Entity Resolution in the Presence of Constraints

Zeyu Shen
Supervisor: Dr Qing Wang

COMP 8790: Software Engineering Project
Australian National University Semester 2, 2013

October 29, 2013
Acknowledgments

I would take this opportunity to thank my project supervisor Dr. Qing Wang for giving me an opportunity to work on this project. I really appreciate her guidance, encouragement and strong support, which were significant in helping me complete the project successfully.

I would also like to thank Dr. Weifa Liang for his guidance on the whole program and teaching on technical writing.

Last but not the least, I would like to thank my parents for everything they gave to me.
Abstract

In this project, I present a declarative framework for entity resolution in the presence of constraints. Constraints occur naturally in various data cleaning domains and can improve the quality of entity resolution. For example, there is a table containing titles which look similar but actually represent different publications. So if two publications’ titles are in that table, then they are two different publications. The main feature of my work is that I incorporate weights into the framework. For pairwise matching, I use rule based matching combined with weighted score to determine whether a pair is a positive pair or negative pair. The rules are based on a declarative Datalog-style language with precise semantics. The weights of rules are measured through experiments on training dataset. For conflicts solving, I also use weights to revise the clustering result. My framework mainly focuses on single entity reference relation. However, I also use other relations to illustrate how collective entity resolution works in my framework. The relationships between different entity reference relations are monodirectional, and which entity reference relation affecting other relations is also determined by weights. My experiments are based on the Cora dataset. From the evaluation result, my framework has a high precision and recall (>0.97) for the clustering result of the entity reference relation. I also infer that quality of one entity reference relation will affect quality of the others.
Contents

1 Introduction ................................................. 5
   1.1 Background .......................................... 5
   1.2 Objectives ............................................ 6
   1.3 Motivating Examples ................................. 6
   1.4 Outline ................................................ 7

2 Related Works ................................................. 8
   2.1 Attribute Similarity ................................. 8
   2.2 Collective Entity Resolution ....................... 8
   2.3 Constraints ............................................ 9

3 Dedupalog Framework ......................................... 11
   3.1 Features of Dedupalog ............................... 11
   3.2 Changes in Framework ............................... 12

4 Methodology .................................................. 15
   4.1 Rules .................................................... 15
   4.2 Data Matching Process ............................. 17
   4.3 Data Preprocessing .................................. 18
   4.4 Indexing ............................................... 20
   4.5 Classification Method ................................. 23
      4.5.1 Step 1: Soften Rules –Prediction ............. 24
      4.5.2 Step 2: Harden Rules –Validation .......... 25
      4.5.3 Step 3: Conflicts Resolve .................... 25
      4.5.4 Ad-hoc Way ..................................... 27
   4.6 Relationships between Entity Reference Relations 28

5 Evaluation .................................................... 30
   5.1 Dataset ................................................ 30
   5.2 Measures .............................................. 32
   5.3 Experiments .......................................... 33
      5.3.1 Compare Quality between Each Step ....... 33
      5.3.2 Compare Quality between Methodologies .. 34
      5.3.3 Quality Propagation ............................ 36
5.4 Discussion ........................................................................... 37
   5.4.1 Positive Rules ............................................................... 37
   5.4.2 Negative Rules ............................................................. 41
   5.4.3 Weights ........................................................................ 42

6 Conclusion and Future Work ................................................. 45
   6.1 Conclusion ....................................................................... 45
   6.2 Future Work .................................................................... 46
Chapter 1

Introduction

1.1 Background

Entity Resolution (ER) is an important problem in Computer Science. *Entity resolution* (sometimes referred to as deduplication, record linkage and data matching) is the process of identifying and merging records that represent the same real-world entity (Benjelloun et al. 2008).

In real-life applications, each entity has some attributes. For example, a student entity can have attributes such as name and address. One student may have a real name, a preferred name and a nick name, so there may be three records for that student. In this case, we need to use entity resolution techniques to identify such records that represent the same real-world entity.

The ER problem is important in many applications. For example, consider an insurance company that has two different customer databases (e.g., one storing customers who bought financial insurance, another storing customers who bought personal insurance), and want to combine them. If there are no unique identifiers for customers, then identifying matching records is challenging because a customer may appear in different ways in each database. In practice, people often have to determine which customers match manually.

Entity resolution is also needed in Web applications. When people use search engine to search related information, it is important for search engine to filter similar or same results and give an aggregate result. This work can be facilitated by using entity resolution techniques.

However, the process of entity resolution is not easy. The main three challenges are as follows (Christen 2012):

**Lack of Unique Identifiers**

As mentioned in the previous examples, if a record has a unique identifier, then this record can be easily referred to the particular customer. But without an identifier, entity resolution process should compare all the other attributes to determine which customer the record refers to.
Computational Complexity

Entity resolution often needs to compare pairs between different databases. When comparing them, it is necessary to compare each pair of records between the databases. So if each database has a large number of records, it will take a long time for just one round comparison between different databases. More commonly, comparison for each pair of record may need further comparisons for other pairs. It is easy for comparison complexity growing exponentially.

Privacy Issues

When entity resolution is used associated with people, privacy and confidentiality are likely to be problems. People's personal information such as name, address and phone numbers are all confidential. Thus the results of entity resolution may cause potential discrimination problems.

1.2 Objectives

It is widely recognized that entity resolution plays a central role in many application domains, such as data cleaning and data integration. State-of-the-art approaches to entity resolution favor similarity-based methods, which often fail to capture constraints describing data semantics. The goal of this project is to develop a declarative framework for entity resolution based on weighted declarative Datalog-style rules. The specific tasks are:

1. To conduct a literature review on constrain-based entity resolution methods.

2. To implement the algorithms used in Dedupalog for entity resolution.

3. To develop a declarative framework by incorporating weighted rules into Dedupalog

4. To analyse the efficiency and effectiveness of the developed framework, and implement the algorithm using real datasets.

The framework developed in this project will be based on the work by Arasu et al. (2009). The main features of my framework are:

- use weighted rules to identify whether a pair of records is a positive pair or a negative pair
- use a novel algorithm for conflicts solving

1.3 Motivating Examples

The general purpose of this part is to illustrate my framework using examples. Considering two publications P1 and P2, each publication has many attributes
such as title, authors, publishers and etc. P1’s title is “Data Matching” and P2’s title is “Data Matching Algorithm”. P1’s publisher is “ANU” and P2’s publisher is “NUST”. I have a rule1 called “publications with similar titles are likely to be the same publication” so that I consider P1 and P2 to be the same publication. This is the rule-based matching, which follows rules to match pairs. There is another rule2 called “publications with different publishers are likely different publications”. If I also follow this rule, then there is a conflict since rule1 prefers to make two publications as the same publication and rule1 prefers to make them different. To solve this problem, I allocate weight to rule1 and rule2. I am confident about rule1 so that rule1 gets a weight of 0.8 and I am not so confident with rule2 so that rule2 gets a weight of -0.5. So this pair of publications get a weight score of 0.3. I also predefine that if a pair’s weight score is larger than 0, they are considered to be the same publication. Thus, this pair of publications are considered to be the same with the help of weighted score.

To explain the problem of conflicts between negative pairs and clustering result, consider another publication P3 and P3 is considered as the same publication with P1. According to the transitivity, P2 and P3 should also be the same publication. However, there is another rule3 called “P2 and P3 are definitely two different publications”. So conflicts happen and my work also finds a way to resolve conflicts like this.

1.4 Outline

The following of the thesis is structured as follows:

- In chapter 2, I present the related works for attribute similarity, collective entity resolution and constraints.
- After that, I introduce Dedupalog language used by Arasu et al. (2009) and show the changes on that language in this project in chapter 3.
- In chapter 4, I present the main methodology for entity resolution in this project following the sequence of data matching process.
- In chapter 5, I present the experiment results to show the effectiveness of my framework. Some discussions are held about rules and weights used in the experiments.
- Finally, conclusion and future work are provided in the last chapter.
Chapter 2

Related Works

2.1 Attribute Similarity

My work on entity resolution has used string similarity method. There are many string similarity methods such as edit distance and variations (Bilenko and Mooney 2003), Jaccard, TF/IDF (Gravano et al. 2003), and Jaro (Jaro 1976). In this work, I chose Jaccard as the string similarity method.

