and Testing Sets in ANN with Dataset 4 (High Noise)

	Training set		
	Non-match	**Review**	**Match**
Non-match	88	0	0
Match	0	0	1064
	Testing set		
Non-match	67	0	0
Match	0	1	425

After using the above two thresholds, the confusion matrices for the training and testing sets were obtained. Table 5.15 shows the most common results among the 20 experiments. All record pairs in the training set were correctly classified, whereas one matching record pair in the testing test was classified as a potential match. Consequently, the accuracy rates for the training and testing set were 100% and 99.8%, respectively. Since the candidate record pairs were classified with a very high accuracy, the weights obtained from the ANN were applicable to match the records in the Bayesian network.

The mean values and the standard deviations of the final hidden-output weights $\boldsymbol{w}$ coming from the ANN are shown in Table 5.16. The Positional Bigram comparison method contributed most to the record linkage task since its mean weight was the highest among the ten comparison

methods. The Trigram method and the Exact String Comparison method were the next two fields that had bigger impacts on records matching.

The input-hidden weights z corresponding to the Positional Bigram comparison method are also shown in Table 5.16. Here, the *DOB* field had the least variation among the eight fields, so it was the most relevant field for separating the matches from the non-matches. Besides *DOB*, *suburb* and *address* were the fields that had fewer variations. For dataset 4, after excluding *DOB*, *name*, *surname*, *address* and *state* were still the first four fields that had fewer variations. Therefore, the roles of these four fields for dataset 4 are similar to the roles among the previous three datasets. Furthermore, the mean values of the optimized weights connecting to *phone number* and *SSI* were the lowest two. Field *DOB* had the fourth highest overall weight. These findings indicated that the Positional Bigram comparison method performed better on text fields than number fields with dataset 4.

Table 5.16. Hidden-output Weights w and Input-hidden weights z Corresponding to Positional Bigram Comparison Function with Dataset 4 (High Noise)

(a) Hidden-output Weights

SyD	Trigram	Bigram	LCS3	**PBigram**	LCS2	SW-D	LE-D	DLE	Exact
0.4426 ± 0.408	0.5199 ± 0.454	0.4113 ± 0.364	0.4949 ± 0.274	**0.5343 ± 0.385**	0.4193 ± 0.262	0.3964 ± 0.274	0.4501 ± 0.325	0.4832 ± 0.289	0.5164 ± 0.455

(b) Input-hidden Weights

DOB	SSI	Name	Surname	Address	Suburb	State	Tel.
0.5629 ± 0.107	0.5075 ± 0.413	0.5554 ± 0.239	0.6003 ± 0.220	0.5536 ± 0.192	0.5664 ± 0.183	0.5826 ± 0.241	0.5226 ± 0.282

The input-hidden and hidden-output weights we selected are shown in Table 5.17. Here, the Trigram comparison method, which is similar to Positional Bigram, influenced most on the record linkage, and it was followed by the Bigram method and Syllable Alignment Distance method. For data fields, *SSI* and *phone number* were the most relevant ones among the eight fields, and they were beyond the other six fields a lot. The normalized weights are also included in Table 5.17. As before, the summation of the normalized hidden-output weights and the summation of the normalized input-hidden weights corresponding to each comparison method were all 1.

Table 5.17. Selected Hidden-output Weights w and Input-hidden weights z Corresponding to Trigram Comparison Function with Dataset 4 (High Noise)

(a) Hidden-output Weights

Normalization	SyD	**Trigram**	Bigram	LCS3	PBigram	LCS2	SW-D	LE-D	DLE	Exact
Before	0.7904	**1**	0.8257	0.6728	0.2482	0.3119	0.3042	0.6984	0.2687	0
After	0.1544	**0.1953**	0.1613	0.1314	0.0485	0.0609	0.0594	0.1364	0.0525	0

(b) Input-hidden Weights

Normalization	DOB	**SSI**	Name	Surname	Address	Suburb	State	Tel.
Before	0.4421	**1**	0.2633	0.2870	0.4422	0.4296	0.2132	0.8144
After	0.1136	**0.2569**	0.0677	0.0738	0.1136	0.1104	0.0548	0.2093

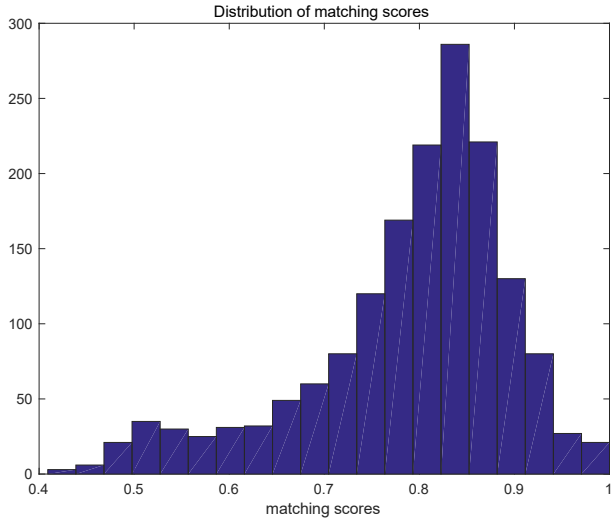

Figure 5.12. Histogram of Matching Scores for Candidate Record Pairs with Dataset 4 (High Noise)

The normalized weights and the 1629 candidate record pairs were imported into the Bayesian network to compute matching scores. The distribution of these matching scores is shown in Figure 5.12. Most matching pairs' scores were within [0.6, 1], where the lower boundary was a little bit lower than those for the previous three datasets. This is because more noise was included in this dataset. For the most non-matches, their matching scores were within [0.4, 0.55]. In the histogram, two bell-shaped distributions were connected in the region [0.55, 0.6]. This range was considered as the region of potential matches. Hence, the boundary values of this range were selected as the thresholds. The record pairs whose matching scores were higher than 0.6 were classified as

matches, and the record pairs that had matching scores lower than 0.55 were identified as non-matches.

