Journal of Machine Learning Research 15 (2014) 3037-3064 Submitted 3/14; Published 10/14

Effective String Processing and Matching for Author Disambiguation

Wei-Sheng Chin D01944006@NTU.EDU.TW
Yong Zhuang R01922139@NTU.EDU.TW
Yu-Chin Juan R01922136@NTU.EDU.TW
Felix Wu B99902090@NTU.EDU.TW
Hsiao-Yu Tung B98901044@NTU.EDU.TW
Tong Yu R01922141@NTU.EDU.TW
Jui-Pin Wang R01922165@NTU.EDU.TW
Cheng-Xia Chang R01944041@NTU.EDU.TW
Chun-Pai Yang R99902109@NTU.EDU.TW
Wei-Cheng Chang B99902019@NTU.EDU.TW
Kuan-Hao Huang B99902059@NTU.EDU.TW
Tzu-Ming Kuo B99902073@NTU.EDU.TW
Shan-Wei Lin B99902023@NTU.EDU.TW
Young-San Lin B97902055@NTU.EDU.TW
Yu-Chen Lu B98902105@NTU.EDU.TW
Yu-Chuan Su R01922159@NTU.EDU.TW
Cheng-Kuang Wei B98901037@NTU.EDU.TW
Tu-Chun Yin D00922023@NTU.EDU.TW
Chun-Liang Li R01922001@NTU.EDU.TW
Ting-Wei Lin R01944011@NTU.EDU.TW
Cheng-Hao Tsai R01922025@NTU.EDU.TW
Shou-De Lin SDLIN@CSIE.NTU.EDU.TW
Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Chih-Jen Lin CJLIN@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan

Editors: Senjuti Basu Roy, Vani Mandava, Martine De Cock

Abstract

Track 2 of KDD Cup 2013 aims at determining duplicated authors in a data set from Microsoft Academic Search. This type of problems appears in many large-scale applications that compile information from different sources. This paper describes our solution developed at National Taiwan University to win the first prize of the competition. We propose an effective name matching framework and realize two implementations. An important strategy in our approach is to consider Chinese and non-Chinese names separately because of their different naming conventions. Post-processing including merging results of two predictions further boosts the performance. Our approach achieves F1-score 0.99202 on the private leader board, while 0.99195 on the public leader board.

Keywords: deduplication, author disambiguation, name matching

1. Introduction

Track 2 in KDD Cup 2013 is a task of name disambiguation. Ideally, a name uniquely identifies an entity (e.g., an author), but practically an entity may correspond to different names. For example, once two data sets assigning an entity two or more names are integrated, the entity may become associated with different names. In this article, we call these names as *duplicates* of the original entity.

The data set is offered by Microsoft Academic Search (MAS). As a search engine, MAS integrates information of authors and their publications from different sources. We have mentioned that duplicates more frequently occur when data sets are integrated to a larger one, so MAS naturally suffers from this issue. The aim of this competition is to find which of around 250,000 authors are duplicates. We are given seven files, `Author.csv`, `Paper.csv`, `PaperAuthor.csv`, `Conference.csv`, `Journal.csv`, `Train.csv`, and `Valid.csv`. The two more important ones are `Author.csv` and `PaperAuthor.csv`, where the former stores author information (e.g., names and identifiers), and the latter gives authorships for around two million publications. Each line, a record, in both `Author.csv` and `PaperAuthor.csv` includes an author identifier and a name. The task is to upload duplicated identifiers in `Author.csv`. Other details of data sets and the competition can be found in Roy et al. (2013). Because no training information is given, the problem is an unsupervised learning task. The evaluation measure is the average of F1-scores over all authors in `Author.csv`. The definition of F1-score is

$$\text{F1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

where

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \text{ and}$$
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negative}}.$$

For example, to find author A's duplicates, if our algorithm returns A, B, C, F, and the true duplicates are A, B, C, D, E, then

$$\text{precision} = \frac{|\{A,B,C\}|}{|\{A,B,C\}| + |\{F\}|} = \frac{3}{4}, \text{ and}$$
$$\text{recall} = \frac{|\{A,B,C\}|}{|\{A,B,C\}| + |\{D,E\}|} = \frac{3}{5}.$$

In the competition, 20% of authors in `Author.csv` are used to evaluate duplicates submitted by participants. We refer to them as results on the public leader board. For the final evaluation, the remaining 80% authors in `Author.csv` are used and the F1-scores are called results on the private leader board. The baseline F1-score on the public leader board is 0.94411 by assuming no duplicates (i.e., all author names are considered as different entities).

Author disambiguation is a difficult problem that is also known as entity resolution (Whang et al., 2013; Köpcke and Rahm, 2010), duplication elimination/detection (Ananthakrishna et al., 2002), object matching (Köpcke et al., 2010), and record linkage (Brizan and Tansel, 2006). Generally speaking, this problem includes two fundamental phases, similarity measurement and record grouping; if the similarity is high enough, then two records should be assigned to a group corresponding to one real entity. Note that a record in a data set refers to a real-world object; for example, in the case of author disambiguation, a record refers to an author of a paper. It has been pointed out by, for example, Köpcke et al. (2010) and Bilenko et al. (2003), that algorithms well-tuned for some data sets may not perform well on other data sets. Thus, we should carefully design suitable measurements and tune parameters for distinct applications. Some data sets intrinsically have a hierarchical structure; for example, customer data sets may contain detailed addresses so selling records can be divided into several subsets according to the countries, states, and cities. Ananthakrishna et al. (2002) proposed a disambiguation method to find duplicates by using hierarchical information. That is, two records are referred to one real-world object if they are considered as duplicates at each level of the hierarchy. In Bhattacharya and Getoor (2004), the authors iteratively merge a pair of highly similar records to be duplicates for reducing the false positive rate using more conservative strategies (e.g., two records are merged if their similarity is larger than a threshold). In each iteration, the duplicates found are used to evaluate the distance between two entries; i.e., the distance between entries depends on the overlap between duplicates found in previous iterations of the two entries. In Bilenko et al. (2003), support vector machine (SVM) is used to integrate measurements on multiple attributes. For example, similarity between two authors can be the weighting sum of Jaro distances between their first names, middle names, last names, and addresses, where the weights correspond to the coefficients of SVM. They report that with the existence of typos the string-based distances (e.g., Levenshtein distance and Jaro distances) are superior to the token-based distances (e.g., Jaccard similarity and cosine distance with TF-IDF). Because typos and other types of noise exist in KDD Cup 2013 track 2 (see Section 2.2), we can expect the difficulty of using token-based techniques. Whang and Garcia-Molina (2012) proposed an iterative framework to solve disambiguation problems when there are different types of entries in the data set. Their framework allows users to set the logical relations between these types, and automatically find the best order to conduct disambiguation for different types. Note that different types of disambiguation can be handled in parallel if they are logically independent (i.e., the result of one type of disambiguation does not provide additional information for the disambiguation of another type). Take the KDD Cup 2013 as an example. The types of entries include author, publication, conference, journal, etc. Obviously, a parallel disambiguation for conferences and journals is possible, but the disambiguation of publications and conferences should be ordered. Treeratpituk and Giles (2009) provide rich measurements on many attributes (e.g., the last name of authors, authors' affiliation and coauthorship) to judge whether two authors correspond to one entity. In KDD Cup 2013, due to missing values, noises, and insufficient information we can not directly generate reliable features using their measurements. Consequently, similar to Torvik and Smalheiser (2009); Bilenko et al. (2003), we consider author-name strings as the major sources to measure the similarity between two authors. Thus all strategies in our algorithm are highly related to string processing. Furthermore, our approach is related to the iterative

Filename	Fields in each line	#Entries
`Author.csv`	AuthorID, Name, Affiliation	247,203
`PaperAuthor.csv`	PaperID, AuthorID, AuthorName, AuthorAffiliation	12,776,670
`Paper.csv`	PaperID, Title, ConferenceID, JournalID, Keyword	2,267,542
`Conference.csv`	ConferenceID, ShortName, FullName, Homepage	4,544
`Journal.csv`	JournalID, ShortName, FullName, Homepage	15,150

Table 1: Brief description of each file.

scheme proposed by Bhattacharya and Getoor (2004) because we also iteratively identify reliable duplicates and then refine results.

The next section summarizes the noise sources so that we can further understand the difficulties in the competition. We describe our approach in Section 3. It is realized in our two implementations described in Sections 4 and 5, respectively. Section 6 proposes a strategy to ensemble the predictions from these two implementations. We handle typographical errors (typos) in Section 7. Our results are summarized in Section 8, and the last section provides a detailed comparison on several successful approaches in this competition. Section 10 concludes this paper and gives some possible directions. Besides, our implementations are available at

`https://github.com/kdd-cup-2013-ntu/track2`.