For multi-attribute similarity measures, two general proposals are to use a weighted score or average of component similarity scores, and to use thresholds. Among all the methods, one classic technique is the Fellegi & Sunter (1976) Model: given a pair of records, $r = (x,y)$ the comparison vector, is $\gamma$. If $P$ is the set of matching pairs of records and $N$ is the set of non-matching pairs of records, linkage decisions are based on the probability of $\gamma$ given $r \in P$ divided by the probability of $\gamma$ given $r \not\in N$. Further, you can decide if a record is a match or not based on error bounds, $\mu$ and $\lambda$ that create thresholds for whether a record is a match, a non-match, or it is simply uncertain. In this work, I use rule based matching combined with weighted score to determine whether a pair is a positive pair or negative pair.

2.2 Collective Entity Resolution

Collective entity resolution in the presence of constraints is relatively a new topic. For example, I have three records, which are “L. James”, “LBJ” and “King James”. We know that these three records direct to the same real-work basketball player, which is Lebron James. Besides, I have another two records which are “Heat” and “Miami Heat”. There are two relationships; one is “L. James” belongs to “Heat” and the other is “LBJ” belongs to “Miami Heat”. These relationships are presented in Figure 1.
As we know that “L.James”, “LBJ” and “King James” are three references of the entity “Lebron James”. Then we can also refer “Heat” and “Miami Heat” to the same entity according to the two relationships. Thus I collectively deduplicates five references into two entities.

Most of the entity resolution algorithms use constraints in an ad hoc way such as work of Dong et al. (2006). There is no formal semantics in their work to describe constraints and it is unclear how to add new ones. Their work is lack of independent semantics for their algorithm. The techniques demonstrated by Bhattacharya et al. (2007) also adopt hard-coded constraints, but do not provide semantics as well.

Collective entity resolution approaches include non-probabilistic approaches like similarity propagation, or probabilistic reference relations including generative frameworks, or simply a hybrid approach. Generative probabilistic approaches are based on directed reference relations, where dependencies match decisions in a generative manner (Bhattacharya and Getoor 2005).

Markov Logic Networks and other related probabilistic models are also used by other works to do collective entity resolution with constraints. Markov logic combines first-order logic and Markov random fields (Richardson and Domingos 2006), with weighted satisfaction testing for efficient inference and a voted perception algorithm for discriminative learning (Singla and Domingos 2005). Markov logic networks are suitable for simple constraints. However, they are not scalable for large data sets yet.

In this work, collective entity resolution will be used in explaining relationships between entity reference relations. An experiment will also be performed to prove the quality effects between entity reference relations.

2.3 Constraints

Constraints pervasively exist in many entity resolution applications. There are several important forms of constraints. \( P_i \) is denoted as the set of matching pairs of records.

- Transitivity: If \( P_1 \) and \( P_2 \) match, \( P_2 \) and \( P_3 \) match, then \( P_1 \) and \( P_3 \) must also match.
• Exclusivity: If P1 matches with P2, then P3 cannot match with P2

• Functional Dependency: If P1 and P2 match, then P3 and P4 must match.

Based on these constraints, transitivity is the key to entity resolution. In Arasu et al. (2009)’s paper, they use transitivity in back propagation process to solve the potential conflicts. A number of previous works solved conflicts of database constraints such as standard and conditional functional dependencies (Bohannon et al. 2007).

Fuxman et al. (2001) consider a slight variant that performs repairs in real-time while answering queries, in the presence of hard constraints. Tung et al. (2001) proposed two kinds of instance-level constraints, so-called must-link and cannot-link constraints. For example, “ICDE” and “Conference on Data Engineering” are totally two different strings, but they are the same conference in reality, or conversely, that “Multivariate decision trees” and “Multivariate versus univariate decision trees” are different publication titles.

Several related works consider clustering in the presence of aggregation constraints (Chaudhuri et al. 2007). Shen et al. (2005) developed a two-layer ER method based on constraints, in which the first layer handles constraints at the group level, while the second layer handles constraints at the level of entity pairs.

In this project, constraints will be used in many parts. First, it will be used in clustering building. For example, if there are pair (a, b) and pair (b, c), I will put a, b and c into the same cluster according to the transitivity. Functional Dependency will be adopted when explaining relationships between different entity reference relations.
Chapter 3

Dedupalog Framework

As my work is based on Arasu et al.’s work, I will explain the framework they used for entity resolution and also explain my framework. Dedupalog is a framework developed by Arasu et al. (2009). Dedupalog is declarative, domain-independent, is expressive enough to encode many constraints considered, and its key feature is allowing users to specify both hard and soft constraints (Arasu 2009). For example, the constraint that “if authors and pages of two publications are the same, then these two publications are the same” is an example of a hard constraint; this constraint must be followed by each clustering result. Example of soft constraint is like “publications with similar titles are likely to be the same publication”. In contrast to the hard constraints, soft constraints can be violated, but they aim to keep the violations of soft constraints in a minimum level.

The process of running a dedupalog is as follows:

- First, a set of input tables are provided, which contain the references that need to be clustered and any additional tables useful in their clustering task (the results of any similarity computation such as TitleSimilar table).
- Second, they define a list of entity references which identify the kinds of entities that need to cluster, such as paper and author.
- Third, they define a dedupalog program. This program defines what should be satisfied when clustering, such as hard constraints.
- Finally, they execute the program and produce a clustering result of data that is aware of the constraints.

3.1 Features of Dedupalog

Dedupalog defines a list of entity references. This is done by declaring a set of entity reference relations that contain the references that user wants to cluster. In their paper, they want to deduplicate the papers in data. So they created
entity reference relation denoted as (Paper!). Each tuple in the relation equals to a single entity reference. Then Dedupalog will work with the knowledge of schema of the relation.

The main feature of Dedupalog is that they define kinds of dedupalog rules for program. For the entity reference relation, they created a clustering relation denoted as $R^*(x,y)$ which includes duplicate pairs. The concept of clustering relation is similar to concept of views in relational database. The aim of their method is to populate the clustering relations, which are the building blocks of Dedupalog rules (Arasu et al. 2009).

There are six kinds of rules are defined in Deduplalog, which are soft-complete rules, soft-incomplete rules, hard rules, complex hard rules, complex negative rules and recursive rules. The concept of hard rules, soft rules, positive rules and negative rules will be explained later.

A complete rule example is like “papers with similar titles are likely duplicates”. I can infer two derived rules from this rule: 1) if two papers have similar titles, then they are likely to be clustered together. 2) if two papers’ titles are not similar, then they are likely to be put into separate clusters. Incomplete rule is one side of complete rule and only gives positive information. If “papers with very similar titles are likely duplicates” is a incomplete rule, I can only use the first derived rule.

A complex rule involves joins with clustering relations. For example, when I cluster two papers, I also need to cluster the journals of those papers. This rule will need a functional dependency between papers and journals, which is a collective entity resolution problem.

A recursive rule example is like “Authors that don not share common coauthors are unlikely to be duplicates” (Arasu et al. 2009). If I want to apply this rule, I must cluster two authors that share a co-author. I need to examine the current clustering result and that requires recursion.

### 3.2 Changes in Framework

In my work, I also use the Datalog-style rules to define constraints. For example, I want to deduplicate the publication, venue and author. For this, I define three entity reference relations: publications (Publication!), venues (Venue!) and authors (Author!) as follows.

- **Publication! (id) :-** Publication (id,-)
- **Venue! (vid) :-** Venues(vid,-)
- **Author! (aid) :-** Author(aid,-)

(\(-\) omits all other attributes in that table)

Same as their paper, I will also create clustering relations, such as Publication $(id, id')$ that contains duplicate pairs of objects. In my work, I split soft rules and hard rules to four kinds: soft positive rules, soft negative rules, hard positive rules and hard negative rules, which means I abandon the usage of complete and
incomplete rules. For single entity reference relation, I only use the following four kinds of rules.

