Real-time collective entity resolution

Jingwei Zhang
u5285681

Supervisor: Dr. Qing Wang

COMP8780 Information and Human Centered Computing Project
Research School of Computer Science
Australian National University
Semester 2, 2013
Acknowledgements

I would like to thank my supervisor Dr. Qing Wang, for her ideas, support, encouragement and feedback.

I would like to thank my families, for everything.
Abstract

Some recent works studied collective entity resolution. However, collective entity resolution has a high computational complexity. In practice, a real-time collective entity resolution approach with fast response and low time cost is often desired. In this project, I develop a real-time collective entity resolution approach. This approach uses a variety of methods and measures to improve the effectiveness and efficiency of entity resolution. I perform several experiments to evaluate my approach. The dataset used by my experiments is from Scopus, a well-know bibliographic database. Instead of generating a single resolution result, my approach can provide three kinds of results: a fuzzy result with high recall, a refine result with high precision and a balanced result with high precision and high recall. The results of the experiments show that my approach has achieved its desired goals.
# Table of Contents

1. Introduction ............................................................................................................. 4  
   1.1 Background .................................................................................................. 5  
   1.2 Motivation .................................................................................................. 6  
   1.3 Objectives and contributions ..................................................................... 7  
2. Related works ...................................................................................................... 9  
3. Methodology ........................................................................................................ 11  
   3.1 Attribute-based blocking ........................................................................... 11  
   3.2 Candidate representative determination .................................................. 12  
   3.3 Attribute-based entity resolution ............................................................... 13  
   3.4 Naïve relational entity resolution ............................................................... 14  
   3.5 Collective entity resolution ...................................................................... 15  
   3.6 Real-time threshold set determination ...................................................... 18  
4. Experiments ......................................................................................................... 20  
   4.1 Blocking experiment .................................................................................. 21  
      Experiment 4.1.1 Blocking thresholds evaluation ....................................... 21  
   4.2 Attribute-based entity resolution experiment .......................................... 22  
      Experiment 4.2.1 Attribute weighting ......................................................... 23  
      Experiment 4.2.2 Attribute similarity modification ................................... 24  
      Experiment 4.2.3 Attribute-based ER evaluation ....................................... 25  
   4.3 Naïve relational ER and collective ER experiment ...................................... 26  
      Experiment 4.3.1 Naïve relational ER evaluation ......................................... 27  
      Experiment 4.3.2 Collective ER evaluation .................................................. 28  
   4.4 Real-time threshold set evaluation ............................................................. 30  
   4.5 Complexity analysis and run time statistics .............................................. 31  
5. Conclusion and future works .............................................................................. 33  
Reference list ............................................................................................................ 35  
Appendix .................................................................................................................. 37
1. Introduction

Data never sleeps, every minute a great amount of data is being generated. By the end of 2011, there was more data produced in that single year than in all the rest of human history combined [1]. This is the age of data, and people are inundated within it.

On the other hand, once analyzed appropriately, data could be a goldmine. Techniques for mining data are desperately needed. “Finding ways to make data useful is one of the great challenges, and opportunities, of the network age,” OCHA’s Network Age Report states [1].

Entity resolution is the problem of determining which records in a dataset refer to the same entities [2]. To be more specific, the entity resolution includes the identification problem and the disambiguation problem. The identification problem is to find the same entity from a set of similar records. As for the disambiguation problem, the task is to figure out different entities from a set of records having same attributes.

Why is entity resolution a problem? In real world, information is widely distributed in various sources, and one entity can be represented in different ways. Meanwhile, one representation can be referred to different entities as well. Therefore, redundancy records are very common in real world data sources, and entity resolution is the task to resolve it.

For a better explanation of entity resolution, in Figure 1.1, there are three different entities, Joshua Zhang, Jing Zhang, and Jingwei Zhang. There are several ways of referring to them, in Figure 1.1, ‘J. Zhang’, ‘Josh Zhang’ and ‘Joshua Zhang’ both refer to the same entity. This may be caused by different reasons, such as different spelling variations, name changes, different names in different organizations or data entry errors [3].

Different entities can have the same initials as well, which means one initials may
refer to different entities. For example, the three entities in Figure1.1 both share the initials J. Zhang. This is a common situation in many domains that use initials to store entities’ names.

Many common names are used by over thousands of people. According to howmanyofme.com, there are 45,806 people named ‘John Smith’ in the United States [4]. The problem is more serious in China, for example, around 300,000 people have the same name ‘Wei Zhang’.

1.1 Background

The history of computer-assist entity resolution started from 1950s [5]. During that period, entity resolution was based on ad-hoc heuristic methods. In 1959, Newcombe and Kennedy laid the probabilistic foundations of modern record linkage theory (also entity resolution) [6].

Along with the extraordinary explosion in big data, the studies on entity resolution become an important area. People are inundated with more and more data that needed to be integrated, aligned and matched before further data mining processing [5]. The techniques used to solve entity resolution problems has been improved, and both traditional attribute-based ER techniques and newly collective ER techniques are been considered.

The attribute-based ER techniques concern the similarity of entities’ attributes, e.g. names and addresses. The similarity measure for the comparison of attributes is based on their string similarity.

One major issue for attribute-based ER techniques is the existence of incomplete data. Real world data contains a lot of missing values, and attribute-based ER techniques perform badly under that situation. Another situation when attribute-based ER techniques perform badly is resolving similar but distinct entities, e.g. couples or brothers with same initials or same address [7].

Some recent works considered the relational similarity in entity resolution, which was referred as relational entity resolution [7].

For a better explanation of relational entity resolution, this report uses an example from Getoor and Machanavajjhala’s paper [7]. In Figure 1.2 (in next page), there are three entities J. Doe, Jon Doe and Jonathan Doe, and each of them has its own relationship. These entities’ relationships are represented as a graph. In the graph, a node indicates an entity and a hyper-edge between an entity and its related entities (or neighbors) indicates a relationship among them.
The relational ER techniques regard the graph as an entity graph and resolve it. Figure 1.3 shows the resolved entities’ relationships of Figure 1.2. It shows that J. Doe, Jon Doe and Jonathan Doe are referred to the same entity.

However, since the naïve relational ER techniques do not actually resolve related entities, they get poor results in situations like most entities’ attributes are similar. It has been stated by Getoor and Bhattacharya that when resolving entities, their related entities need to be resolved as well [7]. That led to collective entity resolution, where the entities are not resolved independently any more.

I still use the example in Figure 1.2 to explain the concept of collective ER. To resolve J. Doe and Jonathan Doe, collective ER resolves two Jason Doe as well, because J. Doe and Jon Doe are both related to Jason Doe. If the collective ER result shows that those two Jason Doe are the same entities, it can be inferred that J. Doe and Jon Doe are the same entities. On the contrary, if the collective ER result indicates that two Jason Doe are two different entities, it can be inferred that J. Doe and Jon Doe are different entities as well.