A preliminary version of this work appears in a conference paper (Chin et al., 2013). The main extensions include three new sections, Section 2, Section 9, and Section 10, and the provision of more details in many places.

2. An Overview of Data sets

In this section, we firstly give a brief introduction about the content in the data set. Then we discuss various types of noise observed in the competition data.

2.1 Data Set Description

The organizers of KDD Cup 2013 release seven files, `Author.csv`, `PaperAuthor.csv`, `Conference.csv`, `Journal.csv`, `Paper.csv`, `Train.csv`, and `Valid.csv`. We do not use `Train.csv` and `Valid.csv` because they are for Track 1. Details of other files are shown in Table 1.

2.2 Noisy Patterns in the Competition Data

A search engine like MAS integrates relationships between authors and papers from different sources. What MAS must do is parsing and merging, but this process of integration results in some unexpected errors. By careful analysis, we observe the following types of noisy data from the sets given by MAS.

- **Alleviation**: Two entries in `Author.csv` are "2071403, Yuri Gurevichy" and "926979, Yuri Gurevichyand." At first glance, they are different persons. However, if we eliminate the last 3 characters "and" from the second author, the resulting author "Yuri Gurevichy"

may be the same as the first one. It is hard to decide because the two may actually be different.

- **Unusual Name**: The entry "894651, 560263, A Case Study," may provide incorrect author information because "A Case Study" is not a common name. However, we can not rule out the possibility that it is someone's name. Besides, in `Author.csv`, an entry "1813899, Ming-Wei Chang Dan Roth" appears. Apparently, this entry concatenates two author names.
- **Inconsistent Information**: In `Author.csv`, we get an entry "1791054, Steven L. Salzberg," which means the 1791054th author is "Steven L. Salzberg." However, in `PaperAuthor.csv`, the entry "310678, 1791054, David Saunders" indicates that the 1791054th author is "David Saunders." Therefore, the information in `Author.csv` and PaperAuthor.csv is not consistent.
- **Typo**: Some names in `Author` may contain typos. For example, "Chih-Jen Lin" becomes "Chi-Jen Lin."
- **Incomplete Name**: For the names "Michael Jordan," "M. I. Jordan," and "Michael I. Jordan," some authors have middle names, while others do not. Besides, for the name "M. I. Jordan," all words except the last name are abbreviated. Then it is difficult to decide if "Michael Jordan" and "Michael I. Jordan" correspond to the same author.
- **Empty Entry**: For example, the 14143th and 56473th entries in `Author.csv` are empty, so we cannot obtain any information.
- **Missing Value**: Some information of an entry is missing. In `PaperAuthor.csv`, each entry contains "PaperID, AuthorID, AuthorName, AuthorAffiliation." However, for the entry "127675, 1360414, Chih-Jen Lin" in `PaperAuthor.csv`, it only tells us that the 1360414th author named Chih-Jen Lin writes the 127675th paper. The affiliation of the author is missing.
- **Nickname**: Instead of using real names, some authors use nicknames in their publications. For example, "532490, 1060199, William R. Gates" and "717206, 1060199, Bill Gates" may be the same author, although the latter uses the nickname "Bill."
- **Wrong matching between authors and papers**: Errors of matching authors and papers exist in `PaperAuthor.csv`. From the entry "1360414, Chih-Jen Lin, National Taiwan University" in `Author.csv`, we check papers written by him in `PaperAuthor.csv`. Although most obtained entries are his papers, some are not. For example, a record "674977, 1360414, Chih-jen Lin" indicates that "Chih-jen Lin" writes the 674977th paper. However, the 674977th paper has the title "Built-In Current Sensor for IDDQ Test in CMOS," and is not written by the "Chih-Jen Lin" from National Taiwan University.
- **Non-English characters**: Some non-English characters like "43416, J. Jürjens" cause difficulties to compare names.

We carefully take these noisy patterns into consideration in designing our approach.

3. Our Approach

In this section, we discuss three key strategies of our approach, and then introduce a framework based on these strategies. Two implementations of the framework were finished by two groups within the team. They are respectively described in Sections 4 and 5.

3.1 The Main Strategies

The first strategy is that we identify duplicates mainly based on string matching; in particular, name matching. Academic authors often use their real names. If two authors are duplicates, then their name strings are similar. Therefore, name matching is very effective for this problem.

The second strategy is that if an author in `Author.csv` has no publication records in `PaperAuthor.csv`, then we assume that this author has no duplicates. In our earlier experiments, this strategy significantly improves the F1-score by around 0.02. Apparently MAS implements this rule internally, so we admit that this strategy may not be useful for other data sets.

The third strategy is to classify an author as Chinese or non-Chinese before any string matching. This strategy is useful because Chinese and non-Chinese names have some crucial differences. First, a western name may be written without the middle name. For example, "Vladimir Naumovich Vapnik" may write his name as "Vladimir Vapnik." In contrast, Chinese generally do not omit any part of their name. For instance, "Chih Lin" and "Chih Jen Lin" are almost surely different authors. Second, Chinese last names provide much less information than non-Chinese ones because some Chinese last names are very common (e.g., "Wang" and "Lin") and the romanization process may map different last names to the same English word (e.g., "林" and "藺" are romanized to "Lin"). The common last names cause difficulties to analyze a shortened Chinese name. For example, there are much more names that can be shortened as "C. J. Lin" than those as "E. W. Dijkstra." Besides, Korean, Vietnamese, and Singaporean names have a similar structure (i.e., two-word first name and a shortened last name), so we include them along with Chinese names. Examples include "Chi Minh Ho" (Vietnamese), "Yong Jun Bae" (Korean), and "Hsien Loong Lee" (Singaporean). Interestingly, we do not consider Japanese names because they often have a longer last name and a one-word first name (e.g., "Ichiro Suzuki").

For easy description, in this paper, we categorize words to two classes, shortened words and full words. A word is considered shortened if it is one single character or includes a period. For example, "C" and "Chris." are shortened words. A word is a full word if it is not a shortened word. Similarly, we consider two types of names. A name should be a shortened name if it consists of at least one shortened word; otherwise, the name is a full name.

3.2 The Framework

Our framework can be divided into six stages. Here we briefly introduce each stage, but leave details in Sections 4 and 5. Note that if not specified, in the sequel "an author" refers to an unique identifier (rather than a name and a human).

1. **Chinese-or-not**. We classify each author as Chinese or non-Chinese.
2. **Cleaning**. To efficiently compare author names, we remove redundant information. For example, "CHIH JEN LIN" is likely to be a duplicate of "chih jen lin," so we lowercase all strings.
3. **Selection**. For each author we select some candidates of possible duplicates. Naively, an author should be compared with all other authors, but the computational cost is high.

Therefore, we select only some candidates for subsequent analysis. At this stage, recall is the main concern because any missed names cannot be recovered in a later stage.

4. **Identification**. For each author, we check if those in the candidate set are duplicates or not.
5. **Splitting**. Because some names are wrongly grouped together after the identification stage, we make corrections by splitting some of them.
6. **Linking**. This stage maps author names back to author identifiers. This step is needed only for our second implementation; see more explanation below.

We illustrate an important difference between our two approaches by the following example.

Author identifier	1001
Name in `Author.csv`	"Chih Jen Lin"
Names in `PaperAuthor.csv`	"C. J. Lin" "Chih Jen Peter Lin" "C. J. P. Lin"

This table shows that, in the given data, one identifier may correspond to different names. Following the competition requirement to find duplicates of each author identifier, in the first implementation, each author is referred to as an author identifier. However, the second implementation treats each name string as an author. That is, the four names in the above examples are considered different. The algorithm must group them together among all name strings. Therefore, the second implementation needs a linking stage in the end.

4. The First Implementation

In this section, we discuss details of the first implementation. Each sub-section corresponds to a stage of the framework. Note that the linking stage is not performed in this implementation.

