ALIAS: Author Disambiguation in Microsoft Academic Search Engine Dataset

Michael Pitts #1, Swapna Savvana #2, Senjuti Basu Roy #3, Vani Mandava *4, Dhineshkumar Prasath #5

Institute of Technology,CWDS, University of Washington Tacoma
[1] pittsm@uw.edu, [2] ssavvana@uw.edu, [3] senjutib@uw.edu, [5] dkp11@uw.edu

* *Microsoft Research, USA*
[4] vanim@exchange.microsoft.com

ABSTRACT

We present a system called ALIAS, that is designed to search for duplicate authors from Microsoft Academic Search Engine dataset. *Author-ambiguity* is a prevalent problem in this dataset, as many authors publish under several variations of their own name, or different authors share similar or same name. ALIAS takes an author name as an input (who may or may not exist in the corpus), and outputs a set of author names from the database, that are determined as duplicates of the input author. It also provides a confidence score with each output. Additionally, ALIAS has the feature of finding a Top-k list of *similar* authors, given an input author name. The underlying techniques heavily rely on a mix of learning, mining, and efficient search techniques, including partitioning, clustering, supervised learning using ensemble algorithms, and indexing to perform efficient search to enable fast response for near real time user interaction. While the system is designed using Academic Search Engine data, the proposed solution is generic and could be extended to other problems in the category of entity disambiguation. In this demonstration paper, we describe different components of ALIAS and the intelligent algorithms associated with each of these components to perform author name disambiguation or similar authors finding.

1. INTRODUCTION

The ability to search literature and collect/aggregate metrics around publications is of pivotal interest in modern research. Specialized search engine technologies (such as Google scholar, Microsoft Academic Search) are designed to cater to the academic and industry researchers across hundreds of scientific disciplines to enable search, browsing, and exploration functionalities over different scientific disciplines and domains. Microsoft Academic Search[1] (henceforth referred to as MAS) allows search and exploration over a wide variety of scientific disciplines, from astronomy to zoology. It currently covers more than 50 million publications and over 19 million authors, with updates added each week. Prevalence of noise is one of the hardest problems in this dataset, as the raw data is collected by crawling and extracting data over hundreds of publishers websites, 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. Thus *author-ambiguity* is a tremendous challenge that exacerbates the overall search experience. Figure 1 and Figure 2 explain these scenarios using MAS dataset examples.

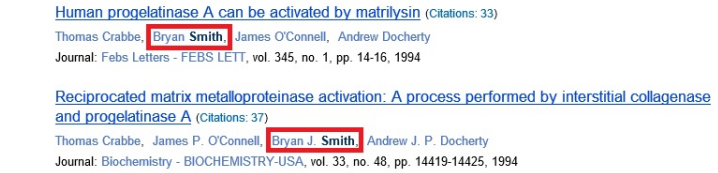


Figure 1: Example of author name variations: the names "Bryan J Smith" and "Bryan Smith" refer to the same person

In the light of this opportunity, we develop ALIAS[2], an intelligent and effective system that determines duplicate authors from the MAS corpus, given an input author. ALIAS makes the following contributions- a) The user interface of ALIAS has autocomplete suggestions based on prefix matching on author name. An author who does not exist in the corpus, she could be added to the database by the user. b) ALIAS outputs a set of duplicate authors, given an input author. While the actual disambiguation algorithm is designed using intelligent supervised learning algorithms, the user interface still accepts incomplete inputs (for input authors who do not exist in the database) and the remaining metadata values are *imputed* using novel data mining solutions. Additional computational challenges arise to enable this functionality, as ALIAS has to handle all possible combinations of incomplete author input. A multi-layer inverted index [7] is designed to enable fast search. ALIAS has an additional feature of finding *similar authors*, where the *similarity* is formalized by considering an exhaustive set of author metadata and an efficient search is conducted using novel *multi-dimensional* indexing techniques. d) The disambiguation and the similar author finding algorithm also output a confidence score with each output. The interface enables side-by-side comparison between each output and the input.

