Ethnicity sensitive author disambiguation using semi-supervised learning

Gilles Louppe[1], Hussein T. Al-Natsheh[2], Mateusz Susik[3], and Eamonn James Maguire[1]

[1] CERN, Switzerland,
{g.louppe, eamonn.james.maguire}@cern.ch,
[2] ISH-LYON (USR 3385), CNRS, France,
hussein.al-natsheh@ish-lyon.cnrs.fr
[3] University of Warsaw, Poland,
msusik@student.uw.edu.pl

Abstract. Author name disambiguation in bibliographic databases is the problem of grouping together scientific publications written by the same person, accounting for potential homonyms and/or synonyms. Among solutions to this problem, digital libraries are increasingly offering tools for authors to manually curate their publications and claim those that are theirs. Indirectly, these tools allow for the inexpensive collection of large annotated training data, which can be further leveraged to build a complementary automated disambiguation system capable of inferring patterns for identifying publications written by the same person. Building on more than 1 million crowdsourced annotations that we publicly release, we propose an automated author disambiguation solution exploiting this data (i) to learn an accurate classifier for identifying coreferring authors and (ii) to guide the clustering of scientific publications by distinct authors in a semi-supervised way. To the best of our knowledge, our analysis is the first to be carried out on data of this size and coverage. With respect to the state of the art, we validate the general pipeline used in most existing solutions, and improve by: (i) proposing phonetic-based blocking strategies, thereby increasing recall; (ii) adding strong ethnicity-sensitive features for learning a linkage function, thereby tailoring disambiguation to non-Western author names whenever necessary; and (iii) showing the importance of balancing negative and positive examples when learning the linkage function.

1 Introduction

In academic digital libraries, author name disambiguation is the problem of grouping together publications written by the same person. It is often difficult because an author may use different spellings or name variants across their career (synonymy) and/or distinct authors may share the same name (polysemy). Most notably, author disambiguation is often more troublesome for researchers from non-Western cultures, where personal names may be traditionally less diverse (leading to homonym issues) or for which transliteration to Latin characters may not be unique (leading to synonym issues). With the fast growth of the scientific literature, author disambiguation has become a pressing issue since

the accuracy of information managed at the level of individuals directly affects: the relevance search of results (e.g., when querying for all publications written by a given author); the reliability of bibliometrics and author rankings (e.g., citation counts or other impact metrics, as studied in [26]); and/or the relevance of scientific network analysis [20]. Thus, even small improvements in the field significantly improve the usability of the digital libraries to some users.

Efforts and solutions to author disambiguation have been proposed from various communities [16]. On the one hand, libraries have maintained authorship control through manual curation, either in a centralized way by hiring professional collaborators or through developing services that invite authors to register their publications themselves (e.g., Google Scholar or Inspire-HEP). Recent efforts to create persistent digital identifiers assigned to researchers (e.g., ORCID or ResearcherID), with the objective to embed these identifiers in the submission workflow of publishers or repositories (e.g., Elsevier, arXiv or Inspire-HEP), would univocally solve any disambiguation issue. As the centralized manual authorship control is expensive and the success of persistent digital identifiers requires large and ubiquitous adoption by both researchers and publishers, fully automated machine learning-based methods have been proposed during the past decade to provide immediate, less costly, and satisfactory solutions to author disambiguation. In this work, our goal is to explore and demonstrate how both approaches can coexist and benefit from each other. In particular, we study how labeled data obtained through manual curation (either centralized or crowdsourced) can be exploited (i) to learn an accurate classifier for identifying coreferring authors, and (ii) to guide the clustering of scientific publications by distinct authors in a semi-supervised way. Our analysis of parameters and features of this large dataset reveal that the general pipeline commonly used in existing solutions is an effective approach for author disambiguation. Additionally, we propose alternative strategies for blocking based on the phonetization of author names to increase recall and ethnicity-sensitive features for learning a linkage function which tailor our author disambiguation to non-Western author names whenever necessary.

The remainder of this report is structured as follows. In Section 2, we briefly review machine learning solutions for author disambiguation. The components of our method are then defined in Section 3 and its implementation described in Section 4. Experiments are carried out in Section 5, where we explore and validate features for the supervised learning of a linkage function and compare strategies for the semi-supervised clustering of publications. Finally, conclusions and future works are discussed in Section 6.

2 Related work

As reviewed in [24,5,15], author disambiguation algorithms are usually composed of two main components: (i) a linkage function determining whether two publications have been written by the same author; and (ii) a clustering algorithm producing clusters of publications assumed to be written by the same author. Approaches can be classified along several axes, depending on the type and amount of data available, the way the linkage function is learned or defined, or the clustering procedure used to group publications. Methods relying

on supervised learning usually make use of a small set of hand-labeled pairs of publications identified as being either from the same or different authors to automatically learn a linkage function between publications [10,11,**?**,30,**?**].

