Effective String Processing and Matching for Author Disambiguation

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, Chih-Jen Lin.
National Taiwan University
{d01944006, r01922136, r01922139, b99902090, b98901044, r01922141, r01922165, r01944041, r99902109, b99902019, b99902059, b99902073, b99902023, b97902055, b98902105, r01922159, b98901037, d00922023, r01922001, r01944011, r01922025}@ntu.edu.tw, {sdlin, htlin, cjlin}@csie.ntu.edu.tw

ABSTRACT

Track 2 in 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.

1. INTRODUCTION

Track 2 in KDD Cup 2013 is a task of name disambiguation. Ideally, a name uniquely identifies an entity, 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 letter gives authorships for around two millions 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 [7]. 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

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

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

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

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 [10] [11] [5], duplication elimination/detection [1], object matching [6], and record linkage [4] [2]. Similar to [8] [9] [3], we consider author name strings as the major feature to measure the similarity between two authors. Thus all strategies in our algorithm are highly related to string processing.

We describe our approach in Section 2. It is realized in our two implementations described in Sections 3 and 4, respectively. Section 5 proposes a strategy to ensemble the predictions from these two implementations. We handle typographical errors (typos) in Section 6. Finally, Section 7

summarizes our work. Our implementations are avaliable at `https://github.com/kdd-cup-2013-ntu/track2`.

2. 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 3 and 4.

2.1 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 "蘭" 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" comparing to those of "E. W. Dijkstra." Besides, Korean, Vietnamese, and Singaporean names have a similar structure (i.e., two-word first name and a short 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").

In this paper, we define a word as a *shortened word* 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. A name is a shortened name if it consists of at least one shortened word; otherwise, it is a full name.

2.2 The Framework

Our framework can be divided into six stages. Here we briefly introduce each stage, but leave details in Sections 3 and 4.

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. It is needed 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"

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 so that in the end we have groups of author identifiers.

3. 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.

3.1 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. Roughly each Chinese word is romanized to four English words, though we omit details here. 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. Moreover, each dictionary consists of two subdictionaries storing last names and syllables respectively. Some words in our Chinese dictionary are also common in non-Chinese names (e.g., "van," "den," and "ben"), so we construct a "banned list."

The flow to determine if an input name is Chinese or not is in Figure 1. To begin, a name is cleaned and then tokenized. 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 bannned 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. We consider all full words which is contained in the banned list as Chinese once any Chinese full words are found. In pratice, Counter_3 always outputs 0 when Counter_2 returns 0. Otherwise, the output of Counter_3 becomes the number of matches of full words and the banned list.

In Figure 1, 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.

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}.
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 returns 0 to be the number of non-Chinese full words, so "Chih Jen Dean H. Lin" is classified as Chinese.

3.2 Cleaning

The purpose of data cleaning is to make name comparisons more accurate. 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, 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. Remove apostrophes and replace punctuations. For example, "o'relly" becomes "orelly." We then replace punctuations except periods with blanks. We keep the period because it is useful to determine if a word is a shortened one or not.
5. Replace European alphabets with similar English alphabets. For example, replace "ä" with "a."

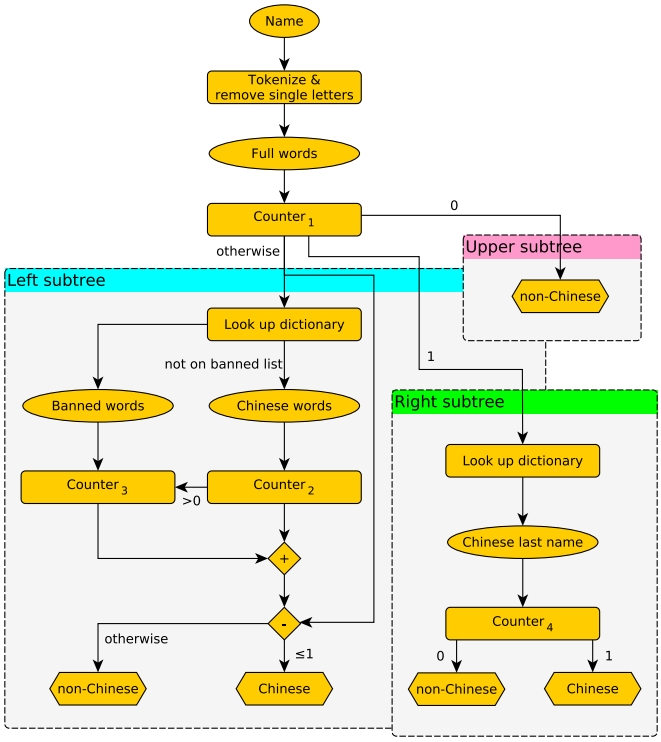

Figure 1: Flow of finding if a name is Chinese or not.

6. Replace common English nicknames. For example, replace "bill" with "william."

3.3 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.

3.4 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 3.4.1, uses many matching functions listed in Section 3.4.2. The two additional procedures are described in Section 3.4.3.

3.4.1 The Main Procedure

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 is to remove authors that were wrongly selected. However, although a strict criterion gives high precision, it many cause low recall. Therefore, we sequentially apply matching functions (listed in Section 3.4.2) 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 is

"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 short 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 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.

3.4.2 Matching Functions

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

```
Data: Each author has a set of candidates.
Result: All authors are split to groups of duplicates.
begin
    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
                authors in P are duplicates
            end
        end
    end
end
```

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

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."
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.

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. Druuzduzel."
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."
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."

3.4.3 The Additional Procedures

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 1. "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 1: 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 "petraquillfel."

3.5 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 3.4.2 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.