Finally, the confusion matrix for the record linkage task with dataset 4 is shown in Table 5.18. In total, 84 candidate record pairs were misclassified, including 59 false positives and 25 false negatives. Besides, 50 candidate record pairs were classified as potential matches. Thus, the accuracy rate, *precision* and *recall* were 91.85%, 93.91% and 97.31%, respectively. The *f-measure* was calculated based on *precision* and *recall*, which was about 95.58%.

Table 5.18. Confusion Matrix for the Final Record Linkage Results with Dataset 4 (High Noise)

	Non-match	**Review**	**Match**
Non-match	61	35	59
Match	25	15	1450

5.2. Analyzing All Datasets

Overall, the accuracy rates for the four datasets with different levels of noise were all above 90%, which can be seen from Table 5.19. However, the datasets with higher levels of noise, viz. dataset 3 and dataset 4, had lower accuracy rates than the other two datasets. Since *f-measure* is the harmonic mean of *precision* and *recall*, it is more suitable for quantifying the quality of record linkage than the *precision* and *recall*. For the four datasets we tested, the *f-measures* were all above or equal to 95%. Therefore, it can be concluded that the proposed model performs well for the record linkage task.

Table 5.19. Performance Measures for Four Datasets

Dataset	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
1	91.29	94.49	96.07	95.27
2	93.09	95.71	96.73	96.22
3	90.79	94.50	95.51	95.00
4	91.85	93.91	97.31	95.58

With respect to weights, *surname*, *address*, *name*, and *state* were the most relevant fields for record linkage since their weights had fewer variations among the four datasets. Meanwhile, the mean values of their weights were also very high. Among the ten comparison functions, Positional

Bigram's weight was the most stable one as its mean values (viz. 0.58, 0.45, 0.54, and 0.53) moved in a tight range across the four datasets. The Longest Common Substring 2 and 3 comparison functions also had small variations on weights. The ranges of these two methods' weights were from 0.42 to 0.56 and from 0.46 to 0.58, respectively. Furthermore, their weights were very close to each other for each of the four datasets. Hence, their roles in records matching were quite similar. For dataset 1, dataset 2 and dataset 4, the mean weights of the Trigram method were always among the top four. And the average value of its four mean weights was 0.57, which was the highest among the ten comparison methods. Hence, the Trigram was the most significant comparison function. Moreover, the Smith-Waterman Edit Distance comparison method played a more import role in the first three datasets than in the dataset 4 since its mean weight for dataset 4 (0.40) was apparently lower than the other three mean weights, viz. 0.51, 0.51 and 0.59.

To get insights into the drawbacks of the proposed model, a detailed analysis for the misclassified record pairs is provided as follows. We selected a set of records from dataset 4 as an example. Recalling that the dataset 4 was split into two files, one file (file A) had 3,000 original records, and the other one (file B) had 1,500 duplicate records and 1,500 original records. The original record we selected from File A had three duplicate records in File B, as shown in Table 5.20. The '*rec-2076-org*' was from File A, and the other three records with '*dup*' were from File B. Each of the duplicates was compared with the original record after records indexing. The '*rec-2076-dup-0*' and '*rec-2076-dup-1*' were classified as matching records since their matching scores were above the upper threshold (also known as *autolink*), 0.6. The '*rec-2076-dup-2*' had a matching score smaller than the lower threshold (also known as *clerical review*), 0.55, so it was classified as a non-matching record. When we looked at the detailed information of the original record and '*rec-2076-dup-2*', we found that the biggest difference was that the content of fields *name* and *surname* was exchanged, which resulted in a low matching score and a wrong decision. Furthermore, a typing error ('*i*' was replaced by an adjacent character '*u*') occurred in field *surname*. In '*rec-2076-dup-0*', two errors existed. First, two numbers were typed in the wrong order in *DOB*. The other error was that the number '3' was replaced by '5' in field *Phone Number* due to an OCR error. Similarly, the '*rec-2076-dup-1*' also had two errors. A number in field *SSI* was entered incorrectly, and an additional character '*c*' was inserted in field *Suburb*. The errors in '*rec-2076-dup-0*' and '*rec-2076-dup-1*' were very minor so that the matching scores did not decrease so much. However, the '*rec-2076-dup-2*' had a big error that two fields' content was

exchanged, which became a 'critical strike', so its matching score was much lower than the other two duplicates. Moreover, a missing field in record linkage is always considered as a non-matching field, so the matching score will decrease when it occurs. It is therefore that the '*rec-2076-dup-0*' and '*rec-2076-dup-1*' did not have high matching scores, like 0.9 or more, even only two minor errors existing in each of them.

Table 5.20. Example 1 of Comparison Results with Dataset 4 (High Noise)

Record ID	DOB	SSI	Name	Surname	Address	Suburb	State	Phone Number	Match Score
rec-2076-org	**19780324**	**2858484**	**sophie**	**aaternir**	**<undefined>**	**roxbydowns**	**sa**	**746439244**	
rec-2076-dup-0	19**8**70324	2858484	sophie	aaternir	<undefined>	roxbydowns	sa	7464**5**9244	0.8672
rec-2076-dup-1	19780324	2858**7**84	sophie	aatern**oi**	<undefined>	roxby**c**dwons	sa	746439244	0.7538
rec-2076-dup-2	19780324	2858484	**aaternur**	**sophie**	<undefined>	roxbydowns	sa	746439244	0.4404

Another misclassified record pair, a false positive, is shown in Table 5.21. The '*record-2473-org*' was from File A and the '*record-3446-org*' was from File B. Apparently, the states in these two records were both '*vic*'. The names in these two records were also quite similar as '*annabel*' and '*annabelle*' had the same pronunciation. In other words, a phonetical error occurred here. Furthermore, field *DOB* in '*record-3446-org*' had four identical numbers with '*record-2473-org*'. Similarly, '*record-3446-org*' and '*record-2473-org*' also had three identical numbers in field *Phone Number*. Therefore, a high matching score, 0.6841, was generated. Since the score was above the *autolink* (0.6), this non-matching record pair was classified into the region of matches.