Training data is usually not easily available, therefore unsupervised approaches propose the use of domain-specific, manually designed, linkage functions tailored towards author disambiguation [18,19,25,**?**,**?**,**?**,**?**]. These approaches have the advantage of not requiring hand-labeled data, but generally do not perform as well as supervised approaches. To reconcile both worlds, semi-supervised methods make use of small, manually verified, clusters of publications and/or high-precision domain-specific rules to build a training set of pairs of publications, from which a linkage function is then built using supervised learning [6,29,15].

Semi-supervised approaches also allow for the tuning of the clustering algorithm when the latter is applied to a mixed set of labeled and unlabeled publications, e.g., by maximizing some clustering performance metric on the known clusters [15].

Due to the lack of large and publicly available datasets of curated clusters of publications, studies on author disambiguation are usually constrained to validating their results on manually built datasets of limited size and scope (from a few hundred to a few thousand papers, with sparse coverage of ambiguous cases), making the true performance of these methods often difficult to assess with high confidence. Additionally, despite devoted efforts to construct them, these datasets are rarely public, making it even more difficult to compare methods using a common benchmark.

In this context, we position the work in this paper as a semi-supervised solution for author disambiguation, with the significant advantage of having a very large collection of more than 1 million crowdsourced annotations of publications whose true authors are identified. The extent and coverage of this data allows us to revisit, validate and nuance previous findings regarding supervised learning of linkage functions, and to better explore strategies for semi-supervised clustering. Furthermore, by releasing our data in the public domain, we hope to provide a benchmark on which further research on author disambiguation and related topics can be built.

3 Semi-supervised author disambiguation

Formally, let us assume a set of publications $\mathcal{P} = \{p_0, ..., p_{N-1}\}$ along with the set of unique individuals $\mathcal{A} = \{a_0, ..., a_{M-1}\}$ having together authored all publications in $\mathcal{P}$. Let us define a signature $s \in p$ from a publication as a unique piece of information identifying one of the authors of p (e.g., the author name, his affiliation, along with any other metadata that can be derived from p, as illustrated in Figure 1). Let us denote by $\mathcal{S} = \{s | s \in p, p \in \mathcal{P}\}$ the set of all signatures that can be extracted from all publications in $\mathcal{P}$.

In this framework, author disambiguation can be stated as the problem of finding a partition $\mathcal{C} = \{c_0, ..., c_{M-1}\}$ of $\mathcal{S}$ such that $\mathcal{S} = \cup_{i=0}^{M-1} c_i$, $c_i \cap c_j = \phi$ for all $i \neq j$, and where subsets c_i, or clusters, each corresponds to the set of all signatures belonging to the same individual a_i. Alternatively, the set $\mathcal{A}$ may remain (possibly partially) unknown, such that author disambiguation boils

Fig. 1. An example signature s for "Doe, John". A *signature* is defined as unique piece of information identifying an author on a publication, along with any other metadata that can be derived from it, such as publication title, co-authors or date of publication.

down to finding a partition $\mathcal{C}$ where subsets c_i each correspond to the set of all signatures from the same individual (without knowing who). Finally, in the case of partially annotated databases as studied in this work, the set extends with the partial knowledge $\mathcal{C}' = \{c'_0, ..., c'_{M-1}\}$ of $\mathcal{C}$, such that $c'_i \subseteq c_i$, where c'_i may be empty. Or put otherwise, the set extends with the assumption that all signatures $s \in c'_i$ belong to the same author.

Our algorithm is composed of three parts, as sketched in Figure 2: (i) a blocking scheme whose goal is to roughly pre-cluster signatures $\mathcal{S}$ into smaller groups in order to reduce computational complexity; (ii) the construction of a linkage function d between signatures using supervised learning; and (iii) the semi-supervised clustering of all signatures within the same block, using d as a pseudo distance metric.

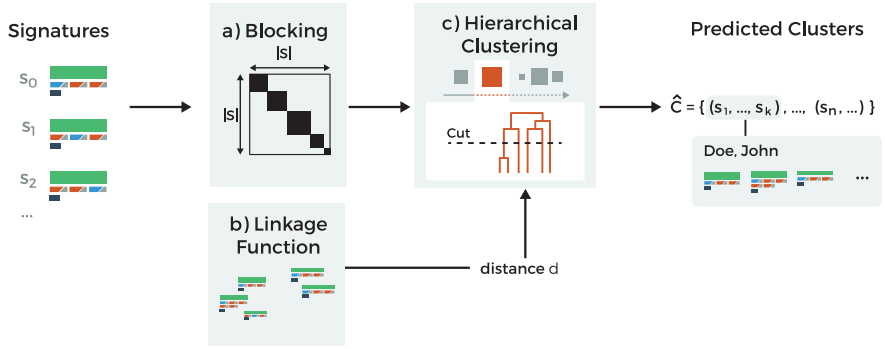


Fig. 2. Pipeline for author disambiguation: (a) signatures are *blocked* to reduce computational complexity, (b) a linkage function is built with supervised learning, (c) independently within each block, signatures are grouped using hierarchical agglomerative clustering.