4.1 Chinese-or-not

An author is classified to be either Chinese or non-Chinese by the flowchart in Figure 1. The process relies on information including the romanization of common Chinese last names and Chinese syllables. We build two dictionaries, which differ in the coverage of Chinese words. The first dictionary contains 2,381 English words that are the romanization of top 100 Chinese last names and 410 syllables. Each Chinese last name is romanized to four English words because there are four different romanization systems in Taiwan,[1] and the list of Chinese syllables is publicly available on Wikipedia.[2] Note that romanization of several Chinese words can be the same. The second dictionary extends from the first by including 853 additional words of the romanization of 406 Chinese last names and 20 Korean last names.[3] Moreover, each dictionary consists of two sub-dictionaries storing last names

1. The romanization tool we used can be found at `http://www.boca.gov.tw/content?CuItem=5609&mp=1`.
2. These Chinese syllables can be found at `http://en.wikipedia.org/wiki/Comparison_of_Chinese_romanization_systems` and `http://www.pinyin.info/romanization/compare/gwoyeu_romatzyh.html`.
3. The additional Chinese last names and the Korea last names are available at `https://en.wikipedia.org/wiki/List_of_common_Chinese_surnames` and `http://mirror.enha.kr/wiki`, respectively

and syllables respectively. Some words in our Chinese dictionary also commonly appear in non-Chinese names (e.g., "van," "den," and "ben"), so we construct a "banned list" according our own knowledge.

The flow to determine if an input name is Chinese or not is in Figure 1. In this figure, hexagons are the final decision of Chinese or non-Chinese, ellipses stand for variables, and rectangles are operations. Every counter directly counts the number of inputs. For example, Counter_2 gives the total number of Chinese words in a name after checking the dictionary. To begin, a name is cleaned and then tokenized. Note that this cleaning step only removes non-alphabet characters and lowercase all remaining letters because we assume that punctuation provides no useful information and the matching between two words should be case-insensitive. Next, we consider three cases according to the number of full words (0, 1, or more) in an author name.

1. If there is no full word in a name (upper sub-tree in Figure 1), words in our Chinese dictionary cannot be used and hence the author is classified as non-Chinese. For example, "C J L" is considered as non-Chinese.
2. If an author has only one full word (right sub-tree in Figure 1), then we consider the author as Chinese if the full word is one of the last names in our dictionary but not in the banned list. This case is mainly for detecting Chinese authors with abbreviated first name such as "C. J. Lin," "C.-J. Lin," and "C. J. P. Lin."
3. For a name with more than one full word (left sub-tree in Figure 1), if it has more than one full word not in our Chinese dictionary, then the name is considered as non-Chinese. For example, "Chih J. Peter Lin" is Chinese if "Peter" is the only full word not in our dictionary. In contrast, if our dictionary does not contain "Chih," then the name becomes non-Chinese because of having two non-Chinese full words. The banned list plays a similar role as in the previous rule; if a word is on the list, then the word is not considered as a Chinese word. However, an exception of not applying the banned list is when the name contains a full Chinese word. This exception helps to recognize "Dan Wang" as a Chinese name and "Dan Roth" as non-Chinese. In detail, "Dan" is regarded as a Chinese word because "Wang" is a Chinese word. However, the other "Dan" is not a Chinese word because "Roth" is not. This idea can be found in the blocking mechanism of Counter_3; that is, if there is at least one Chinese word, the output of Counter_3 is the number of banned words. Otherwise, the output of Counter_3 is 0.

We illustrate our Chinese classification using an author with four full words. Given "Chih Jen Dean H. Lin," "H." is removed first, and then four words "Chih," "Jen," "Dean," and "Lin" are obtained. Counter_1 in Figure 1 returns 4 because of four full words. Then we use the left sub-tree in Figure 1. Assume that

1. our Chinese dictionary contains common last names {lin, wang} and common Chinese words {wang, chih, dean, and jen}, and
2. the banned list is {dean}.

Counter_2 returns 3 for one match of the last name "lin" and two matches of syllables "chih" and "jen." Because the output of Counter_2 is larger than 0 (some Chinese full words are found), Counter_3 is activated and returns 1 for the match of "dean." Then the adder returns 4. Finally, the subtracter, which subtracts the total number of full words from the result returned by the adder, returns 0 to be the number of non-Chinese full words, so "Chih Jen Dean H. Lin" is classified as Chinese.

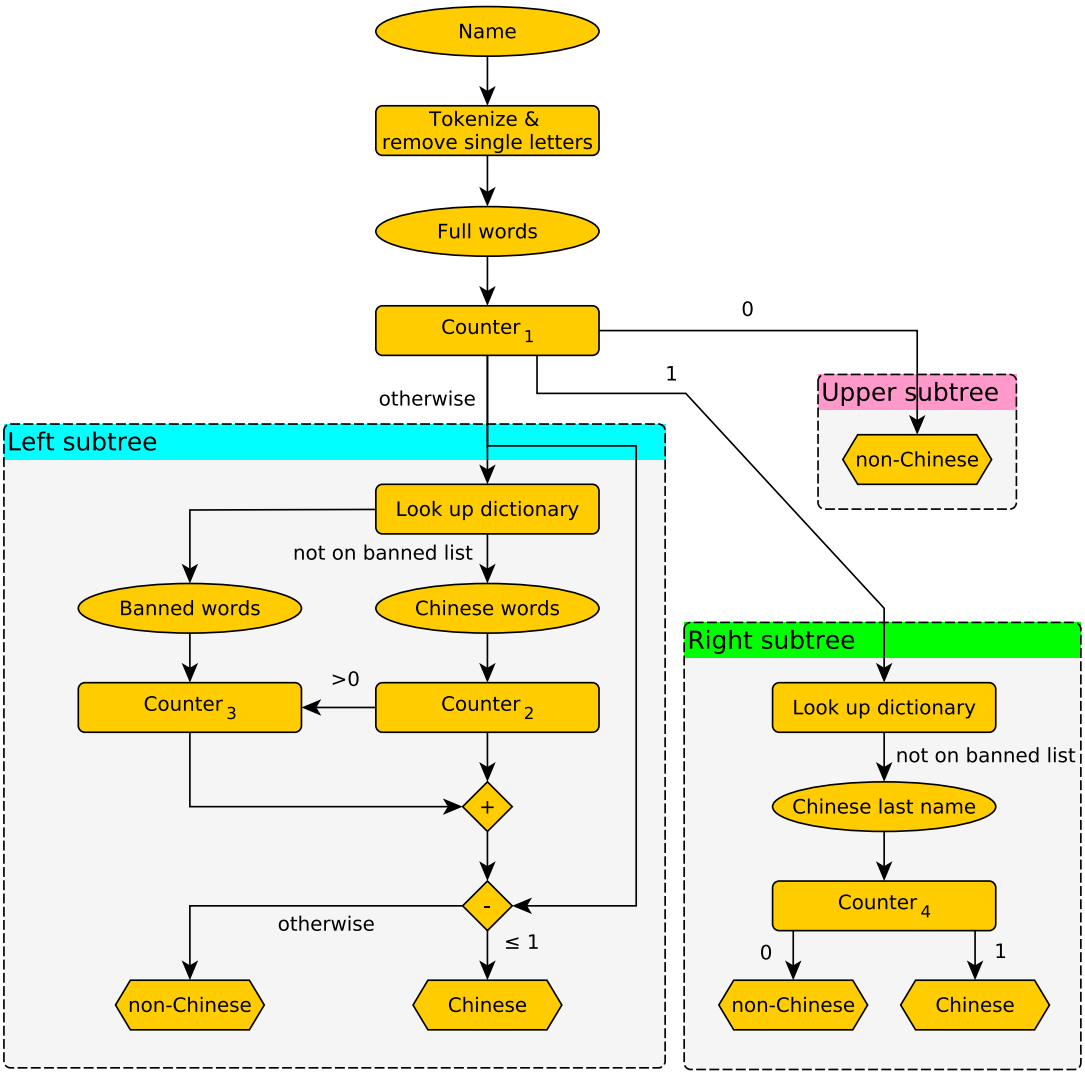

Figure 1: Flow of finding if a name is Chinese or not.

4.2 Cleaning

The purpose of data cleaning is to make name comparisons more accurate.[4] This process consists of the following steps.

1. Split two consecutive uppercase characters because we suspect that each character is an abbreviation of a word. For instance, we replace "CJ" with "C J."
2. Remove English honorifics (e.g., "Mr." and "Dr."), and some suffixes such as "Jr." and "III."
3. Transform uppercase to lowercase.

4. Note that in Section 4.1 a simpler cleaning step was conducted before identifying Chinese names.

4. Remove apostrophes and replace punctuation. For example, "o'relly" becomes "orelly." We then replace punctuation except periods with blanks. For example, "Chih-Jen" becomes "Chih Jen." We keep the period because it is useful to determine if a word is shortened or not.
5. Replace European alphabets with similar English alphabets. For example, we replace "ä" with "a."
6. Replace common English nicknames.[5] For example, we replace "bill" with "william."