Table 5.21. Example 2 of Comparison Results with Dataset 4 (High Noise)

Record ID	DOB	SSI	Name	Surname	Address	Suburb	State	Phone Number	Match Score
rec-2473-org	**19100120**	**8371008**	**annabel**	**<undefined>**	**shackleton circuit**	**pascoevale**	**vic**	**878490690**	
rec-3446-org	**19860728**	6718759	**annabel**le	<undefined>	elvireplace	mentone	**vic**	**87**2394383	0.6841

5.3. Eliminating Two Comparison Methods

From the overall hidden-output weights for the synthetic datasets, as shown in Table 5.3, Table 5.8, Table 5.12 and Table 5.16, we can see that the Syllable Alignment Distance comparison method and the Exact String comparison method had the most unstable performance in record linkage since the standard deviations of their weights were relatively significant and were very close to the mean values of the weights. This means that they made tiny contributions to the linkage of records and might even reduce the linkage accuracy. Therefore, we eliminated these two less significant comparison methods in the proposed record linkage model and repeated the experiments with the same data to see if the accuracy was increased or not.

After elimination, we only had eight comparison methods, viz. $c = 8$. Hence, the ANN had 64 neurons in the input layer, eights neurons in the hidden layer, and one neuron in the output layer. Similarly, the Bayesian network had 64 nodes in the first layer, eight nodes in the second layer, and a single node in the third layer. As the indexing method was not changed, the candidate record pairs remained the same as before. After using the proposed model (the ANN model and the Bayesian network model) to link the records, the experimental results are described as follows. The histograms of matching scores are shown in Figure 5.13, Figure 5.14, Figure 5.15 and Figure 5.16, where the matches were further separated from the non-matches, compared with the experiments using ten comparison methods. From these four histograms, we can see that as more noise was added into the data, the region of matching record pairs moved left, which means the scores for matching record pairs were decreased. Furthermore, the scores for most non-matching record pairs were around 0.5. For most matching record pairs, their scores were higher than 0.6. For dataset 1 (low-noise data) and dataset 2 (medium-low-noise data), we set the thresholds as 0.55 and 0.6 since there was a gap within [0.55, 0.6]. For the rectangles located below 0.55, their shape was similar to the right side of a bell-shaped distribution, thus they are considered as the region of non-matching records pairs. For dataset 3 (medium-high-noise data) and dataset 4 (high-noise data), the boundary between the matches and non-matches was not clear. However, the first rectangle was still considered as the region of non-matches as most non-matches had matching scores around 0.5. The heights of the second and third rectangles were relatively lower compared with their ‘neighbors’, so they were considered as the region connecting the matches and non-matches (viz. the region of potential matches). As the interval containing the second and third bins was [0.52, 0.57], its boundary values (0.52 and 0.57) were selected as the thresholds.

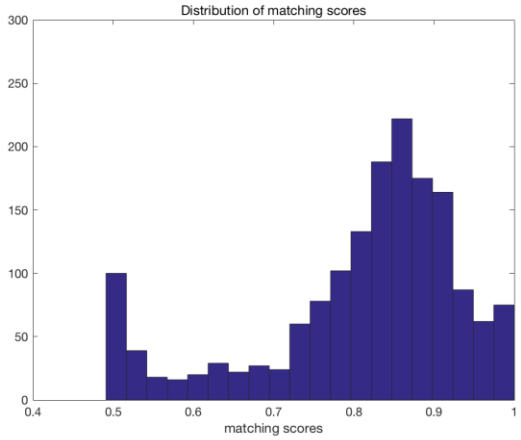

Figure 5.13. Histogram of Matching Scores for Experiments with Less Comparison Methods and Dataset 1 (Low Noise)

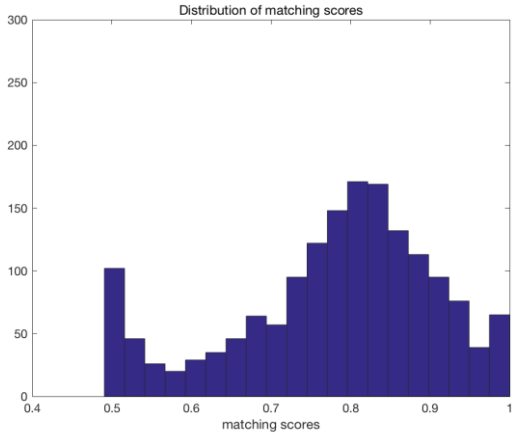

Figure 5.14. Histogram of Matching Scores for Experiments with Less Comparison Methods and Dataset 2 (Medium-low Noise)

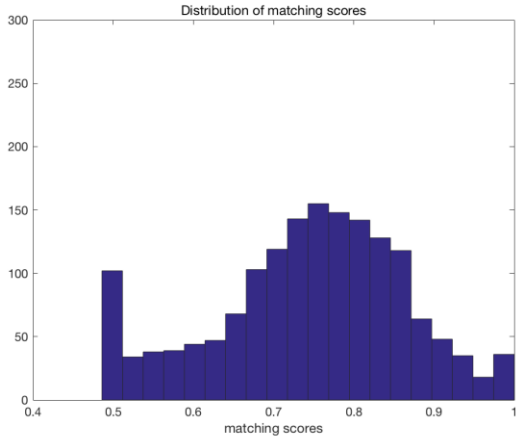

Figure 5.15. Histogram of Matching Scores for Experiments with Less Comparison Methods and Dataset 3 (Medium-high Noise)

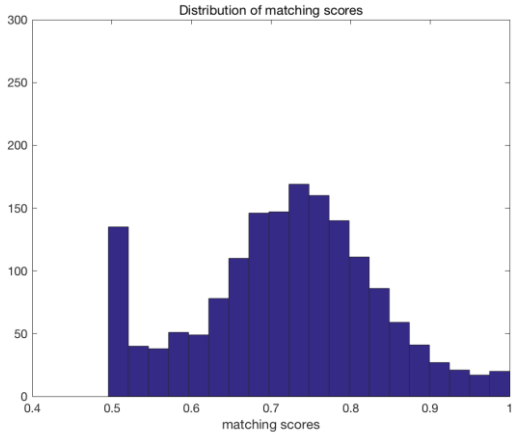

Figure 5.16. Histogram of Matching Scores for Experiments with Less Comparison Methods and Dataset 4 (High Noise)

After selecting the thresholds, the confusion matrixes can be obtained, as shown in Table 5.22. For dataset 1, 41 candidate record pairs were misclassified, including 10 false positives and 31 false negatives, and 34 candidate record pairs were classified as potential matches that needed clerical review. Dataset 2 had seven false positives and 28 false negatives. 42 candidate record pairs would be checked by a clerk. Regarding dataset 3 and dataset 4, more candidate record pairs were classified as potential matches, 63 and 79, respectively. Dataset 3 had the most misclassified

record pairs, including 19 false positives and 35 false negatives. For dataset 4, only 45 record pairs were misclassified, which is similar to dataset 1.