3.1 Blocking

As in previous works, the first part of our algorithm consists of dividing signatures $\mathcal{S}$ into disjoint subsets $\mathcal{S}_{b_0}, ..., \mathcal{S}_{b_{K-1}}$, or *blocks* [4], followed by carrying out author disambiguation on each one of these blocks independently. By doing so, the computational complexity of clustering (see Section 3.3) typically reduces from $O(|\mathcal{S}|^2)$ to $O(\sum_b |\mathcal{S}_b|^2)$, which is much more tractable as the number of signatures increases. Since disambiguation is performed independently per block, a good blocking strategy should be designed such that signatures from the same author are all mapped to the same block, otherwise their correct clustering would not be possible in later stages of the workflow. As a result, blocking should be a balance between reduced complexity and maximum recall.

The simplest and most common strategy for blocking, referred to hereon in as *Surname and First Initial (SFI)*, groups signatures together if they share the same surname(s) and the same first given name initial (e.g., *SFI*("Doe, John") == "Doe, J"). Despite satisfactory performance, there are several cases where this simple strategy fails to cluster related pairs of signatures together, including:

1. There are different ways of writing an author name, or signatures contain a typo (e.g., "Mueller, R." and "Muller, R.").
2. An author has multiple surnames and some signatures place the first part of the surname within the given names (e.g., "Martinez Torres, A." and "Torres, A. Martinez").
3. An author has multiple surnames and, on some signatures, only the first surname is present (e.g., "Smith-Jones, A." and "Smith, A.")
4. An author has multiple given names and they are not always all recorded (e.g., "Smith, Jack" and "Smith, A. J.")
5. An authors surname changed (e.g., due to marriage).

To account for these issues we propose instead to block signatures based on the phonetic representation of the normalized surname. Normalization involves stripping accents (e.g., "Jabłoński, Ł" → "Jablonski, L") and name affixes that inconsistently appear in signatures (e.g., "van der Waals, J. D." → "Waals, J. D."), while phonetization is based either on the Double Metaphone [22], the NYSIIS [27] or the Soundex [28] phonetic algorithms for mapping author names to their pronunciations. Together, these processing steps allow for grouping of most name variants of the same person in the same block with a small increase in the overall computational complexity, thereby solving case 1.

In the case of multiple surnames (cases 2 and 3), we propose to block signatures in two phases. In the first phase, all the signatures with a single surname are clustered together. Every different surname token creates a new block. In the second phase, the signatures with multiple surnames are compared with the blocks for the first and last surname. If the first surnames of an author were already used as the last given names on some of the signatures, the new signature is assigned to the block of the last surname (case 2). Otherwise, the signature is assigned to the block of the first surname (case 3). Finally, to prevent the creation of too large blocks, signatures are further divided along their first given name initial. Cases 4 and 5 are not explicitly handled.

3.2 Linkage function

Supervised classification. The second part of the algorithm is the automatic construction of a pair-wise linkage function between signatures for use during the clustering step which groups all signatures from the same author.

Formally, the goal is to build a function $d : \mathcal{S} \times \mathcal{S} \mapsto [0, 1]$, such that $d(s_1, s_2)$ approaches 0 if both signatures s_1 and s_2 belong to the same author, and 1 otherwise. This problem can be cast as a standard supervised classification task, where inputs are pairs of signatures and outputs are classes 0 (same authors), and 1 (distinct authors). In this work, we evaluate Random Forests (RF, [1]), Gradient Boosted Regression Trees (GBRT, [7]), and Logistic Regression [3] as classifiers.

Input features. In most cases, supervised learning algorithms assume the input space $\mathcal{X}$ to be numeric (e.g., $\mathbb{R}^p$), making them not directly applicable to structured input spaces such as $\mathcal{S} \times \mathcal{S}$. Following previous works, pairs of signatures (s_1, s_2) are first transformed to vectors $v \in \mathbb{R}^p$ by building so-called similarity profiles [30] on which supervised learning is carried out. In this work, we design and evaluate fifteen standard input features based on the comparison of signature fields, as reported in the first half of Table 1. As an illustrative example, the *Full name* feature corresponds to the similarity between the (full) author name fields of the two signatures, as measured using as combination operator the cosine similarity between their respective (n, m)-*TF-IDF* vector representations[4].

