Improve Record Linkage Using Active Learning Techniques

Chong Feng
u4943054

supervised by
Dr.Qing Wang and Dr.Dinusha Vatsalan

COMP8715 Individual Project (12 units)
Research School of Computer Science
Australian National University

October 28, 2016
Abstract

Record linkage is a crucial step in data integration and data mining. It has been studied extensively in the past several decades due to its wide-spread applications in e-commerce, healthcare, social sciences and so on. However, many record linkage models are not satisfactory in practice as they neglect an important aspect, which is how to repair the errors that are detected later on in the linkage results. In this report I develop a record linkage model that is integrated with active learning techniques to effectively repair errors and thereby improving the model itself and the linkage results incrementally over time. The model can accept user detected linkage errors, locate the errors in itself and repair it automatically. Also, with each error being repaired, the classification process within the record linkage model is refined, thus the overall clustering results can be improved. I have evaluated the record linkage model on two real-world databases and the results show that the clustering result can be effectively improved over time.
Chapter 1

Introduction

Many relational databases contain uncertain and imprecise record references to the same real-world entities [1]. For example, a database table that stores registered user information may have “John Smith” and “J.Smith” as separate records, but they actually refer to the same person. This often happens when multiple databases are integrated from different or distributed sources into a single database, as different sources may use different naming conventions or coding methods for their records. Having multiple references representing the same entities (duplicates) in a database can not only lead to data redundancy, but also cause inaccuracies in the analytical processes such as query processing and knowledge extraction [1]. Therefore, eliminating such redundancy and correctly identifying real-world entities from a database is a crucial step for data integration and data cleaning [2].

Record linkage, also known as entity resolution (ER), data linkage, or duplicate detection, is the process of identifying and matching records that represent the same real-world entity [3]. With the earliest work going back to 1950s, record linkage remains an active research area until today [4]. Its outcome has been applied to many fields, including e-commerce, healthcare, social sciences, and fraud detection [5].

Typically, as shown in Figure 1.1, record linkage includes the following core steps [5, 6]:

1) Cleaning and standardising data into a well defined and consistent form.

2) Dividing data records into blocks using a subset of the attributes as blocking keys and only comparing records within the same block, assuming that records in different blocks are unlikely to match.
3) Generating pair-wise comparison weight vectors of data records using functions that calculate the similarities between attribute values.

4) Classifying weight vectors into matches or non-matches using either supervised or unsupervised classification methods. Matches are pairs of records that refer to the same entity and non-matches are record pairs that refer to different entities.

5) Clustering weight vectors based on classification results such that each record cluster is meant to represent a single real-world entity.

Figure 1.1: Core record linkage steps
1.1 Research Problem

There are two main problems in the record linkage process which we aim to address in this study. The first one is the quality of record linkage, and the second one is the volume of training data needed for record linkage.

The quality of record linkage result is often not satisfactory in practice. One of the reasons is that the data quality cannot be guaranteed. When there is dirty data (i.e. missing values, incorrect values or badly formatted values), the classification and other linkage processes will likely be affected and the result may contain errors that cannot be detected at the time of performing the linkage task [7]. These errors can either be true matches being mislabelled as non-matches (false non-matches) or non-matches being mislabelled as matches (false matches). For example, if we do record linkage on a database given in Table 1.1 that contains four records \( \{r_1, r_2, r_3, r_4\} \), we may end up having two record clusters \( \{(r_1, r_2, r_3), (r_4)\} \) and each of the clusters represents a single real person. However, a possible error of this scenario is that \( r_1 \) and \( r_3 \) actually refer to two different persons and the correct record clusters should be \( \{(r_1, r_2), (r_3), (r_4)\} \). The error might have been introduced in the classification stage where the weight vector of \( (r_1, r_3) \) was misclassified as a ‘match’, and therefore \( r_3 \) was added to \( r_1 \)’s cluster. This kind of error is normally inevitable in most of the record linkage applications. It is beneficial if we can leverage the errors detected by users to refine the classification model incrementally whenever an error is found and repaired, and improve the linkage result accordingly.

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td>J.Smith</td>
<td>22333</td>
</tr>
<tr>
<td>r_2</td>
<td>John Smith</td>
<td>22333</td>
</tr>
<tr>
<td>r_3</td>
<td>Josh Smith</td>
<td>35111</td>
</tr>
<tr>
<td>r_4</td>
<td>David Brown</td>
<td>67882</td>
</tr>
</tbody>
</table>

Table 1.1: An example database table

On the other hand, in the linkage process, many unsupervised and supervised learning techniques have been proposed for classification in the past years [8]. While supervised learning techniques usually outperform unsupervised techniques in terms of classification accuracy and overall linkage quality, they require training data in the form of true matches and true non-matches, which are often not available. Generating training data costs more time and
human labour [5]. Furthermore, record linkage is generally performed or conducted on databases that do not have entity identifiers. The linkage therefore needs to utilise the attributes that contain partially identifying information such as names, addresses, or phone numbers to help determine if two records refer to the same entity [8]. Given two data records and their comparison weight vector, an oracle (e.g., a human annotator) is usually needed to manually classify them into ‘matches’ or ‘non-matches’ for generating training data. As labelling (classifying) cost is often expensive and it is proportional to the number of weight vectors to be labelled, we want to ensure that the overall manual labelling effort can be controlled at an acceptable level.