### 1.2 Motivation

Data is widely dispersed over different sources, and that makes data integration and linkage necessary. This is one of the major motivations of entity resolution studies.
Entity resolution techniques can be used to deal with many other data problems, like handling redundant data or detecting data conflicts [3].

Rather than being part of the data integration process, entity resolution has significant implications in some other domains as well, such as advertising, online shopping, database management and network science [5].

One of the newly techniques of entity resolution is the collective ER. However, collective ER techniques have their shortness. Since each record’s collective ER result also affects other records’ collective ER results, the collective ER techniques need traverse all records. As a result, the computational complexity for collective ER techniques is high. In situations like querying entities or updating records in database, people often prefer a real-time result of entity resolution.

Another point is that results of collective ER techniques are resolved offline. Given the fact that data is updating frequently nowadays, results from collective ER can be out of date quickly. For instance, many popular bibliographic databases like Scopus, IET Electronic Library and Science Citation Index, update their data less than once a week.

It can be concluded that the collective entity resolution is not very practical in some situations and a real-time collective entity resolution approach conforms to the actual needs of entity resolution users nowadays.

1.3 Objectives and contributions

The objectives of this project are as follows:

1) To develop a real-time collective entity resolution approach.
2) To implement some techniques into this approach to improve the efficiency and effectiveness of entity resolution.
3) To analyze experimental results of the developed approach.

There are two major challenges for this project. Firstly, it is hard to find ideal datasets for my experiments, since my experiments need the original datasets for testing, as well as their gold-standard sets for evaluation. Secondly, my approach is hard to be domain-independent. The similarity measures, weighting measures and name patterns used in my approach are closely related to the features of bibliographic domain.

The contributions of this project are as follows:

1) This project developed a real-time collective ER approach based on the framework of collective ER proposed by Getoor and Machanavajjhala [7].
2) This project implemented the developed approach and conducted several
experiments to evaluate the developed approach’s performance on a bibliographic dataset.

The rest of this report is organized as follows. In Chapter 2, this report gives a brief introduction on related works about entity resolution, including their features and differences comparing to this project. In Chapter 3, this report explains the methodology of the developed approach. In Chapter 4, this report presents several experiments for the developed approach, and evaluates its efficiency and effectiveness. Finally, the conclusion of this project and some advises for further works are presented in Chapter 5.
2. Related works

Entity resolution is an area that attracts a lot attention for studies. Singla and Domingos proposed a solution to the entity resolution problem based on Markov logic [2]. Their work combines first-order logic and probabilistic graphical models by attaching weights to first-order formulas, and viewing them as templates for features of Markov networks.

Some studies focused on relational entity resolution. Getoor and Bhattacharya proposed a novel relational clustering algorithm [7]. It uses both attribute and relational information for determining the underlying domain entities.

One of the problems for relational entity resolution is its high computational complexity. Techniques for reducing the complexity of relational entity resolution are needed. Steven Euijong Whang, etc. proposed an iterative blocking framework where the entity resolution results of blocks were reflected to subsequently processed blocks [8]. Their blocks are iteratively processed until no block contains any more matching records.

Several studies incorporated graph patterns into the entity resolution techniques. Lingfeng Niu etc. proposed an unsupervised entity resolution algorithm that utilized both the attributes and graph of the entity [9]. Their approach propagates the similarity of each entity pair based on the entity graph in the similar way as in PageRank of Google.

The entity resolution techniques sometimes need external domain knowledge to improve their performance, and this makes entity resolution techniques hard to be domain-independent, thus some studies works on domain-dependence of entity resolution. Zhaoqi Chen, etc. presented an entity resolution approach for making it self-adaptive to the underlying data, which minimizes the required participation from the domain-analyst [10].

There are several studies focused on the scalability of entity resolution in real-time. Christen and Gayler investigated an inverted indexing technique, and employed it for real-time entity resolution [11]. They presented two variations of the traditional inverted index approach, aimed at facilitating fast approximate matching.

To conform the studies of entity resolution on practical needs, some studies focused on query-time entity resolution. Getoor and Bhattacharya developed new algorithms for collective ER to answer entity resolution queries at query-time [12]. Their approach first extracts the related records for a query using two novel expansion operators, and then resolves the extracted records collectively.
The approach developed by this project builds on the framework of collective entity resolution proposed by Getoor and Bhattacharya, which combines attributed-based ER and relational ER together [7]. However, this project significantly differs from their works in following aspects.

Firstly, instead of collectively resolving all records, my approach removes all irrelevant records that should not be resolved using a blocking approach and produces a candidate set for further resolution.

Secondly, rather than weighting attributes based on their frequency of occurrence in attribute-based ER, my approach builds up an attribute-weighting table to rank relevant attributes and assign them with different weights.

Thirdly, my approach uses attribute missing value percentages to adjust attribute’s weights in real-time.

Fourthly, instead of generating one result each time, my approach sets up a threshold set consisting of three thresholds, which can produce three different results: a fuzzy ER result, a refine ER result and a balanced ER result.

Finally, my approach determines the three thresholds based on the combination similarity and the gap monitor in the collective ER process. Therefore, the threshold set is determined in real-time.
3. Methodology

The main steps of my real-time collective entity resolution approach are listed in Table 3.1, and this chapter explains each step in the following sections.

<table>
<thead>
<tr>
<th>step</th>
<th>process</th>
<th>method</th>
<th>result</th>
<th>experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Attribute-based blocking</td>
<td>Name similarity comparison</td>
<td>Candidate set</td>
<td>4.1.1</td>
</tr>
<tr>
<td>2</td>
<td>Candidate representative determination</td>
<td>Manually queries refinement</td>
<td>Candidate representative</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Attribute-based ER</td>
<td>Weighting &amp; attribute similarity comparison</td>
<td>Clustering based on attribute similarity</td>
<td>4.2.1, 4.2.2, 4.2.3</td>
</tr>
<tr>
<td>4</td>
<td>Naïve relational ER</td>
<td>Naïve relational similarity comparison</td>
<td>Clustering based on naïve relational similarity</td>
<td>4.3.1</td>
</tr>
<tr>
<td>5</td>
<td>Collective ER</td>
<td>Neighborhood similarity comparison &amp; hierarchical clustering</td>
<td>Clustering based on combination similarity</td>
<td>4.3.2</td>
</tr>
<tr>
<td>6</td>
<td>Real-time threshold set determination</td>
<td>Combination similarity &amp; gap observation</td>
<td>Three kinds of result based on the threshold set</td>
<td>4.4.1</td>
</tr>
</tbody>
</table>

Table 3.1 Main steps of real-time collective entity resolution approach

3.1 Attribute-based blocking

As been pointed out in Section 1.2, it is often impractical to traverse all records in the real-time collective ER approach, because of collective entity resolution’s high complexity.

To reduce the computational complexity and to get a real-time result, my approach narrows down the number of records that need to be considered, and produces a candidate set by using the attribute-based blocking. This blocking process compares one attribute of records and removes those records with low similarities. My approach uses entities’ name to be the considered attribute.