Author names from different cultures, origins or ethnic groups are likely to be disambiguated using different strategies (e.g., pairs of signatures with French author names versus pairs of signatures with Chinese author names) [31,**?**]. To support our disambiguation algorithm, we added seven features to our feature set, with each evaluating the degree of belonging of both signatures to an ethnic group. [MS: Explain here ethnicity features (ethnicities come from some kind of US Census. Name those groups (as I removed them from the table))]

More specifically, using census data extracted from [23], we build a support vector machine classifier (using a linear kernel and one-versus-all classification scheme) for mapping the $(1, 5)$-TF-IDF representation of an author name to one of the seven ethnic groups. Given a pair of signatures (s_1, s_2), the proposed ethnicity features are each computed as the estimated probability of s_1 belonging to the corresponding ethnic group, multiplied by the estimated probability of s_2 belonging to the same group. In doing so, the expectation is for the linkage function to become sensitive to the actual origin of the authors depending on the values of these features. Indirectly, let us also note that these features hold discriminative power since if author names are predicted to belong to different ethnic groups, then they are also likely to correspond to distinct people.

Building a training set. The distinctive aspect of our work is the knowledge of more than 1 million crowdsourced annotations $\mathcal{C}' = \{c'_0, ..., c'_{M-1}\}$, indicating together that all signature $s \in c'_i$ are known to correspond to the same individual a_i. In particular, this data can be used to generate positive

[4] (n, m) denotes that the *TF-IDF* vectors are computed from character n, $n + 1$, ..., m-grams. When not specified, *TF-IDF* vectors are otherwise computed from words.

Table 1. Input features for learning a linkage function

Feature	**Combination operator**
Full name	Cosine similarity of $(2,4)$-TF-IDF
Given names	Cosine similarity of $(2,4)$-TF-IDF
First given name	Jaro-Winkler distance
Second given name	Jaro-Winkler distance
Given name initial	Equality
Affiliation	Cosine similarity of $(2,4)$-TF-IDF
Co-authors	Cosine similarity of TF-IDF
Title	Cosine similarity of $(2,4)$-TF-IDF
Journal	Cosine similarity of $(2,4)$-TF-IDF
Abstract	Cosine similarity of TF-IDF
Keywords	Cosine similarity of TF-IDF
Collaborations	Cosine similarity of TF-IDF
References	Cosine similarity of TF-IDF
Subject	Cosine similarity of TF-IDF
Year difference	Absolute difference
Any ethnicity feature	Product of estimated probabilities

pairs $(x = (s_1, s_2), y = 0)$ for all $s_1, s_2 \in c'_i$, for all i. Similarly, negative pairs $(x = (s_1, s_2), y = 1)$ can be extracted for all $s_1 \in c'_i, s_2 \in c'_j$, for all $i \neq j$.

The most straightforward approach for building a training set on which to learn a linkage function is to sample an equal number of positive and negative pairs, as suggested above. By observing that the linkage function d will eventually be used only on pairs of signatures from the same block S_b, a further refinement for building a training set is to restrict positive and negative pairs (s_1, s_2) to only those for which s_1 and s_2 belong to the same block. In doing so, the trained classifier is forced to learn intra-block discriminative patterns rather than inter-block differences. Furthermore, as noted in [14], most signature pairs are non-ambiguous: if both signatures share the same author names, then they correspond to the same individual, otherwise they do not. Rather than sampling pairs uniformly at random, we propose to oversample difficult cases when building the training set (i.e., pairs of signatures with different author names corresponding to same individual, and pairs of signatures with identical author names but corresponding to distinct individuals) in order to improve the overall accuracy of the linkage function.

3.3 Semi-supervised clustering

The last component of our author disambiguation pipeline is clustering, that is the process of grouping together, within a block, all signatures from the same individual (and only those). As for many other works on author disambiguation, we make use of hierarchical clustering [32] for building clusters of signatures in a bottom-up fashion. The method involves iteratively merging together the two most similar clusters until all clusters are merged together at the top of the hierarchy. Similarity between clusters is evaluated using either complete, single or average linkage, using as a pseudo-distance metric the probability that s_1 and

s_2 correspond to distinct authors, as calculated from the custom linkage function d from Section 3.2.