The problem of record matching and de-duplication has been studied in the past [1, 2, 3]. Most of the previously studied techniques rely heavily on approximate string matching to find the appropriate solution. Specific to the interest of academic publication and related applications, a recent work proposes BibPro [4], a ci-

[1] http://academic.research.microsoft.com/

[2] Youtube video link: http://youtu.be/LxA4Ms0U75U

10.5441/002/edbt.2014.65

Figure 2: Example of a paper that is co-authored by an author with an ambiguous name and missing affiliation

tation 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. Conversely, our proposed approach primarily relies on a series of techniques - such as, partitioning, clustering, and supervised learning algorithms with the objective to learn *the importance of different author metadata in the disambiguation task*. The aforementioned techniques are designed with the help of rigorous feature engineering that captures explicit as well as implicit characteristics of author metadata. We note that much of our proposed solutions can be generalized to other problems in the category of entity disambiguation. Furthermore, ALIAS accepts author names as inputs that do not need to exist in the database, or the user could input incomplete author metadata. Finally, ALIAS enables near real-time interaction with the user by using novel indexing techniques.

To summarize, ALIAS makes the following contributions:

- We design ALIAS, a novel author disambiguation system using MAS dataset that accepts an author input in the user interface, and outputs duplicate authors (along with their respective confidence scores) from MAS corpus.
- ALIAS also returns a ranked list of k similar authors and respective confidence score, given an input author.
- To guarantee real-time interaction, ALIAS is designed using an offline and an online layer. The technical solution of ALIAS constitutes a mix of various learning techniques and mining algorithms, combined with indexing techniques. In particular, the offline layer consists of partitioning, clustering, supervised learning techniques, a multi-layer inverted index and a multi-dimensional index. The online layer makes use of the indexes to perform efficient *duplicate detection* or *similarity search*.
- Unlike existing work on entity disambiguation, the duplicate detection task is formulated as a supervised learning problem, by designing an ensemble of classification techniques with the objective to learn appropriate functions leveraging author metadata to determine duplicates.

Next, we first describe the technical specifications of ALIAS and then demonstrate application scenarios with the system.

2. TECHNICAL SPECIFICATION

2.1 System Overview

ALIAS is designed as a desktop or a web application that has two primary features: 1) **Duplicate Detection** and 2) **Similar Author Finding**. The majority of the system components are precomputed and stored to increase the speed of the application. Figure 3 presents the system overview for Duplicate Detection feature. Since different authors may have *similar* names, the author names of the MAS corpus are first partitioned based on name similarity, as described in Section 2.2. After that, a set of clusters are designed inside each partition. Next, the duplicate detection task is defined

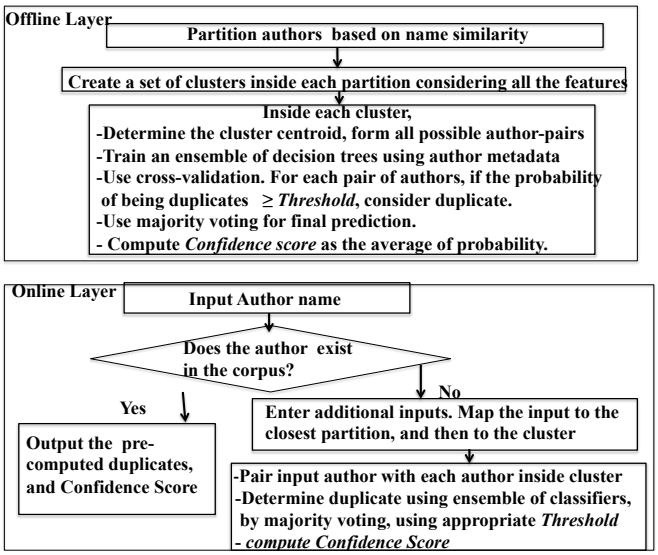
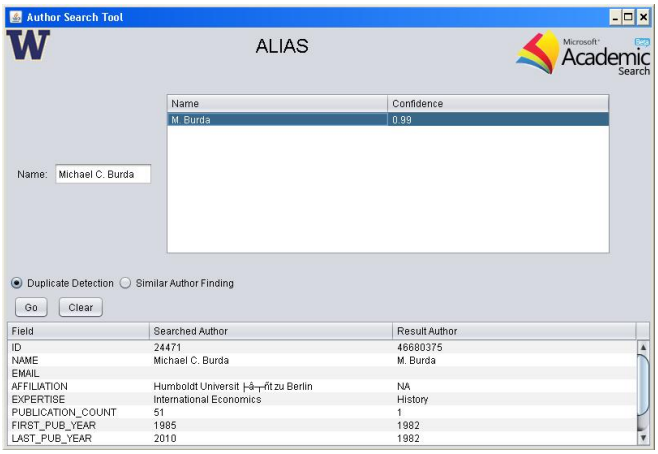