In regard to the above two outstanding record linkage issues, this report aims to propose and develop a mechanism that can effectively improve record linkage quality while maintaining the burden of labelling cost at a desired level using active learning techniques.

Active learning, also known as ‘query learning’ or ‘optimal experimental design’, is a subfield in machine learning and artificial intelligence [9]. Compared with traditional supervised learning algorithms that use whatever training data that are available to them, an active learning method only selects the most informative training data points and it can improve the model itself by requesting more informative training data iteratively based on the knowledge it gained from each learning iteration [10]. Using active learning approach can often yield in a training set that is small but still sufficient to achieve high linkage accuracy [5].

With active learning techniques, the research problem of this report can essentially be converted to finding the most informative weight vectors to refine the classification model based on user-detected clustering errors. We aim to have the classifier improved incrementally to reach high record linkage accuracy after a certain number of refinement iterations.

1.2 Objectives

The overall aim of this project is to use active learning techniques to develop a record linkage model that can repair errors and thereby improve the quality of record linkage over time while keeping the total labelling effort at a low level. In particular, the following objectives are required:

- Design and develop a record linkage model that is able to accept linkage errors, locate the errors in the weight vector space and repair the errors.
The record linkage model is also able to refine and improve itself whenever an error is detected and repaired using active learning techniques.

Experiment the record linkage model using large real world datasets, evaluate the model and justify the results.

1.3 Contribution

In this project, I have made the following contributions:

- I have developed a record linkage model that can handle user provided linkage errors. The model can look up the errors in its weight vector space and fix it if possible.

- The linkage model can also improve its classification model using the errors iteratively and incrementally.

- I experimented the record linkage model on two real world datasets and the experiment results show that the model can effectively improve the clustering results over time.

In Chapter 2, I will discuss the related work of my research problem. The actual methods and algorithms will be described in Chapter 3. In Chapter 4, I will explain the experiment methodology and evaluate the record linkage model based on the results. Finally, I will conclude with a list of future work in Chapter 5.
Chapter 2

Related Work

Record linkage has been studied extensively in the past years due to its high importance in data integration and data mining fields [7, 8, 11]. The problem was first studied by Newcombe et al. in 1959 [12], since then numerous approaches have been proposed and developed including supervised approaches and unsupervised approaches [2]. One important research area of record linkage nowadays focuses on the use of active learning to minimise the need for labelled data [11]. Active learning techniques allow the classifier to pick a dynamic subset of data records which is small but still sufficient to provide high information gain to the classifier. Another research area focuses on the scalability of record linkage on large databases, where the researchers aim to improve the performance by avoiding the quadratic number of record attribute comparisons [11].

However, most of the related works are focused on preventing errors in record linkage results, rather than repairing record linkage errors that are detected after the linkage process [7]. Since classification is a crucial step for many record linkage approaches, many researchers view record linkage as essentially a classification problem: given a weight vector of similarity scores between the attributes of two data records, classify it as either a ‘match’ or a ‘non-match’ [11]. Therefore, for this study, we want to continue with the first research area to develop a record linkage model that can benefit from the idea of active learning and use it to improve the model itself over time. Specifically, the record linkage model should be able to fix the errors detected in linkage results, and utilise these errors to improve its classification process, and thus the final linkage results.
Chapter 3

Overall Architecture

The overall architecture of the record linkage model is shown in Figure 3.1. The model requires two data inputs: a training dataset, and a test dataset on which record linkage will be performed. The training dataset contains an extra ground truth attribute whose value is used to store the ground truth of the data record. If two records have the same ground truth value, they refer to the same real world entity despite that there might be inconsistencies among other attribute values. The test dataset will not have this ground truth attribute. Note that both the training dataset and the test dataset can consist of records from one or many databases, and the difference of having one or many databases will be explained in Section 3.1.

The test dataset will need to go through blocking process and pair-wise records comparison process. These two processes will generate a set of weight vectors such that each weight vector represents the similarity between two data records. The weight vectors generated by the training dataset will be labelled as either matches or non-matches based on their ground truth identifiers, then these labelled weight vectors will be used as the training data to train the weight vector classifier. After the classifier is trained, it will be passed to the test dataset to classify the weight vectors generated by the test dataset which do not have ground truth identifiers. After the classification process, two sets of weight vectors will be created which are matches and non-matches. The clustering process will use these match vectors and non-match vectors to generate a number of record clusters and each cluster is supposed to represent one single real world entity.

As we discussed in the earlier chapters, the record clusters are likely to contain errors, meaning some clusters may contain data records that refer to
entities that are different from other data records in the same cluster. Therefore, in our record linkage model, errors will be used by the active learning process whenever they are detected. The active learning process consists of three components: user feedback, training data selection process and reclassification process. Also, this record linkage model keeps a weight vector black list which is updated by the active learning process, and the weight vectors stored in the black list will affect the clustering process. Section 3.3 will explain the active learning process more comprehensively.