To form flat clusters from the hierarchy, one must decide on a maximum distance threshold above which clusters are considered to correspond to distinct authors. Let us denote by $\mathcal{S}' = \{s | s \in c', c' \in \mathcal{C}'\}$ the set of all signatures for which partial clusters are known. Let us also denote by $\widehat{\mathcal{C}}$ the predicted clusters for all signatures in $\mathcal{S}$, and by $\widehat{\mathcal{C}}' = \{\widehat{c} \cap \mathcal{S}' | \widehat{c} \in \widehat{\mathcal{C}}\}$ the predicted clusters restricted to signatures for which partial clusters are known. From these, we evaluate the following semi-supervised cut-off strategies, as illustrated in Figure 3:

- *No cut:* all signatures from the same block are assumed to be from the same author.
- *Global cut:* the threshold is chosen globally over all blocks, as the one maximizing some score $f(\mathcal{C}', \widehat{\mathcal{C}}')$.
- *Block cut:* the threshold is chosen locally at each block b, as the one maximizing some score $f(\mathcal{C}'_b, \widehat{\mathcal{C}}'_b)$. In case $\mathcal{C}'_b$ is empty, then all signatures from b are clustered together.

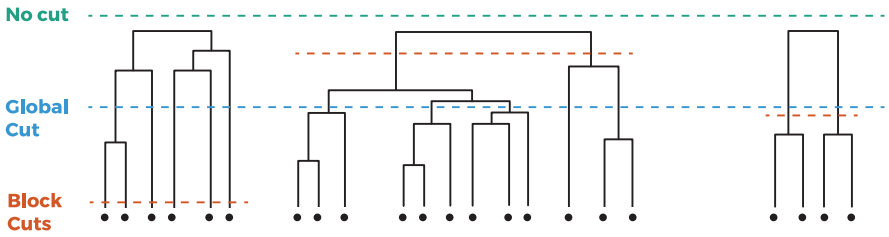


Fig. 3. Semi-supervised cut-off strategies to form flat clusters of signatures.

4 Implementation

As part of this work, we developed a stand-alone application for author disambiguation, publicly available online[5] for free reuse or study. Our implementation builds upon the Python scientific stack, making use of the Scikit-Learn library [21] for the supervised learning of a linkage function and of SciPy for clustering. All components of the disambiguation pipeline have been designed to follow the Scikit-Learn API [2], making them easy to maintain, understand and reuse. Our implementation is made to be efficient, exploiting parallelization when available, and ready for production environments. It is also designed to be runnable in an incremental fashion in which our approach is considered to be scalable. We adopt a divide-and-conquer strategy for clustering (called blocking in this line of

[5] `https://github.com/glouppe/beard`

work) in order to reduce the computational complexity from $O(N^2)$ to $O(\sum N_i^2)$, which in practice tends to $O(N)$ when $N_i << N$. This also means that instead of having to run the disambiguation process on the whole signature set, the process could be run only on specified blocks if desired.

5 Experiments

5.1 Data

The author disambiguation solution proposed in this work, along with its enhancements, are evaluated on data extracted from the *INSPIRE* portal [8], a digital library for scientific literature in high-energy physics. Overall, the portal holds more than 1 million publications $\mathcal{P}$, forming in total a set $\mathcal{S}$ of more than 10 million signatures. Out of these, around 13% have been *claimed* by their original authors, marked as such by professional curators or automatically assigned to their true authors thanks to persistent identifiers provided by publishers or other sources. Together, they constitute a trusted set $(\mathcal{S}', \mathcal{C}')$ of 15388 distinct individuals sharing 36340 unique author names spread within 1201763 signatures on 360066 publications. This data covers several decades in time and dozens of author nationalities worldwide.

Following the *INSPIRE* terms of use, the signatures $\mathcal{S}'$ and their corresponding clusters $\mathcal{C}'$ are released online[6] under the CC0 license. To the best of our knowledge, data of this size and coverage is the first to be publicly released in the scope of author disambiguation research.

5.2 Evaluation protocol

Experiments carried out to study the impact of the proposed algorithmic components and refinements, follow a standard 3-fold cross-validation protocol, using $(\mathcal{S}', \mathcal{C}')$ as ground-truth dataset. To replicate the $|\mathcal{S}'|/|\mathcal{S}| \approx 13\%$ ratio of claimed signatures with respect to the total set of signatures, as on the INSPIRE platform, cross-validation folds are constructed by sampling 13% of claimed signatures to form a training set $\mathcal{S}'_{\text{train}} \subseteq \mathcal{S}'$. The remaining signatures $\mathcal{S}'_{\text{test}} = \mathcal{S}' \setminus \mathcal{S}'_{\text{train}}$ are used for testing. Therefore, $\mathcal{C}'_{\text{train}} = \{c' \cap \mathcal{S}'_{\text{train}} | c' \in \mathcal{C}'\}$ represents the partial known clusters on the training fold, while $\mathcal{C}'_{\text{test}}$ are those used for testing.