1) **Soft Positive Rules:**

The first type of my Dedupalog rule is called soft positive rule and I denote it using the soft sign (\(<\)-). It is the basic rule that will be used in this project. An example soft positive rule is “papers with similar titles are likely to be clustered together”. To implement this, I must have a string similarity function for the attribute of title. Instead of creating another table to include similar titles, I directly perform string similarity function (Jaccard) on the pairs. I can write the soft positive rule as follows:

\[
\text{Publication}*(\text{id},\text{id}') \leftarrow \text{Publication} (\text{id},\text{t}), \text{Publication} (\text{id}',\text{t'}),
\quad \text{TitleSimilar}(\text{t},\text{t'})
\]

This rule says that publications with similar titles are preferred to be clustered together. It is a positive rule because if this rule is applied, many positive pairs will be generated. It is soft because if a pair of publications are approved to be clustered together after this rule, but they are actually allocated to different clusters due to some reasons, then this behavior is allowed. It is a soft rule so that it can be violated.

2) **Soft Negative Rules:**

In contrast to soft positive rules, I also define the soft negative rules. For example, if one publication has some value in tech and the other does not have, then these two publications should be put into different clusters. In Dedupalog format, this rule can be represented as

\[
\neg\text{Publication}*(\text{id},\text{id}') \leftarrow \text{Publication} (\text{id},\text{tech}), \text{Publication} (\text{id}',\text{_})
\]

Negative rules are used to separate pairs. It is soft rule so that if two publications should not be in the same cluster because one have tech value and the other doesn’t, but they are clustered into one cluster due to other positive rules in the end, then it is also allowed.

3) **Hard Positive Rules:**

The simplest examples of hard rules are “If authors and pages of two publications are similar, then these two publications should be put into one cluster”.

\[
\text{Publication}*(\text{id},\text{id}') \subseteq \text{Publication} (\text{id},\text{a},\text{p}), \text{Publication} (\text{id}',\text{a}',\text{p'}),
\quad \text{PageSimilar} (\text{p},\text{p'}), \text{AuthorSimilar} (\text{a},\text{a'})
\]

It is a hard rule because once two publications are put into the same cluster by this rule. Then this pair will not be separated by any other negative rules.
4) **Hard Negative Rules:**

Hard negative rules are a little difficult to find. In this paper, I choose the rule “If pages of two publications are not similar, then these two publications should be put into different clusters.” as a hard negative rule.

\[
\neg {\text{Publication}}(\text{id, id}') \leq {\text{Publication}}(\text{id, p}), \text{Publication}(\text{id', p'}),
\]

\[
\text{PageNotSimilar} (\text{p, p'})
\]

Once a hard rule separates a pair, this pair will not be clustered due to any reasons. The hard negative rule I chose in this paper cannot ensure also the separations are correct. I will discuss this problem and solution in 4.5.4 “Adhoc Way”.

**Assumption**

I assume that there will no conflicts between hard negative rules and hard positive rules as in the work of Arasu et al. (2009).
Chapter 4

Methodology

4.1 Rules

I declare four types of rules: soft positive rule, soft negative rule, hard positive rule and hard negative rule. Hard rules must be obeyed. Soft rules can be violated. The following 10 rules are based on the publication table and they are also main rules used in my work.

- Positive Soft Rules
  
  - Publications_rule1
    
    If titles of two publications are similar, then these two publications should be put into one cluster.
    
    \[
    \text{Publication}^*(id, id') \leftarrow \text{Publication} (id, t), \text{Publication} (id', t'), \\
    \text{TitleSimilar} (t, t')
    \]
    
    The attribute used in this rule is “title” from publications table.

  - Publications_rule2
    
    If titles, authors and year of two publications are similar, then these two publications should be put into one cluster.
    
    \[
    \text{Publication}^*(id, id') \leftarrow \text{Publication} (id, t, a, y), \text{Publication} (id', t', a', y'), \\
    \text{TitleSimilar} (t, t'), \\
    \text{AuthorSimilar} (a, a'), \text{YearSimilar} (y, y')
    \]
    
    The attributes used in this rule are “title”, “authors” and “year” from publication table.
If titles and authors of two publications are similar, then these two publications should be put into one cluster.

\[
\text{Publication}^{*}(id,id') \leftarrow \text{Publication} (id,t, a), \text{Publication} (id',t', a'), \\
\text{TitleSimilar} (t,t'), \text{AuthorSimilar} (a,a')
\]

The attributes used in this rule are “title” and “authors” from publication table.

- **Publications_rule4**

If titles and booktitle of two publications are similar, then these two publications should be put into one cluster.

\[
\text{Publication}^{*}(id,id') \leftarrow \text{Publication} (id,t, b), \text{Publication} (id',t', b'), \\
\text{TitleSimilar} (t,t'), \text{BooktitleSimilar} (b,b')
\]

The attributes used in this rule are “title” and “booktitle” from publication table.

- **Publications_rule6**

If titles and journal of two publications are similar, then these two publications should be put into one cluster.

\[
\text{Publication}^{*}(id,id') \leftarrow \text{Publication} (id,t, j), \text{Publication} (id',t', j'), \\
\text{TitleSimilar} (t,t'), \text{JournalSimilar} (j,j')
\]

The attributes used in this rule are “title”, and “journal” from publication table.

- **Positive Hard Rules**

- **Publications_rule5**

If authors and pages of two publications are similar, then these two publications should be put into one cluster.

\[
\text{Publication}^{*}(id,id') \leftarrow \text{Publication} (id,a, p), \text{Publication} (id',a', p'), \\
\text{PageSimilar} (p,p'), \text{AuthorSimilar} (a,a')
\]

The attributes used in this rule are “authors”, and “pages” from publication table.
**Negative Soft Rules**

- **Negative_rule1**

If a pair of titles of two publications are the same as a pair of titles in the table `title_neq`, then these two publications should be put into different clusters.

\[ \neg \text{Publication}^*(\text{id}, \text{id}') \leftarrow \text{Publication} (\text{id}, \text{t}), \text{Publication} (\text{id}', \text{t}'), \text{TitleNEQ} (\text{t}, \text{t}') \]

- **Negative_rule3**

If one publication has value of `tech` and the other does not have, then these two publications should be put into different clusters.

\[ \neg \text{Publication}^*(\text{id}, \text{id}') \leftarrow \text{Publication} (\text{id}, \text{tech}), \text{Publication} (\text{id}', \_ ) \]

- **Negative_rule4**

If one publication has value of `journal` and the other does not have, then these two publications should be put into different clusters.

\[ \neg \text{Publication}^*(\text{id}, \text{id}') \leftarrow \text{Publication} (\text{id}, \text{j}), \text{Publication} (\text{id}', \_ ) \]

**Negative hard rules**

- **Negative_rule2**

If pages of two publications are not similar, then these two publications should be put into different clusters.

\[ \neg \text{Publication}^*(\text{id}, \text{id}') \leftarrow \text{Publication} (\text{id}, \text{p}), \text{Publication} (\text{id}', \text{p}'), \text{PageNotSimilar} (\text{p}, \text{p}') \]

4.2 **Data Matching Process**

![Data Matching Process Diagram](image)

**Figure 2: Data Matching Process**
An overview of the data matching process is shown in Figure 2 (Christen 2012). In this project, I cover all five steps of data matching process that he described. The first step is data preprocessing. This step is to clean and normalize the data from data source. If the data are from different sources, the data will be in same format. My data preprocessing work is about normalizing the data and filling the empties in the data, which aims to improve the accuracy of string comparison.

The second step is indexing. It aims to decrease the computational complexity of the raw dataset through deleting the pairs of records which are apparently different. After this step, the raw data will improve to candidate data. In my work, I use positive rules to generate candidate pairs, which extremely enhances the efficiency of data processing.

The third step is the actual data matching process. It compares the pair of records through the string similarity function or other methods. I also use rule-based method to compare the concrete pairs of records.

In the classification phase, I add weights to each rule and use the weight score to determine the result of a pair. If the pair of record is above a threshold, than it is seems to be a potential match. Otherwise, it is marked as non-match. However, if a pair of records is identified by a hard positive rule, then it will be marked as match. And for those potential match pairs, I will use conflict solving process to verify the correctness of classification.

Last, in the evaluation phase, measures such as precision, recall and F1-measure will be used to evaluate the quality of clustering.

### 4.3 Data Preprocessing

Data preprocessing is required in many entity resolution tasks. I also conduct data normalization and data filling in my project.

I normalize data to enhance the accuracy of comparison. For attributes in a table (e.g., title, authors, publisher), data normalization involves converting all strings to upper or lower case, and replacing all useless symbols with space. My measure of similarity between two strings is based on Jaccard coefficients so that I should keep the spaces among the characters to improve the accuracy. For some attributes that mainly contain numbers (e.g., pages, year), besides of converting all strings to same case, I only keep numbers and spaces.