In the following sections, I will discuss the above processes in more detail.
3.1 Blocking and Record Pairs Comparison

3.1.1 Blocking

The aim of blocking is to group data records into small blocks based on some criteria and the records within the same block will be similar in some way [8]. In the blocking process, a subset of attributes can be used as the blocking keys, and a blocking algorithm is used on these blocking keys to generate a number of blocks for the dataset. Each block has an unique block identifier and a block can contain one or many data records. As similar records would have similar blocking key values, we can assume that records in the same block would have high probability to represent the same real entity, while records in different blocks are unlikely to match. For example, consider the example database table shown in Table 3.1, we can do blocking using the Soundex algorithm on attributes “Last Name” and “Suburb”. The result is shown in Table 3.2.

<table>
<thead>
<tr>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Suburb</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>1224</td>
<td>Andy</td>
<td>Murray</td>
<td>Fayetteville</td>
</tr>
<tr>
<td>r2</td>
<td>1523</td>
<td>David</td>
<td>Vincent</td>
<td>Chapel Hill</td>
</tr>
<tr>
<td>r3</td>
<td>6234</td>
<td>Fred</td>
<td>Meyer</td>
<td>Fayetteville</td>
</tr>
</tbody>
</table>

Table 3.1: An example database before blocking

<table>
<thead>
<tr>
<th>Block ID</th>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Suburb</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>m600f314</td>
<td>r1</td>
<td>1224</td>
<td>Andy</td>
<td>Murray</td>
<td>28301</td>
</tr>
<tr>
<td></td>
<td>r3</td>
<td>6234</td>
<td>Fred</td>
<td>Meyer</td>
<td>28304</td>
</tr>
<tr>
<td>v525c144</td>
<td>r2</td>
<td>1523</td>
<td>David</td>
<td>Vincent</td>
<td>27514</td>
</tr>
</tbody>
</table>

Table 3.2: An example database after blocking

An important benefit of blocking is that it can significantly reduce the number of pair-wise record comparisons. If we do record linkage on two large databases without blocking, we need to compare every single record in the first database to every single record in the second database, which could be very inefficient considering the time and memory space needed to generate all the comparison weight vectors. With proper blocking, the number of pair-wise comparisons can be greatly reduced.
To illustrate this, suppose \( R_1 \) and \( R_2 \) are two databases that both contain a finite number of records and we want to do record linkage on these two databases. The sizes of these two databases are \( s_1 \) and \( s_2 \) accordingly. If there is no blocking mechanism, the number of pair-wise comparisons needed to generate all the weight vectors would be \( s_1 \times s_2 \), i.e. the product of the sizes of two databases. However, suppose we do blocking on both of the databases and they generate two block sets \( B_1 \) and \( B_2 \), \( B_1 = \{ b_1^1, b_1^2, \ldots, b_1^n \} \) and \( B_2 = \{ b_2^1, b_2^2, \ldots, b_2^n \} \), the size of these block sets are \( n_1 \) and \( n_2 \) respectively. Let \( I_{1,2} \) denote the intersection of \( B_1 \) and \( B_2 \), which is a set of common blocks whose identifiers exist in both \( B_1 \) and \( B_2 \). In this scenario, assuming there is no overlapping between the blocks within a dataset, the approximate average number of comparisons \( c_{1,2} \) would be:

\[
c_{1,2} = s_1 \times s_2 \times \frac{|I_{1,2}|}{n_1 \times n_2}
\]

(3.1)

For example, if \( n_1 = 100 \) and \( n_2 = 100 \), the size of the common sets \( |I_{1,2}| = 50 \), the number of comparisons with blocking would approximately be around 1/200 of the number of comparisons without blocking.

### 3.1.2 Record Pairs Comparison

Similarity weight vectors are generated by comparing two data records within the same block and calculating the similarities between a subset of their common attributes using various similarity functions. Suppose we have two example databases as show in Table 3.3, and we want to generate weight vectors for these two databases.

<table>
<thead>
<tr>
<th>DB</th>
<th>Block ID</th>
<th>ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Suburb</th>
<th>Postcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>h400r420</td>
<td>( r_1 ) 5620</td>
<td>James</td>
<td>Hawley</td>
<td>Raleigh</td>
<td>27609</td>
</tr>
<tr>
<td></td>
<td>h651m621</td>
<td>( r_2 ) 6725</td>
<td>Margaret</td>
<td>Hornback</td>
<td>Mooresville</td>
<td>28117</td>
</tr>
<tr>
<td>B</td>
<td>h400r420</td>
<td>( r_3 ) 7152</td>
<td>James</td>
<td>Hill</td>
<td>Raleigh</td>
<td>27607</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( r_4 ) 9845</td>
<td>Bryan</td>
<td>Hill</td>
<td>Raleigh</td>
<td>27601</td>
</tr>
</tbody>
</table>