Figure 3: Overview of duplicate detection process in ALIAS

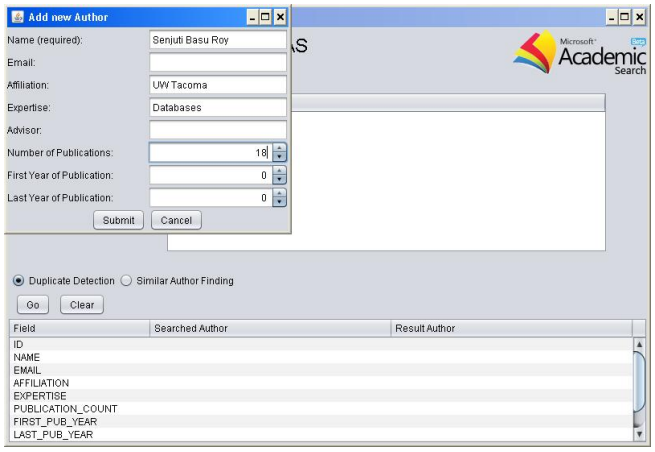
as a binary classification task inside each cluster, where the objective is to leverage author metadata to identify a pair of authors as duplicates or not. An ensemble of decision tree classifiers [7] are designed inside each cluster. MAS website allows authenticated users (i.e., authors) to make direct edits in the author-profiles by adding, confirming, or deleting respective publications metadata, affiliation, and area of interests information, or by sending requests to merge multiple author profiles. This author edit data constitutes the *ground truth* in ALIAS and is used to create the labeled data for supervised learning (details in Section 2.2). The classification algorithms use author meta-data that a user may not have entered as inputs in the interface. Therefore, ALIAS handles such cases by *imputing* the missing author metadata using clustering techniques, such that the classification algorithm could make use of it for duplicate detection. In particular, ALIAS creates a set of clusters (more details in Section 2.3 and 2.2) and maintains a *centroid* for each cluster, which is used to *impute* the remaining missing attribute values. Given an author name that does not exist in the corpus, ALIAS first detects the *most similar partition*, and then use other author metadata to perform efficient matching to map the inputs to its closest cluster inside the partition. Efficient partition finding and centroid matching are the only two online processes that take place at runtime. For Similar Author Finding feature, each author record is transformed to a point in a multi-dimensional space, and the entire search space is indexed using multidimensional indexing [8] for efficient similarity search. This index is created offline, and a top-k nearest neighbor search interface is designed for the online computation. Section 2.2 describes the offline component, whereas Section 2.3 presents the online computations.

ALIAS is primarily implemented in Java, using R-studio and SQL Server. The raw author corpus and related metadata are stored as relational tables using SQL-Server. Java provides the main user interface functionality to the client and also runs on the server to allow program interaction. R-studio is used for partitioning, clustering, supervised learning algorithm implementations, the results of which are stored in a SQL database server and later retrieved. The multidimensional indexing is primarily implemented using Java that interacts directly with the author records stored in the SQL database.

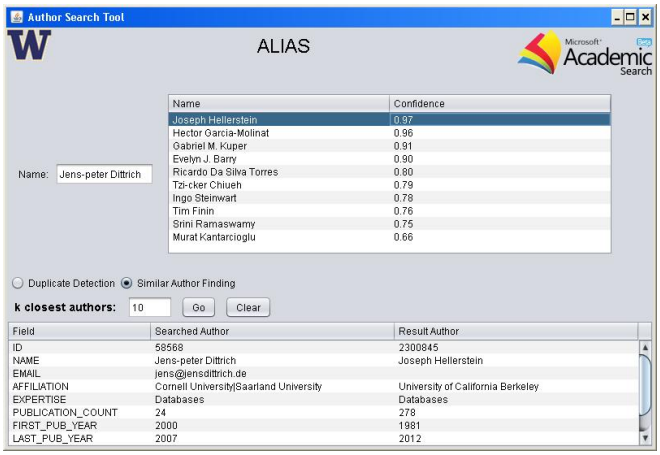


(a) Duplicate Detection -Scenario 1

(b) Duplicate Detection -Scenario 2 (a)