4.3 Selection

In this stage, for each author, a set of possible duplicates are obtained. The purpose is to reduce the quadratic complexity of subsequent string matchings to linear. Our method includes two phases. In the first phase, we build a dictionary of (key, value) pairs. Each key is a set of words, while each value is a set of authors containing the key. To generate keys, we consider all word combinations of an author name. For example, "Chih Jen Lin" leads to the following keys.

"Chih"	"Jen"	"Lin"
"Chih Jen"	"Jen Lin"	"Chih Lin"
"Chih Jen Lin"		

Note that the order does not matter, so "Chih Lin" is considered the same as "Lin Chih." To avoid too many keys, we do not consider the combination of more than four words. In the second phase, for each given author, we examine his/her (key, value) pairs. If the "value" contains no more than 17 authors, then they are included as possible duplicates. The reason to discard a "value" with too many authors is because the corresponding key is too common. For example, two authors shall not be suspected as duplicates merely because their first name is "Lin." Including these names does not improve the recall much, but significantly increases the running time. In our experience, changing the threshold from 17 to 100 results in a three-fold increase of the running time. Moreover, a higher recall may not lead to a better final result.

4.4 Identification

Because the criteria to select candidates in the previous stage is loose, many authors were wrongly selected. In this stage, we find a subset as duplicates by applying a main procedure and two additional procedures. The main procedure, described in Section 4.4.1, uses many matching functions listed in Section 4.4.2. The two additional procedures are described in Section 4.4.3.

4.4.1 The Main Procedure for Identification

We iteratively apply matching functions to identify duplicates from the candidate set. Each matching function checks if two given authors are similar to each other. Criteria used in matching functions here should be stricter than those in the previous stage because the aim

5. The list of all nicknames we considered is available at `http://www.cc.kyoto-su.ac.jp/~trobb/nicklist.html` and `http://mentalfloss.com/article/24761/origins-10-nicknames`.

is to remove authors that were wrongly selected. However, although a strict criterion gives a high precision, it many cause a low recall. Therefore, we sequentially apply matching functions (listed in Section 4.4.2) from the strictest to the loosest.

Each iteration consists of two steps. First, between an author and any member of its candidate set, a matching function returns if the two names are duplicates or not. For a name "Chih Jen Lin," if his candidate set includes

"Chih Jen Lin," "Lin Chih Jen," "Chih Jen M. Lin," and "Chih Jen K. Lin,"

and the matching function requires that two names have the same word set, then "Chih Jen Lin" and "Lin Chih Jen" are considered as duplicates.

In the second step of an iteration, we make some corrections because duplicates obtained after applying matching functions may be wrong. For example, assume the following duplicates have been identified.

"C. J. Lin," "Chih Jen Lin," and "Chen Ju Lin."

Obviously they are not the same author because "Chih Jen Lin" and "Chen Ju Lin" are very different. We design a dry-run procedure to detect if a set of selected duplicates contains some very different names. If such names exist, then the set is discarded. For describing the dry-run function, we say that two author names are *loosely identical* if one of the following conditions holds.

1. One author has at least a shortened word and one author's first-character set of words is a subset of the other.
2. Both authors contain full words only and one author's first-three-character set of words is a subset of the other.

For example, "C J Lin" and "Chih Jen Lion" are loosely identical, while "Chih Jen Lin" and "Chen Ju Lin" are not.

In the dry-run procedure, we divide the selected set of duplicates to two sets: the longest-name set and the shorter-name set, where the former contains names with the largest number of words, while the latter contains the rest. The dry-run procedure is passed if the following conditions hold.

1. In the longest-name set, any two names are loosely identical.
2. In the shorter-name set, any name is loosely identical to at least one name in the longest-name set.

The identification procedure is shown in Algorithm 1.

4.4.2 Matching Functions

For easy description, we define the following relationship between two author names.

1. **The same name:** Two names share the same set of words.
   Example: "Chih Jen Lin" and "Lin Chih Jen."
2. **A shortened name:** The first author is a shortened name of the second one if the following conditions hold.
   - The full-word set of the first is a subset of the second.
   - Each shortened word in the first name is the prefix of a word in the second.

   Example: "Ch. J. Lin" and "Chih Jen Lin."

```
Data: Each author has a set of candidates.
Result: All authors are split to groups of duplicates.
begin
    for a ∈ all authors do
        set {a} as a's set of duplicates
    end
    for f ∈ matching functions do
        for a ∈ all authors do
            P ← {duplicates already identified for a}
            for c ∈ {candidates of a} do
                if f(a, c) then
                    P ← P ∪ {c's identified duplicates}
                end
            end
            if P passes the dry-run procedure then
                all authors in P are considered as duplicates
            end
        end
    end
end
```

Algorithm 1: The main procedure in the identification stage of the first implementation.

3. **A partially shortened name:** The first author is a shortened name of the second and each word in the first name is the prefix of a word in the second.
   Example: "C. J. Lin" and "C. J. Lint."
4. **Alias:** Name A in `PaperAuthor.csv` is an alias of name B in `Author.csv` if A and B have the same author identifiers, and B is a shortened name of A.
   Example: "C. J. Lin" is in `Author.csv`, while "C. Jen Lin" and "Chih Jen Lin" are in `PaperAuthor.csv`. If they have the same author identifiers, then "C. Jen Lin" and "Chih Jen Lin" are aliases of "C. J. Lin."

Now we introduce the following matching functions, each of which checks two input names.

1. Two authors have the same words in their names.
   Example: "Chih Jen Lin," and "Lin Chih Jen."
2. One is a shortened name of the other and both have the same initial-character set of words. However, this rule is not applied if
   - both authors are Chinese and each has at least one shortened word, or
   - one of the authors is Chinese and contains no more than two words.

   Example: "John S. Nash" and "John Smith Nash."
3. (Non-Chinese only) One author name is a shortened name of the other.
   Example: "Michael Jordan" and "Michael I. Jordan."
4. (Non-Chinese only) One author name is a partially shortened name of the other.
   Example: "Marek J. Druzdze" and "Marek J. Druzduzel."

5. Two authors have at least one identical alias.
   Example: Suppose that the name "1273890, Thomas J. Webb" in `PaperAuthor.csv` is an alias of the author "1273890, Thomas Webb" in `Author.csv`, while "207564, Thomas J. Webb" is an alias of the author "207564, Thomas J. Webb." Then "207564, Thomas J. Webb" and "1273890, Thomas Webb" are considered as duplicates.
6. (Non-Chinese only) Two author names are loosely identical and both have at least one identical paper or affiliation.
   Example: "571595, Alex Pentland" and "993869, Alex Pentland Perceptual" are loosely identical and both have the same affiliation "MIT Media Laboratory."
7. The two authors satisfy the following conditions.
   - Each author name has at least three words.
   - Two author names are the same after removing their respective last word.
   - The last word of each name should be the same after removing the last character of the longer word.

   Example: "Chih Jen Linu" and "Chih Jen Lin."
8. (Non-Chinese only) Only one author name has middle name and their last names differ in the last two characters.
   Example: "Michael I. Jordan" and "Michael Jordann."
9. (Chinese only) Two authors have at least one identical alias and one identical affiliation.
   Example: Assume "Chih Jen Lin" in `Author.csv` has the same identifier with "Chih Jen Lin" and "C. J. Lin" in `PaperAuthor.csv`. Similarly, "Chih J. Lin" has "Chih Jen Lin" and "C. J. Lin" in `PaperAuthor.csv`.
   The two authors have the same alias "C. J. Lin." If both have the same affiliation, then they are considered as duplicates.
10. (Chinese only) Each author has at least three words, but has no shortened words. Moreover, one author's word set is a subset of the other.
    Example: "Michael Chih Jen Lin" and "Chih Jen Lin."
11. (Chinese only) Both author names have more than three words and their lists of initial characters are the same.
    Example: "Michael Chih Jen Lin" and "M. Chih Jen Lin." Their lists of initial characters are both "m c j l."
12. (Chinese only) Both author names have more than three words, neither has a shortened word, and the full-word set of one's name is a subset of the other.
    Example: "Michael Chih Jen Lin" and "Michael Jackson Chih Jen Lin."
13. (Chinese only) The two authors satisfy one of the following conditions.
    - Each has three words and both have the same list of initial characters.
    - Neither has a shortened word and one author's full-word set is a subset of the other.

    Example: "Lin Chih Jen" and "C. Jen Lin."