Table 3.3: Two example databases for calculating similarity weight vectors
The first step is to find the intersection set of blocks between database A and database B. As can be seen from Table 3.3, block “h400r420” is the common block between these two databases. Every record in “h400r420” from database A will need to be compared with every record in “h400r420” from database B. Therefore, the record pairs \((r_1, r_3)\) and \((r_1, r_4)\) will be generated. To get the similarity weight vector of each record pair, we need to calculate the similarity value between each pair of the attribute values. A number of string similarity functions can be used in this scenario, including edit distance, q-gram similarity, etc. For example, the attributes “First Name”, “Last Name”, “Suburb” and “Postcode” together with the q-gram similarity can be used to generate the weight vectors, thus the weight vector of \((r_1, r_3)\) will be \([1.0, 0.167, 1.0, 0.667]\). Note that the value of each weight in a weight vector is between 0 and 1, with 0 meaning two attributes are totally different and 1 meaning they are identical or nearly identical based on the similarity function it uses.

Also, as mentioned earlier, our record linkage model supports one or more training or test datasets (databases) as data input. There is a slight difference between generating weight vectors on one dataset and multiple datasets. If there is only one dataset, each block that contains more than one record will need to be compared with the block itself to generate the weight vectors, whereas if there are more than one dataset, a block of records in each dataset will need to be compared with all the datasets including itself. For example, if there are N databases, the total number of database pairs is \(\left(\frac{N^2 - N}{2}\right)\) and for each database pair, all records from the first database block will need to be compared with all records from the second database block if two blocks have the same block identifier.

### 3.2 Classification and Clustering

After the record pair comparison step, all the generated weight vectors from the test dataset will be passed to the classifier and they will be classified as either matches or non-matches. The classifier is trained using the labelled weight vectors generated by the training data. As we have ground truth identifier in the training data, we can label each training weight vector by checking if two records of that weight vector have the same ground truth identifier. In previous studies, a variety of classifiers have been used for record linkage including decision trees, SVMs, and k-nearest neighbours [8].
In the implementation of our record linkage model, I specifically used SVM as the classifier.

Our record linkage model keeps a list of weight vector sets as the weight vector space. Each set in the list corresponds to a subspace in the weight vector space. In the classification process, one can choose to either classify the whole space (i.e. all available weight vectors) or a subspace (i.e. a weight vector set) of the weight vector space. If a subspace is to be classified, it will first be removed from the weight vector space list, and the weight vectors in it are added to either a new match vector set or a new non-match vector set based on the classification results. Then these two new sets are pushed to the weight vector space list.

The clustering process takes place after the classification process. Given a set of classified weight vectors $W$, the clustering algorithm will generate a set of clusters each containing two or more data records, and each cluster is supposed to represent a single real world entity.

As shown in Algorithm 1, the clustering algorithm will first initialise an empty list $C$ which will be used to store all the clusters (line 1). For each of the classified weight vectors, it will first check if it is on the black list or not (line 4 to 8). The black list will be used to hold weight vectors that are either outliers or not fixable in the active learning process. The idea behind the black list will be explained in Section 3.3. Suppose a vector is on the black list, and the vector does not have a label already, the clustering algorithm will ask an oracle (e.g. human annotator) to manually check if two records of the vector actually represent the same real entity and store the result in the black list. If they do, it will return 1 to the variable $m$, or 0 if they are non-matches (line 5). If the weight vector is not on the black list, the algorithm will check the classification result and return it to variable $m$ (line 7).

If the weight vector $w$ is a match (i.e. $m$ equals 1, line 9), the algorithm will loop through all the clusters $c$ within the cluster list $C$. Each cluster is a set that stores the IDs of data records. If $r_1$ or $r_2$ that belong to $w$ is found in a cluster (line 12), these two data records will be added to this cluster (line 13). If $r_1$ or $r_2$ is not found in any of the clusters (line 17), a new cluster $m$ will be created and these two records will be added into this cluster (line 18 to 19). Afterwards, this new cluster $m$ will be appended to the cluster list $C$ (line 20). After looping through all the weight vectors, we will have a list of clusters, but it is possible that some of the clusters are identical, meaning they have the same records in them. We need to remove
Algorithm 1 Clustering algorithm

Input:
A list of weight vectors: \(W\)
A black list of weight vectors: \(B\)
A dictionary that stores the classification result of all weight vectors: \(P\)

Output:
A list of clusters, each represent a single entity: \(C'\)

1: \(C = []\)
2: \(C' = []\)
3: for weight vector \(w \in W\) do
4:   if \(w \in B\) then
5:     \(m = \text{CheckGroundTruth}(w)\)
6:   else
7:     \(m = P[w]\)
8:   end if
9:   if \(m == 1\) then
10:      added = False
11:      for \(c \in C\) do
12:         if \(r_1 \in c\) or \(r_2 \in c\) then
13:            \(c = c \cup \{r_1\}\), \(c = c \cup \{r_2\}\)
14:            added = True
15:         end if
16:      end for
17:      if added == False then
18:         \(m = \{\}\)
19:         \(m = m \cup \{r_1\}\), \(m = m \cup \{r_2\}\)
20:         \(C = C \cup m\)
21:      end if
22:   end if
23: end for
24:
25: for \(c \in C\) do
26:   if \(c \notin C'\) then
27:     \(C' = C'.add(c)\)
28:   end if
29: end for
30: return \(C'\)
the duplicate clusters and only keep one in the cluster list (line 25 to 29). Finally, the cluster list $C'$ that does not contain duplicate clusters will be returned.