After data normalization, the next thing is to fill empty values. The original dataset is not as completed as enough so that many attributes’ values are missing. This brings difficulties to string comparison because if one of the two strings is empty, the string similarity function will identify them to be different strings. Thus, the filling function will improve the completeness of the dataset and maintain the correctness and accuracy of filling values at the same time.
Figure 3: Filling Function

Figure 3 shows the algorithm of filling function. First, I generate candidate pairs using all positive rules and my raw table (without filling). Then I use “clustering building method” in 4.4 to build a clustering result so that every record in that table belongs to a cluster. Then, I have a traversal on all the clusters. For each of the clusters, I also have a traversal on all attributes of that table. For each of the attributes, I select a value of that attribute which appears the most frequently in that cluster. I use that value to fill all the other blanks of that attribute. After one attribute is filled, I continue until all the attributes have been filled. Then I move on to the next cluster and repeat the same things.

In my work, the filling functions are applied on the 15 attributes of publication table. Attributes “tech” and “page” are not filled. The reason for this is that both page and tech have strong ability to separate two publications. For example, if two publications’ pages are totally different, then I have 99% confidence to separate these two publications into different clusters. Likewise, if one publication has some value in tech and the other does not have, then I know that one publication is technique report and the other is another format publication so that I can separate them. If “pages” and “tech” are filled with wrong values, then the false negatives will be increased considerably.

Figure 4 shows the differences between “before filling function” and “after filling function” on my publication table. We can see that completeness of most of attributes increases a lot.
4.4 Indexing

There are 1879 records in the publication table. If these records are directly formed into pairs, there will be 3.5 million pairs need to be dealt with. Thus, indexing process will remove the pairs which are unlikely in one cluster and only keep the potential matching pairs, which are called candidate pairs. I will apply three steps on the candidate pairs to improve the efficiency.

In this project, I do the indexing based on the filled table. From Figure 5, the filling function generates raw candidates based on the raw table and my positive rules at the beginning. Then it uses these pairs to generate clustering result and directs to the concrete filling process. After the process, a filled table was generated. Then the indexing process will generate candidate pairs directly based on the filled table and all positive rules. The process “generate candidate pairs” actually consists of two steps: generate single rule based positive pairs and generate candidate pairs. I need to generate candidate pairs for each positive rule at first and then combine them together to generate final candidate pairs.
Generate Single Rule Based Positive Pairs

Generating positive pairs of single rule is basic part of indexing. I need to generate positive pairs based on each positive rule. The main process is that I have a traversal on all the pairs in the filled table. For each pair of records, I use string similarity function to determine the similarity of this pair. If the measure of similarity is above a threshold, then this pair is considered as a positive pair. The concrete comparison method is as follows:

```
Input a rule r
Input the filled table t
r has a set of attributes A*
attribute attr ∈ A*
String similarity function is S(attr,p)
Threshold for attr is t_{attr}
1 Foreach pair p in the t
2       If sizeof(A*)=1
3           If (S(attr,p)>=t_{attr})
4              Then p is marked as a positive pair
5              Else p is marked as a negative pair
6      Else
7          Foreach attr in A*
8             If (S(attr,p)<t_{attr})
9                Then p is marked as a negative pair
10               Else continue
11     p is marked as a positive pair
```

Figure 5: Relationship between Filling Function and Indexing

Figure 6: String Comparison Process
For each rule, I will give a threshold for each attribute of the table in advance. The threshold is determined by experiments on a small training dataset. I will justify this later in 5.4.1. The string similarity function will compare the values of the attributes between two records sequentially. The results of the comparisons are decimals between 0 and 1. 1 means absolutely a match and 0 means absolutely non-match.

If the rule only has one attribute, then the string comparison function will use the only result comparing with the threshold that has been set. If the result is not smaller than the threshold, then the two records are considered as similar records and this pair of records will write into a rule result table in the database.

If the rule has more than one attribute, then the function will use each of the results comparing with its corresponding threshold. If all of the results are not smaller than their thresholds, then the two records are considered as similar records and this pair of records will write into a rule result table in the database.

**Generate Candidate Pairs**

As the single rule based pairs have been generated, then I repeat the steps above to get all rules’ positive pairs. To combine all these positive rules, I use SQL union language. For example, in my project, I combine all six positive rules’ pairs:

```sql
select * from publications_rule1 union select * from publications_rule2 
union ... union select * from publications_rule6
```

This will ensure that there will be no duplicates pairs in the candidate pairs. And these candidate pairs are ready to be clustered or be further processed before clustering.

**Clustering Building Algorithm**

The clustering method used in this project is based on the ids of pairs. In all of the dataset tables, id is the primary key of a record. So I only use id to represent a particular record of a table. The algorithm for clustering is as follows.
For each pair in the table, I first check whether id1 is in any of existed clusters. If id1 is found in one cluster, then I check whether id2 is also in that cluster. If id2 is not in that cluster, then I add id2 to that cluster. Analogously, if id1 is not found in any of existed clusters, I try to find whether id2 is in any of current clusters. If id2 is found in one cluster and id1 is not in that cluster, id1 is added to that cluster. However, if both of them are not in any of existed clusters, I build a new cluster for them. After all positive pairs in the table have been processed, the clustering result is generated. The clustering result includes all clusters and all ids in each cluster.

4.5 Classification Method

The main part of the classification method consists of three steps. I start with candidate pairs using the method of “generate candidate pairs”. The main procedure of my method is shown in Figure 8.
Step 1 will consider all rules as soft rules and give each rule a weight. Each pair will be applied with all rules and gets a weight score, which determines whether it is a positive pair or negative pair. After this step, my candidate pairs will be split into positive pairs and negative pairs and stored in the softened table.

Step 2 will add hard rules on these processed pairs. If a pair is identified by a hard rule, then it will not be affected by any other rules and will be directly written into the hardened table. After Step 2, I will generate clustering result using all the positive pairs in the hardened table.

Step 3 is a conflicts solving process. Some pairs in the hardened table are marked as hard negative pairs but are put into the same cluster by other rules. This step solves these conflicts. After these three steps, I will get a final clustering result \( R^* \) for the entity reference relation based on the rules.

### 4.5.1 Step 1: Soften Rules – Prediction

The purpose of the Step 1 is to give each pair a weight score and determine whether each pair is a positive pair or a negative pair. Originally, I have hard rules and soft rules. But in this step, I consider all of these rules as soft rules even if some of them are hard rules. Then I give a weight to each rule. The weight for each rule is from the experiment results based on a training dataset. I will justify each weight later in 5.4.3. For positive rules, the range of weights is \([0,1]\). For negative rules, the range of weights is \([-1,0]\). The algorithm for determining whether a pair is a positive pair is as follows:

**Figure 9: Soften Rules**

For each candidate pair, I apply positive rules and negative rules on the pair. The initial weight score is 0. If a positive rule returns true, then the weight of the positive rule will be added into the weight score. If a negative rule return true, the weight of the negative rule will be added into the weight score (minus...
the absolute value of weight). After all rules have been performed, the candidate pair will have a final weight score. If the score is larger than the threshold I set, then the pair is marked as positive rule and written to the softened table. Otherwise, a negative pair is written to the softened table. After this step, I get the softened table in which all candidate pairs are split to positive pairs and negative pairs and the total number of pairs does not change. Each pair is also associated with a weight.

4.5.2 Step 2: Harden Rules – Validation

The main purpose of Step 2 is to improve the effectiveness of the clustering result. In this step, hard rules will directly have impact on the pairs table. For each pair in the softened table, I apply hard positive rules and hard negative rules. If a hard positive rule returns true, then the pair will be marked as positive pair even if it is originally marked as a negative pair. If a hard negative rule returns true, then the pair will be marked as a negative pair even if it is originally marked as a positive pair. All processed pairs will be written into hardened table. If a pair is not affected by all hard rules, then it will be directly written into hardened table with its original mark.

![Figure 10: Harden Rules](image)

After this step, the precision of the clustering result will be improved as well as recall, which means false positives are reduced according to the hard negative rules and false negatives are reduced according to the hard positive rules. The hardened table records all the positive pairs and negative pairs. Besides, the pairs identified by hard positive rules or hard negative rules are marked.