As commonly performed in author disambiguation research, we evaluate the predicted clusters over testing data $\mathcal{C}'_{\text{test}}$, using both B3 and pairwise precision,

[6] `https://github.com/glouppe/paper-author-disambiguation`

recall and F-measure, as defined below:

$$P_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{|c(s) \cap \widehat{c}(s)|}{|\widehat{c}(s)|} \tag{1}$$

$$R_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \frac{|c(s) \cap \widehat{c}(s)|}{|c(s)|} \tag{2}$$

$$F_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) = \frac{2P_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})R_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})}{P_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S}) + P_{\mathrm{B3}}(\mathcal{C}, \widehat{\mathcal{C}}, \mathcal{S})} \tag{3}$$

$$P_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{|p(\mathcal{C}) \cap p(\widehat{\mathcal{C}})|}{|p(\widehat{\mathcal{C}})|} \tag{4}$$

$$R_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{|p(\mathcal{C}) \cap p(\widehat{\mathcal{C}})|}{|p(\mathcal{C})|} \tag{5}$$

$$F_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}}) = \frac{2P_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}})R_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}})}{P_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}}) + R_{\mathrm{pairwise}}(\mathcal{C}, \widehat{\mathcal{C}})} \tag{6}$$

and where $c(s)$ (resp. $\widehat{c}(s)$) is the cluster $c \in \mathcal{C}$ such that $s \in c$ (resp. the cluster $\widehat{c} \in \widehat{\mathcal{C}}$ such that $s \in \widehat{c}$), and where $p(\mathcal{C}) = \cup_{c \in \mathcal{C}} \{(s_1, s_2) | s_1, s_2 \in c, s_1 \neq s_2\}$ is the set of all pairs of signatures from the same clusters in $\mathcal{C}$. The F-measure is the harmonic mean between these two quantities. In the analysis below, we rely primarily on the B3 F-measure for discussing results, as the pairwise variant tends to favor large clusters (because the number of pairs is quadratic with the cluster size), hence unfairly giving preference to authors with many publications. By contrast, the B3 F-measure weights clusters linearly with respect to their size. General conclusions drawn below remain however consistent for pairwise F.

5.3 Results and discussion

Baseline. The simplest baseline against which we compare our results consists in grouping all signatures sharing the same (normalized) surname(s) and the same (normalized) first given name initial. As it does not require the usage of any machine learning component, this baseline provides a simple and fast solution yielding decent results, as reported at the top of Table 2

State-of-the-art. Most methods proposed in related works have released neither their software, nor their data, making a fair comparison very difficult. Yet, we believe solutions reported in the literature can be closely matched to our generic pipeline, provided the blocking strategy, the linkage function and the clustering algorithm are properly aligned. In particular, we consider hereon as the *state-of-the-art* solution the following combination of components:

- Blocking: same surname and the same first given name initial strategy (SFI) and with name normalization;
- Linkage function: all 22 features defined in Table 1, gradient boosted regression trees as supervised learning algorithm and a training set of pairs built from $(\mathcal{S}'_{\mathrm{train}}, \mathcal{C}'_{\mathrm{train}})$, by balancing easy and difficult cases.
- Clustering: agglomerative clustering using average linkage and block cuts found to maximize $F_{\mathrm{B3}}(\mathcal{C}'_{\mathrm{train}}, \widehat{\mathcal{C}}'_{\mathrm{train}}, \mathcal{S}'_{\mathrm{train}})$.

Table 2. Average precision, recall and f-measure scores on test folds. Underlined components correspond to the state-of-the-art choices.

	B3			Pairwise		
Description	P	R	F	P	R	F
Baseline	0.9024	0.9828	0.9409	0.8298	0.9776	0.8977
Blocking = SFI	0.9901	0.9760	0.9830	0.9948	0.9738	0.9842
Blocking = Double metaphone	0.9856	0.9827	0.9841	0.9927	0.9817	0.9871
Blocking = NYSIIS	0.9875	0.9826	**0.9850**	0.9936	0.9814	**0.9875**
Blocking = Soundex	0.9886	0.9745	0.9815	0.9935	0.9725	0.9828
No name normalization	0.9887	0.9697	0.9791	0.9931	0.9658	0.9793
Name normalization	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**
Classifier = GBRT	0.9901	0.9760	0.9830	0.9948	0.9738	0.9842
Classifier = Random Forests	0.9909	0.9783	**0.9846**	0.9957	0.9752	**0.9854**
Classifier = Linear Regression	0.9749	0.9584	0.9666	0.9717	0.9569	0.9643
Training pairs = Non-blocked, uniform	0.9793	0.9630	0.9711	0.9756	0.9629	0.9692
Training pairs = Blocked, uniform	0.9854	0.9720	0.9786	0.9850	0.9707	0.9778
Training pairs = Blocked, balanced	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**
Clustering = Average linkage	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**
Clustering = Single linkage	0.9741	0.9603	0.9671	0.9543	0.9626	0.9584
Clustering = Complete linkage	0.9862	0.9709	0.9785	0.9920	0.9688	0.9803
No cut (baseline)	0.9024	0.9828	0.9409	0.8298	0.9776	0.8977
Global cut	0.9892	0.9737	0.9814	0.9940	0.9727	0.9832
Block cut	0.9901	0.9760	**0.9830**	0.9948	0.9738	**0.9842**
Combined best settings	0.9897	0.9827	**0.9862**	0.9954	0.9805	**0.9879**

Below we study each component individually and discuss results with respect to the underlined state-of-the-art solution.