4.4.3 Additional Procedures for Identification

We conduct two additional procedures to improve the identification of duplicates. Instead of fitting them to the main procedure, we find that separately considering them is more suitable.

Same paper title: Because data are collected from different sources, some papers have an identical title after removing non-alphabet characters. For any two such papers, if an author of one paper is a partially shortened name of an author of the other paper, then the two authors are considered as duplicates. For example, assume two papers are shown in Table 2. "C. C. Chang" is a partially shortened name of "Chih-Chung Chang," while "C. J. Lin" is that of "Chih J. Lin" and "Chih Jen Lin." Therefore, authors 1 and 2 are regarded as duplicates, and so are authors 3 and 5.

Paper IDs	1	2
PaperName	LIBSVM	lib-svm
AuthorList	1, C. C. Chang 3, Chih Jen Lin 5, C. J. Lin	2, Chih-Chung Chang 3, Chih J. Lin

Table 2: An example of using papers with the same title to identify duplicates.

Parsing alleviation: Because of incorrect string parsing, some names such as "Chih JenLin," and "Chih Jen LinAN" may appear although the correct one is "Chih Jen Lin." To find duplicates for these ill-formed names, we map each name to some keys and group names with the same key as duplicates.

To obtain keys and identify duplicates, we run two separate steps. The first one uses two keys. After some duplicates have been identified, the second uses three keys to get more duplicates.

The step of using two keys generates keys for each author. Words in a name are concatenated and lowercased to get the first key. By removing the last character of this key, we generate the second key if one of the following conditions holds.

1. The name has one shortened and two full words.
2. The name has more than two full words and the length of the name is greater than 12.
3. The length of each word in the name is greater than four.

For example, "Chih Jen Lin" has a key "chihjenlin," while "Chih J. Lint" has keys "chihjlint" and "chihjlin."

In the step of using three keys, the first key is the same as that in the previous step. The second key is also by removing the last character of the first key, but it is generated if one of the four conditions holds. These four conditions are similar to the three conditions used in generating the second key in the case of using two keys except that the first condition is modified to the following two rules:

1. The name has one shortened and two full words. Moreover, the length of the last word is greater than four.
2. The name has more than two full words and the length of the last name is greater than four.

Then we remove the last two characters of the first key to get a new key if one of two conditions holds.

1. The length of the name is greater than 15 and that of the last word is greater than five.
2. The number of full word in one's name is greater than two and the length of the last word is greater than five.

For example, "Petra QuillfeldtA" has keys "petraquillfeldta," "petraquillfeldt," and "petraquillfeld."

4.5 Splitting

In some groups of duplicates that have been identified, we still observe very different author names. For example, the following authors are considered as duplicates after the identification stage.

"k. kobayashi" "keven w. kobayashi"
"kazuo kobayashi" "kunikazu kobayashi"

When the third matching function mentioned in Section 4.4.2 is applied to the author "k. kobayashi," the above four authors are considered as duplicates. Then because "keven w. kobayashi" is the longest name in the set, and "kazuo kobayashi" and "kunikazu kobayashi" are loosely identical to "keven w. kobayashi," they pass the dry-run function. Obviously the grouping is incorrect. Therefore, in this splitting stage, we check the number of incorrectly identified pairs (details described below) in every set of duplicates. If the number exceeds three, then we dissolve the group and each element goes back to be an individual. We say two authors are incorrectly identified if they satisfy that
1. Each author name is a full name with two words.
2. Neither author name is a partially shortened name of the other.
For example, "kazuo kobayashi" and "kunikazu kobayashi" satisfy the above criteria because "kazno" is not a prefix of "kunikazu" and vice versa. We do not consider names with more than two words because "Alex Pentland Perceptual" and "Alexander Pentland" may be unexpectedly treated as incorrect duplicates.

5. The Second Implementation

In this section, we introduce the second implementation following the framework in Section 3. As mentioned in Section 3.2, we treat all names in `PaperAuthor.csv` and `Author.csv` as individuals and find groups of duplicates. Only in the end we obtain groups of author identifiers as requested by the competition.

5.1 Chinese-or-not

In contrast to Section 4.1, the implementation is simpler here. We build a Chinese dictionary that consists of 694 words of romanized Chinese syllables[6] and a banned list including some manually-selected words such as "ben", "dan", and "long."[7] For each author name after tokenization, we check if any word is on the Chinese list but not on the banned list. If such a word exists, we classify the name as Chinese. Our Chinese-or-not techniques were developed by two groups independently, so the dictionaries and the banned lists are slightly different while their designing spirits are identical. Note that the dictionary in the second

6. Related information can be found at `http://www.chineseinla.com/lastname/key_ng.html` and `http://irw.ncut.edu.tw/general/chen813`.
7. For the full list, see `https://github.com/kdd-cup-2013-ntu/track2/blob/master/main2/chinese_name_list_v4.py`.

implementation collects less Chinese words. Because of using less Chinese information and simpler rules, results here may be worse than those obtained by the Chinese-or-not procedure in the first implementation.

5.2 Cleaning

This stage goes through three phases: character-based filtering, word-based filtering, and parsing alleviation.

1. **Character-based filtering:** This phase replaces European alphabets to English alphabets and we remove all punctuation except the blank.
2. **Word-based filtering:** This phase splits two consecutive uppercase characters (e.g., "CJ" → "C J"), removes English titles and some suffixes, transforms uppercase to lowercase, and replaces common English nicknames with formal names.[8] In addition, the nobility particles (e.g., "von" and "de") are removed, and we split each two-Chinese-character word to two words (e.g., "ChihJen" → "Chih Jen").
3. **Parsing alleviation:** This phase addresses incorrect string parsing by going through two steps. First, among names that are the same after blank removal, we keep only the longest one. For instance, if "Joseph Douglas Horton," "Josephdouglas Horton," and "JosephDouglasHorton" appear in the database, we keep only "Joseph Douglas Horton." The second step handles typos caused by incorrect string parsing; see examples in Section 4.4.3. For any pair of two names, if the longer one differs from the shorter one in less than four characters, then we remove the longer one. The threshold four is chosen by the scores on the leader board.

5.3 Selection

Recall that in this stage for each author name we obtain a candidate set. One author name is a candidate of another (and vice versa) if one of the following conditions holds.

1. Both names are exactly the same regardless of the order of words. **Example:** "Chih Jen Lin" and "Lin Chih Jen."
2. Both names are Chinese with more than two words or neither is Chinese. Further, the set of initial characters of words in a name is a subset of the other and so is the set of full words. **Example:** "Chih Jen Lin" and "Chih Jen K. Lin."
3. The set of initial characters of words in a name is a subset of the other. The shorter name has only one full word not in the longer, but the word is the prefix of a word in the longer name.
   Example: "Ch Jen Lin" and "Chih Jen Lin."

5.4 Identification

In contrast to the first implementation, we believe that the candidates selected in Section 5.3 are of high credibility, so this stage is not performed. That is, the candidates selected in Section 5.3 are identified as duplicates.

8. The nickname list is available at `https://code.google.com/p/author-dedupe/`.

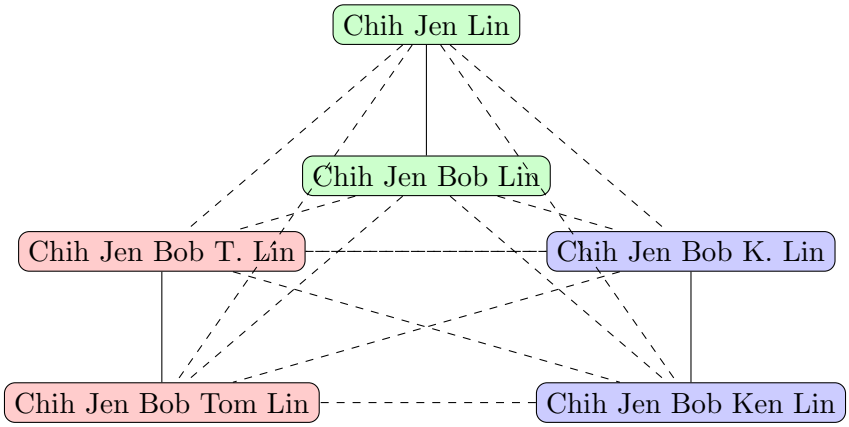

Figure 2: An example to illustrate the splitting procedure. Dashed edges are removed from the figure. In the end, the graph is split to three sub-graphs, each of which is considered as a set of duplicates.

5.5 Splitting

After Section 5.4, some names are still wrongly grouped as duplicates (e.g., "Chih Jen Lin," "Chih K. Lin," and "Chih H. Lin"). In this stage, we split such groups to improve precision. For easy explanation, we define the following terms.