After the clustering process, a list of clusters is generated and each cluster represents a real world entity. Note that in the original dataset, only data records that are duplicated will correspond to a record cluster, singleton records do not correspond to any record clusters.

### 3.3 Active Learning

As mentioned in the first chapter, many record linkage applications are far from perfect in practice [7]. Errors can be introduced in the classification process of record linkage, and these errors may have negative impact on the final clustering results. Therefore, it is often inevitable for a cluster to have records referring to different real world entities.

Our active learning model consists of three main components: user feedback, training data selection process and reclassification process. User feedback provides the model with the information of errors that are found in clustering results. With the errors, the active learning process will locate the weight vector that caused the error in the weight vector space, and select informative training data from that particular space, retrain the classifier and reclassify that weight vector space. The newly classified weight vectors will be sent to the clustering process together with weight vectors from other weight vector spaces to generate new clusters.

Whenever an error is detected by users, it often implies that there is inaccuracy in the record linkage model. For example, suppose we have a cluster that contains $\{r_1, r_2, r_3\}$, where $r_3$ might have been introduced to this cluster by weight vector $(r_1, r_2)$ or $(r_1, r_3)$. If $r_3$ refers to an entity that is different from $r_1$ and $r_2$, it implies that either $(r_1, r_2)$ or $(r_1, r_3)$ is a false match, and we can trace it back to our classification model to find out why it was incorrectly classified. A linkage model is desired if it can fix errors detected by users and utilise the errors to refine its classification model to achieve better clustering result incrementally.

Figure 3.2 illustrates a small example of active learning used to improve classification result. After initial classification, weight vectors are classified into a match vector set or a non-match vector set. A vector set corresponds to a subspace as shown in Figure 3.2. It is possible that false matches exist
Figure 3.2: Classification with active learning
in the match vector space or false non-matches exist in the non-match vector space.

![Diagram: Generating possible erroneous vectors](image)

**Figure 3.3: Generating possible erroneous vectors**

When an error is found in the clustering result, the first step is to generate all the possible erroneous weight vectors that contain the error for that cluster. For example, as shown in Figure 3.3, if \( r_4 \) is detected that it should not belong to a cluster (false match error) that contains \( r_1, r_2 \) and \( r_3 \), or it should belong to that cluster (false non-match error), a set of possible erroneous weight vectors \((r_1, r_4), (r_2, r_4), \) and \((r_3, r_4)\) will be generated. These weight vectors will be searched in the vector space. As shown in Figure 3.2, if an error vector is found in a vector subspace, it implies that there may be more errors in that particular space, thus the space will need to be further classified. After several active learning iterations, we can see from step 4 in Figure 3.2 that most weight vectors are correctly classified, and thus it can improve the quality of clustering.

Algorithm 2 is the main active learning algorithm. The algorithm takes a list of weight vector spaces \( \mathbf{V} \) as input, each space \( v \) contains a number of weight vectors that are classified as either matches or non-matches. Given an error vector, the algorithm first searches all the vector spaces to see if the error vector exists in any of the spaces (line 3). If it does not exist in the whole vector space, it implies that the error cannot be fixed through classification thus it will be disregarded. If it is found in vector space \( v_i \), the
Algorithm 2 Active learning algorithm

\textbf{Input:}
A list of weight vector spaces: \( V \)
A list of training vector spaces: \( T \)
A black list of weight vectors: \( B \)
An oracle: \( \text{Oracle} \)
A classifier: \( \text{Classifier} \)
An erroneous weight vector: \( e \)

\textbf{Output:}
A set of newly classified weight vectors: \( C \)

1: \( C = \{\} \)
2: \textbf{for} vector space \( v_i \in V \) \textbf{do}
3: \quad \textbf{if} \( e \in v_i \) \textbf{then}
4: \quad \quad \( S = \text{MixedSelect}(v_i, e) \)
5: \quad \quad \textbf{if} \( S == \emptyset \) \textbf{then}
6: \quad \quad \quad B.append(e)
7: \quad \quad \textbf{else}
8: \quad \quad \quad \( S^m, S^n = \text{Oracle}.\text{label}(S) \)
9: \quad \quad \quad \( S^m = S^m \cup T^m_i, S^n = S^n \cup T^n_i \)
10: \quad \quad \quad \text{Classifier}.\text{train}(S^m, S^n)
11: \quad \quad \textbf{if} \( |W^m| = 0 \) or \( |W^n| = 0 \) \textbf{then}
12: \quad \quad \quad B.append(e)
13: \quad \quad \quad v_i.pop(e)
14: \quad \quad \textbf{else}
15: \quad \quad \quad \( V.pop(i), V.append(W^m), V.append(W^n) \)
16: \quad \quad \quad \( T.pop(i), T.append(S^m), T.append(S^n) \)
17: \quad \quad \quad \textbf{end if}
18: \quad \quad \textbf{end if}
19: \quad \textbf{end if}
20: \textbf{end for}
21: \textbf{return} \( C \)
error \( e \) together with the space \( v_i \) will be passed as parameters to a selection function called \( \text{MixedSelect}(\cdot) \) (line 4).

