Project for Michael Pitts
Course TCSS 702A
University of Washington Tacoma
Institute of Technology

ALIAS: A Tool for Disambiguating Authors in Microsoft Academic Search

Under supervision of : Dr. Senjuti Basu Roy
Co-advisor: Dr. Ankur Teredesai

Table of Contents

Abstract

With the plethora of existing and new academic publications a serious problem is determining if computer records have incorrectly attributed papers to the wrong author. Having papers falsely attributed to the wrong author, or not having all of an author's papers correctly attributed can be damaging and disruptive This project was developed in two parts; (a) a random forest algorithm that would calculate the similarity between two authors and (b) a graphical user interface that would allow a user to access and search the author data via either an application or an embedded web tool. The user could search by name and find either potential duplicate entries or search for similar authors; those who share expertise and experience with a a given author.

Introduction

The ability to search articles and papers and collect data about publications is of critical interest in modern research. Specialized search engines, such as Google scholar and Microsoft Academic Search, cater to researchers across all scientific disciplines, to enable search, browsing, and exploration of academic publications over different scientific disciplines and domains. Microsoft Academic Search[1] (henceforth referred to as MAS) is one such search engine that has one of the largest repositories of academic publications, from astronomy to zoology. It currently covers more than 50 million publications and over 19 million authors across a variety of domains, with updates added each week. Prevalence of noise is one of the problems in this data set, as the raw data is collected by crawling and extracting data over hundreds of publishers websites[2], each adhering to its own format and standard of representations. The problem gets further magnified, as many authors publish under several variations of their own name, or different authors share similar or same name. Consequently author-ambiguity is a tremendous challenge that exacerbates the overall search experience. Figure 1 and Figure 2 explain these scenarios using MAS data set examples.

Fig. 2 Example of different authors with the same name

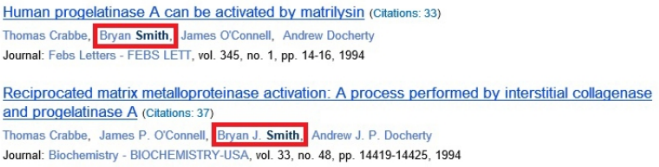

Fig. 1 When an author has a variation in their name, Bryan Smith vs Bryan J. Smith.

Microsoft has been kind enough to supply a large sample set from their own academic search which includes a set of ground truth values; author ids that were indicated to be duplicates of other authors. With this data it has been possible to test the effectiveness of algorithms as well as develop a proof of concept.

On the front end, ALIAS provides a consistent user interface, whether the user is running ALIAS as a stand alone application or embedded in a web page. The user can first search on a particular author by name, and then retrieve data relevant to that author. The data can be potential duplicate entries for that author, which can be compared with the original record side by side. The data retrieved could also be similar authors, those who have similar interests, expertise, conference attendance, and publications.

1 http://academic.research.microsoft.com/
2 http://academic.research.microsoft.com/About/Help.htm#5

On the back end, ALIAS relies on a large data set containing both individual author data, such as the author email, affiliation, journals published to, and conferences attended, and in some cases if the author had flagged another author object as a duplicate. This data was initially provided by MAS. It also has a good amount of pair wise author data, such as given two specific author objects, are they affiliated with the same organizations, do they attend the same conferences and have the same co-authors. This data was constructed from the initial, individual author data and then aggregated to allow later algorithmic analysis.

Author pairs were then evaluated for similarity, resulting in a pairing similarity value being generated between zero and one. This algorithm was primarily developed by Swapna Savvana, under the supervision of Professor Senjuti Basu Roy using both the initial data provided by MAS as well as meta-data that was generated by Michael Pitts.

The problem of record matching and de-duplication has been studied in the database literature in the past [1], [2], [3]. What differentiates ALIAS from the rest of the work is its underlying data model assumes a co-authorship graph, precluding these solutions to be inapplicable in our settings. Additionally, unlike existing works, ALIAS accepts author names as inputs that do not need to exist in the database. Finally, ALIAS enables near real-time interaction with the user, returning the duplicate set of authors with respective confidence score to the user almost instantaneously. Specific to the interest of academic publication and related applications, a recent work proposes BibPro [4], a citation parser that extracts components of a citation string, considering sequence alignment. Similarly, the web intelligence community has addressed author name disambiguation problems [5], [6] and some of its variants. To the best of our knowledge, no existing works tackles the problem considering scale, nor do they accept inputs that are incomplete and not present in the underlying data set.