4.5.3 Step 3: Conflicts Resolve

After Step 2, I have the hardened table including all positive pairs and negative pairs with hard pairs marked. Then I generate a clustering result denoted as
$R_p^*$ based on all the positive pairs in the hardened table. Due to the transitivity and symmetry properties, I only need parts of the pairs to build a cluster. For example, based on three positive pairs (1,2), (1,3), (1,4), I can build the cluster (1,2,3,4). But actually, the cluster (1,2,3,4) can infer 6 pairs, which are (1,2), (1,3), (1,4), (2,3), (2,4) and (3,4). However, if (2,3) is a negative pair marked by a hard negative rule, then 2 and 3 should be in separate clusters. In this case, there will be a conflict between the hard negative pair (2,3) and the cluster (1,2,3,4). Thus, in Step 3, I solve such conflicts using the algorithm as follows.

```
Input a table $t_p$ including all hard negative pairs in hardened table
Input a table $t_p$ including all positive pairs in hardened table
Input a Clustering $R_p^*$ based on $t_p$
Output Clustering $R^*$

Cluster $(V, \emptyset) \in$ Clustering $R_p^*$
1 Foreach pair $p$ in $t_p$
  2 If id1 and id2 of $p$ are in the same cluster $(V, \emptyset)$
  3 Then generate new cluster Cluster1 and add id1 to Cluster1
  4 And generate new cluster Cluster2 and add id2 to Cluster2
  5 And Foreach id in V
  6     find weight $w_1$ of pair(id1,id) from $t_p$
  7     find weight $w_2$ of pair(id2,id) from $t_p$
  8     If $w_1 \geq w_2$
  9     Then add id to Cluster1
 10     Else add id to Cluster2
 11     delete Cluster $(V, \emptyset)$ from $R_p^*$
 12     add Cluster1 to $R^*$
 13     add Cluster2 to $R^*$
 14 Add all clusters in $R_p^*$ into $R^*$
```

**Figure 11: Conflicts Solve**

The table $t_p$ including all positive pairs is generated by using SQL to select all the positive pairs from the hardened table. The table $t_n$ including all hard negative pairs generated by using SQL to select all hard negative pairs from the hardened table. The clustering result $R_p^*$ is generated based on positive pairs in table $t_p$ using the “Clustering Building Method”. For each of hard negative pairs in $t_n$, I will check whether the two ids of the pair are in the same cluster. If yes, it means there is a conflict and I should separate the two ids into different clusters. I firstly generate two clusters Cluster1 and Cluster2 and add the two ids (id1 and id2) to them respectively (id1 is in cluster1 and id2 is in cluster2). Then I grab all the other ids in that cluster. For each id, I find the weights for pair (id1,id) and pair (id2,id) from table $t_p$. And if the weight for (id1,id) is larger than (id2,id), then I add the id to the cluster1. Otherwise, I add the id to cluster2. I repeat these steps until all the other ids in that cluster have been disposed. Then, I delete the cluster from clustering result $R_p^*$ and add the two new clusters to the final clustering result $R^*$. When all hard negative pairs have been processed, I add all clusters in $R_p^*$ to $R^*$ so that $R^*$ contains all clusters.
To explain the algorithm further, there is a simple example for it. (2,4) is a hard negative pair in table \( t_n \) and its corresponding cluster is \( (1,2,3,4) \). I firstly grab 1 and compare weights between pair (2,1) and pair (4,1). If (2,1)’s weight is larger than (4,1), then 1 belongs to 2. Then I grab 3 and compare weights between (2,3) and (4,3). If (4,3)’s weight is larger than (2,3), then 3 belongs to 4. Finally, I delete cluster (1,2,3,4) from \( R_p^* \) and add cluster (1,2) and cluster (3,4) to the \( R^* \).

4.5.4 Ad-hoc Way

After solving all the conflicts and adding new clusters, I get a new clustering result \( R^* \) and it have lower false positives even if false negative increases a little bit. However, my hard negative rules are not really “hard”, which means if I apply my hard negative rules directly on candidate pairs, some of pairs split by my rule are actually in the same cluster. In other words, there will be false negatives if I apply my hard negative rules on candidate rules.

So if I use my hard negative rules in Step 3, the true positive may decrease and false negative may increase considerably. It can be deduced that my hard negative rules make some positive pairs separate but they are actually positive pairs. In this case, some of the conflicts I found are actually caused by the false separation by my hard negative rules.

Thus, to verify that my algorithm will indeed decrease the false positives. I use an adhoc way to apply the algorithm. After I found all the conflicts in Step 3, I add a review step. The review step will find the conflicts which are real conflicts according to the Cora_gold. The real conflicts are pairs that are actually negative pairs but are clustered into same cluster. I pick up all the real conflicts and use my conflicts resolve algorithm in Step 3 to deal with them.
4.6 Relationships between Entity Reference Relations

There are three entity reference relations in my project: publication, venue and author. The relationships between them are mono-directional as shown in Figure 13;

![Figure 13: Relationships between Entity Reference Relations](image)

From Figure 13, we can see that each of the three entity reference relations can use string comparison based rules to generate clustering results by themselves. For author reference relation, though it only has one attribute called “name”, I can also generate positive pairs by using a rule “if the names of two authors are similar, then these two authors should be in one cluster” and generate clustering result using these positive pairs.

However, when it comes to the relationships between entity reference relations, I only consider the impacts from one reference relation to the other. In Figure 13, we can see there is only one direction from one reference relation to the other, which means mutual effects between two relations are not allowed.

For example, between the publication relation and author relation, either publication relation affects the clustering result of author relation or author relation affects clustering result of publication relation.

The rule “r1” in Figure 13 is like “if two publications are same, their authors in the same position should be same”. It can be presented in the datalog format as follows:

\[
\text{Author*}(\text{aid}, \text{p}, \text{aid’}, \text{p’}) \leftarrow \text{Publication*}(\text{id}, \text{id’}), \text{SamePos}(\text{p}, \text{p’})
\]

The rule “r3” in Figure 13 is like “if two publications have similar title and authors in same position are the same, then they are the same publications”. It can be presented in the datalog format as follows:

\[
\text{Publication*}(\text{id}, \text{id’}) \leftarrow \text{Publication} (\text{id}, \text{t}, \text{id’}, \text{t’}), \text{TitleSimilar} (\text{t}, \text{t’}), \\
\text{Author*}(\text{aid}, \text{p}, \text{aid’}, \text{p’})
\]

However, whether the relationship between publication reference relation and author reference relation determined by r1 or r3 depends on the average weights
of their positive rules. In my project, if average weight of all positive rules in publication relation is larger than that in author relation. Then r1 is the relationship between two relations. Otherwise, r3 is the relationship. In this project, the average weight for publication reference relation is 0.92. The average weight for author reference relation is 0.90. The average weight for venue reference relation is 0.36. Thus, publication reference relation will affect the clustering results of venue reference relation and author reference relation, which is the same as Figure 13.
Chapter 5

Evaluation

5.1 Dataset

The dataset used in this project is the Cora dataset, which is a collection of machine learning papers. There are three tables in the Cora dataset: Publication table, Author table and Venue table. All the tables are extracted from a Cora version of XML file. Publication table has 1879 records. Author table has 4571 records. Venue table has 1536 records. Tables 1-3 describe attributes in each of these three tables.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (Primary key)</td>
<td>Publication ID</td>
</tr>
<tr>
<td>authors</td>
<td>All author names included in the publication</td>
</tr>
<tr>
<td>title</td>
<td>Publication title</td>
</tr>
<tr>
<td>publisher</td>
<td>Publisher</td>
</tr>
<tr>
<td>address</td>
<td>Address of publisher</td>
</tr>
<tr>
<td>note</td>
<td>Note</td>
</tr>
<tr>
<td>pdate</td>
<td>Publication date</td>
</tr>
<tr>
<td>journal</td>
<td>Journal name</td>
</tr>
<tr>
<td>volume</td>
<td>Volume</td>
</tr>
<tr>
<td>pages</td>
<td>Pages</td>
</tr>
<tr>
<td>tech</td>
<td>Whether the publication is a technical report</td>
</tr>
<tr>
<td>institution</td>
<td>Institution name</td>
</tr>
<tr>
<td>booktitle</td>
<td>Venue</td>
</tr>
<tr>
<td>editor</td>
<td>Editor</td>
</tr>
<tr>
<td>year</td>
<td>Publication year</td>
</tr>
<tr>
<td>type</td>
<td>Type of publication</td>
</tr>
<tr>
<td>month</td>
<td>Publication month</td>
</tr>
</tbody>
</table>

Table 1: Publication Table Description
Completeness of Dataset

The following tables show the completeness of each table.