1. **An extended/abbreviated name:** For two names satisfying the conditions in Section 5.3, if A is not shorter (not longer) than B, then A is an extended (abbreviated) name of B.
   Example: "Chih Jen Lin" is an abbreviated name of "Chih Jen Bob Lin," while "Chih Jen Bob Lin" is an extended name of "Chih Jen Lin."
2. **Common extended names (CEN):** In a set of duplicates, B is a CEN of A if B is an extended name of A and every A's extended name is either B's abbreviated or extended name. Note that any name is a common extended name of itself.
   Example: In Figure 2, assume "Chih Jen Bob Lin" and five other names are grouped as duplicates. All names except "Chih Jen Lin" are its extended names because of equal or longer length. We see that "Chih Jen Bob T. Lin" is not a CEN of "Chih Jen Bob Lin" because "Chih Jen Bob K. Lin" is an extended name of the latter, but is neither an extended nor abbreviated name of the former. Therefore, "Chih Jen Bob Lin" is the only CEN of "Chih Jen Bob Lin." Another example is in Figure 3. Further, Table 3 and Table 4 respectively list CENs of some authors in Figures 2 and 3.
3. **Longest common extended name (LCEN)**: A name B is an LCEN of A if B is a CEN of A and has the largest number of abbreviated names among A's CENs.
   Example: In Figure 3, assume "Chih Jen Bob Lin" and five other names are grouped as duplicates. For "Chih Jen Lin," whose CENs include "Chih Jen Lin," "Chih Jen Bob Lin," and "Chih Jen Bob Tom Ken Lin." In these CENs, "Chih Jen Bob Ken Lin" has three abbreviated names, more than the other two CENs. Therefore, "Chih Jen Bob Tom Ken Lin" is an LCEN of "Chih Jen Lin."

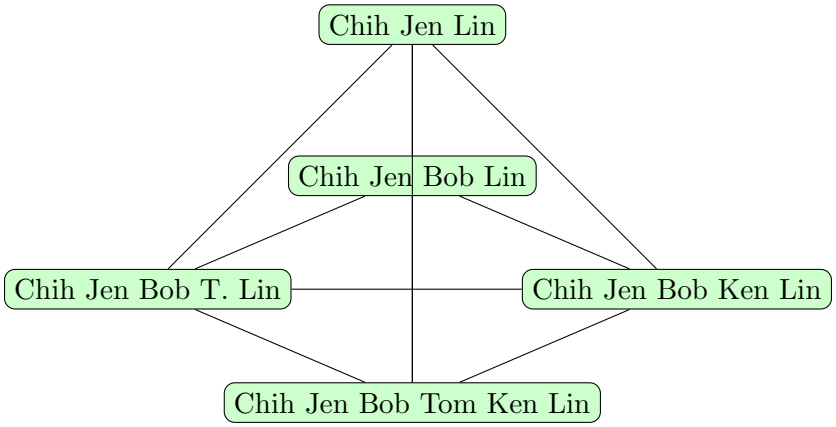

Figure 3: An example to illustrate the splitting procedure. All edges are preserved, so all names in this figure are considered as duplicates.

Author Name	CEN	LCEN
Chih Jen Bob Lin	Chih Jen Bob Lin	✓
Chih Jen Bob T. Lin	Chih Jen Bob T. Lin Chih Jen Bob Tom Lin	✓

Table 3: An part of examples for CEN and LCEN for Figure 2.

With the above definition, we describe the splitting procedure. For each group of duplicates, we construct an undirected graph so that nodes are names and any two names are connected by an edge. For each name, we get the corresponding LCEN. We then split any edge whose two nodes have different LCENs. Next, names in each connected sub-graph are considered as a set of duplicates.

Consider an example in Figure 2, where six names are considered as duplicates after the procedure in Section 5.4. The following example shows how we split edges. The LCEN of "Chih Jen Bob T. Lin" is "Chih Jen Bob Tom Lin," which differs from "Chih Jen Bob Lin" of "Chih Jen Bob Lin." Therefore, the link between "Chih Jen Bob T. Lin" and "Chih Jen Bob Lin" is removed. We remove many other edges by the similar reason. In the end only three edges remain.

We give another example in Figure 3, in which any two names have the same LCEN "Chih Jen Bob Tom Ken Lin." Therefore, we keep all edges. All names in this figure are then considered as duplicates.

5.6 Linking

As mentioned in Section 3.2, previous stages group duplicated names rather than identifiers. However, the competition task is to group duplicated identifiers, so some transformation is needed. Recall that each record in `Author.csv` and `PaperAuthor.csv` is a (name, identifier) pair. Our procedure starts from removing pair in `PaperAuthor.csv` that conflict with pairs in `Author.csv`. For example if "C J Lin" in `PaperAuthor.csv` and "C C Lin" in `Author.csv`

Author Name	CEN	LCEN
Chih Jen Lin	Chih Jen Lin	
	Chih Jen Bob Lin	
	Chih Jen Bob Tom Ken Lin	✓
Chih Jen Bob T. Lin	Chih Jen Bob T. Lin	
	Chih Jen Bob Tom Ken Lin	✓

Table 4: An part of examples for CEN and LCEN for Figure 3.

have the same identifier, we remove "C J Lin" because of the name mismatching. Next, for any group of names considered as duplicates, we construct an undirected graph so that each node is a name. For any node, we link it to all nodes satisfying that their (name, identifier) pairs appear in either `Author.csv` and or `PaperAuthor.csv`. In the end, if one identifier appears in two connected components of the graph, then the two groups are put together as duplicates. We consider the following example

Group_1		Group_2	
name	identifier	name	identifier
"C J Lin"	9_A, 41_A	"Chih Lin"	9_A, 41_A
"Chihjen Lin"	75_A	"C Lin"	8_{PA}, 10_{PA}
"Chih Jen Lin"	12_{PA}, 28_{PA}		

Assume that each group corresponds to a connected component of the constructed undirected graph. The subscript of an identifier indicates the source of the (name, identifier) pair, where "A" and "PA" denotes `Author.csv` and `PaperAuthor.csv`, respectively. Because the two groups share identifiers 9 and 41, all author identifiers in this table are considered as duplicates.

6. Ensemble

Because the two implementations in Sections 4 and 5 detect different sets of duplicates, an ensemble of their results may improve the performance. In this section, we propose a conservative setting to accurately find more duplicates by using background information such as an author's affiliation and field of study. The main idea is that if two authors have a similar background, then we are more confident in saying that they are duplicates.

For the two predictions generated by our implementations, we denote the prediction by the first implementation as the *major prediction*, while the other as the *auxiliary prediction*. This naming comes from the fact that the first implementation generally gives a better prediction; see Table 6. Given an author a, we say a' is an additional duplicate if a and a' are considered as duplicates only in the auxiliary prediction. We use a *filter* to check if a and a' have a similar background. If they pass the filter, then we consider a' as a possible duplicate of a. By an approach similar to that in Section 4.4.1, we choose duplicates from these possible duplicates by a dry-run function. That is, possible duplicates becomes real duplicates only if they pass the dry-run function. Moreover, in our practical experience, if a high-frequency word such as "Lin" exists in names considered as duplicates, the precision

file	author identifier	duplicates
major	10	10,11
auxiliary	10	10,11,12,13,14
ensembled	10	10,11,12,14

Table 5: An example of ensembling duplicates.

is often low. The reason is that people with a common last name are in general different. Therefore, we discard names having high-frequency words.[9]

Table 5 shows an example, where the additional duplicates of author 10 are 12, 13, and 14. Therefore, we apply the filter to pairs (10, 12), (10, 13), and (10, 14). Assume 12 and 14 pass the filtering. We then check if 10, 11, 12, and 14 could be duplicates by the dry-run function, and examine if high-frequency words exist in the names of these authors. In this example, we assume that the two checks are passed, so 12 and 14 are added as duplicates of 10.

In Section 6.1, we discuss the collection of background information, while in Section 6.2, we describe the filter. The ensemble procedure is summarized in Algorithm 2.

6.1 Collection of Background Information

For each author, we collect two sets of words: the affiliation-word set and the field-word set. The affiliation-word set is collected from affiliation information in `Author.csv` and `PaperAuthor.csv`; the field-word set is collected from paper titles and keywords in `Paper.csv`. The procedure can be divided into three stages: cleaning, stop-word removal, and collection.