Project Planning

During the two quarters ALIAS was developed, we meet regularly under the direction of Professor Senjuti Basu Roy. At first this was a scheduled weekly meeting where Professor Basu Roy would help to set goals for the coming week and help divide labor among participants. Later, as work roles and data were better understood face to face meetings were less critical.

In particular, Professor Basu Roy and Michael Pitts planned out the expected interface for ALIAS, designing the look and feel that was to come. These sketches and paper prototypes gave a good idea of what was to be expected in terms of both appearance and functionality and made the eventual construction process cleaner and removed many potential mistakes in communication.

Technical Specification

ALIAS works primarily in Java, which handles the graphical user interface and networking aspects of the project. It also uses SQL queries to access and change author data. It also uses the R programming language to run its author similarity algorithm.

Front End

Was developed in Java so that it could be developed as a stand alone application, but easily transferred to a web based embedded applet. It would function identically in either setting.

When first started, the user would enter a name in the Name field. As the user types ALIAS searches its Author data base and provides suggestions, which the user can select. They can then select “Duplicate Detection” and click Go. This will provide a list of potential author objects that based on calculated data indicate a strong chance that the listed authors are duplicates of the search on author. Next to each author in the list is a Confidence score which can range from zero to 1.0, with a higher confidence indicating a stronger chance that the two authors are duplicates. In testing, confidence scores greater than 0.97 often indicated a duplicate author had been found. By selecting the potential duplicate, the user can then compare the details of each author side by side. Figure 3 demonstrates such a search and examination.