The \( \text{MixedSelect}(v, e) \) function takes two parameters as input: a vector space \( v \) and an error vector \( e \). The function selects and returns predefined \( k \) number of weight vectors as training vectors. \( k \) can be set as a constant or a fraction of the size of the vector space \( v \). Based on the error \( e \), the function will first calculate the distance between the error vector and every single weight vector within \( v \). Then the distances are sorted and a small number of nearest weight vectors (i.e., smallest distances) will be selected first as the seed samples to calculate the purity. The number of seed samples \( N \) are much smaller than \( k \). The purity is defined as follows:

- If the error vector \( e \) is a false match:
  \[
  \text{purity} = \frac{|N^m|}{|N|},
  \]
  (3.2)

- If the error vector \( e \) is a false non-match:
  \[
  \text{purity} = \frac{|N^n|}{|N|},
  \]
  (3.3)

where \( N^m \) denotes the true matches of \( N \) and \( N^n \) denotes the true non-matches of \( N \). If the purity is high, meaning that most of the surrounding weight vectors of the error vector \( e \) have opposite labels, it is sufficient to regard the error vector \( e \) as an outlier and the \( \text{MixedSelect}(v, e) \) function will stop and return an empty set. If the purity is not high, the function will select the remaining farthest weight vectors based on the distances so that the total number of weight vectors selected equals to \( k \). These weight vectors are stored in a set and the set will be returned by the \( \text{MixedSelect}(v, e) \) function.

Continuing from line 5 of the active learning algorithm, if the \( \text{MixedSelect}(v_i, e) \) function returns an empty set, the error weight will be considered as an outlier and added to the black list \( B \), and it will also be removed from the vector space \( v_i \). As explained earlier, the labels of the vectors in the black list will be manually checked and stored in the clustering process.

If the \( \text{MixedSelect}(v_i, e) \) does return a non-empty set of weight vectors (line 8), these weight vectors will be manually labelled by an oracle (line 10) and they will be used as the training data together with previous training
data that fall in that subspace to train the classifier (line 10 and 11). Then the classifier will classify the weight space $v_i$. Two sets of weight vectors $W^m$ (match), $W^n$ (non-match) will be generated (line 12). Sometimes it is possible that either $W^m$ or $W^n$ is empty, meaning the vector space $v_i$ is not dividable and the error cannot be fixed (line 13), and therefore the error vector $e$ will be added to the black list and removed from the vector space. If $v_i$ is dividable, the original space $v_i$ will be removed from the vector space list $V$ and two new generated vector spaces $W^m$ and $W^n$ will be added to $V$ (line 17). The training vector space $T$ is updated in a similar way (line 18). The output of this algorithm is a set of newly classified weight vectors $C$ that contains $W^m$ and $W^n$ (line 19 and 24).

In an iteration of active learning process, an error vector can either be fixed, not fixed or added to the black list. If it is fixed, meaning the new classification result of the error vector is opposite to the original result, and the error vector is placed into the correct vector subspace. The record linkage model will then enter the clustering process to cluster all the weight vectors based on their new labels. Similarly, if it is added to the black list, the weight vectors will be re-clustered and the clustering process will use the ground truth of the weight vectors stored in the black list. If it is not fixed, meaning the new classification result is the same with the original result, the error vector is incorrectly classified and placed into a wrong vector subspace, and therefore the active learning process will repeat with the same error vector until it is fixed or added to the black list.

Therefore, the active learning process can essentially divide the weight vector space iteratively based on the errors found, and it only selects training data that are locally informative and representative for a specific weight vector subspace. After several iterations of active learning process, we can expect higher classification accuracy, and thus better clustering quality compared to the initial result.
Chapter 4

Experiments

The chapter presents the results of the experiments conducted in this study. The experiments were to evaluate the performance of the proposed record linkage model developed in this project.

4.1 Experiment Setup

The prototype of the proposed record linkage model is developed in Python based on *Febrl*, which is an open source record linkage system developed by Dr. Peter Christen from the Australian National University \(^1\). The implementation of the record linkage model used some functions of the *Febrl*, such as comparison functions, blocking functions and phonetic encodings.

The experiment setup is shown as follows:

4.1.1 Software and Hardware

<table>
<thead>
<tr>
<th>Operating System</th>
<th>ubuntu 16.04 LTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS Type</td>
<td>64-bit</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Python 2.7.12</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Core i5-3470 CPU at 3.20 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>8GBytes RAM</td>
</tr>
</tbody>
</table>

Table 4.1: Software and Hardware Information

\(^1\)Available from: https://sourceforge.net/projects/febrl/
4.1.2 Datasets

The experiments used two real-world datasets: the CORA publication dataset\(^2\) and the North Carolina Voter Registration (NVCR) dataset\(^3\). The CORA publication dataset contains 1878 records of machine learning publications, each of the records contains the publication’s author, publication title, name of the conference, volume number and date. The NCVR dataset contains registration information of voters from North Carolina, where each of the records contains a voter’s information including his/her given name, surname, suburb and postcode. Note that both datasets have ground truth stored as attribute values, meaning the record linkage program can look up the ground truth values of records to check if they refer to the same entity. The ground truth values are also used to calculate performance measures.