Table 5.22. Confusion Matrixes for Experiments with Less Comparison Methods

	Dataset 1 (Low Noise)		
	Non-match	**Review**	**Match**
Non-match	114	17	10
Match	31	17	1452
	Dataset 2 (Medium-low Noise)		
Non-match	131	13	7
Match	28	29	1442
	Dataset 3 (Medium-high Noise)		
Non-match	82	35	19
Match	35	28	1430
	Dataset 4 (High Noise)		
Non-match	102	39	14
Match	31	40	1419

With the confusion matrixes, the performance measures, viz. accuracy rate, *precision*, *recall*, and *f-measure*, can be calculated. The performance measures for the original record linkage model where ten comparison methods were used and the revised record linkage model where eight comparison methods were used are shown in Table 5.23. From this table, we can see that the revised model outperformed the original model on accuracy, *precision*, and *f-measure*. As more noise was added into the data, the range of advancement was decreased. Overall, the revised model performed better than the original model on record linkage, and the accuracy was augmented at the same time.

Table 5.23. Performance Measures for the Original Model and Revised Model

Dataset	**Accuracy (%)**		**Precision (%)**		**Recall (%)**		**F-measure (%)**	
	Original	**Revised**	**Original**	**Revised**	**Original**	**Revised**	**Original**	**Revised**
1	91.29	95.43	94.49	98.17	96.07	96.80	95.27	97.48
2	93.09	95.33	95.71	98.63	96.73	96.20	96.22	97.40
3	90.79	92.82	94.50	96.36	95.51	95.78	95.00	96.07
4	91.85	92.46	93.91	96.40	97.31	95.23	95.58	95.81

5.4. Indexing Only the Field 'Given Name'

In the real world, when we want to find a person in two or more databases, we usually search the records according to his/her given name, and then identify which records refer to the same entity. Therefore, indexing the given name can be considered as a technique of records indexing since it is commonly used in practice. Here, we selected dataset 3 to complete the record linkage task with indexing the given name. And we still used the original proposed record linkage model (the model with eight data fields and ten comparison methods) to link the records. After indexing, the records that had the same give name became the candidate record pairs. Specifically speaking, if we selected the first record in File A, the File B would be searched to find which records had the same given name with the record coming from File A. Assume that *L* records were found in File B, then these *L* records were respectively compared with the record coming from File A, and *L* matching scores were obtained in the end. This process proceeded iteratively until the comparisons between the last record in File A and its 'look-alikes' were completed. The dataset 3 was selected to perform this experiment, which has 2 maximum duplicates per record, 3 maximum modifications per field, and 4 maximum modifications per record. The dataset was split into two files, where File A has 3,000 original records, and File B contains 1,500 duplicate records and the other 1,500 original records. Table 5.24 shows an example of the comparison results, where the record in the first row came from File A and the other four records were the searching results from File B based on given name indexing. From the searching results, we can see that the two duplicate records had higher matching scores, around 0.9. On the contrary, the other two original records had lower matching scores, about 0.6.

Table 5.24. An Example of Given Name Indexing with Dataset 3 (Medium-high Noise)

Record ID	DOB	SSI	Name	Surname	Address	Suburb	State	Phone Number	Match Score
rec-4092-org	**20080723**	**3994703**	**sachin**	**drulkak**	**osburndrive**	**wynnum**	**qld**	**61347108564**	
rec-4092-dup-1	20080723	3994703	sachin	dru|kak	osburndrive	wynnum	qld	61347108564	0.9335
rec-2139-org	19770319	3726886	sachin	garikapaty	Williamwebb drive	kirwan	sa	705937275	0.6054
rec-4092-dup-0	20080723	3994703	sachin	drulkak	osburndrive	wynnm	qld	6134718564	0.8932
rec-3638-org	19931221	9177120	sachin	kawrungruang	Coningham street	marcushill	vic	853663699	0.6091

In total, 31,763 candidate record pairs were obtained after searching all records in File A and File B with indexing the given name. The distribution of the matching scores for all candidate record pairs is shown in Figure 5.17, where the scores of most matching record pairs were within [0.8, 1.0], and most non-matching records' scores were less than 0.75. From the figure, we can also see that there was a large bell-shaped distribution with a higher peak value within [0.4, 0.75] and a small bell-shaped distribution with a lower peak value within [0.8, 1]. It seemed that a gap existed between 0.75 and 0.8, which might be the region of potential matches. Thus, we set 0.75 and 0.8 as the thresholds to classify the candidate record pairs into three classes, *non-match*, *potential match*, and *match*. These two thresholds are also known as *clerical review* and *autolink*. If the matching score fell below the *clerical review*, the record pair would be classified as a non-match. A matching score above the *autolink* indicated that the corresponding record pair was a match. When a record pair's matching score was between the *clerical review* and the *autolink*, it would be classified as a potential match and be reviewed by a clerk.