An entity’s name can be presented in different ways, for example, in different orders or using initials. In order to handle different situations, this blocking process uses three comparisons: comparison of initials, comparison of last name and comparison of
full name.

To be more specific, this blocking process starts with the comparison of initials. If the similarity of entities’ initials is high, then this process compares entities’ last name and full name, and the final name similarity is the summation of the last name similarity and the full name similarity.

For instance, to compare names J. Zhang and Jing Wei Zhang, the blocking process includes two steps:

1) Comparing their initials, J. Z. and J. W. Z., and in this case there are similar to each other.
2) Comparing their last name, Zhang and Zhang and their full names, J. Zhang and Jing Wei Zhang

For each step above, this blocking process uses approximate string comparison to compare entities name’s edit-distance. The result combines two comparisons in the second step together. Records with an under-threshold name similarity are discarded, and only candidates are left in a set, which is referred as the candidate set.

To minimize the false negatives in the candidate set, my approach sets the threshold for the name similarity relatively low. The evaluation for this blocking process is presented in Experiment 4.1.1.

3.2 Candidate representative determination

After the attribute-based blocking process, similar records are gathered in the candidate set. To further resolve these records, my approach applies clustering algorithms for further resolutions.

However, the number of clusters is unknown. Rather than using semi-automatic methods to determine the number of clusters, my approach uses the following way to solve this problem.

Here my approach applies some manually refinements and lets users determine a representative of each candidate set. For example, when people want to resolve an entity, they first get a candidate set from the blocking process. Then they can determine one of the records in the set to be the candidate representative based on their attributes, e.g. entities’ name.

After the candidate representative is determined, the next move is to remove different records in the candidate set based on the similarity between the candidate representative and the rest of records in the candidate set.
3.3 Attribute-based entity resolution

This report will discuss three different processes for computing the similarities between different records. I discuss the attribute-based entity resolution process in this section.

There are two methods for comparing entities’ attributes: the full value comparison and the keyword comparison. However, both methods have their shortness. For full value comparison, the result’s quality is sensitive to the existence of short function words and lexical words. For keyword comparison, it is difficult to extract keywords correctly without access to external domain knowledge, and an indiscreet extraction can end up with semantic loss as well. In this project, I applied the full value comparison.

Moreover, it is illogical that if all attributes are treated equally, thus the relevant attributes need to be assigned with different weights. Thus, I built an attribute-weighting table to determine the weights for different attributes.

In this attribute-weighting table, attributes are classified based on their different owners. Then attributes in the same class are graded based on their importance to their owners. After that, attributes in each grade are ranked based on their ability to identify entities. Table 3.2 presents an example attribute-weighting table in bibliographic domain.

<table>
<thead>
<tr>
<th>Class</th>
<th>Grade</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author attributes</td>
<td>1st grade: Author name;</td>
<td>1. Author name</td>
</tr>
<tr>
<td></td>
<td>Author email</td>
<td>2. Author email</td>
</tr>
<tr>
<td></td>
<td>2nd grade: Author affiliations; Author address</td>
<td>1. Author affiliations</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Author address</td>
</tr>
<tr>
<td>Publication attributes</td>
<td>1st grade: Index-Keywords;</td>
<td>1. Index-Keywords</td>
</tr>
<tr>
<td></td>
<td>Publication-title</td>
<td>2. Publication-title</td>
</tr>
<tr>
<td></td>
<td>2nd grade: Source title;</td>
<td>1. Source title</td>
</tr>
<tr>
<td></td>
<td>Source volume</td>
<td>2. Source volume</td>
</tr>
</tbody>
</table>

Table 3.2 Example of attribute-weighting table in bibliographic domain

Attributes’ weights are also adjusted in real-time based on the percentage of missing values in the attributes. More details are presented in Experiment 4.2.1.

In each of the attribute-based ER process, my approach calculates the attribute similarity between the candidate representative and one of the other records in the candidate set. This process considers attribute values as strings and uses approximate string comparison to calculate each value pair’s edit-distance respectively. Then it calculates the products of each edit-distance and corresponding attributes’ weights.
Finally, the attribute similarity is the summation of these products and then divided by the summation of weights of all attributes concerned.

The formula for attribute similarity is shown below. $E_i$ stands for the candidate representative, $E_j$ stands for one of the other records in each attribute-based ER process, $S_a$ stands for the similarity of each relevant attributes and $\omega$ is the weight for each attribute.

\[
\text{Attribute similarity} = \frac{\sum(\omega \times S_a(E_i, E_j))}{\sum(\omega)}
\]

In the attribute-based ER process, my approach sets an attribute similarity threshold to remove different records. That means records with an under-threshold attribute similarity will be removed out of the candidate set after this process.

The attribute-based ER process is insufficient to generate desirable results in some cases. The shortness of the attribute-based ER process is that real-world entities’ attributes are often insufficient to determine entities’ similarity. Also, the attribute-based similarity is sensitive to missing values, which is fairly common in real-world data.

To improve the effectiveness, instead of removing any records in attribute-based ER process, my approach keeps all the results (i.e. attribute similarities) in this process, and combines them with records’ relational similarity.

3.4 Naïve relational entity resolution

In this section, I discuss the second process for computing the similarities between records, which is the naïve relational entity resolution process.

As discussed by Getoor and Bhattacharya’s [7], the naïve relational ER process uses entities’ relationships to resolve entities, which means it treats the related entities as additional attributes for matching.

However, the naïve relational ER process has its shortness. Using Figure 3.1 (in the next page) as an example, when the naïve relational ER process resolves Zhang W. of Paper 1 and Zhang W. of Paper 3, it compares two authors’ attributes, their related papers’ attributes, and their co-authors’ names. Since two Zhang W.’s co-authors both contain Wang W. and Li Y., their relational similarity will be increased. However, these two Wang W. and two Li Y. are actually two different entities.
It has been pointed out by Getoor and Bhattacharya [7], the performance of naïve relational ER is unsatisfactory when most entities’ names are common and their relationship is dense. This project implements the naïve relational ER process and evaluates its performance in Experiment 4.3.1.

### 3.5 Collective entity resolution

Rather than using naïve relational ER process, my approach resolves the related entities as well, which means to resolve entities and their neighbors collectively. This is referred as collective entity resolution [7], and it is the third process for computing the similarities between records.

Consider the example in Figure 3.1 again, when the collective entity resolution process resolves Zhang W. of Paper 1 and Zhang W. of Paper 3, it compares two authors’ attributes and their papers’ attributes, meanwhile it also resolves their co-authors (i.e. neighbors). Different from the naïve relational ER technique, the collective ER technique resolves two Wang W. and two Li Y. as well. As a result, two Zhang W.s’ relational similarity will not be increased, because the collective ER result shows that two Wang W. and two Li Y. are different entities. Hence, the quality of collective ER’s result can be improved. This report provides a comparison between the collective ER process and the naïve relational ER process in Experiment 4.3.2.