The NCVR dataset used in the experiments has been modified, only a portion of the original dataset records were extracted and experimented. Also, there are two versions of the NCVR datasets, one is corrupted and one is non-corrupted. The corrupted NCVR dataset contains 2 random corruptions (edits and inserts, phonetic variations and misspellings) in each record [13], while the non-corrupted one does not contain any corruptions, meaning that two non-corrupted records are identical if they have the same ground truth values.

Also, as the NCVR dataset is relatively large, I did blocking on it otherwise it would be impossible to generate all the weight vectors. The blocking keys of the NCVR dataset are “surname” and “suburb”. For the CORA dataset, as there are only 1878 records, I did not do blocking on it thus each record is compared with all other records to generate the weight vectors. Based on the blocking keys, **pairs completeness** can be calculated. Pairs completeness is defined as the ratio of the matched records pairs found in the reduced comparison space, to the number of matched record pairs in the entire comparison space\([14]\). In our case, the pairs completeness equation is defined as

\[
P C = \frac{|C_m|}{|C|},
\]

where \(|C_m|\) is the number of clusters that have 2 or more records, and the records in the clusters have the same block identifier. \(|C|\) is the total number

\(^2\)Available from: https://www.cs.umass.edu/mccallum
\(^3\)Available from: ftp://alt.ncsbe.gov/data
of true clusters which have 2 or more records. A high pairs completeness value implies that most of the duplicate records are blocked into the same block and have the same block identifier. As when generating the weight vectors, we only compare records that have the same block identifier, high pairs completeness means that there is a higher chance that all necessary weight vectors needed to form a complete cluster will be generated, while low pairs completeness means that it is likely that some true match weight vectors are missing from the weight vector space. The clustering results are affected by pairs completeness.

The initial training data were generated using 5% of the dataset. The training weight vectors are labelled automatically using their ground truth values.

Some key characteristics of the datasets are shown in Table 4.2

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of records</th>
<th>Number of weight vectors</th>
<th>Number of clusters</th>
<th>Min. size of clusters</th>
<th>Pairs completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCVR corrupted</td>
<td>100,000</td>
<td>314,663</td>
<td>5,000</td>
<td>2</td>
<td>0.5346</td>
</tr>
<tr>
<td>NCVR non-corrupted</td>
<td>100,000</td>
<td>350,766</td>
<td>5,000</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>CORA</td>
<td>1,878</td>
<td>1,764,381</td>
<td>120</td>
<td>2</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 4.2: Software and Hardware Information

### 4.2 Experiment Results

To evaluate the performance and effectiveness of the record linkage model, I used measures including Recall, precision and F-measure. These measures were gathered from both weight vector classification results and clustering results.

In the experiments, to simulate real-world user feedback, all false match vectors and false non-match vectors in the weight vector space were collected first, and I used a random number generator to randomly select errors from the false match vector list and false non-match vector list. Afterwards, these randomly generated errors were passed to the active learning process to execute.
Some parameters used in the experiments were: purity of the \textit{MixedSelection()} function $p = 0.8$, number of training weight vectors selected in each active learning process $k$: 1\% of the number of the weight vectors in that particular vector space. For the classifier, I used the SVM classifier from the Scikit-learn library\textsuperscript{4} with \textit{RBF} kernel and default parameter settings.

With each dataset, I ran the program a number of times and for each time, 10 randomly generated erroneous weight vector were given to the program to repair. I collected the measures for both classification results and clustering results after each active learning iteration. I also generated a summary graph for all the experiment runs. The summary graph compares the final measures with the baseline measures, which are the original measures before any active learning processes are applied. The results are analysed as follows:

For the non-corrupted NCVR dataset, as the records are identical if they represent the same real world entity, the pairs completeness of this dataset is 1. We can assume that the classification results and clustering results would already be good without active learning process and the experiments proved our assumption. As can be seen from the summary graphs of Figure 4.1 and Figure 4.2, given that the initial measurement values are all over 0.9, we can still see improvements in all measures for both classification results and clustering results as they reach to the maximum value 1.0 after several active learning iterations.

For the corrupted NCVR classification, the pairs completeness of this dataset is 0.5346, which means the recall and precision of the clustering results are limited by it. We can see from the summary graphs of Figure 4.3 and Figure 4.4 that the F-measure increases significantly for both classification results and clustering results, but we are not able to achieve better results. Using a different blocking strategy for this dataset might improve the results at the cost of more weight vectors.

For the CORA dataset, the overall results are good. But we can see from Figure 4.5 and Figure 4.6 that sometimes the measures drop at the first several active learning iterations, and come back to a steady level afterwards. This is understandable as the weight vector subspaces are large initially, many records may be misclassified. As we divide the subspaces further and further, it yields smaller subspaces with more records being correctly classified, thus the overall measures are improved.