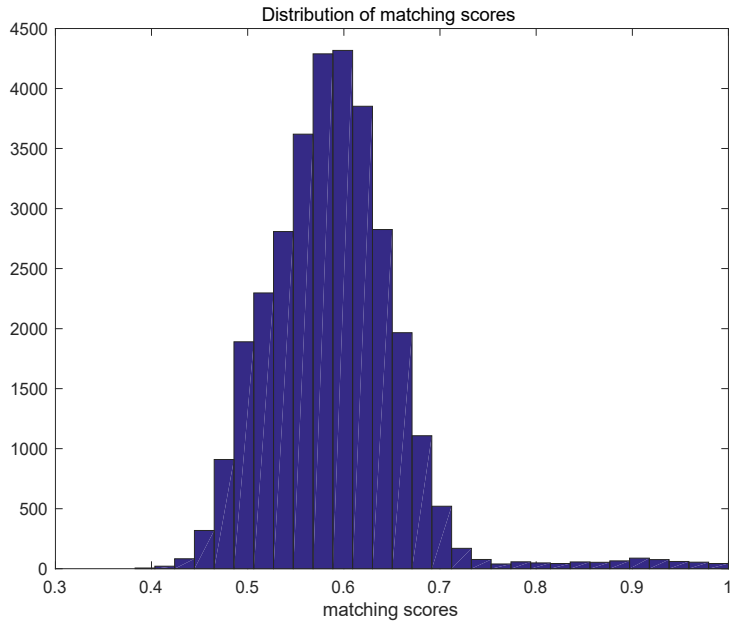

Figure 5.17. Distribution of Matching Scores for All Candidate Record Pairs Based on Given Name Indexing with Dataset 3 (Medium-high Noise)

After setting the two thresholds, the confusion matrix for this record linkage task was obtained, as shown in Table 5.25. Here, the accuracy rate was 98.72%. The *precision*, *recall*, and *f-measure* were 94.52%, 60.47%, and 73.75%, respectively. 288 candidate record pairs were misclassified,

including 281 false negatives and 7 false positives. Moreover, 117 candidate record pairs were assigned to the region of potential matches, and they would be reviewed manually to determine their categories.

Table 5.25. Confusion Matrix for the Record Linkage Based on Given Name Indexing with Dataset 3 (Medium-high Noise)

	Non-match	**Review**	**Match**
Non-match	30789	26	7
Match	281	91	569

For the matching record pairs classified as non-matches, viz. false negatives, the main reason is that there were missing fields in these records dues to the noise. The matching scores for the missing fields became 0, and as a result, the matching scores for the corresponding record pairs were lower than the normal values. Even if some records were compared with their duplicates, the matching scores were still very low.

Table 5.26. Searching Results based on the Given Name '*Eliza*' with Dataset 3 (Medium-high Noise)

Record from File A	**Record from File B**	**Matching Score**
rec-2643-org	rec-4026-org	0.6231
rec-2643-org	rec-2243-org	0.6287
rec-2643-org	rec-615-org	0.5836
rec-2643-org	rec-1783-org	0.5579
rec-2643-org	rec-2883-org	0.6232
rec-2643-org	rec-2643-dup-0	0.5755
rec-2643-org	rec-106-dup-0	0.6208
rec-2643-org	rec-609-org	0.6018
rec-2643-org	rec-2643-dup-1	0.7539
rec-2643-org	rec-3516-org	0.6195

An example is shown in Table 5.26 and Table 5.27 to illustrate this kind of situation, where field *state* was missing in both the original record and two duplicates. '*Rec-2643-org*' was from File A. After searching the File B based on indexing the given name '*eliza*', the indexing results were obtained, as shown in Table 5.26. Ten candidate record pairs were obtained after indexing. From this example, we can also see that even some records in File B were the duplicates, their matching scores were still lower than the normal values (about 0.8 to 1). The main reason is that

the missing fields existed in the record pair. Furthermore, for '*rec-2643-dup-0*', the address and suburb were filled out with the wrong order. Thus, '*rec-2643-dup-0*' and '*rec-2643-dup-1*' were classified as a non-match and a potential match, respectively. So, when one or more fields are missing in the record pair, the matching score will be lower than that under the same condition but without missing fields. This demonstrates why some matching record pairs were classified as non-matches or potential matches.

Table 5.27. The Record Information of the Original and Duplicate Records

Record ID	DOB	SSI	Name	Surname	Address	Suburb	State	Phone Number
rec-2643-org	**19951112**	**9890381**	**eliza**	**fairclough**	**millerstreet**	**castlemaine**	**<undefined>**	**244911181**
rec-2643-dup-0	19951112	9890381	eliza	fairclough	**osreetmiller**	**castlemaine**	<undefined>	244911181
rec-2643-dup-1	19951112	9890381	eliza	fairclough	millerstreet	castlemaine	<undefined>	244911181

To visualize the record linkage results based on given name indexing, the candidate record pairs and their matching scores were imported into Neo4j, which is a powerful graph database that can run fast queries on complex graph datasets [34]. With this tool, graph sets can be constructed manually with setting nodes, directed edges, and relationships. Each node could have several properties and a label. Each edge could have one or two directions to represent the relationships between the nodes it connects to. Moreover, edges could have several properties to show their characteristic. If needed, a node can have several edges connecting to other nodes.

In this study, we separated the information of records into three parts, personal information, home information and phone information. The personal information contains fields *name*, *surname*, *DOB* and *SSI*, and the home information refers to fields *address*, *suburb* and *state*. The phone information is precisely the *phone number.* For each record, each part of its information was represented by a node in the graph set, and we labeled these three parts of information as '*PersonA*'/'*PersonB*', '*HomeA*'/'*HomeB*' and '*PhoneA*'/'*PhoneB*' depending on which dataset did the record belong to. Furthermore, we had the ID of each record when generating the data, so another type of node was added to show the record' ID. In this way, we could know which records are the duplicates and which are the original records. Similarly, the fourth type of node was labeled as '*idA*' or '*idB*'. In total, there were four types of nodes in the graph set we constructed. For

'*Person*' nodes, they had four properties, *name*, *surname*, *DOB*, and *SSI.* '*Home*' nodes had three properties, *address*, *suburb*, and *state*. Apparently, both '*Phone*' nodes and '*id*' nodes had a single property, *phone number* and *ID*, respectively. Afterward, edges (relationships) between these nodes were built. Here, we set the relationship between '*Person*' nodes and '*Home*' nodes as '*LIVE_IN*', the relationship between '*Person*' nodes and '*Phone*' nodes as '*TEL*', the relationship between '*Person*' nodes and '*id*' nodes as '*ID*'. All these relationships (edges) started from the '*Person*' nodes and pointed to the '*Home*'/'*Phone*'/'*id*' nodes. In this way, the four types of nodes were connected with the three kinds of relationships.