In each of the collective ER process, the same records from the candidate set are clustered into a new cluster together with the candidate representative, and this new cluster is referred as the collective cluster.

My approach adopts hierarchical clustering in the collective ER process. The reason is that the structure of hierarchic clustering is more informative than the unstructured set of clusters returned by flat clustering [13]. By using hierarchical clustering, my approach can cut the hierarchy at different points by setting up different thresholds,
and more details about the cutting points will be discussed in next section.

To be more specific, my approach uses hierarchical clustering with the bottom up structure in the collective ER process. In the first hierarchy, each of the records in the candidate set are re-considered as a new cluster, and the cluster that contains the candidate representative is the collective cluster here. In each following hierarchy, my approach uses greedy algorithm that calculates the similarities between each of new clusters and the collective cluster. The cluster with the highest similarity is merged into the collective cluster. The merged cluster becomes the new collective cluster for next hierarchy, and that highest similarity is referred as the merged cluster’s combination similarity.

The reason for incorporating the bottom-up structure with the greedy algorithm in this collective ER process is that the bottom-up structure makes decisions based on local patterns without initially taking into account the global distribution [13]. Since the candidate representative is determined, it is more accurate to use the bottom-up structure based on local pattern (i.e. $C_i$ and $C_j$) than use the top-down structure, which considers information about the global distribution (i.e. all clusters).

Figure 3.2 shows an example of the hierarchical clustering with the bottom up structure in the collective ER process.

Figure 3.2 Example of the hierarchical clustering in the collective ER process
In Figure 3.2, there are four records in the candidate set originally: Candidate representative, Record A, Record B and Record C, and each of them is considered as a new clusters in the first hierarchy. The new cluster contains Candidate representative is the collective cluster. In the second hierarchy, the similarities between each of new clusters and the collective cluster are calculated, and it turns out that the collective cluster and the new cluster1 have the highest similarity. Thus, the collective cluster and the new cluster1 are merged into the new collective cluster. In the third hierarchy, the collective cluster and the new cluster2 are merged. In the fourth hierarchy, since the highest similarity, i.e. the similarity between the collective cluster and new cluster3 is under the threshold, thus the cutting point is reached. As a result, the Record C is removed and the collective cluster becomes the final cluster.

A high level pseudo-code for the collective ER process is provided in Figure 3.3.

1. Get candidate set using attribute-based blocking
2. Determine representative of the candidate set (i.e. \( C_i \)) by manually refinement
3. For clusters \( C_i, C_j \) in the candidate set
4. Insert \( \{ \text{sim} (C_i, C_j), C_i, C_j \} \) into priority queue
5. While stop conditions (cutting points) not meet
6. Extract \( \{ \text{sim} (C_i, C_j), C_i, C_j \} \) from queue
7. If \( \text{sim}(C_i, C_j) < \text{threshold} \), then stop
8. Merge \( C_i \) and \( C_j \) to collective cluster \( C_{ij} \)
9. Remove entries for \( C_i \) and \( C_j \) from queue
10. For each cluster \( C_k \) in the candidate set
11. Insert \( \{ \text{sim} (C_{ij}, C_k), C_{ij}, C_k \} \) into queue
12. For each cluster \( C_n \) neighbor of \( C_{ij} \)
13. For \( C_k \) such that similar\((C_k, C_n)\)
14. Update \( \text{sim} (C_k, C_n) \) in queue

Figure 3.3 High level pseudo-code of the collective ER process

Here I apply the formula proposed by Getoor and Bhattacharya to calculate the similarities of two clusters \( C_i \) and \( C_j \) [7]:

\[
\text{Sim} (C_i, C_j) = (1 - \alpha) \times \text{Sim}_A (C_i, C_j) + \alpha \times \text{Sim}_C (C_i, C_j) \quad (0 \leq \alpha \leq 1)
\]

The similarity of two clusters is the summation of two clusters’ attribute similarity and collective similarity, and these two similarities are assigned with different weights \( \alpha \) and \( 1 - \alpha \). My approach uses cluster’s neighborhood similarity to measure the collective similarity of two clusters.

This project adopts one of the measures from Liben-Nowell and Kleinberg’s study
and uses the *Jaccard coefficient* as two clusters’ neighborhood similarity. The Jaccard coefficient measures the similarity between clusters’ neighbors $C_{iN}$ and $C_{jN}$. It is defined as the size of the intersection divided by the size of the union of the neighbors [14]:

$$\text{Jaccard} \ (C_i, C_j) = \frac{C_{iN} \cap C_{jN}}{C_{iN} \cup C_{jN}}$$

After the neighborhood similarity of two clusters $C_i$ and $C_j$ is obtained, my approach looks up the *corresponding attribute similarity* of these two clusters from the attribute-based ER step. This corresponding attribute similarity is the attribute similarity between the candidate representative and the record in another cluster $C_j$. Finally, the similarity of two clusters $C_i$ and $C_j$ is the summation of their attribute similarity and neighborhood similarity.

This project implements the collective ER process and evaluates its performance in Experiment 4.3.2.

### 3.6 Real-time threshold set determination

The collective ER process needs thresholds to control the combination similarity of the merged cluster, which are also the cutting points for the hierarchical clustering as this report discussed in Section 3.5.

However, different users sometimes may need different kinds of results. For instance, expert users prefer a high recall result when they are doing professional novelty searches, but normal users prefer a high precision result for a common querying. In consideration of those situations, my approach sets up a *threshold set* for different kinds of real-time collective ER results: a low threshold for a fuzzy ER result with high recall, a middle threshold for a balanced ER result with high balanced F-measure and a high threshold for a refine ER result with high precision. That also means there are three cutting points for the hierarchical clustering in the collective ER process.

The method this project used to determine the cutting points is based on the fundamental assumption of hierarchical clustering that the merge operation is monotonic, and the combination similarity in each hierarchy is decreasing [13]. My approach determines the three thresholds in the threshold set based on the first combination similarity (also the highest), which occurred in the second hierarchy in the collective ER process. Therefore, the threshold set is determined in real-time.

My approach also observes the gap (i.e. the difference) between every two successive combination similarities in two hierarchies, and uses it to assist with determining the threshold set in real-time. The cutting point of the hierarchical clustering is reached when the gap suddenly becomes very large, which means adding one more cluster
into collective cluster (i.e. the final cluster) will decrease the quality of the result significantly. The evaluation of the threshold set is presented in Experiment 4.4.1.

Consider the example in Figure 3.2 again, the first merging occurs in its second hierarchy, and the highest similarity in the second hierarchy belongs to the collective cluster and the new cluster1. Then this highest similarity becomes the combination similarity of the newly merged collective cluster in this hierarchy, and our approach also uses this highest similarity to determine the threshold set in this collective ER process. In addition, the gap that I mentioned in the last paragraph is the difference between the combination similarities of the second hierarchy and the third hierarchy.