Also, the variations between each experiment run are caused by the errors

\textsuperscript{4}\textit{Available at: http://scikit-learn.org/stable/}
being identified and submitted to the model to repair. The errors determine how the weight vector space is classified and divided, they hence have significant impact on the quality of our record linkage model. As we generated erroneous weight vectors randomly, we could not guarantee that the errors were generated in the most meaningful way, and it is why there are variations between each experiment run on the same dataset.
Figure 4.1: Classification measures for non-corrupted NCVR dataset
Figure 4.2: Cluster measures for non-corrupted NCVR dataset
Figure 4.3: Classification measures for corrupted NCVR dataset
Figure 4.4: Cluster measures for corrupted NCVR dataset
Figure 4.5: Classification measures for CORA dataset
Figure 4.6: Cluster measures for CORA dataset
Chapter 5
Conclusion and Future Work

In this project, I have developed a record linkage model using active learning techniques. The model can accept errors detected by users in the record clusters and use the errors to improve its classification model as well as the clustering results iteratively over time. The experiments conducted on the CORA and NCVR datasets validated that the model can effectively improve classification results and clustering results, but the improvement is limited by other factors which are beyond the scope of this project.

Therefore, as for future work, there are a number of directions one may continue to explore:

1) As can be seen from the experiments, clustering results are restricted by the quality of blocking, which is measured by pairs completeness. It is beneficial if one can investigate more effective blocking techniques, and incorporate such effective techniques into this record linkage model to achieve better clustering results.

2) The weight vector space is divided using user provided errors. Therefore, the selection of the errors and the order of the errors that are provided to the record linkage model determine how the weight vector space is divided. One may investigate how to arrange the order of errors submitted to the model so that the weight vector space is divided in the most appropriate way.

3) Investigate the configurations of the classifier. In the experiments, I used the SVM classifier with RBF, Poly and Linear kernels with default parameters (the experiment results of Poly and Linear kernels are not
included in this report as they are outperformed by the RBF kernel). However, one may continue to explore how to effectively configure the classifier to achieve the best result.
Bibliography


Appendices
Appendix A

Project Description
1 Project Title
Repairing Data Linkage

2 Project Description
The goal of the project is to use active learning techniques to improve the quality of data linkage models. The specific tasks are:

1. Conduct a literature review on data linkage and active learning techniques;

2. Using support vector machines (SVM) to build a learning model for data linkage, which can group records into different clusters such that only records in the same cluster refer to the same real-world entity;

3. Develop active learning methods to actively select samples for repairing errors in linkage results and for improving the performance of the developed SVM learning model;

4. Conduct experiments on two data sets (CORA and NCVR), and evaluate the effectiveness and efficiency of the developed methods;

   • The CORA data set contains 1,878 machine learning publications and is publicly available together with its ground truth.
   • The NCVR data set is a public voter registration data set from North Carolina.

5. Write up a project report.

3 Learning Objectives
On the completion of the project, the following learning objectives are expected to achieve:

• Have a good understanding on the literature of data linkage and active learning techniques;

• Develop an active learning based method to resolve errors in linkage results and improve the quality of data linkage models;

• Be able to conduct experiments and analyze their results;

• Be able to effectively communicate about data and project results understandably, using adequate indicators, tables, and graphs.
Appendix B

Project Contract
INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

<table>
<thead>
<tr>
<th>UniID:</th>
<th>u4943054</th>
</tr>
</thead>
<tbody>
<tr>
<td>SURNAME:</td>
<td>FENG</td>
</tr>
<tr>
<td>FIRST NAMES:</td>
<td>CHONG</td>
</tr>
<tr>
<td>PROJECT SUPERVISOR (may be external):</td>
<td>Qing Wang, Dinusha Vatsalan</td>
</tr>
<tr>
<td>COURSE SUPERVISOR (a RSCS academic):</td>
<td>Weifa Liang</td>
</tr>
<tr>
<td>COURSE CODE, TITLE AND UNIT:</td>
<td>COMP 8715  12UNITS</td>
</tr>
<tr>
<td>SEMESTER</td>
<td>□ S1  □ S2  YEAR: 2016</td>
</tr>
</tbody>
</table>

PROJECT TITLE:

See attached.

LEARNING OBJECTIVES:

See attached.

PROJECT DESCRIPTION:

See attached.

Research School of Computer Science

Form updated Jun-12
ASSESSMENT (as per course's project rules web page, with the differences noted below):

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: name style:</td>
<td>45%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e.g. research report, software description...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artefact: name kind:</td>
<td>45%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(e.g. software, user interface, robot...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation:</td>
<td>10%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

............................... 15/07/2016
Signature Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

............................... 15/07/2016
Signature Date

REQUIRED DEPARTMENT RESOURCES:

SECTION C (Course coordinator approval)

.........................................................
Signature Date

SECTION D (Projects coordinator approval)

............................... 15/7/16
Signature Date

Research School of Computer Science

Form updated Jun-12