Publication Table

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Number of non-empty</th>
<th>Completeness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id (Primary key)</td>
<td>1879</td>
<td>100%</td>
</tr>
<tr>
<td>authors</td>
<td>1879</td>
<td>100%</td>
</tr>
<tr>
<td>title</td>
<td>1841</td>
<td>98%</td>
</tr>
<tr>
<td>publisher</td>
<td>706</td>
<td>38%</td>
</tr>
<tr>
<td>address</td>
<td>740</td>
<td>39%</td>
</tr>
<tr>
<td>note</td>
<td>72</td>
<td>4%</td>
</tr>
<tr>
<td>plate</td>
<td>505</td>
<td>27%</td>
</tr>
<tr>
<td>journal</td>
<td>635</td>
<td>34%</td>
</tr>
<tr>
<td>volume</td>
<td>835</td>
<td>44%</td>
</tr>
<tr>
<td>pages</td>
<td>1235</td>
<td>66%</td>
</tr>
<tr>
<td>tech</td>
<td>150</td>
<td>8%</td>
</tr>
<tr>
<td>institution</td>
<td>248</td>
<td>13%</td>
</tr>
<tr>
<td>booktitle</td>
<td>905</td>
<td>48%</td>
</tr>
<tr>
<td>editor</td>
<td>457</td>
<td>24%</td>
</tr>
<tr>
<td>year</td>
<td>1225</td>
<td>65%</td>
</tr>
<tr>
<td>type</td>
<td>113</td>
<td>6%</td>
</tr>
<tr>
<td>month</td>
<td>30</td>
<td>2%</td>
</tr>
</tbody>
</table>

Table 4 : Publication Table Completeness

Author Table

The author table’s completeness is 100% before filling function, which means all records have values in every attribute.
Venue Table
As the venue table only have one attribute “venue” except for vid, id and cluster id. So this project only considers the non-empty venues. The total number of non-empty venues is 1536.

5.2 Measures
When a clustering result is generated, I will use it to compare with the ground-truth in Cora_gold. The differences between two kinds of clustering results are presented in true positive, false positive and false negative. These three counts are widely used in evaluation of entity resolution results.

True Positive (TP)
If a pair of records should be put into the same cluster and the result actually does so, then this pair of records are marked as True Positive (TP).

False Positive (FP)
If a pair of records should be put into different clusters but the result actually puts them into the same cluster, then this pair of records are marked as False Positive (FP).

False Negative (FN)
If a pair of records should be put into the same cluster but the result actually puts them into different clusters, then this pair of records are marked as False Positive (FN).

In this project, I use three measurements: precision, recall and F1-measure. The definition of these measurements are as follows:

Precision
Precision is the proportion of retrieved documents that are relevant

\[Precision = \frac{TP}{TP + FP}\]

Recall
Recall is the proportion of relevant documents retrieved by the system

\[Recall = \frac{TP}{TP + FN}\]
**F1-measure**

A measure that combines precision and recall is the harmonic mean of precision and recall,

\[
F_1 - measure = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}
\]

This is also known as the F1-measure, because recall and precision are evenly weighted.

### 5.3 Experiments

I ran all experiments on an Intel Core i5-3470 at 3.2 Ghz with 8GB of RAM running Windows 8 pro (64 bit). All data for the experiments was stored in PostgreSQL 9.2. The language for implementation is Java SE 1.7 and the IDE for Java is myEclipse 10.0.

#### 5.3.1 Compare Quality between Each Step

The first experiment is to evaluate the effectiveness of every step in my three steps method in 4.5. For this, I compare the results between clustering result after the three steps and clustering result before the three steps. The clustering result before the three steps is built based on the candidate pairs.

- In Step 1, I added a set of soft rules on candidate pairs and allocated weights to each of the rules. These rules can be positive or negative rules. Though they are all soft rules, they also improve the clustering result a lot because each pair got a weight score which made classifying pairs more rational.
- In Step 2, I applied a set of hard rules to the pairs generated by Step 1.
- In Step 3, the conflicts solving function was applied to deal with the conflicts. The results are presented in Figure 14.
Figure 14: Quality Improvement in Each Step

From the clustering result before the three steps, I can see that it has a good recall, which is larger than 0.95. However, its precision does not reach 0.8, which means that there are many false positives.

The result of Step 1 indicates a significant improvement in precision. The negative rules actually separate many false positives without increasing number of false negative. In contrast, the recall also has an increase due to the implementation of weight score. Due to the both improvements in precision and recall, the F1-measure increased to about 0.97.

From the result of Step 2, this change also has positive effects on the clustering result. The hard positive rules revised some false negatives and hard negative rules revised some positive rules so that both precision and recall has a further improvement.

From the result of Step 3, though the number of conflicts is actually small comparing to the total number of pairs. After the conflicts are solved, the precision of clustering result is indeed improved further. Even though there is a small drop in recall, the F1-measure is improved to about 0.98.

5.3.2 Compare Quality between Methodologies

The second experiment is to compare quality horizontally between my methodology and methodology described in Arasu et al (2009)’s paper. In that paper, the authors deduplicated data which is aware of constraints by using a dedupalog program.

Each dedupalog program is evaluated in two phases: forward-voting and back-propagate.

Forward-voting is similar to my Step 1 and Step 2. Instead of giving each rule a weight, they have a count for each pair. If a pair is identified by a soft positive rule, then this pair’s count will add one. In contrast, this pair’s count will subtract by one if it is identified by a soft negative rule. After all rules have been applied, if the count of the pair is no less than a threshold, then the pair is recognized as a positive pair.

Back-propagate is similar to my aim of Step 3. It is also a conflict solving process. Instead of dealing with hard negative pairs, their method deals with each cluster. Each cluster is like a graph composed with nodes and edges. For each cluster, they firstly add soft positive edges and then add hard positive edges and hard negative edges. Then they infer the third edge by these two rules

1. If edge \((u, v)\) is a hard positive edge and edge \((v, w)\) is also a hard positive edge, then edge \((u, w)\) is a hard positive edge

2. If edge \((u, v)\) is a hard positive edge and edge \((v, w)\) is a hard negative edge, then edge \((u, w)\) is a hard negative edge

After that, there is a “hardening” process. They pick soft edges in a defined sequence and makes soft positive edge to hard positive edge and makes soft
negative edge to hard negative edge. Then they redo the inference process using the two rules. In the end, the nodes connected by hard positive edges will be new clusters.

I implemented my method and the paper's method based on the publication entity reference relation. I denote my method as method1 and their method as method2.

Figure 15: Quality of Method1 in Different Thresholds

Figure 16: Quality of Method2 in Different Counts

Figure 15 presents the results of method1, running in different threshold settings. The first conclusion to draw is that my clustering method gives both high precision and recall (>0.95) for reasonable settings of the threshold.
Figure 16 presents the results of method2, running in different count settings. The three settings are: count of pair larger than 0, count of pair larger than 1 and count of pair larger than 2. As all the operations in their method are integer related operations. So their thresholds are also integers. From the result, it can be inferred that only when count larger than 1, their method will give a high precision and recall (>0.95).

Figure 17: Quality between Method 1 and Method2

Figure 17 compares the best results of two methods which are count larger than 1 in method2 and threshold equals to 0.83 in method1. According to the results, both methodologies give high precision and recall (>0.95). The two methods give a very similar precision. However, my method has a better performance on recall, which means my method makes more pairs clustered together.

5.3.3 Quality Propagation

The final experiment is to study effects on author reference relation and venue reference relation due to different quality of publication clustering results. For simplicity, I use one soft positive rule for each of author reference relation and venue reference relation.

- For the author table, the rule is “if two publications are same, their authors in the same position should be the same”.
- For the venue table, the rule is “if two publications are same, their venues should be the same”.