(c) Duplicate Detection -Scenario 2 (b)



(d) Similar Author Finding

Figure 4: Different Functionalities of ALIAS

2.2 Offline Layer

In this section, we present the details of the five offline components: partitioning based on name similarity, clustering inside each partition, classification algorithm inside each cluster, multi-layer inverted index, and multidimensional index for similarity search.

Partitioning: ALIAS pre-computes a set of partitions based on author name similarity. The partition is designed considering individual substring matching on first, middle, and last names between author pairs, and aggregating them at the end. A pair of authors are placed into the same partition if their *aggregated Levenshtein distance* [7] is smaller than a particular threshold value β. The objective is to detect duplicates efficiently, and avoid all pair match.

Clustering: Inside each partition, ALIAS pre-computes a set of clusters based on all other remaining metadata. ALIAS uses following metadata (i.e., attributes): Affiliation, Expertise, Primary expertise, Advisor Information, Publication Count, Collaboration count, First year of publication, Last year of publication. Clustering is designed to expedite online computation and to handle incomplete author input. The intuition is to compare an input author only with a subset of *similar* authors in the MAS corpus and block comparison with the rest. Our implementation uses k-Medoid clustering algorithm [7] for clustering. Our initial experimental results have demonstrated that the best results are obtained when $k = 10$ clusters are created on an average inside each partition. As we shall see in Section 2.3, the centroids of the clusters are used to *impute* incomplete author inputs. Intuitively, given an input author whose "Expertise" field is left blank in the interface, ALIAS imputes that with the value "Database", if this author is *most similar* to a cluster centorid which has the "Expertise=Database".

Ensemble of Classifiers: Inside each cluster, the duplicate detection solution is designed as a *binary classification task* for an author pair. Given an author-pair, appropriate metadata are extracted and a single *vector* is created that contains individual characteristics of each of the two authors, as well as some characteristics that capture relative similarity between an author pair. The individual characteristics are extracted using author metadata as described above, whereas, the latter type of characteristics are captured considering the following additional factors: Conference overlaps, Journal overlaps, Keyword overlap, and Domain overlap[3]. These characteristics are typically known as *features*. The overall task is to design a function that accurately classifies an author-pair as duplicates or otherwise. Mathematically, given author-pair features X, the task is to predict the class label Y (0= not-duplicates/1=duplicates) by designing a classifier C,

$$C : X \rightarrow C(X)$$

[3]Overlap is calculated as the Jaccard Index.

such that the error in classification $P(C(X) \neq Y)$ is minimized. A single classifier designed for such purpose comes inadequate to make accurate classification due to noise and imbalance in the dataset. Instead, we propose a series of classifiers (ensemble) considering random partitions of the feature space. Each classifier outputs its own decision; the idea is to combine these individual decisions at the end to accurately classify new examples. Previous work [9] has demonstrated that *majority voting* is guaranteed to achieve an overall higher accuracy of an ensemble method when the individual classifiers have accuracy > 0.5, which we also use to combine individual classification decisions.

In our implementation, we design an ensemble with 200 decision trees using majority voting, where each decision tree makes an independent decision. A pair of authors are marked duplicate by a classifier, when the probability of being duplicate is greater than a threshold (α is empirically chosen to be 0.8). The overall *confidence score* is chosen to be the average probability over all the classifiers that classifies the pair to be duplicate.

Multi-layer Inverted Index: Two layers of inverted indices are designed over the pre-computed partitions, and on the clusters inside each partition to efficiently match an author input to its closest cluster inside a partition for duplicate detection.