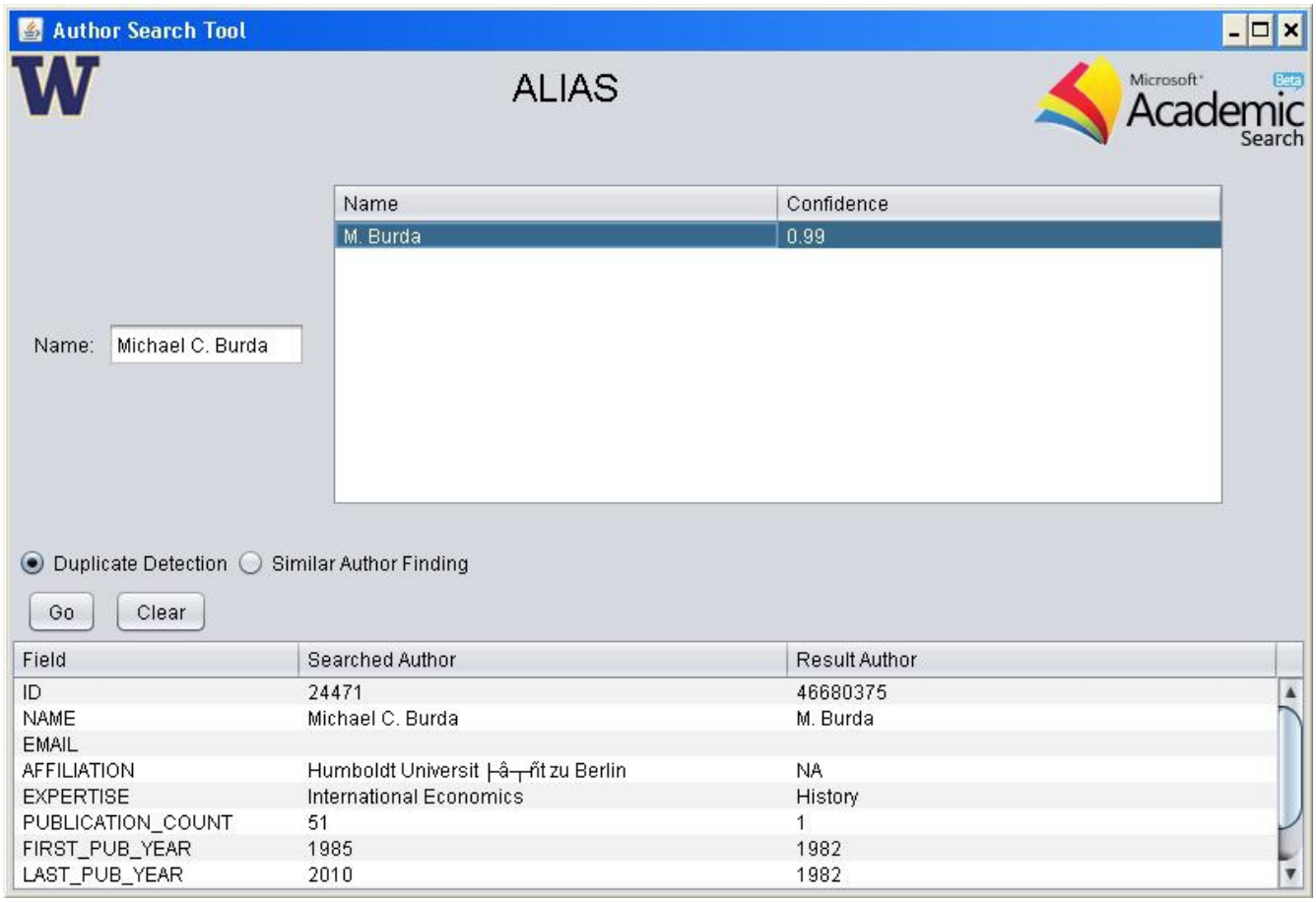

Fig. 3 Two duplicate authors, found despite mangled or missing data

The author data is parsed and searched without any cleaning or restriction. Even if fields contain damaged data, or a re missing data all together ALIAS can still generate a confidence score. This gives ALIAS the strength to operate on a multitude of data sources, with little or no intermediate work being done to “clean” the incoming author data.

The Clear option is provided so that a user may remove previous search results and comparisons.

The user can also search for similar authors, by selecting “Similar Author Finding”. They can then fill in how many similar authors they would like to find. By selecting Go the user initiates a search for similar authors, which from testing was found to be author pairs with a confidence less than 0.97. Selecting Go will bring up a list of similar authors who can be examined in the same way duplicate authors were. Figure 4 gives such an example.