Blocking choices The good precision of the state-of-the-art (0.9901), but its lower recall (0.9760) suggest that the blocking strategy might be the limiting factor to further overall improvements. Our experiments showed the maximum $B3$ recall (i.e., if within a block, all signatures were clustered optimally) for SFI is 0.9828, which corroborates the estimation of this technique on real data by [29]. At the price of fewer and therefore slightly larger blocks, the proposed phonetic-based blocking strategies show better maximum recall (all around 0.9905), thereby pushing further the upper bound on the maximum performance of author disambiguation. Let us remind however that the reported maximum recalls for the blocking strategies using phonetization are also raised due to the better handling of multiple surnames, as described in Section 3.1.

As Table 2 shows, switching to either Double metaphone or NYSIIS phonetic-based blocking allows to improve the overall F-measure score, trading precision for recall. In particular, the NYSIIS-based phonetic blocking shows to be the most effective when applied to the state-of-the-art (with an F-measure of 0.9850) while also being the most efficient computationally (with 10857 blocks versus 12978 for the baseline).

Finally, as discussed previously, the seemingly insignificant step of normalizing author names (stripping accents, removing affixes), as performed in the

state-of-the-art, is shown to be important. Results from Table 2 clearly suggest that not normalizing significantly reduces performance (yielding an F-measure of 0.9830 when normalizing, but decreasing to 0.9791 when raw author name strings are used instead).

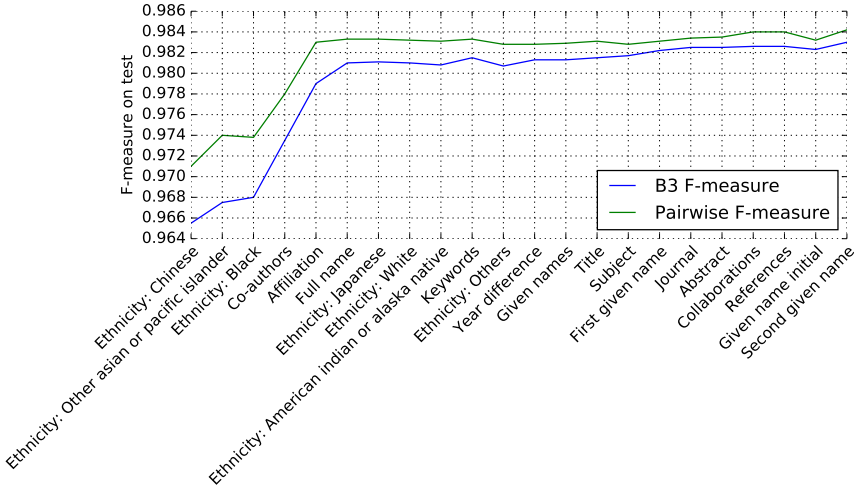
Linkage function choices Let us first comment on the results regarding the supervised algorithm used to learn the linkage function. As Table 2 indicates, both tree-based algorithms appear to be significantly better fit than Linear Regression (0.9830 and 0.9846 for GBRT and Random Forests versus 0.9666 for Linear Regression). This result is consistent with [30] which evaluated the use of Random Forests for author disambiguation, but contradicts results of [15] for which Logistic Regression appeared to be the best classifier. Provided hyper-parameters are properly tuned, the superiority of tree-based methods is in our opinion not surprising. Indeed, given the fact that the optimal linkage function is likely to be non-linear, non-parametric methods are expected to yield better results, as the experiments here confirm.

Second, properly constructing a training set of positive and negative pairs of signatures from which to learn a linkage function yields a significant improvement. A random sampling of positive and negative pairs, without taking blocking into account, significantly impacts the overall performance (0.9711). When pairs are drawn only from blocks, performance increases (0.9786), which confirms our intuition that d should be built only from pairs it will be used to eventually cluster. Finally, making the classification problem more difficult by oversampling complex cases proves to be relevant, by further improving the disambiguation results (0.9830).

