Author-Profiling

Zhiming Zhou 5140309059

May 16, 2017

Abstract

During the semester, we found that author name ambiguity is a frequently encountered problem in Acemapdifferent authors may publish under the same name, while the same author could publish under various names due to abbreviations, nicknames, etc. This report is about the work we have done and a new direction of its development we have found that should be more effective and practical to solve the problem.

1 INTRODUCTION

Acemap is a novel academic information system, and the website has already online with lots of useful, interesting, beautiful, powerful, intuitive and unique function. But while searching for academic papers by a specific author name, you may found a large variety of papers ranging from different fields under the author name you search, and it is really tough to distinguish the correct paper you are looking for from others. When you glance through the list and still could not find a paper you are particularly interested in, it suddenly occurs to you that the missing paper was in fact published under another name which uses abbreviations of the author.

Author name ambiguity is indeed an often encountered problem in digital publication libraries. Author profiling is important for an apparent reason: it helps refine search results for end users and thus improves user experience and productivity. Another reason is that it empowers more accurate bibliometric analysis such as mining academic social networks.

The following part of this report is organized as follows. In section 2, I will show what we have had tried to do. In section 3 I will illustrate a more effective algorithm we have found which has been proved useful on author profiling. I will show the detail of the major components in the system in section 4, followed by conclusion in section 5.

2 PREVIOUS WORK

At first, we have no idea what to do, because we can not find anything as a reference. So we just do some simple try and we hope we can make some adjustment by observing the effect of the result. The easiest method is to cluster the author with co-authors. For an example, if we want to cluster author A, we could first collect all papers that published under the name A, then, we extract their co-authors, and compare them. If the first author has co-authors B and C, and the second author has co-author C and D, we can initially identify that the two author is the same person. After we have complete the code which can be used in our database, I adjust it to another smaller database that have already done the author profiling job to evaluate this method. Unfortunately, the result was not very satisfactory. But there is always a way out for we have found an algorithm which fits our problem during the test of our previous code, and the algorithm had ranked in the second place in KDD Cup Data Mining Contest 2013.

3 SYSTEM OVERVIEW

We mainly have two steps to first maximize the recall and then maximize the precision. the two steps are as follows:

Maximize the recall: enlarge candidate pool of duplicates. For each indexing author ID, we should try to pull out all the author IDs whose author names are possible variations of the indexing author name. Ideally this pool covers all the duplicates and is as compact as possible. To come up with the pool we should take into account a number of cases where names can mutate or be disturbed. For instance, Mike Lewisg can be safely assumed to be a noisy variant of Mike Lewis by an extra g, while it is not the case for Chinese names Wei Lin and Wei Ling, as both Lin and Ling are valid last names.

Maximize the precision: trim the candidate pool based on authors publication features (i.e., meta-paths) we have calculated. Examples of publication features include co-author network, publication venues, years, titles words and keywords. These features turn out to

be discriminative for identifying real duplicates from the candidates pool.

And we should also have a pre-processing procedure to handle noisy and missing values and a post-processing step to remove unconfident duplicates. In the following section, I will illustrate the algorithms in details.

4 ALGORITHM

4.1 Pre-processing

Before going deeper to the recall and precision steps, we need to first clean the data to recover part of author names. Here I list two main types of noises:

Noisy First or Last Names: Some examples of noisy names belonging to this type are Eytan H. Modiano and Eytan Modianoy, Nosrat O. Mahmoodo and Nosrat O. Mahmoodiand, UniversityofBonn Andreu Mas Colell and Andreu Mas Colell. To recover the correct first and last names from the noisy observations, we can first build statistics of single name units. Later for rare last and first names, we should compare them with the possible sub-strings by discarding the characters at the beginning or end of the name unit. If theses substrings appear frequent in the data set, we believe these name strings should get recovered.

Mistakenly Separated or Merged Name Units: Name units are defined as elements split by whitespace or other punctuation marks like dash as delimiters. Sometimes, different name units are mistakenly separated or merged in the data set. Examples include Sazaly Abu Bakar and Sazaly AbuBakar, Vahid Tabataba Vakili and Vahid Tabatabavakili. To handle this kind of noise, similar to the previous type, we can still build statistics of name units. But this time co-occurrences of name units appearing within the same author name are recorded. By referring to the times of appearance of concatenated name units, we are able to separate or merge name units if they are originally mistakenly merged or separated.

4.2 Improving recall

We propose to enhance the recall via two main categories of considerations:

4.2.1 String-based Consideration:

Levenshtein Edit Distance: the Levenshtein distance is a string metric for measuring the difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single-character edits (insertions, deletions or substitutions) required to change one word into the other.

Soundex Distance: Soundex algorithm is a phonetic algorithm that indexes words by their pronunciation in English.

Overlapping Name Units: Two strings are viewed as similar if one name string shares most of its name units with the other.

4.2.2 Name-Specific Consideration

Name Suffixes and Prefixes: Name prefixes like Mr and Miss, generational titles like Jr, I, II are ignored during name comparison.