In the end, my real-time collective ER approach generates three different kinds of results for each time, and the results are ranked by the combination similarity produced in the collective ER process.
4. Experiments

In this chapter, I present the experiment results and evaluate the efficiency and effectiveness of my approach.

The efficiency of my approach is evaluated by simple complexity analysis and experiment programs’ run time counting, which are presented in Section 4.5.

The effectiveness of my approach is evaluated by assessing the quality of resolved records, and this report uses the following three measures to determine the quality of each result:

1) Precision = number of correct records being extracted / number of records been extracted
2) Recall = number of correct records being extracted / number of records in the gold-standard set
3) Balanced F-Measure = 2 × (Precision × Recall / Precision + Recall)

Considering the multiple parameters involved in this real-time collective ER approach, I performed several experiments to control parameters for a better evaluation. In each experiment, the values of the three measures were the mean of all results.

Due to the limitation of time, my project only applied on bibliographic database. The reason why I chose bibliographic database was because the problem of entities’ resolution was particularly serious in that domain. Since entities’ name are often presented in initials in bibliographic databases.

The dataset used in the experiments was from Scopus, a well-known bibliography database. I stored the experiment dataset into two tables: scopus_authors table and scopus_publications table. These two tables contained 47333 author records and 10784 publications records respectively. Besides, there were 8 attributes for scopus_authors table, and 19 attributes for scopus_publications table. The author attributes are listed in Table 4.1, and publication attributes are listed in Table 4.2.

<table>
<thead>
<tr>
<th>author</th>
<th>email</th>
<th>address</th>
<th>authorid</th>
<th>affiliation</th>
<th>pid</th>
<th>gold</th>
<th>aid</th>
<th>authororder</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

Table 4.1 Author attributes table

<table>
<thead>
<tr>
<th>page_end</th>
<th>issn</th>
<th>issue</th>
<th>cited_by</th>
<th>author_keywords</th>
<th>conference_name</th>
<th>conference_date</th>
<th>id</th>
<th>page_start</th>
<th>authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

Table 4.2 Publication attributes table
My experiments needed records’ gold-standard, thus I added a gold-standard field in the scopus_authors table, add removed records without gold-standard. After that, the number of records in these two tables were reduced to 19527 (for scopus_authors table) and 10784 (for scopus_publications table) respectively.

The experiment’s software environments included Myeclipse IDE (for java programming) and Postgre 9.2 database (for data & experiment result storage). The operation system that my experiments ran on was Windows 7 Home Premium 64-bit. In addition, the hardware environment for this experiment was a laptop with Intel Core i5 M460 2.53GHz processor and 4GB of RAM.

The SQL codes for creating all the database tables in my experiments are listed in the Appendix. Diagrams for the high-level structures of my experiment programs are listed in the Appendix as well.

4.1 Blocking experiment

In this section, this report evaluates the effectiveness of the blocking process using different blocking thresholds.

Experiment 4.1.1 Blocking thresholds evaluation

In this experiment, I gradually increased the blocking threshold from 0.5 to 0.9, and tested this blocking process on 1000 records. Then, I watched the changes of results’ (i.e. candidate sets) precision, recall and balanced F-measure. The results were shown in Figure 4.1 in the next page. The values of results’ recall remained around 0.99. For a better observation, I standardized all the recall values by deducting 0.99 and magnified it 1000 times.

Through the observation of Figure 4.1 in next page, I noticed that the increasing turning point for precision and F-measure was both occurred around 0.6 to 0.8, and the standardized recall also met its decreasing turning point in that range. Thus I narrowed down the range of the threshold to 0.6-0.8. Then, I listed the average size of the candidate set generated by the blocking process, the average size of gold-standard, number of records with 0-0.5 recall and 0.5-0.99 recall was shown in Table 4.3 in next page.
In Table 4.3, the results of threshold 0.5 to 0.9 had almost the same recall distributions, while the decreasing of average candidate set size slowed down gradually. I decided to set 0.7 as the final blocking threshold with the average candidate set size at 100+ levels. The candidate sets produced by this blocking process were considered as the baseline of my approach and used for following experiments.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>Average Gold size</th>
<th>Average candidate set size</th>
<th>Number of records with 0-0.5 recall</th>
<th>Number of records with 0.5-0.99 recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>7.932</td>
<td>424.927</td>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>0.6</td>
<td>7.932</td>
<td>289.911</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>0.7</td>
<td>7.932</td>
<td>179.237</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>0.8</td>
<td>7.932</td>
<td>98.596</td>
<td>4</td>
<td>34</td>
</tr>
<tr>
<td>0.9</td>
<td>7.932</td>
<td>68.697</td>
<td>5</td>
<td>33</td>
</tr>
</tbody>
</table>

Table 4.3 Average size of the candidate set generated by the blocking process, the average size of gold-standard, number of records with 0-0.5 recall and 0.5-0.99 recall

4.2 Attribute-based entity resolution experiment

In this section, I analyzed the attributes’ weighting of my approach in Experiment 4.2.1. I modified the attribute similarity algorithm used by my approach in Experiment 4.2.2, and analyzed the performance of the attribute-based ER process in Experiment 4.2.3.
Experiment 4.2.1 Attribute weighting

The attributes used by the attribute-based ER process were listed in Table 4.4, which was the attribute-weighting table in this experiment.

<table>
<thead>
<tr>
<th>rank</th>
<th>Author attributes</th>
<th>Publication attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1̣th grade</td>
<td>2̣nd grade</td>
</tr>
<tr>
<td>1</td>
<td>Author email</td>
<td>Author address</td>
</tr>
<tr>
<td>2</td>
<td>Author name</td>
<td>Affiliation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4 Attribute weighting table in Experiment 4.2.1

This attribute-weighting table showed author and publication attributes’ grades and ranks. These attributes were graded based on their ability to identify entities. Then attributes in each grade were ranked based on their correlations with their owners, and the correlations were adjusted by domain experts.