In the cleaning stage, we remove common punctuation and handle several synonyms in the sources. From our statistics, some frequent words in affiliation sources are synonyms. For example, "univ" and "universidade" frequently appear in the data set, but they are equivalent to "university." For each set of synonyms, we replace all words with the most frequent one. Totally we consider three sets of synonyms, which are respectively transformed to words "university," "center," and "department."

In the stop-word removal stage, we generate two stop-word lists for affiliation and fields, respectively. Each includes a stop-word list and some high-frequency words (words occurred more than 1,704 and 32,000 in affiliation and field sources, respectively). In addition, for affiliation, we include several common country names. For fields, we include words that appear only once because such words are not very informative. After the two lists are generated, we remove all stop words.

Finally, in the collection stage, for each author all collected strings are split by space. The resulting two sets of words on affiliations and fields are then used by the filter in the ensemble process.

9. We call a word as a high-frequency one if it appears in `Author.csv` more than 1,200 times.

Data: A major prediction and an auxiliary prediction denoted by P_m and P_s, respectively.
Result: P_m and P_s are ensembled.
begin
  background information collection
  for $a \in$ all authors **do**
    $D_m \leftarrow$ duplicates of a from P_m
    $D_s \leftarrow$ duplicates of a from P_s
    $P \leftarrow D_m$
    for $a' \in D_s - D_m$ **do**
      if `filter` (a,a') **then**
        $P \leftarrow P \cup \{a'$ and its duplicates in $P_m\}$
      end
    end
    if any author in P has high-frequency words in its name **then**
      continue
    end
    if P passes the dry-run procedure **then**
      authors in P are duplicates
    end
  end
end

Algorithm 2: Ensemble of two results.

6.2 Filter

The filter considers that two authors have a similar background if the following conditions hold.

1. Two authors have at least two common words in their affiliation-word sets and at least one common word in their field-word sets, or they have at least one empty affiliation-word set and at least two common field words.
2. The two authors' field-word sets have no more than 75 common words.

The first condition implies that authors must share some words on affiliations or fields for having a similar background. The second condition addresses some special situations where two authors have papers in various fields. For such cases data tend to be more noisy.

7. Typo Correction

In this competition, typos occur in many places such as author names and paper titles. We focus on typos in author names because they are directly related to author disambiguation. From our observation, names in `PaperAuthor.csv` are too noisy, so we only handle typos in author names of `Author.csv`. To begin, we pre-process data by replacing all non-word characters with blanks and converting strings to lowercase. We also remove 11 manually selected

method	F1-score on leader board public	private	submitted
baseline	0.94411	0.94352	yes
implementation 1	0.99186	0.99198	yes
implementation 2	0.99071	0.99083	no
ensemble	0.99192	0.99201	no
ensemble + typo correction	0.99195	0.99202	yes

Table 6: F1-scores by our approach. Baseline means that we assume no duplicates at all. A submitted result means that it was uploaded during the competition. For unsubmitted results, we obtain F1-scores through the help of the competition organizers.

common words of author affiliations in `Author.csv` such as "department," "university," and "institute."

Because typos rarely occur, we assume that a word which appears at least twice in all author names is not a typo. Based on this principle, we split all words of author names in `Author.csv` to two sets. The first one includes words that appear only once as typo candidates, while the second includes all others. Next, for any typo candidate in the first set, we find their corrections from the second set. Specifically, a word in the second set is called a correction of a typo candidate if they differ in only one character. Note that a typo may have several corrections. For example, corrections of "cocn" may include "coin," "corn," and "conn."

After obtaining (typo, correction) pairs, the remaining task is to find duplicates. Two author names are considered as duplicates if

1. their word sets are the same after treating each typo the same as after its correction, and
2. their affiliations share at least one common word.

The first rule identifies "Lin Chih Jen" and "Litn Chih Jen" as duplicates if ("lint", "lin") is a (typo, correction) pair. However, the same rule also identifies "Lin C J" and "Litn C J" as duplicates, though the two names are likely different. Therefore, we impose the second rule. In the end, about 10 pairs of duplicates are obtained.

Finally, we merge the newly founded duplicates with results obtained in Section 6. Two author groups are combined if they share at least one author name.

8. Results

In our experiments, the used platform includes 96GB memory and two Intel Xeon E5-2620 2.0 Hz processors of which each has 6 physical cores. The running time of the first and the second implementations is around 15 minutes and 3 hours, respectively. The significant time gap is caused by the difference between the selection steps in the two implementations. The first implementation restricts the maximum number of authors in a group of possible duplicates to be less than a constant 17, but the other implementation does not put any

	F1-score on leader board		
method	public	private	submitted
implementation 1	0.99186	0.99198	yes
without Chinese-or-not	0.99109	0.99125	no
without dry-run	0.99097	0.99112	no
without both	0.98891	0.98934	no

Table 7: Evaluations of the first implementation with/without Chinese-or-not and/or dry-run.

limitation. The ensemble of the two implementations takes about 10 minutes, while the typo correction discussed in Section 7 needs about 7 minutes.

Table 6 presents the results (F1-score) on both public and private leader boards. Our first implementation gives slightly higher F1-scores than the second. After the post-processing procedure in Sections 6 and 7, the result is further boosted. Our approach gives the best F1-score in this competition, while the first implementation gives the second best (see the submitted results in Table 6). In Table 6 we see that some results were not uploaded during the competition because each team is not allowed to make more than two submissions per day. Therefore, we carefully submit results that may lead to the largest improvement. Several factors attribute to the success of our approach. Important ones include the identification of Chinese/non-Chinese names and effective string matching procedures to find duplicates with few of ad hoc parameters. Taking Chinese-or-not and dry-run procedure as examples, Table 7 shows the degeneration of implementation 1 if we do not include them into our approach. The degeneration is significant because the rank is lowered to the third if either the Chinese-or-not or the dry-run procedure is not applied. The rank is further lowered to the fourth if neither is applied.

9. Comparison on Approaches in KDD Cup 2013

At the KDD Cup workshop, four of the top 10 teams presented their work. These teams are

- 1st place: Algorithm @ National Taiwan University (Chin et al., 2013)
- 2nd place: SmallData @ UIUC (Liu et al., 2013)
- 4rd place: BS Man&Dmitry&Leustagos (Solecki et al., 2013)
- 7th place: SEU_WIP_AD (Zhao et al., 2013)

In this section, we conduct a comparison on them. Results are summarized in Table 9. Except the 7th that uses a semi-supervised strategy, all others are unsupervised and consider many heuristic rules. Compared with other approaches, ours highly relies on the information provided by `Author.csv`; in other words, we believe that `Author.csv` is trustworthy. The three unsupervised approaches can be described by the following four stages: pre-processing, finding candidates of duplicates for all authors, determining if the candidates are trustworthy, and post-processing. Based on Table 9, subsequently we highlight important differences between the four teams.

Aspect Considered	1st	2nd	4th	7th
Use `Author.csv`	√	√	√	√
Use `Paper.csv`	√	√	√	√
Use `PaperAuthor.csv`	√	√	√	√
Use `Conference.csv` and `Journal.csv`		√		√
Remove English honorifics	√	√	√	
Consider stop words	√	√	√	√
Split two consecutive uppercase characters	√		√	
Treat different types of punctuation using different strategies	√		√	
Transform European alphabets into English	√	√	√	
Remove nobility particles	√	√	√	
Replace nicknames using a public list	√	√	√	
Correct typos and accidentally added letters within every name	√	√	√	
Identify the last and the first name in a name	√	√		
Identify Asian authors	√	√		
Perform name disambiguation on authors' IDs (i.e., identify duplicated IDs)	√	√	√	√
Perform name disambiguation on authors' names (i.e., identify duplicated names)	√			
Consider name frequencies	√	√	√	√
Calculate similarity between authors using string distances		√	√	√
Use topic model			√	
Calculate similarity between authors using an bibliographic graph		√	√	
Compare authors using heuristic rules	√			√
Consider different naming conventions	√	√		
Use support vector machine/logistic regression				√
Select possible duplicates to reduce computational complexity	√	√	√	√
Check conflict names after duplicates are selected	√	√		
Use duplicated papers	√		√	
Refine result iteratively		√		√
Ensemble results generated by different algorithms	√			√

Table 8: Comparison on four of the top 10 teams

The pre-processing stage includes tasks such as cleaning data and correcting strings. This step is not directly related to author disambiguation but significantly improves the final results. However, some pre-processing procedures may bring additional noise into the data set or remove useful information. Therefore, in our approach we merely applied few conservative rules, so the data is changed in only a minor way. The most risky rule we

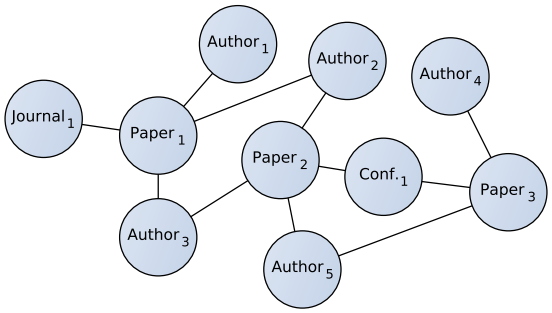

Figure 4: An example of bibliographic graph.

used is the nickname replacement. The 2nd-place team used more aggressive strategies than us because they tried to replace a name unit with a similar word which appears more frequently in the data set. The most aggressive pre-processing was conducted by the 4th-place team. They consider 16 steps including reversing the order of name units, replacing nickname, removing single letters within a name, etc. Therefore, ours is the most restrictive one among the three teams using purely unsupervised strategies. For the 7th-place team, they do not conduct any cleaning technique for pre-processing.