Nicknames: We use the search of the team for common nicknames from the world wide web to form our nickname knowledge base because some researchers prefer to use nicknames and original names for different papers.

Name Initials: Due to the diverse format of research papers, it is not surprising to see name initials used in title, in the content, and/or the citation of research papers. One thing that one needs to be careful with is that some initials refer to the same name unit even if they are not the same(such as nicknames).

Asian Names and Western Names: Asians have different name structure from westerns. Thus, we cannot use the same approach for names from different regions. In order to differentiate Asian names and Western names, we use the teams definition of different rules to extract names. Additionally, Asian and western names usually have totally different settings for the above mentioned ideas.(edit distance, soundex distance)

4.3 Improving The Precision

It is questionable for considering two author IDs to be the duplicates even though they share the same or similar names. Now I will introduce a ranking-based method in cascaded stages to solve these two problems simultaneously.

4.3.1 Meta-Path-based Similarity

In the network schema extracted from the data set, two authors can be connected via different paths. For example, two authors can be connected via author-paper-author path, author-paper-venue-paper-author path, and so on. Intuitively, the semantics underneath different paths imply different similarities. Formally,

these paths are called meta-paths.

For a single relation, the measure matrix is an adjacency matrix M of the sub-network. Given a composite relation in a meta-path, the measure matrix can be calculated by the matrix multiplication combined with normalization of the partial relations.

The team selected meta-paths and corresponding weights simply based on our prior knowledge. The selected meta-paths are APA, AOA, APAPA, APVPA, APKPA, APTPA and APYPA.("A" is author, "P" is paper, "O" is Org., V is venue, "K" is keyword, "T" is title, and "Y" is year) The weights for them are decreasing progressively.

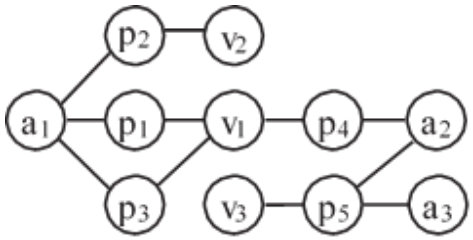

Figure 1: Illustration for computing meta-path- based similarity

Assume we are given a tiny sub-network as shown in Figure 1 composing three authors, five papers and three venues. It is easy to check the adjacency matrix $M_{A,P}$ and $M_{P,V}$ depicting the relations for Author-Paper and Paper-Venue are:

$$M_{A,P} = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccccc} p_1 & p_2 & p_3 & p_4 & p_5 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad M_{P,V} = \begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \end{array} \begin{array}{c} \begin{array}{ccc} v_1 & v_2 & v_3 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array}$$

Then, the measure matrix between Author and Venue is:

$$M_{A,V} = Normalize(M_{A,P} * M_{P,V})$$

From now on, the similarity scores for any pairs of authors can be simply referred to by looking up the corresponding matrix values.

4.3.2 Meta-Path-based Similarity

We do a scan from the top ranked ID pair to the lower ranked ones to help infer the author entity. And we will skip the conflict IDs, find one that has high similarity but also passes the name matching comparison, we believe these two IDs having high probability to be the real duplicate. After that, if A is the duplicate of B and B is the duplicate of C, we will consider that a is the duplicate of C. Another important strategy is to expand the author names corresponding to the IDs once we are confident about two IDs to be the duplicate. This idea is useful because it can help avoid the mistakenly detected conflicts.

4.4 Post-processing

Unconfident duplicate author IDs should be removed even though their names are compatible and their meta-path- based similarity scores are acceptable. This step is crucial in that the later iterative framework requires highly confident output to gradually refine the results. It is also tricky since through empirical study, and it is difficult to define unconfident.

4.5 Iterative Framework

Assume we are able to find highly confident duplicate author IDs using the aggressive strategy introduced in the last subsection, what can we do with this to further improve the performance? The answer is clear: An iterative framework which takes the detected duplicates of the last iteration as part of the input for the following resons:First, we are able to generate much better meta-path-based similarity scores by adding up the elements of the duplicate author IDs in the adjacency matrices describing different relations. This definitely can help the p-step to increase the precision.Second, recall the name expansion module introduced at the end of the p-step. One shortcoming of that module is the irreversibility of the conflict detection. That is, once we detect conflicts within a group of author IDs, the merge must be rolled back even though later the conflicted name strings are expanded to be compatible. However, if we adopt this iterative framework and expand author names at the very beginning, it is less possible to miss this performance gain.

5 Conclusion

We have tried to disambiguation the author name, and we have found a better algorithm which is undoubtedly practical in KDD Cup Data Mining Contest 2013. But there is still lots of work need to be done. We have a problem that we originally intended to deal with all the data in our database, but after trying to run the program, we found out that the RAM of the computer is not enough to deal with all of the data at once. So, in the future, we want to change the code to deal with one author name at one time, and we need to change some of the parameters according to the result to obtain the best result. I am looking forward to the day we complete the

work, and I am firmly believed that our work will turn out to be an improvement of the Acemap.