I calculated the percentage of author attributes’ missing values and the percentage of publication attributes’ missing values in scopus_authors and scopus_publications tables respectively. The results were listed in Table 4.5a, Table 4.5b and 4.5c.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>author</th>
<th>affiliation</th>
<th>address</th>
<th>email</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>0/19527</td>
<td>0/19527</td>
<td>3266/19527</td>
<td>2997/19527</td>
</tr>
<tr>
<td>Percentage</td>
<td>0%</td>
<td>0%</td>
<td>16.73%</td>
<td>15.35%</td>
</tr>
</tbody>
</table>

Table 4.5a Author attributes’ missing values percentage

<table>
<thead>
<tr>
<th>Attribute</th>
<th>title</th>
<th>year</th>
<th>source title</th>
<th>volume</th>
<th>issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>3/10784</td>
<td>91/10784</td>
<td>0/10784</td>
<td>474/10784</td>
<td>944/10784</td>
</tr>
<tr>
<td>Percentage</td>
<td>0.02%</td>
<td>0.84%</td>
<td>0%</td>
<td>4.4%</td>
<td>8.75%</td>
</tr>
</tbody>
</table>

Table 4.5b Publication attributes’ missing values percentage

<table>
<thead>
<tr>
<th>Attribute</th>
<th>author keywords</th>
<th>index keywords</th>
<th>conference name</th>
<th>conference date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing value</td>
<td>4566/10784</td>
<td>1579/10784</td>
<td>10308/10784</td>
<td>10310/10784</td>
</tr>
<tr>
<td>Percentage</td>
<td>42.34%</td>
<td>14.64%</td>
<td>95.59%</td>
<td>95.6%</td>
</tr>
</tbody>
</table>

Table 4.5c Publication attributes’ missing values percentage

The steps of setting author attribute weights and publication attribute weights included:

Step 1: Remove less relevant attributes in the dataset (e.g. the 3̣rd grade of publication
attributes in Table 4.4).

Step 2: Set the same weights to each attributes, and keep their summation as 1.
Step 3: Adjust weights based on attribute grades.
Step 4: Adjust weights based on attribute ranks.
Step 5: Adjust weights based on attributes’ missing values percentages.

The weights’ variations of author attributes and publication attributes after Step 2, Step 4 and Step 5 were listed in Table 4.6 and Table 4.7 respectively. This experiment used the weights after Step 5 as the final weights.

<table>
<thead>
<tr>
<th>Author attributes</th>
<th>Email</th>
<th>Author name</th>
<th>Address</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
</tr>
<tr>
<td>Step 4</td>
<td>0.4</td>
<td>0.3</td>
<td>0.2</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>0.35</strong></td>
<td><strong>0.3</strong></td>
<td><strong>0.2</strong></td>
<td><strong>0.15</strong></td>
</tr>
</tbody>
</table>

Table 4.6 Author attributes’ weights after Step 2, Step 4 and Step 5

<table>
<thead>
<tr>
<th>Publication attributes</th>
<th>Index keywords</th>
<th>Author keywords</th>
<th>Publication title</th>
<th>Source title</th>
<th>Conference name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>Step 4</td>
<td>0.3</td>
<td>0.25</td>
<td>0.2</td>
<td>0.15</td>
<td>0.1</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>0.3</strong></td>
<td><strong>0.2</strong></td>
<td><strong>0.25</strong></td>
<td><strong>0.2</strong></td>
<td><strong>0.05</strong></td>
</tr>
</tbody>
</table>

Table 4.7 Publication attributes’ weights after Step 2, Step 4 and Step 5

**Experiment 4.2.2 Attribute similarity modification**

The dataset being used in this experiment included one author table and one publication table, which had different levels of correlations to my approach. To modify the attribute similarity algorithm being used by my approach based on the dataset, I split the algorithm into two parts: the author attribute similarity and the publication attribute similarity. The attribute similarity became the summation of author attribute similarity $S_A$ and publication attribute similarity $S_P$ with different weights $(1 - \alpha)$ and $\alpha$. The modified formula of attribute similarity:

\[
\text{Attribute similarity} = (1 - \alpha) \times S_A + \alpha \times S_P
\]

In this experiment, I set the proportion for the author attribute similarity and publication attribute similarity with 2:1 (i.e. $\alpha$ is 0.33), by considering author attributes were more relevant to the attribute similarity.

As I emphasized in Section 3.3, ER in real-time was one of the feature of my approach. Hence there were some adjustments to my approach in real-time, and there were listed below.
The first adjustment was to set up a missing value threshold to control attributes’ missing value in real-time, and any attribute with an over-threshold missing value percentage was assigned with zero weight. Through the observation of Table 4.5a, Table 4.5b and Table 4.5c, I concluded that attributes’ missing value percentages could be grouped into two classes: much less than 20% and much greater than 20%. Therefore, I used 20% as the threshold for missing values control in this experiment.

The second adjustment was to change the parameter $\alpha$ based on the missing value ratio between author attributes and publication attributes in real-time. Here I add two extra parameters into the attribute similarity formula:

1) The ratio between the summations of author attributes and summations of all attributes, $R_A$.
2) The ratio between the summations of publication attributes and summation of all attributes, $R_P$.

Two ratios were both standardized using a fraction $k \in (0,1)$ to avoid it excessive influencing the parameter $\alpha$. The final formula for attribute similarity was shown below:

$$\text{Attribute similarity} = (1 - \alpha + k \times R_A) \times S_A + (\alpha + k \times R_P) \times S_P$$

**Experiment 4.2.3 Attribute-based ER evaluation**

In this experiment, I tested the attribute-based ER process using the 1000 candidate sets produced in Experiment 4.1.1. I gradually increased the attribute similarity threshold from 0.5 to 0.7, and evaluated the effectiveness of the attribute-based ER process.

Figure 4.2 in the next page showed the variations of precision, recall and F-measure as the attribute similarity threshold increased. In general, the increasing of precision and balanced F-measure came with the decreasing of recall. More details were listed as follows:

1) The highest precision occurred at threshold 0.7, which was 0.8861.
2) The highest recall occurred at threshold 0.5, which was 0.9883.
3) The highest balanced F-measure was 0.8778 and occurred at threshold 0.6.
4) There was a significant decreasing of recall and balanced F-measure after threshold 0.65.
This experiment extracted some records that had a high precision but a low recall for further analysis. Their corresponding author attribute records and publication attribute records were presented in Table 4.8 and Table 4.9. Apparently, many records contained missing values. That proved the two statements about the shortness of the attribute-based ER process in Section 3.3.

1) Real world attributes are often insufficient to determine entities’ similarity.
2) Attribute-based ER approach is highly sensitive to records’ missing values.

<table>
<thead>
<tr>
<th>aid integer</th>
<th>author varying(179)</th>
<th>affiliation varying(252)</th>
<th>address varying(231)</th>
<th>email varying(51)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kieser J.</td>
<td>Department of Oral S</td>
<td></td>
<td>jules.kiesse</td>
</tr>
<tr>
<td>2</td>
<td>Hood K.A.</td>
<td>Wakefield Biomedical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Moltedo A.</td>
<td>Department of Ophtha</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Hornby C.C.</td>
<td>Dept. of Human Nutri</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Hughes A.R.W.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8 Author attribute records

<table>
<thead>
<tr>
<th>index_keywords</th>
<th>author_keywords</th>
<th>title varying(260)</th>
<th>source_title varying(205)</th>
<th>conference_name varying(191)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>antiangiogloss</td>
<td>2D-DIGE: CRC: LCM:</td>
<td>M Protonic analysis</td>
<td>Journal of Protonic</td>
</tr>
<tr>
<td>2</td>
<td>acetazolamide:</td>
<td>Otago glaucoma surge</td>
<td>Investigative Ophtha</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>adult: article:</td>
<td>Adult: Decisional</td>
<td>Validated scales to</td>
<td>Applepie</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Correlation between</td>
<td>Journal of Ophthalmology</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>anatomy: article:</td>
<td>Clinical anatomy:</td>
<td>D: Professional story:</td>
<td>Anatomical Sciences</td>
</tr>
</tbody>
</table>

Table 4.9 Publication attribute records

4.3 Naïve relational ER and collective ER experiment

In this section, I presented the experiment result for the naïve relational ER process in
Experiment 4.3.1 and the experiment result for collective ER process in Experiment 4.3.2, and then compared these two results.