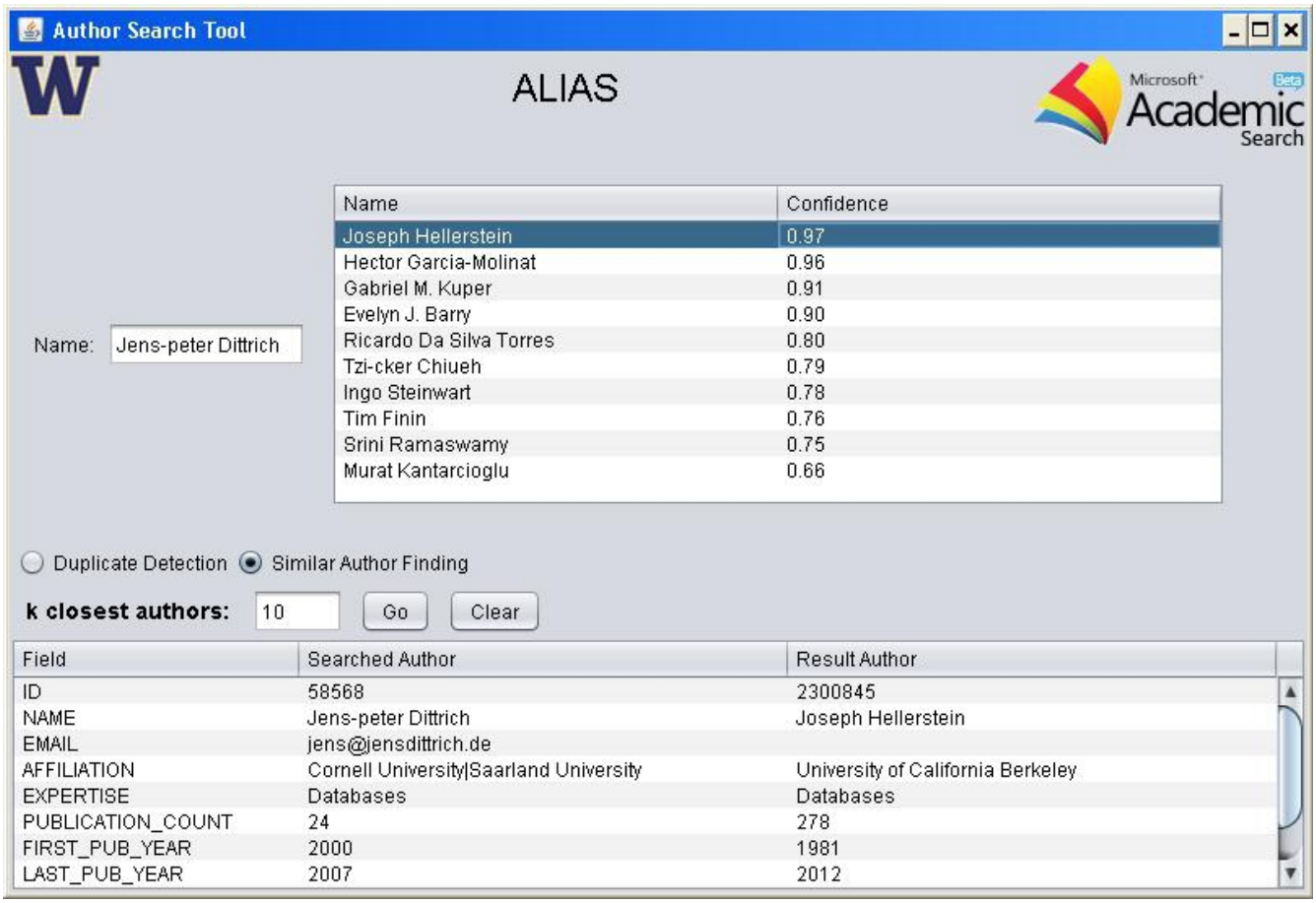

Fig. 4 A list of similar authors, note the same expertise

Back End

We have used unsupervised learning methods to disambiguate authors from a large data set. The useful traits were keywords present in an author's papers, the authors co-authors, conferences attended, areas of expertise, journals published to, and years in which two authors may have published.

All author data is stored in an instance of Microsoft SQL Server 2012, with author pair data stored in a single table named “ALIAS_TREE”, after the random forest algorithm used to assign confidence scores. This information was aggregated from a series of tables that contained both author, paper or article data, as well as tables with conference and journal information. The author pair confidence values were precomputed so ALIAS will run quickly, and when a search or request is made the table is queried over the network. Thus multiple users can access the data in a distributed fashion simultaneously

Implementation Details

A DBSource object was created to interface between the user interface and the data base. It was designed to be implementation independent, so that it could be extended to talk to a generic server rather than a SQL server. Currently DBSource interfaces directly with the SQL server and uses hard coded implementation independent SQL queries. It stores author data in a an Author class that extends a HashMap, mapping author data to character String

values.

The data was generated by first running a Java program that collected all pertinent data about an author from the various source tables, such as the journals they had published to and conferences attended. It then, for each possible pairings of authors found the similarity for each field. In the case of expertise this was a zero or one matching. However, for years published, journals, and conferences an overlap was calculated which was the number of items both authors had in common divided by the smaller cardinality of the two author membership sets. Once this data was computed for all author pairs, the random forest algorithm was on the data and the results put into ALIAS_TREE.

The GUI uses a series of classes that extend the JPanel class from the java Swing package. The two main panels are the AuthorSearchPanel and AuthorDisplayPane, which are

AuthorSearchPanel maintains the top half of the GUI; author name entry and display of the list of potentially duplicate or similar authors. It also stores and displays the command buttons that activate duplicate and similarity searches. It uses a SuggestionList object to create and pop up the suggestions when a user enters and searches on an author name, and set the selected author as the selected author, via the AuthorSettable object.

The AuthorSearchPanel uses an AuthorTableModel to format and control the table that displays duplicate or similar authors. This allows selections made to be updated immediately and keeps the width of columns reasonable. Finally, AuthorSearchPanel uses a TableCoumnAdjuster to dynamically update the contents of it's table, including headers.

The AuthorDisplayPane manages the lower half of the user interface, which displays both the selected, searched by name, author details and the duplicate or similar author details. It uses a DisplayModel, which extends the AbstractTableModel and provides the formatting for the details display table. The TableColumnAdjuster class is also used to maintain good column width and dynamically update table data.

The overview of the gui classes is provided in Figure 5.