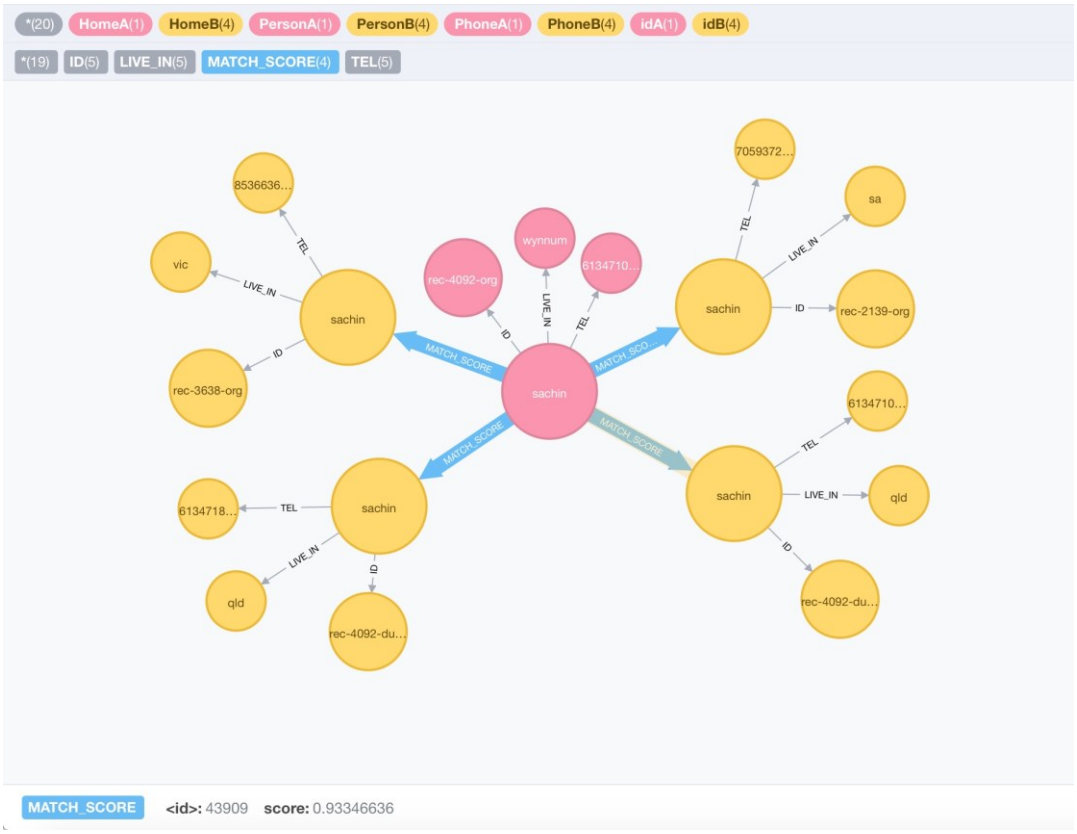

Figure 5.18. An Example from the Neo4j Graph Set Based on Given Name Indexing with Dataset 3 (Medium-high Noise)

To visualize the graph clearly, the records from File A and File B were represented by pink nodes and yellow nodes, respectively, as shown in Figure 5.18. The graph set in Figure 5.18 displayed the linkage results based on the example shown in Table 5.24, where a record in File A had four candidate duplicates in File B. From the Bayesian network, the matching scores of the

candidate record pairs were obtained, which could indicate how likely did each record pair refer to the same entity. So, we imported the matching scores into the graph set, as shown in Figure 5.18. For each candidate record pair, an edge (relationship) named '*MATCHING_SCORE*' was added, which started from the record in File A and pointed to the record in File B. This kind of relationship only had one property, *matching score*, to indicate the probability of being a match for the record pair. In this way, the whole graph set was constructed. From Figure 5.18, we can see that the record at the bottom right was probably a duplicate of the record in File A since their matching score (0.9335) was very close to 1, as shown at the bottom of the figure. Similarly, as the matching score between the pink '*sachin*' and the yellow '*sachin*' at the bottom left was 0.8932, there was a high probability that these two records referred to the same entity.

It is worth mentioning that in this powerful graph database when we click on a node or edge, the toolbar at the bottom can show all its properties. For example, the *matching score* property and its value were both shown at the bottom of Figure 5.18 when we clicked on the '*MATCHING_SCORE*' edge. If we click on the '*Person*' node, all the four properties and their values will be shown at the bottom. Moreover, we can choose any property of the nodes to be shown in the graph. In Figure 5.18, *name* was chosen to be shown on '*Person*' nodes; *suburb* was selected for '*Home*' nodes. From the ids and the matching scores of the records, we can easily find out whether the decision made by the proposed record linkage model is consistent with the truth. In addition, queries can be done easily based on the graph set we built. The graph in Figure 5.18 is a search result from the big graph set we built. In detail, we searched the graph set based on which records had the name '*sachin*' and extracted all the nodes and relationships. In practice, people usually search databases according to a name, an address, or a social security ID. With Neo4j and the graph set constructed, the queries based on the above information can be done quickly and accurately. Anyway, this tool helps us a lot on visualizing our results and doing queries about the results.

6. Conclusions and Future Studies

In this study, a PGM for record linkage was proposed. Several weights, which were previously determined by a supervised gradient-descent learning scheme, were added to the constructed PGM to improve the linkage results. The complete process of record linkage was shown, including fields selection, selection of comparison methods, construction of the gradient-based model, optimization of weights, determination of conditional probability distributions, construction of the Bayesian network, and the determination of the matching scores (probabilities). The indexing method, Standard Blocking, and the selected comparison methods were also fully discussed. Furthermore, a modified python script included in the FEBRL tool was used to generate the synthetic datasets. The similarity scores for fields with different comparison methods were also obtained with the FEBRL tool. These similarity scores were regarded as the input for the ANN and the Bayesian model (PGM) to obtain the optimized weights and matching probabilities, respectively. Moreover, a decision model was used to inspect the quality of the record linkage, based on our model.