**Experiment 4.3.1 Naïve relational ER evaluation**

In this experiment, I discarded the attribute similarity threshold, instead, I considered the attribute similarities generated in attribute-based ER process as a semi-similarity. Here my approach combined attribute similarity and relational similarity together, and these two similarities were assigned with different weights. The weight for relational similarity was generated in real-time based on the number of record’s co-authors, which meant the more co-authors the record had, the larger weights the relational similarity got. The range for relational similarity’s weight was from 0 to 0.5.

This experiment tested the naïve relational ER process on the same dataset used in Experiment 4.2.3, which contained 1000 candidate sets. I gradually increased the naïve relational similarity threshold from 0.3 to 0.5, and observed the changes of results that generated by the naïve relational ER process.

![Figure 4.3 Variations of precision, recall and balanced F-measure in Experiment 4.3.1](image)

Figure 4.3 showed the changes of precision, recall and F-measure as the naïve relational similarity threshold increased. In general, the increasing of precision and balanced F-measure came with the decreasing of average recall. More details were listed as follows:

1) The highest precision occurred at threshold 0.5, which was 0.9646.
2) The highest recall occurred at threshold 0.3, which was 0.9775.
3) The highest balanced F-measure was 0.7921 and occurred at threshold 0.4.
4) There was a significant increasing of precision and balanced F-measure after threshold 0.3.

Comparing with Figure 4.2 in Experiment 4.2.3, I got worse results when implemented the naïve relational ER process into my approach. It indicated that the performance of naïve relational ER process was dissatisfactory over the dataset of Scopus database.

**Experiment 4.3.2 Collective ER evaluation**

In this experiment, my approach replaced the naïve relational ER process to the collective ER process. This experiment used the 1000 records’ attribute similarities from the attribute-based ER process in Experiment 4.2.3, and treated them as the semi-similarities. Here the similarities for two clusters combined attribute similarity with cluster similarity, like I mentioned in Section 3.5. These two similarities were assigned with the same weight in this experiment (i.e. \( \alpha \) is 0.5), which referenced from \( \alpha \)’s evaluation in Getoor and Bhattacharya’s paper [7].

At the beginning of the experiment, I tended to test the collective ER process on the same dataset used in Experiment 4.2.3. However, due to the high time complexity for the collective ER process, I decided to narrow down the size of the experiment dataset to 100 candidate sets.

In this test, I gradually increased the similarity threshold for clusters from 0.2 to 0.4 (also the cutting point of the hierarchical clustering in this process), and observed the changes of results that generated by the collective ER process.

![Graph showing variations of precision, recall, and F-measure](image)

Figure 4.4 Variations of precision, recall and balanced F-measure in Experiment 4.3.2
Figure 4.4 showed the variations of precision, recall and F-measure as the collective similarity threshold increased. In general, the increasing of precision and balanced F-measure came with the decreasing of average recall. More details were listed as follows:

1) The highest precision occurred at threshold 0.4, which was amazingly 1.0 (i.e. 100%).
2) The highest recall occurred at threshold 0.2, which was 0.9991.
3) The highest balanced F-measure was 0.9185 and occurred at threshold 0.3.

A detailed comparison between the results from Experiment 4.2.2 (the attribute-based process), Experiment 4.3.1 (the naïve relational ER process) and Experiment 4.3.2 (the collective ER process) was presented in Figure 4.5. Here I evaluated the three processes using three new measures:

1) Highest precision in the condition of an over 90% recall among all thresholds
2) Highest recall in the condition of an over 90% precision among all thresholds
3) Highest balanced F-measure among all thresholds

Figure 4.5 Highest values of the three new measures in Experiment 4.3.2

Figure 4.5 showed that the highest values of the three measures listed above were all belonged to Experiment 4.3.2 (the collective ER process). Therefore, the performance of my approach had been improved by implementing the collective entity resolution process.
4.4 Real-time threshold set evaluation

In this section, I evaluated the threshold set I discussed in Section 3.5. These three thresholds for each candidate set were generated in real time, based on the highest combination similarity occurred in the second hierarchy of the collective ER process. This experiment used the highest combination similarity as the base value, and the three real-time thresholds took different percentage of it.

The three thresholds were considered as different cutting points for the hierarchical clustering in the collective ER process. Hence, I also implemented a technique to monitor the gap (or the difference) between every two successive combination similarities in two hierarchies, as I explained in Section 3.5.

To evaluate the effectiveness of the real-time threshold set, this experiment used different percentages from 75\% to 85\% of the highest combination similarity, to generate 11 thresholds for each candidate sets. This experiment ran the developed approach on the same dataset in Experiment 4.3.2 by using these 11 thresholds, and watched the precision, recall and balanced F-measure variations of the results. The results were shown in Figure 4.6.

Figure 4.6 Variations of precision, recall and balanced F-measure in Experiment 4.4

Figure 4.6 showed that there was a monotonic increasing for the precision and it reached the highest point at percentage 85\%. Besides, there was a monotonic decreasing for the recall, and its highest point was at percentage 75\%. In addition, the balanced F-measure reached its peak in the middle (i.e. percentage 80\%). Therefore, I finalized the three different thresholds used in the threshold set, which were 0.75 for
the fuzzy ER result, 0.8 for the balanced ER result and 0.85 for the refine ER result. I tested them on three different datasets, and each datasets contains 100 different candidate sets. The results were listed in Table 4.10, Table 4.11 and Table 4.12.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Precision for threshold 0.75</th>
<th>Precision for threshold 0.8</th>
<th>Precision for threshold 0.85</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-100</td>
<td>0.8935</td>
<td>0.9291</td>
<td>0.9537</td>
</tr>
<tr>
<td>101-200</td>
<td>0.7712</td>
<td>0.8192</td>
<td>0.8417</td>
</tr>
<tr>
<td>201-300</td>
<td>0.8437</td>
<td>0.8951</td>
<td>0.9111</td>
</tr>
</tbody>
</table>

Table 4.10 Precision for real-time collective ER approach tested on three different datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Recall for threshold 0.75</th>
<th>Recall for threshold 0.8</th>
<th>Recall for threshold 0.85</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-100</td>
<td>0.9587</td>
<td>0.9272</td>
<td>0.8759</td>
</tr>
<tr>
<td>101-200</td>
<td>0.9868</td>
<td>0.9662</td>
<td>0.9022</td>
</tr>
<tr>
<td>201-300</td>
<td>0.9897</td>
<td>0.9801</td>
<td>0.9033</td>
</tr>
</tbody>
</table>