The “before” clustering result for publications reference relation is the clustering result based on candidate pairs. The “after” clustering result for publication reference relation is the clustering result after the three steps in 4.5. Based on the two rules and the two kinds of clustering results of publication reference relation, I generate clustering results for author and venue
Figure 18: Quality Impact on Venue and Author

Figure 18 shows the effects on author reference relation and venue reference relation using the precision measure. From the result, when the precision of publication increased from below 0.8 to above 0.95, both venue clustering result and authors clustering result have an increase in precision, which means the true positive raises or false positives drops. It also proves that publication reference relation really affects the clustering results of venue reference relation and author reference relation.

5.4 Discussion

In this part, I justify the positive rules and negative rules. For the positive rules, I justify the contributions of them to generate candidate pairs. Besides, the parameters and thresholds I chose for string similarity function of each positive rule are also discussed. For the negative rules, I explain where they come from through case study.

I also justify the weights I used for each rule. In general, weights can be determined by using a training dataset, or domain knowledge from experts.

5.4.1 Positive Rules

Rule Contribution Justification

The positive rule determination is mainly based on experience and some experiments. As all of the positive rules are string comparison based rules, I made some combinations of the attributes in publication table to form the six rules. I conducted an experiment to show that every positive rule has contributions of enhancing result of clustering.

I firstly generate positive pairs for publications_rule1 and generate a clustering result. After that, I compare it with Cora_gold clustering result and record the TP, FN and FP. Then I generate positive pairs for publications_rule2 and union with pairs generated by publications_rule1. Then I use the same procedure to generate clustering results and compare and record the results. I denote the clustering result obtained by applying a set of rules \( \{r_1, \ldots, r_n\} \) as “\( \bigcup_{i=1}^{n} Pair(r_i) \)”
The experiment result shows that publications_rule1, 2, 3, 4 and 5 all have contributions on the clustering result, since there are changes in TP, FP and TN. However, the publications_rule6 seems to have no contributions on the clustering result as there is no change from $\bigcup_{i=1}^{5} \text{Pair}(r_i)$ to $\bigcup_{i=1}^{6} \text{Pair}(r_i)$.

**Rule Threshold Justification**

After I choose the attributes for each rule, the key point is to determine the threshold for string similarity function of each attribute. For example, the publications_rule1 is “Publication*(id,id’) <- Publication (id,t), Publication (id’,t’), TitleSimilar(t,t’)”. There is only one attribute in this rule, which is title. To determine the threshold for the similarity between titles, I used the “softening” process. As we know, if the threshold for similarity is very high, then the precision of the result can also be very high. However, this may also result in a low number of total true positives, because the smaller number of titles will be involved. For example, if I set threshold as 0.95 then the precision may be 0.9, but the actual number of TP is 9 and number of FP is 1. If I set threshold as 0.8 then the precision may be 0.75, but now I have 750 in TP and 250 in FP. Therefore, I should soften these thresholds to increase candidate pairs as much as possible when maintaining the precision of certain level. It may generate some false positives, but the number of records of pairs involved will be large. So I have some experiments to determine the thresholds. I firstly extract part of the dataset as my training dataset. Then I generate pairs for each rule based on the training dataset. But instead of combining pairs of each rule, I directly generate clustering result for each positive rule. After the clustering result is built, it is used to compare with the Cora_gold, which is real clustering result for that table. Then I get the true positive, false positive and false negative and generated precision, recall and F1-measure. After that, I adjust the threshold for the attributes in the positive rule and repeat building clustering results and compare the results with Cora_gold. After
that, I get a list of measurements for that positive rule, I compare the precision and recall to find a proper set of thresholds of attributes. Besides, I should also keep the clustering result precise to some extent. So I set 0.85 for the threshold of precision and record the threshold set of attributes. I repeat the above process to get the measurements for all positive rules.

The guideline of choosing threshold for similarity comparison of attribute is as follows:

1. If the precisions of threshold settings are above 0.85, then these threshold settings are chosen as candidate threshold settings.

2. Among all the candidate threshold settings, I choose the setting with the highest recall. However, if the highest recall is only larger than the second highest recall for less than 1% but setting of the second highest recall gives a higher precision than the setting of the highest recall. Then the threshold setting of the second highest recall is chosen for that rule.

The following tables are the results I got and the concrete threshold settings I chose.

**Publications_rule1**

<table>
<thead>
<tr>
<th>Round</th>
<th>title</th>
<th>Basic Measures</th>
<th>Derived Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>1</td>
<td>0.9</td>
<td>49265</td>
<td>7209</td>
</tr>
<tr>
<td>2</td>
<td>0.8</td>
<td>52815</td>
<td>7416</td>
</tr>
<tr>
<td>3</td>
<td>0.7</td>
<td>58964</td>
<td>12713</td>
</tr>
</tbody>
</table>

For publications_rule1, I follow the guideline 1 to set 0.8 as the threshold for title because it has a precision of 0.8769, which is above 0.85 and the recall 0.8178 is higher than 0.7629.

**Publications_rule2**

<table>
<thead>
<tr>
<th>Round</th>
<th>title</th>
<th>authors</th>
<th>year</th>
<th>Basic Measures</th>
<th>Derived Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
<td>FN</td>
<td>Precision</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>41218</td>
<td>591</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>44909</td>
<td>860</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>46501</td>
<td>1285</td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>46062</td>
<td>4903</td>
</tr>
</tbody>
</table>

For publications_rule2, the results for precision are all very good and above 0.85. So in this situation, I follow the guideline 2 and picked the threshold with the highest recall. So for publications_rule2, thresholds for title, author and year are all 0.3.

**Publications_rule3**
For publications_rule3, only one combination has a precision above 0.85. So I follow guideline1 and set both title’s threshold and authors’ threshold to 0.7.

**Publications_rule4**

<table>
<thead>
<tr>
<th>Round</th>
<th>title</th>
<th>booktitle</th>
<th>Basic Measures</th>
<th>Derived Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>25697</td>
<td>3401</td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td>0.3</td>
<td>26614</td>
<td>3486</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>0.3</td>
<td>26632</td>
<td>4388</td>
</tr>
</tbody>
</table>

For publications_rule4, even though the round 3’s precision is just above 0.85, the round 2’s precision increases a lot without decreasing recall by 1%. Besides, round2’s F1-measure is the highest. So I follow the guideline 2 and set title to 0.4 and booktitle to 0.3.

**Publications_rule6**

<table>
<thead>
<tr>
<th>Round</th>
<th>title</th>
<th>journal</th>
<th>Basic Measures</th>
<th>Derived Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>0.8</td>
<td>37037</td>
<td>1250</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>0.7</td>
<td>39570</td>
<td>1319</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>0.6</td>
<td>41122</td>
<td>2741</td>
</tr>
<tr>
<td>4</td>
<td>0.5</td>
<td>0.5</td>
<td>41164</td>
<td>4462</td>
</tr>
</tbody>
</table>

For publications_rule6, the results for precision are all very good and above 0.85. Even though the round 4’s precision is just above 0.85, the round 3’s precision increases a lot without decreasing recall by 1%. So for publications_rule6, thresholds for title and journal are all 0.6.

**Publications_rule5**

<table>
<thead>
<tr>
<th>Round</th>
<th>authors</th>
<th>pages</th>
<th>Basic Measures</th>
<th>Derived Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>1</td>
<td>0.6</td>
<td>0.5</td>
<td>38117</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>0.5</td>
<td>38970</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.4</td>
<td>0.4</td>
<td>41852</td>
<td>1907</td>
</tr>
<tr>
<td>4</td>
<td>0.3</td>
<td>0.3</td>
<td>40822</td>
<td>7676</td>
</tr>
</tbody>
</table>
For publications_rule5, as it is a hard positive rule, I should choose the threshold setting that makes true positive to 0. It means that all positive pairs identified by rule5 are meant to be matched. So I choose 0.5 as the threshold for both authors and pages.

5.4.2 Negative Rules

I have 4 negative rules for publication entity reference relation. To determine these negative rules, I build a clustering result based on the candidate pairs. Then I compare my clustering result with Cora_gold to see the differences. The comparison result is shown in Table 5.

<table>
<thead>
<tr>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>Precision</th>
<th>Recall</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>62184</td>
<td>16269</td>
<td>2394</td>
<td>0.7926</td>
<td>0.9629</td>
<td>0.8695</td>
</tr>
</tbody>
</table>

Table 5: Clustering Result based on Candidate Pairs

From the result, I found that FN is limited to a small number, which is good. However, false positive is quite a lot and made precision limited to 0.79. Thus, I did a case study to analyse the false positives and tried to find proper negative rules. The pairs of publications listed below are all false positives, which means they should be in different clusters by Cora_gold but the raw clustering result put them into the same cluster.