From the experimental results, several interesting conclusions can be derived. First, the experiments showed that the proposed model could help us find out which comparison methods are more significant and which fields are more relevant. These are important influencing factors that help determine the final matching scores. *Name*, *surname*, *address*, and *state* were the most relevant fields since their mean weights were relatively high among the selected fields for all tested datasets. Regarding the comparison methods, Positional Bigram was the most stable one given that its mean weight was commonly optimized to around 0.52 in most experiments. Longest Common Substring 2 and 3 also reached a similar value in most experiments. Second, after several experiments on datasets with different levels of noise, the results showed that the proposed record linkage model is stable and robust since the fluctuation range of the accuracy rates from different experiments was relatively narrow. Moreover, the indexing of the entities effectively reduced the number of record pairs to be compared, which sped up the record linkage process. Next, the accuracy rates and the values of *f-measure* on the tested datasets were all above 90% and 95%, respectively. From this, we can see that the proposed model had an acceptable performance. Furthermore, with the revised record linkage model where the Syllable Alignment Distance comparison method and the Exact String comparison method were eliminated, the accuracy and *f-measure* on the tested datasets were higher than those with the original proposed record linkage

model. This indicates that removing the comparison methods that are less significant in record linkage will help to augment the accuracy of the linkage of records. Finally, a graph (generated in Neo4j) was shown in this study, which visually displayed the classification results based on the indexing a single given name. A system combining the proposed model with the graph could not only contribute to entity resolution, but also discover relations among individuals. For example, if we want to track which people are named '*Chris*', we can query the system, and then the graph will show all the entities that have this name, including personal information and relationships among them. This kind of tool could be very useful in the real world.

Beyond this study, some interesting ideas can be pursued. First of all, some other widely used comparison functions can be added into the model, e.g. the Jaro-Winkler comparison function, the Bag Distance comparison function, or the Editex comparison function. Next, the identity fields that show little contributions to record linkage can be eliminated, and/or some other fields can be added into the model, such as gender, citizenship, etc. After changing several fields or comparison methods, an updated model can be generated. By comparing the experimental results from the original model and the updated model, we can potentially see whether the new model performs better than the original one or not. Moreover, we can look for some other methods to determine conditional probability distributions, and check if they can augment the classification accuracy/*f-measure* or not. Furthermore, it would be interesting to test our model's performance with the real-world data. As of now, the proposed model requires labeled data. Real-world data does not contain labels, therefore we have to turn to unsupervised classification, such as K-Means or Fuzzy C-Means.

References

[1] L. Hongmei, H. Wenning, G. Wenyan, and C. Gang, "Survey of Probabilistic Graphical Models," in 2013 10th Web Information System and Application Conference, pp. 275–280, November 2013. DOI: 10.1109/WISA.2013.59.

[2] D. Koller and N. Friedman, Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning. The MIT Press, 2009.

[3] H. B. Newcombe, J. M. Kennedy, S. J. Axford, and A. P. James, "Automatic Linkage of Vital Records," Science, vol. 130, no. 3381, pp. 954–959, Oct. 1959. DOI: 10.2307/2286061

[4] I. P. Fellegi and A. B. Sunter, "A Theory for Record Linkage," Journal of the American Statistical. Association, vol. 64, no. 328, pp. 1183–1210, 1969. DOI: 10.2307/2286061.

[5] P. Langley, W. Iba and, and K. Thompson, "An Analysis of Bayesian Classifiers," in Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, California, pp. 223–228, 1992.

[6] D. Q. Department and D. Quass, "Record Linkage for Genealogical Databases," in KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation, pp. 40–42, 2003.

[7] Y. Liang, "A Comparative Experiment on Record Match Algorithms: Naive Bayesian Versus Distance," in Proceedings of the 45th Acm Southeast Conference; Annual Southeast Regional Acmse, Red Hook, New York, NY, USA, pp. 539–540, 2007. DOI: 10.1145/1233341.1233453.

[8] D. L. Olson and D. Delen, Advanced Data Mining Techniques, Berlin: Springer, 2008.

[9] P. Christen, Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. New York: Springer, July 5th, 2012.

[10] Y. Yao, "Three-Way Decision: An Interpretation of Rules in Rough Set Theory," in Proceedings of International Conference on Rough Sets and Knowledge Technology, Berlin: Springer, pp. 642–649, July 14th, 2009. DOI: 10.1007/978-3-642-02962-2_81.

[11] Y. Yao, "Three-way decisions with probabilistic rough sets," Information Sciences, vol. 180, no. 3, pp. 341–353, Feb. 2010. DOI: 10.1016/j.ins.2009.09.021.

[12] V. S. Verykios, G. V. Moustakides, and M. G. Elfeky, "A Bayesian Decision Model for Cost Optimal Record Matching," VLDB Journal., vol. 12, no. 1, pp. 28–40, May 2003. DOI: 10.1007/s00778-002-0072-y.

[13] J. P. Ferry, D. Lo, and T. Seaquist, “A Bayesian idealization of entity resolution,” in 2015 18th International Conference on Information Fusion (Fusion), pp. 150–157, July 2015.

[14] J. Nin and V. Torra, “New Approach to the Re-identification Problem Using Neural Networks,” in Proceedings of International Conference on Modeling Decisions for Artificial Intelligence, pp. 251–261, April 3rd, 2006. DOI: 10.1007/11681960_25.

[15] D. R. Wilson, “Beyond Probabilistic Record Linkage: Using Neural Networks and Complex Features to Improve Genealogical Record Linkage,” in the 2011 International Joint Conference on Neural Networks, pp. 9–14, July 2011. DOI: 10.1109/IJCNN.2011.6033192.

[16] L. Srivastava, S. N. Singh, and J. Sharma, “Parallel Self-organising Hierarchical Neural Network Based Estimation of Degree of Voltage Insecurity,” Computer & Electrical Engineering, vol. 29, no. 5, pp. 589–602, Jul. 2003. DOI: 10.1016/S0045-7906(02)00044-7.

[17] P. Christen, “Automatic Record Linkage Using Seeded Nearest Neighbour and Support Vector Machine Classification,” in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 151–159, 2008. DOI: 10.1145/1401890.1401913.

[18] J. Murillo, D. Abril, and V. Torra, “Heuristic Supervised Approach for Record Linkage,” in Proceedings of the 9th International Conference on Modeling Decisions for Artificial Intelligence, Berlin: Springer, pp. 210–221, 2012. DOI: 10.1007/978-3-642-34620-0_20.