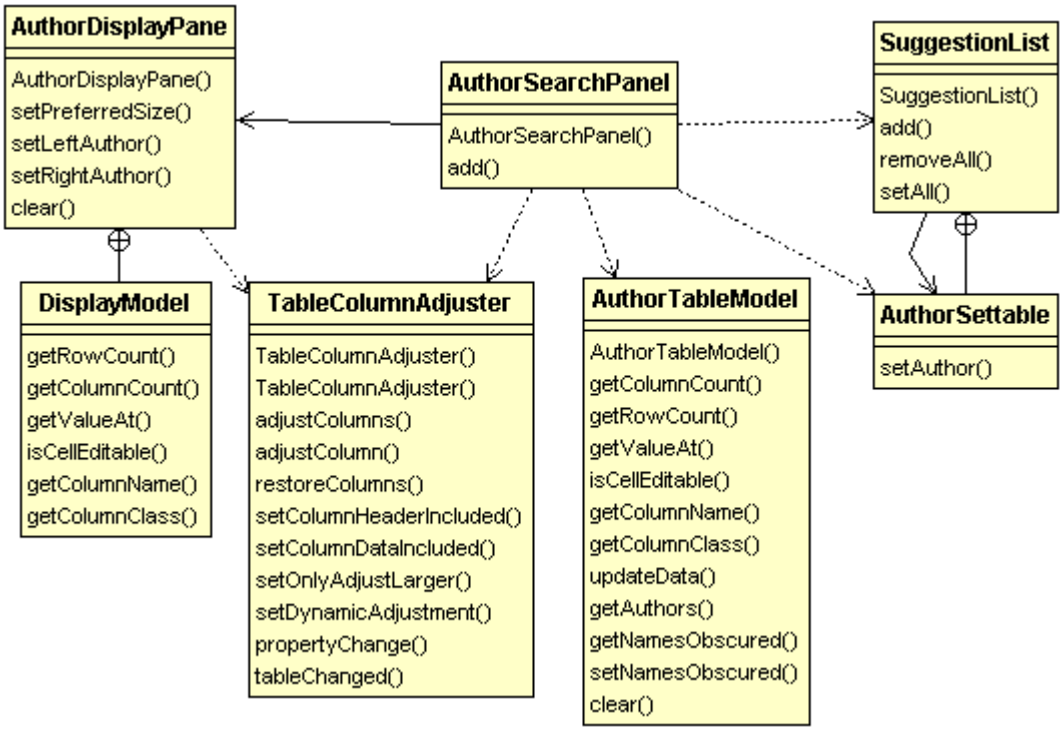

Fig. 5 The UML diagram for the GUI classes.

Future Work

ALIAS currently does not examine author name when determining similarity. There was an algorithm developed that could take a pair of character Strings, break them into two sets of Strings and then find the optimal pairing of Stings from one set with the other. Optimal was defined as the total Levenshtein [8] distance for each pairing. Initial testing showed that for author pairings that were not tagged as duplicates by the authors had an average distance of 14, while those that had been tagged as duplicates had an average distance of 1.6. Adding name matching could improve duplicate detection in future versions of ALIAS.

Also, due to the large number of published authors available, there are plans to cluster the data. This would allow new authors to be added much more quickly, without having to compare a new author with all existing authors.

Acknowledgments

The author is grateful to his adviser Professor Basu Roy for her leadership and guidance through out the project, she was generous with both her time and patience. The author is also grateful to the Microsoft Academic Search team for providing the crucial date set and providing much needed support. The author would also like to thank Professor Teredesai of the Web and Data Analytics center who helped make the initial connections and provided much needed advice and counsel Finally, the author would like to thank Dr. Friedman, the director of the University of Washington, Tacoma's Institute of Technology, who's organizational and linguistic skill added in the drafting of the original proposal.

References

[1] A. Arasu, M. G¨otz, and R. Kaushik, “On active learning of record matching packages,” in SIGMOD Conference, 2010, pp. 783–794.

[2] A. Arasu and R. Kaushik, “A grammar-based entity representation framework for data cleaning,” in SIGMOD Conference, 2009, pp. 233–244.

[3] A. Arasu, S. Chaudhuri, and R. Kaushik, “Transformation-based framework for record matching,” in ICDE, 2008, pp. 40–49.

[4] C.-C. Chen, K.-H. Yang, C.-L. Chen, and J.-M. Ho, “Bibpro: A citation parser based on sequence alignment,” IEEE Trans. Knowl. Data Eng., vol. 24, no. 2, pp. 236–250, 2012.

[5] K.-H. Yang and Y. H. Wu, “Author name disambiguation in citations,” in Web Intelligence/IAT Workshops, 2011, pp. 335–338.

[6] R. Zhang, D. Shen, Y. Kou, and T. Nie, “Author name disambiguation for citations on the deep web,” in Proceedings of the 2010 international conference on Web-age information management, ser. WAIM’10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 198–209. [Online]. Available: http://dl.acm.org/citation.cfm?id=1927585.1927609

[7] J. Han and M. Kamber. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2000.

[8] Levenshtein VI (1966). "Binary codes capable of correcting deletions, insertions, and reversals". Soviet Physics Doklady 10: 707–10.