4. THE SECOND IMPLEMENTATION

In this section, we introduce the second implementation following the framework in Section 2. As mentioned in Section 2.2, 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.

4.1 Chinese-or-not

In contrast to Section 3.1, the implementation is simpler here. We build a Chinese dictionary that consists of 694 words of romanized Chinese syllables and a banned list. 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 Chinese.

4.2 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 punctuations 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. 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 3.4.3. For any pair of 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.

4.3 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, 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."

4.4 Identification

In contrast to the first implementation, we believe that the candidates selected in Section 4.3 are of high credibility, so this stage is not performed. That is, the candidates selected in Section 4.3 are identified as duplicates.

4.5 Splitting

After Section 4.4, 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.

- **An extended/abbreviated name:** For two names satisfying the conditions in Section 4.3, 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."

- **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 2 and Table 3 respectively list CENs of some authors in Figure 2 and 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,

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

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

"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."

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 Section 4.4. We will show how edges are split.

Take an example to show 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.

4.6 Linking

As mentioned in Section 2.2, previous stages group duplicated names together 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` 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
Each group corresponds a connected component component

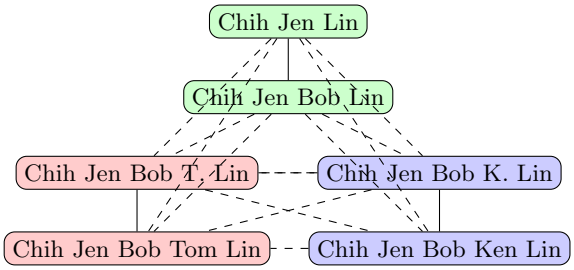

Figure 2: An example to illustrate the splitting procedure. Dashed edges are removed from the figure. In the end, the graph is split to three sub-graphs, each of which is considered as a set of duplicates.

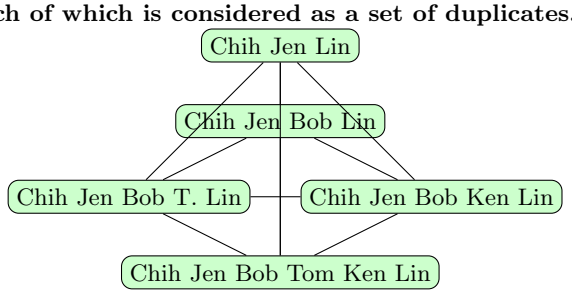

Figure 3: An example to illustrate the splitting procedure. All edges are preserved, so all names in this figure are considered as duplicates.

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}		

of the undirected graph. The subscript of an identifier indicates the source of the (name, identifier), 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.

5. ENSEMBLE

Because the two implementations in Sections 3 and 4 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 similar background, then we are more confident in saying that they are duplicates.

For the two predictions generated by our implementations, we call the prediction with better performance (see Table 5) generated by the first implementation as the *major prediction*, while the other as the *auxiliary prediction*. 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 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 3.4.1, we choose from these possible duplicates by a dry-run function. Moreover, in our practical experience, if a high-frequency

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

Table 4: An example of ensembling duplicates.

word such as "Lin" exists in names considered as duplicates, the precision 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.[1]

Table 4 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 5.1, we discuss the collection of background information, while in Section 5.2, we describe the filter. The ensemble procedure is summarized in Algorithm 2.

5.1 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 punctuations 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.

5.2 Filter

The filter considers that two authors have 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.

Data: A major prediction and a 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. 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 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, where a typo and its corrections are considered as the same, and

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

2. their affiliations are required to 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 5. Two author groups are combined if they share at least one author name.

7. RESULTS AND CONCLUSIONS

Table 5 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 5 and 6, 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 5).

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 6 shows the degeneration of implementation 1 if we do not include them into our approach. Therefore, we expect that these techniques can be useful for other applications of author disambiguation.

	F1-score on leader board		
method	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	0.99195	0.99202	yes

Table 5: F1-scores by our approach. Baseline means that we assume no duplicates at all. Typo is the abbreviation of typo correction. A result is submitted if and only if it was uploaded during the competition.

	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 6: Evaluations of implementation 1 without Chinese-or-not and/or dry-run.

8. ACKNOWLEDGEMENT

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.

9. REFERENCES

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *Proceedings of the VLDB Endowment*, pages 586–597, 2002.

[2] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD '04: Proceedings of the 9th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 11–18, 2004.

[3] M. Bilenko, R. Mooney, W. Cohen, P. Ravikumar, and S. Fienberg. Adaptive name matching in information integration. *Intelligent Systems, IEEE*, 18(5):16–23, 2003.

[4] D. G. Brizan and A. U. Tansel. A Survey of Entity Resolution and Record Linkage Methodologies. *Communications of the IIMA*, 6(3):41–50, 2006.

[5] H. Köpcke and E. Rahm. Frameworks for entity matching: A comparison. *Data & Knowledge Engineering*, 69(2):197–210, 2010.

[6] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. In *Proceedings of the VLDB Endowment*, pages 484–493, 2010.

[7] S. B. Roy, M. D. Cock, V. Mandava, B. Dalessandro, C. Perlich, W. Cukierski, and B. Hamner. The Microsoft academic search dataset and KDD Cup 2013. In *ACM SIGKDD KDD-Cup WorkShop*, 2013.

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

[9] P. Treeratpituk and C. L. 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.

[10] S. E. Whang and H. Garcia-Molina. Joint entity resolution. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, pages 294–305, 2012.

[11] S. E. Whang, D. Marmaros, and H. Garcia-Molina. Pay-as-you-go entity resolution. *IEEE Transactions on Knowledge and Data Engineering*, 25(5):1111–1124, 2013.