Project 2 - Author Disambiguation

Matt Cialini, U50060838

Username: Matt Cialini

Write-up/Algorithm

Introduction

This assignment is centered around disambiguating authors from the Microsoft Academic Search database. Each author is associated with an id number, and about 6% of these authors are actually the same person. The task is to find out which authors are duplicates.

Pre-Processing

My code begins with some minor modifications of the data to set the stage for determining duplicates. I normalize all names by transforming the european characters into ascii, removing all punctuation marks and titles such as Jr., II, III, and putting it all in uppercase. Another fairly major optimization included is to only group authors who have written at least one paper in PaperAuthor. This is implemented using a hash table named "index" where the key is the author ID from the file and the value is a list of all names that appear under that ID. These values, or "aliases", are trusted and used in the recall phase.

Algorithm

This algorithm begins by reading through the Author.csv file, and uses three separate hash tables to handle its data:

authornames {key = name, value = list of ids with that name},

authorids {key=id, value=name under that id},

duplicates {key=id, value = list of duplicate ids}.

For every ID in Author.csv, we add an entry in duplicates for this ID with itself as a duplicate, and add an entry in authorids with the ID and the name it appears with. Additionally, if the ID has also written a paper, we want to include this author in authornames as well. If this hash table doesn't have an entry with this author's name, we add one and set the value to [ID]. Otherwise, we assume two authors with the same exact name are duplicates, so we append this ID to the IDs already listed under that author. In this case, we also append all of the IDs with this name to this author's entry in duplicates, and append this author's ID to the others' entries in duplicates. Once the entire Author file has been processed, the next stage is to go through every author in authornames and generate variations of this name to search for in authornames. If a given variation is found, we return the ids associated with this variation in authornames. The types of variations are as follows:

Aliases: First we look through all of the aliases associated with an authorid from PaperAuthor

and search for each one in our hash table.
Merged subnames: We generate all possible merges of consecutive subnames of an author's full name. For example, for the author Robi David Mitra we search for RobiDavid Mitra and Robi DavidMitra. This merging appears frequently, so it is an effective addition.
Nonsequenced subnames: We generate all possible permutations of an author's subnames and search for them. For example, for the author Sing Bing King we search for Sing King Bing, Bing Sing King, etc.
Concatenated initials: There are several authors who appear in Author.csv with initials, and some of these are not separated by spaces, so when periods are removed initially the initials become one word. So for example, if we have John David Baker, we search for JD Baker.
Abbreviations: A frequent occurrence is an author's name is abbreviated, so we search for all possible abbreviations of their name, excluding the last name. For example, for the author Ralph Edward Griswold, we search for R. Edward Griswold, R. E. Griswold, Ralph E. Griswold.
Dropped subnames: Additionally, if an author has at least three subnames, we search for every possible version of their name where one subname is dropped. For example, for Harvey Kent Dave Bowen, we search for Harvey Bowen, H. Bowen, Harvey Kent Bowen, Harvey K. Bowen, etc.

Once a list of ids is returned from this variation function for ID, we go through these ids and ensure there's only one copy of each id. Then an additional check is if there are more than 7 ids returned. In this case, it's likely that many are false positives, such as for a name like J. L, so we ignore them all to try and maintain the balance of recall to precision. Otherwise, we add these duplicate ids to the current author's entries in authornames and duplicates, and also we add the current author's ID to all duplicate author's entries in those tables.

Testing/Evaluation

All of my testing was done on my Windows 7 machine using the Eclipse PyDev environment. Besides simply producing solution test files and submitting them to see my evaluation, occasionally I also printed an extra line into the solution file of the names corresponding to the duplicate ids. I manually searched through these and sought out entries which were definitely incorrect, and added additional code to account for these cases.