Case 1

<table>
<thead>
<tr>
<th>id</th>
<th>authors</th>
<th>title</th>
<th>pdate</th>
</tr>
</thead>
</table>

From this case, I can see that their titles are quite similar but actually they represent different publications. So there should be a table to record pairs of titles that looks similar but actually are different. This is similar to cannot-link constraints (Tung et al. 2001). A table will record two strings that look same but are actually different. When I compare strings, I will refer to this table and determine whether the two strings are non-match.

Case 2

<table>
<thead>
<tr>
<th>id</th>
<th>authors</th>
<th>title</th>
<th>pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>385</td>
<td>S. E. Fahlman and C. Lebiere</td>
<td>&quot; The cascade-correlation learning architecture, &quot;</td>
<td>2, pp. 524-532,</td>
</tr>
<tr>
<td>396</td>
<td>Fahlman, S.E.</td>
<td>The recurrent cascade-correlation architecture.</td>
<td>190-205. &quot;</td>
</tr>
</tbody>
</table>
In this case, the two publications’ pages are totally different. Thus, if the similarity of two publications’ pages is smaller than a threshold, then they are different publications.

Case 3

<table>
<thead>
<tr>
<th>id</th>
<th>authors</th>
<th>title</th>
<th>tech</th>
<th>booktitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>418</td>
<td>Scott E. Fahlman</td>
<td>The Recurrent Cascade-</td>
<td>Technical Report CMU-</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correlation Architecture,</td>
<td>CS-91-100,</td>
<td></td>
</tr>
<tr>
<td>413</td>
<td>Fahlman, Scott E.</td>
<td>The Recurrent Cascade-</td>
<td></td>
<td>in Advances in Neural</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Correlation Architecture</td>
<td></td>
<td>Information Processing Systems</td>
</tr>
</tbody>
</table>

In this case, the two publications look the same except that one has some value in tech and the other have some value in booktitle, which means one is technique report and the other is another format of publication. So if one publication has some value in tech and the other is a publication in different format, then they should be two different publications.

Case 4

<table>
<thead>
<tr>
<th>id</th>
<th>authors</th>
<th>title</th>
<th>journal</th>
<th>booktitle</th>
</tr>
</thead>
</table>

In this case, the two publications look the same except that one has journal value and the other has booktitle value, which means one is published in journal and the other is published in a conference workshop. So if one publication has journal value and the other is published in some Proceedings, then they should be two different publications.

5.4.3 Weights

The weights for all the different rules are measured from experiments on training dataset or historical data.

Weights of Positive Rules

The weight of positive rules should represent how accuracy the rule will cluster pairs. So the precision is adopted as the weight for soft positive rules.

As I mentioned in justification of positive rules in 5.4.1, I chose a threshold set for each of the rules and these threshold sets are based on their precisions. Since I used the threshold sets for my positive rules, I also should use the corresponding precisions as my rules’ weights.
So the final weights of the positive rules used in my experiments are as follows.

<table>
<thead>
<tr>
<th>Positive Rule</th>
<th>Weight (to 2dp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.88</td>
</tr>
<tr>
<td>2</td>
<td>0.97</td>
</tr>
<tr>
<td>3</td>
<td>0.85</td>
</tr>
<tr>
<td>4</td>
<td>0.88</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.94</td>
</tr>
</tbody>
</table>

Table 6: Weights of Positive Rules

Weights of Negative Rules

As mentioned before, if applying functions on raw data to measure the weights, it may take a long time. Thus the idea to measure the weights of negative rules is to measure the effects of negative rules on candidate pairs.

For the weights of negative rules, I first generate candidate pairs as mentioned before. Then I generate a raw clustering result based on the candidate pairs. Then I apply the negative rules on the candidate pairs respectively. The negative rule will separate a positive pair if it returns true. So after the negative rule, the number of positive pairs will be reduced but the clustering quality may increase.

Following this method, I apply negative rules on the candidate pairs to see the effectiveness. And I also need to build clustering results and compare them with Cora_gold.

The ideal result on measures is that there will be an increase on true positive and a decrease on false negative and false positive. If the negative rule has all these impacts, then its weight will be -1.

<table>
<thead>
<tr>
<th></th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>Diff-TP</th>
<th>Diff-FP</th>
<th>Diff-FN</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>original</td>
<td>62184</td>
<td>16269</td>
<td>2469</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative_rule1</td>
<td>62210</td>
<td>13102</td>
<td>2368</td>
<td>+26</td>
<td>-3167</td>
<td>-101</td>
<td>-1</td>
</tr>
<tr>
<td>Negative_rule2</td>
<td>62178</td>
<td>16229</td>
<td>2400</td>
<td>-6</td>
<td>-40</td>
<td>-69</td>
<td>-0.95</td>
</tr>
<tr>
<td>Negative_rule3</td>
<td>63685</td>
<td>7429</td>
<td>893</td>
<td>+1501</td>
<td>-5800</td>
<td>-1475</td>
<td>-1</td>
</tr>
<tr>
<td>Negative_rule4</td>
<td>62108</td>
<td>13628</td>
<td>2470</td>
<td>-76</td>
<td>-2641</td>
<td>+1</td>
<td>-0.97</td>
</tr>
</tbody>
</table>

Table 7: Weights for Negative Rules

“Diff-TP”, “Diff-FP” and “Diff-FN” represents the differences between original TP, FP, FN and TP, FP, FN with negative rules applied. “+” means there is an increase against the original number. “-” means there is a decrease against the original number.

From the result of experiment, we can see that negative_rule1 and negative_rule3 makes true positive increase, false positive decrease and false negative decrease, which means they have positive effects on all these three measures. So their weights are -1.
Negative_rule2 has a decrease on true positive, false positive and false negative, which means it has two positive effects and one negative effect on the three measures. So I calculate the weight by \(-\frac{40+69}{6+40+69} = -0.95\).

Negative_rule3 has a decrease on true positive, false positive and has an increase on false negative, which means it has one positive effect and two negative effects on the three measures. So I calculate the weight by \(-\frac{2641}{16+1+2641} = -0.97\).
Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this project, I develop a new framework for entity resolution in the presence of constraints based on Arasu et al. (2009)’s work. I adopted some terminologies from their work such as “entity reference relation” and “clustering relation”. However, I also denote some different types of rules.

This framework covers all five steps in entity resolution: data preprocessing, indexing, record comparison, classification and evaluation. In detail, data preprocessing shows the data normalization process and filling function. Indexing contains two steps: generate single rule based positive pairs and generate candidate pairs. The candidate pairs generated by indexing are widely used in my work including classification and evaluation. The classification includes the three main steps of my algorithm. The first step is a prediction phase. I soften all rules to soft rules and give each rule a weight. The weight score of a pair determines whether a pair is a positive pair or a negative pair. In this step, I try to include more positive pairs to increase the recall. The second step is to validate the correctness of classification held by the first step. I harden some of the rules in the first step. In this step, I try to increase both precision and recall through the implementation of hard rules. The third step is a conflicts resolving phase. I solve the conflicts between current clustering result and hard negative pairs through separating clusters with the help of weight.

The experiment results show that there is an improvement in effectiveness after each step in classification method (4.5). My method gives a high precision and recall on clustering result. For the relationships between entity reference relations, weights determine the functional dependencies between different relations. In my work, relation with high average weight of positive rules will affect other relations. When the dependencies are determined, the quality of clustering of one entity reference relation will affect the quality of clustering of others.
6.2 Future Work

As the Cora dataset only have a small number of records. Future work may need to apply the framework on a big dataset to measure the efficiency of the method, especially I cannot find some real hard negative rules.

In my work, positive rules are all based on string similarity function. So in the future work, more sophisticated positive rules can be implemented. Especially rules across multiple tables can be used for collective entity resolution.

For the relationships between different entity reference relations, I only consider mono-direction between relations. In reality, the effects between relations should be mutual effects. The future work can investigate a method to connect relations more closely.
Bibliography


Appendix: Software Structure

Graphical Representation

Data Preprocessing
Indexing
Classification Method