Multidimensional Indexing: The objective is to perform efficient *similarity* search on author metadata. We transform each author metadata (excluding name and email id) to a point in the multidimensional space, and the entire search space is indexed. A top-k nearest neighbor search interface is designed that takes an author input as the query and returns a ranked list of k-most similar authors. For our implementation purpose, we use Euclidean distance [7] as the distance metric. The *confidence score* is computed as the normalized similarity (1 - normalized distance) between the input author and an output.

2.3 Online Layer

As the user types the name, a list of auto-suggestion shows up based on prefix matching. For duplicate detection, if the author name already exists in the database, its corresponding duplicate list is retrieved from the stored results along with the computed *confidence score*. Otherwise, additional inputs are requested from the user (refer to Figure 4b and 4c). The author name is first mapped to the *closest* pre-computed partition and then to the *closest* cluster. Technically, the process thus becomes computing the *similarity (or distance)* [7] between the cluster-centroid and the input values and selecting that centroid that has the highest similarity (smallest distance) with the inputs. While we are aware of several measures to compute similarity or distance, our current implementation considers *Cosine Similarity* [7] for that purpose. To allow incomplete inputs, the total number of materialized cluster centroids is exponential inside each partition(more specifically, $n \times 2^m$, where n-clusters are materialized for each possible subset of m attributes), the challenge is to be able to perform this similarity computation efficiently at run-time. The multi-layer inverted index described above is used to efficiently identify a partition and then the cluster inside the partition. The advantage of using this indexing scheme is that it avoids the exponential number of similarity computation at run-time. In particular, `ALIAS` identifies the closest partition in constant time and only performs n comparisons inside a partition to find out the centroid with the highest Cosine Similarity. The cluster centroid along to impute the missing values along with the input author metadata constitutes a complete test record. The trained classifiers are then used to detect the duplicate list.

For similar author finding feature, the user has to additionally specify an integer value k. A call goes to the pre-computed multidimensional index structure to return the top-k ranked list of authors who are most similar to the input author.

3. SYSTEM DEMONSTRATION

MAS dataset is used to demonstrate `ALIAS`. It consists of $250,000$ unique author records and related metadata, selected randomly from the larger MAS corpus, but only for Computer Science domain. Our demonstration contains three main processes: 1) Flexible Input; 2) Duplicate Detection; 3) Similar Author Finding.

As shown in Figure 4a, Step 1 involves typing an author name. A list of autosuggestion shows up based on prefix matching. The user can then select either of the features. If the author exists in the database, based on the selected feature (Duplicate Detection or Similar Author Finding), the appropriate output shows up (Figure 4a or 4d respectively), leading to Step 2 or 3.

If the input author does not exist in the corpus (refer to Figure 4b and 4c), an alert window pops up, and a form is provided to enter additional metadata values for that input author. After that, the user can choose either of the features, and based on that, she would be lead to Step 2 or Step 3 (e.g., Figure 4c shows the case when the user has chosen Duplicate Detection Feature.).

For both Step 2 and Step 3, an additional window at the bottom provides attribute-by-attribute comparison between the input author and any author in the output that the user has clicked on.

4. CONCLUSION

In this paper we propose `ALIAS`, a system that aims to perform duplicate author detection and top-k similar author finding using MAS dataset. `ALIAS` uses a mix of mining techniques, learning algorithms and indexing techniques including partitioning, clustering, classification, multi-layer inverted indices, and multidimensional indexing. `ALIAS` outputs a confidence score for duplicate detection and similar author finding tasks and accepts incomplete author metadata as inputs. To the best of our knowledge, `ALIAS` is the first ever tool with these improved functionalities. The proposed solutions of `ALIAS` are generic and could be adopted to address the task of entity disambiguation and related problems.

5. REFERENCES

[1] A. Arasu, M. Götz, 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.

[6] R. Zhang, D. Shen, Y. Kou, and T. Nie, "Author name disambiguation for citations on the deep web," ser. WAIM'10, 2010, pp. 198–209.

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

[8] A. Guttman, "R-trees: a dynamic index structure for spatial searching," *SIGMOD Rec.*, vol. 14, no. 2, Jun. 1984.

[9] L. Lam and S. Y. Suen, "Application of majority voting to pattern recognition: an analysis of its behavior and performance," *Trans. Sys. Man Cyber. Part A*.