[19] J. Whittaker, Graphical Models in Applied Multivariate Statistics. Chichester, West Sussex, England; Toronto: Wiley, 1990.

[20] Q. Li and J. Zhao, “A Novel Approach to Object/Background Segmentation Based on the Probabilistic Graphical Model,” in 2009 Fifth International Conference on Image and Graphics, pp. 162–167, September 2009. DOI: 10.1109/ICIG.2009.14.

[21] T. Tamanaha and H. Nakamaya, “Unsupervised Cosegmentation Based on Global Graph Matching,” in Proceedings of the 23rd ACM International Conference on Multimedia, New York, NY, USA, pp. 1203–1206, 2015. DOI: 10.1145/2733373.2806317.

[22] L. Liu, Y. Lu, and C. Y. Suen, “Document Image Matching Using Probabilistic Graphical Models,” in Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), vol. 1, pp. 637–640, November 2012.

[23] S. Chen, W. Zhu, and H. Leung, “Thermo-visual Video Fusion Using Probabilistic Graphical Model for Human Tracking,” in 2008 IEEE International Symposium on Circuits and Systems, pp. 1926–1929, May 2008. DOI: 10.1109/ISCAS.2008.4541820.

[24] G. Nicolai, M. A. Islam, and R. Greiner, “Native Language Identification using probabilistic graphical models,” in 2013 International Conference on Electrical Information and Communication Technology (EICT), pp. 1–6, February 2014. DOI: 10.1109/EICT.2014.6777864.

[25] D. Putthividhya, H. T. Attias, S. S. Nagarajan, and T. W. Lee, “Probabilistic Graphical Model for Auto-Annotation, Content-Based Retrieval, and Classification of TV Clips Containing Audio, Video, and Text,” in 2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP ’07, vol. 2, pp. II-789-II-792, April 2007. DOI: 10.1109/ICASSP.2007.366354.

[26] P. Baldi and S. Brunak, “Probabilistic Graphical Models in Bioinformatics,” in Bioinformatics, second edition: The Machine Learning Approach, MIT Press, pp. 225–263, 2001.

[27] D. E. Xing, “Probabilistic Graphical Models-Theory, Algorithm, and Application,” in Sixth International Conference on Machine Learning and Applications (ICMLA 2007), pp. xvii–xvii, December 2007. DOI: 10.1109/ICMLA.2007.126.

[28] M. Khademi and N. S. Nedialkov, “Probabilistic Graphical Models and Deep Belief Networks for Prognosis of Breast Cancer,” in 2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA), pp. 727–732, December 2015. DOI: 10.1109/ICMLA.2015.196.

[29] K. AlJadda, M. Korayem, C. Ortiz, T. Grainger, J. A. Miller, and W. S. York, “PGMHD: A Scalable Probabilistic Graphical Model for Massive Hierarchical Data Problems,” in 2014 IEEE International Conference on Big Data (Big Data), pp. 55–60, October 2014. DOI: 10.1109/BigData.2014.7004213.

[30] B. D’Ambrosio References"AIAA-2004-6286 Bayesian Variable Resolution Modeling for Autonomous Satellite Cluster Data Fusion," AIAA Intelligent Systems Technical Conference, pp. 458-472, 2004.

[31] L. E. Sucar, “Probabilistic Graphical Models and Their Applications in Intelligent Environments,” in 2012 Eighth International Conference on Intelligent Environments, pp. 11–15, June 2012. DOI: 10.1109/IE.2012.66.

[32] P. Singla and P. Domingos, “Entity Resolution with Markov Logic,” in Sixth International Conference on Data Mining (ICDM’06), pp. 572–582, December 2006. DOI: 10.1109/ICDM.2006.65.

[33] F. van Harmelen, F. van Harmelen, V. Lifschitz, and B. Porter, Handbook of Knowledge Representation. San Diego, USA: Elsevier Science, 2007.

[34] R. van Bruggen and P. Mohanta, Learning Neo4j. Birmingham: Packt Publishing, 2014.

[35] M. A. Jaro, “Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida,” Journal of the American Statistical Association, vol. 84, no. 406, pp. 414–420, 1989. DOI: 10.2307/2289924.

[36] W. E. Winkler, String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. 1990.

[37] V. S. Verykios and G. V. Moustakides, “A Generalized Cost Optimal Decision Model for Record Matching,” in Proceedings of the 2004 International Workshop on Information Quality in Information Systems, New York, NY, USA, pp. 20–26, 2004. DOI: 10.1145/1012453.1012457.

[38] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, “A Comparison of String Distance Metrics for Name-matching Tasks,” in Proceedings of the International Conference on Information Integration on the Web, pp. 73–78, August 2003.

[39] P. Christen, “A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication,” IEEE Transactions Knowledge and Data Engineering, vol. 24, Issue 9, pp. 1537–1555, Sep. 2012. DOI: 10.1109/TKDE.2011.127.

[40] X. Li et al., “Implementation of an Extended Fellegi-Sunter Probabilistic Record Linkage Method Using the Jaro-Winkler String Comparator,” in IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), June 2014, pp. 375–379. DOI: 10.1109/BHI.2014.6864381.

[41] P. Christen, “Febrl: A Freely Available Record Linkage System with a Graphical User Interface,” in Proceedings of the Second Australasian Workshop on Health Data and Knowledge Management - Volume 80, Darlinghurst, Australia, Australia, pp. 17–25, 2008.

[42] P. Christen, “Febrl -: An Open Source Data Cleaning, Deduplication and Record Linkage System with a Graphical User Interface,” in Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, pp. 1065–1068, 2008. DOI: 10.1145/1401890.1402020.

[43] A. Ankan and A. Panda, “pgmpy: Probabilistic Graphical Models Using Python,” in Proceedings of the 14th Python in Science Conference, pp. 6–11, 2015.

[44] A. Ankan and A. Panda, Mastering Probabilistic Graphical Models Using Python: Master Probabilistic Graphical Models by Learning through Real-World Problems and Illustrative Code Examples in Python. Birmingham, UK: Packt Publishing, 2015.

[45] J. A. Snyman, Practical Mathematical Pptimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-based Algorithms. New York : Springer, 2005.