In the second stage, all teams select some possible candidates of duplicates. The aim is to reduce the computational complexity. The $O(n^2)$ cost of pairwise comparisons for n authors is prohibitively expensive. The selection stage begins with binning authors in the data set into different groups using some keys generated from their names. Similar to our strategies shown in Section 4.3, the 2nd-place team assign the authors with same keys (e.g., initial letters and identical name units) to the same bin. Furthermore, these two teams consider set comparisons, so a name's order of tokens does not effect the comparison result. The other two teams generate keys of an author using its first name and its last name. If we permute the tokens in a name, different results may be generated. The reason is that they generate keys in according to the location of the first name and the last name of an author. In addition, all these approaches find candidates for every author ID except our 2nd implementation that searches for possible candidates for every author name.

After we collect possible candidates for every author, the next step is to determine if these candidates are duplicates. In general, all proposed methods compare an author with all its candidates. The 4th place team's comparison and ours are purely string matching. In contrast, a bibliographic graph is often used to describe the complicated relation between authors, publications, and other information; see an illustration in Figure 4. Interestingly, these two teams did not build any bibliographic graph probably because data in `PaperAuthor.csv` are too noisy. That is, the links between authors and publications are unreliable. Note that Figure 2 and Figure 3 are not bibliographic graphs because they are built using our name comparison rules for checking conflict names. The other two teams measure differences between authors using meta-paths in their bibliographic graphs. If two authors are considered as duplicates then all their duplicates are merged. The operation is risky because the merge of two groups is determined by only a pair of authors without considering their duplicates. A simple remedy is to check the similarities between authors

in the merged group. If they are not similar enough, then the merged group would be split. Only the 2nd-place team and our approach conduct such a check. See the dry-run procedure in our 1st implementation, the splitting procedure in our 2nd implementation, and the ranking-based merging stage in the 2nd place team's solution for details. As mentioned before, semi-supervised learning techniques only appear in the 4th-place team's solution. They generate training data using some heuristic rules, and train SVM to decide if two authors are duplicates. Unfortunately, it seems that the data set is too noisy to provide reliable training data. A possible solution is to refine the result generated by SVM using hand-made heuristics (e.g., rules in Section 4.4).

Post-processing techniques mainly include identifying confident information and ensembling models. The 2nd-place team identifies confident information to avoid error propagation in their iterative framework. Ensemble is naturally considered by teams which have multiple approaches. We carefully designed rules to merge our two implementations and conduct typo corrections, while the 4th-place team applied a rule, called transient assumption, to aggressively merge their results.

10. Conclusion and Future Works

We can make the following observations and conclusions following the description of our approaches in this paper and the comparison in Section 9. First, we try our best to keep all information and delay the modification on the data set because the provided data set is noisy and incomplete. For instance, our typo correction is the last step of the whole procedure. In contrast, the 4th-place team conducts a similar process in their pre-processing stage. Second, an important advantage of using rule-based approaches is that we can easily trace the change of results after a rule is added. We can check the effectiveness of rules and identify useful information. For example, we find that different name conventions play an important role in author disambiguation when we add the matching function "one is a shortened name of the other and both have the same initial-character set of words." In contrast, some complicated methods like LDA are hard to analyze. Furthermore, human brain is still very powerful to analyze large-scale data sets if we have systematic schemes to filter out redundant information.

Overall, our experiments show that proposed techniques may be useful for other applications of author disambiguation. For example, others can apply our Chinese-or-not procedure when the separation of Chinese and non-Chinese authors is needed. Further, they can employ our rules designed for each of the two groups. Outputs of rules can then be used to train a binary classifier for determining whether two authors are duplicates. The parallelization of our algorithm is another interesting issue because the sizes of available data sets are growing. Besides, we are also interested in the behavior of graph-based approaches on noisy bibliographic data sets.

Acknowledgments

We thank the organizers for holding this interesting competition. We also thank the College of Electrical Engineering and Computer Science as well as the Department of Computer

Science and Information Engineering at National Taiwan University for their supports and for providing a stimulating research environment. The work was also supported by National Taiwan University under Grants NTU 102R7827, 102R7828, 102R7829, and by National Science Council under Grants NSC 101-2221-E002-199-MY3, 101-2628-E002-028-MY2, 101-2628-E002-029-MY2.

References

Rohit Ananthakrishna, Surajit Chaudhuri, and Venkatesh Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the VLDB Endowment*, pages 586–597, 2002.

Indrajit Bhattacharya and Lise Getoor. Iterative record linkage for cleaning and integration. In *Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 11–18, 2004.

Mikhail Bilenko, Raymond Mooney, William Cohen, Pradeep Ravikumar, and Stephen Fienberg. Adaptive name matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, 2003.

David Guy Brizan and Abdullah Uz Tansel. A survey of entity resolution and record linkage methodologies. *Communications of the IIMA*, 6(3):41–50, 2006.

Wei-Sheng Chin, Yu-Chin Juan, Yong Zhuang, Felix Wu, Hsiao-Yu Tung, Tong Yu, Jui-Pin Wang, Cheng-Xia Chang, Chun-Pai Yang, Wei-Cheng Chang, Kuan-Hao Huang, Tzu-Ming Kuo, Shan-Wei Lin, Young-San Lin, Yu-Chen Lu, Yu-Chuan Su, Cheng-Kuang Wei, Tu-Chun Yin, Chun-Liang Li, Ting-Wei Lin, Cheng-Hao Tsai, Shou-De Lin, Hsuan-Tien Lin, and Chih-Jen Lin. Effective string processing and matching for author disambiguation. In *Proceedings of the KDD Cup 2013 Workshop*, pages 7:1–7:9. ACM, 2013.

Hanna Köpcke and Erhard Rahm. Frameworks for entity matching: a comparison. *Data and Knowledge Engineering*, 69(2):197–210, 2010.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems. In *Proceedings of the VLDB Endowment*, pages 484–493, 2010.

Jialu Liu, Kin Hou Lei, Jeffery Yufei Liu, Chi Wang, and Jiawei Han. Ranking-based name matching for author disambiguation in bibliographic data. In *Proceedings of the KDD Cup 2013 Workshop*, pages 8:1–8:8. ACM, 2013.

Senjuti B. Roy, Martine D. Cock, Vani Mandava, Brian Dalessandro, Claudia Perlich, William Cukierski, and Ben Hamner. The Microsoft academic search dataset and KDD Cup 2013. In *Proceedings of the KDD Cup 2013 Workshop*, pages 1:1–1:6. ACM, 2013.

Benjamin Solecki, Lucas Silva, and Dmitry Efimov. KDD Cup 2013: author disambiguation. In *Proceedings of the KDD Cup 2013 Workshop*, pages 9:1–9:3. ACM, 2013.

Vetle I. Torvik and Neil R. Smalheiser. Author name disambiguation in MEDLINE. *ACM Transactions on Knowledge Discovery from Data*, 3(3):11:1–11:29, 2009.

Pucktada Treeratpituk and C. Lee Giles. Disambiguating authors in academic publications using random forests. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, pages 39–48, 2009.

Steven Euijong Whang and Hector Garcia-Molina. Joint entity resolution. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 294–305, 2012.

Steven Euijong Whang, David Marmaros, and Hector Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1111–1124, 2013.

Jianyu Zhao, Peng Wang, and Kai Huang. A semi-supervised approach for author disambiguation in KDD CUP 2013. In *Proceedings of the KDD Cup 2013 Workshop*, pages 10:1–10:8. ACM, 2013.