Table 4.11 Recall for real-time collective ER approach tested on three different datasets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>F-measure for threshold 0.75</th>
<th>F-measure for threshold 0.8</th>
<th>F-measure for threshold 0.85</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-100</td>
<td>0.9033</td>
<td>0.9088</td>
<td>0.8885</td>
</tr>
<tr>
<td>101-200</td>
<td>0.8146</td>
<td>0.8446</td>
<td>0.8229</td>
</tr>
<tr>
<td>201-300</td>
<td>0.8788</td>
<td>0.9082</td>
<td>0.8738</td>
</tr>
</tbody>
</table>

Table 4.12 Balanced F-measure for real-time collective ER approach tested on three different datasets

The highest results in Table 4.10, Table 4.11 and Table 4.12 were bold. To sum up, the performances of my approach using the three thresholds in the threshold set included:

1) By using threshold 0.85, my approach provided results with a high precision (around 0.85 to 0.95), and the precisions were between 0.87 and 0.9.
2) By using the threshold 0.75, my approach provided results with a high recall (around 0.95 to 0.98), and the precisions were between 0.77 and 0.89.
3) By using threshold 0.8, my approach provided results with a high balanced F-measure (around 0.85 to 0.9), which meant both recall and precision were relatively high in this situation.

4.5 Complexity analysis and run time statistics

In this section, I briefly explain the variations of the complexity of my approach. Then I present the run time counting results in Section 4.4.
To reduce the complexity, my approach only resolves one entity in each real-time running. Assume the number of entities in the dataset is N. In each time, my approach only focuses on resolving one entity. Hence, the complexity of my approach is reduced N times (comparing to normal collective ER).

In addition, my approach adopts the blocking process and using bottom-up structure in the collective ER process. By using them, the complexity of my approach reduced as well.

The average times used for each collective entity resolution by using my real-time collective ER approach with different threshold are listed in Table 4.11.

<table>
<thead>
<tr>
<th>Threshold</th>
<th>0.85</th>
<th>0.8</th>
<th>0.75</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time used(average)</td>
<td>1981.13ms</td>
<td>2004.38ms</td>
<td>2033.24ms</td>
</tr>
</tbody>
</table>

Table 4.11 Average time used for each query using my real-time collective ER approach with different threshold

Through the table, I can conclude that the average time required by my approach for a single run is around 2 seconds.
5. Conclusion and future works

In this project, I developed a real-time collective entity resolution approach based on the framework of collective entity resolution from Getoor and Bhattacharya’s paper [7], and evaluated its efficiency and effectiveness over a dataset from Scopus.

In this real-time collective ER approach, I used attribute-based blocking techniques, manually refinements and real-time stop conditions. There was an efficiency improvement after implementing these techniques into my approach. By using the collective ER process in my approach, there was an effectiveness improvement as well.

For the collective ER results, my approach generated three different kinds of results each time in order to enhance its flexibility. In this project, I evaluated the quality of the results generated by my approach and presented the run time they used.

To sum up, the experiment results were positive. However, some processes in my approach are still needed to be refined and improved:

1) The attribute-based blocking process in my approach only used name similarity measure. I can improve this process by considering multiple attributes.

2) I can incorporate data preprocessing into my approach to improve the performance, like implementing TF-IDF in attribute weighting, adding stop-word list or using stemming for attribute similarity comparison.

3) The candidate representative determination process involves manually queries refinement. It is possible to develop an automatic queries refinement technique that makes my approach without any manual operations.

4) The attribute weighting process in attribute-based entity resolution is tailored to certain domain. Future studies can investigate its domain-independent ability.

5) In the attribute-based blocking process and attribute-based entity resolution process, my approach only considered using approximate string comparison to compare entities name’s edit-distance. Future studies can implements other methods in my approach as well.

6) In the collective entity resolution process, my approach only adopted Jaccard coefficient measure for neighborhood similarity, and there are other options like Adamic/Adar similarity or higher-order neighborhoods similarity. Future studies can implement these methods in my approach as well.
7) Due to the limitation of time, I only tested my approach using one dataset from Scopus. My approach could be tested on datasets from different data source to see the performance variations.
Reference list


Appendix

CREATE TABLE scopus_authors
{
    aid integer NOT NULL,
    pid integer NOT NULL,
    author character varying(179) NOT NULL,
    affiliation character varying(252),
    authororder character varying(7),
    authorid character varying(13),
    address character varying(224),
    email character varying(51),
    gold text,
    CONSTRAINT scopus_authors_pkey PRIMARY KEY (aid),
    CONSTRAINT scopus_authors_pid_fkey FOREIGN KEY (pid)
        REFERENCES scopus_publications (id)
        MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
}
WITH {
    OIDS=FALSE
};
ALTER TABLE scopus_authors
OWNER TO postgres;

Figure 1. SQL of creating scopus authors table

CREATE TABLE scopus_publications
{
    id integer NOT NULL,
    authors character varying(260) NOT NULL,
    title character varying(260) NOT NULL,
    year character varying(9),
    source_title character varying(265),
    volume character varying(115),
    issue character varying(89),
    pagc_start character varying(10),
    pagc_end character varying(54),
    cited_by character varying(9),
    authors_with_affiliations character varying(260),
    author_keywords character varying(260),
    index_keywords character varying(260),
    correspondence_address character varying(260),
    conference_name character varying(191),
    conference_date character varying(158),
    issn character varying(136),
    isbn character varying(235),
    doi character varying(159),
    CONSTRAINT scopus_publications_pkey PRIMARY KEY (id)
}
WITH {
    OIDS=FALSE
};
ALTER TABLE scopus_publications
OWNER TO postgres;

Figure 2. SQL of creating scopus publications table
CREATE TABLE author_attribute
(
    author text,
    email text,
    address text,
    authorid text,
    affiliation text,
    pid text,
    gold text,
    aid text,
    authororder text
)
WITH (OIDS=FALSE);
ALTER TABLE author_attribute
OWNER TO postgres;

Figure 3. SQL of creating author attribute table

CREATE TABLE publication_attribute
(
    page_end text,
    issn text,
    issue text,
    cited_by text,
    author_keywords text,
    conference_name text,
    conference_date text,
    id text,
    page_start text,
    authors text,
    source_title text,
    title text,
    authors_with_affiliations text,
    isbn text,
    volume text,
    year text,
    doi text,
    correspondence_address text,
    index_keywords text
)
WITH (OIDS=FALSE);
);
ALTER TABLE publication_attribute
OWNER TO postgres;

Figure 4. SQL of creating publication attribute table
Figure 5. Real-time collective ER experiment program Java classes list
Figure 6. High-level structure of my real-time collective ER experiment program

Figure 7. Example of the result file ‘log1_to_100_8_80’