Using Recursive Feature Elimination [9], we next evaluate the usefulness of all fifteen standard and seven additional ethnicity features for learning the linkage function. The analysis consists in using the state-of-the-art algorithm first using all twenty two features, to determine the least discriminative from feature importances [17], and then re-learn the state-of-the-art algorithm using all but that one feature. That process is repeated recursively until eventually only one feature remains. Results are presented in Figure 4 for one of the three folds with the state-of-the-art, starting from the far right, *Second given name* being the least important feature, and ending on the left with all features eliminated but *Chinese*. As the figure illustrates, the most important features are ethnic-based features (*Chinese*, *Other Asian*, *Black*) along with *Co-authors*, *Affiliation* and *Full name*. Adding the remaining other features only brings marginal improvements, with *Journal*, *Abstract*, *Collaborations*, *References*, *Given name initial* and *Second given name* being almost insignificant. Overall, these results highlight the added value of the proposed ethnicity features. Their duality in modeling both the similarity between author names and their origins make them very strong predictors for author disambiguation. The results also corroborate those from [12] or [6], who found that the similarity between co-authors was a highly discriminative feature. If computational complexity is a concern, this analysis also shows how decent performance can be achieved using only a very small set of features, as also observed in [30] or [15].

Semi-supervised clustering choices The last part of our experiment concerns the study of agglomerative clustering and the best way to find a cut-off

Fig. 4. Recursive Feature Elimination analysis.



threshold to form clusters. Results from Table 2 first clearly indicate that average linkage is significantly better than both single and complete linkage.

Clustering together all signatures from the same block (i.e., baseline) is the least effective strategy (0.9409), but yields anyhow surprisingly decent accuracy, given the fact it requires almost no computation (i.e., both learning a linkage function and running agglomerative clustering can be skipped – only the blocking function is needed to group signatures). In particular, this result reveals that author names are not ambiguous in most cases[7] and that only a small fraction of them requires advanced disambiguation procedures. On the other hand, both global and block cut thresholding strategies give very good results, with a slight advantage for block cuts (0.9814 versus 0.9830), as expected. In case $\mathcal{S}'_b$ is empty (e.g., because it corresponds to a young researcher at the beginning of his career), this therefore suggests that either using a cut-off threshold learned globally from the known data or using SFI would in general give satisfactory results, only marginally worse than if claimed signatures had been known. Trying out 3 clustering methods, the results show that *average linkage* was the best one.

Combined best settings When all best settings are combined (i.e., Blocking = NYSIIS, Name normalization, Classifier = Random Forests, Training pairs = blocked and balanced, Clustering = Average linkage, Block cuts), performance reaches 0.9862, i.e., the best of all reported results. In particular, this combination exhibits both the high recall of phonetic blocking based on the NYSIIS algorithm and the high precision of Random Forests.

[7] This holds for the data we extracted, but may in the future, with the rise of non-Western researchers, be an underestimate of the ambiguous cases.

Execution time Our implementation takes around 20 hours to process the complete set of the data (for 10M signatures, on a 16 cores machine with 32GB of RAM). Related work [13] reports execution times around 24 hours to cluster 4M signatures. Note also that shorter execution times can be achieved, at the expense of worse results, by reducing the set of the features used.

6 Conclusions

In this work, we have revisited and validated the general author disambiguation pipeline introduced in previous independent research work. The generic approach is composed of three components, whose design and tuning are all critical to good performance: (i) a blocking function for pre-clustering signatures and reducing computational complexity, (ii) a linkage function for identifying signatures with coreferring authors and (iii) the agglomerative clustering of signatures. Making use of a distinctively large dataset of more than 1 million crowdsourced annotations, we experimentally study all three components and propose further improvements. With regards to blocking, we suggest to use phonetization of author names to increase recall while maintaining low computational complexity. For the linkage function, we introduce ethnicity-sensitive features for the automatic tailoring of disambiguation to non-Western author names whenever necessary. Finally, we explore semi-supervised cut-off threshold strategies for agglomerative clustering. For all three components, experiments show that our refinements all yield significantly better author disambiguation accuracy.

Overall, these results all encourage further improvements and research. For blocking, one of the open challenges is to manage signatures with inconsistent surnames or inconsistent first given names (cases 4 and 5, as described in Section 3.1) while maintaining blocks to a tractable size. As phonetic algorithms are not yet perfect, another direction for further work is the design of better phonetization functions, tailored for author disambiguation. For the linkage function, the good results of the proposed features pave the way for further research in ethnicity-sensitive author disambiguation. The automatic fitting of the pipeline to cultures and ethnic groups for which standard author disambiguation is known to be less efficient (e.g., Chinese authors with many homonyms) indeed constitutes a direction of research with great potential benefits for the concerned scientific communities.

As part of this study, we also publicly release the annotated data extracted from the *INSPIRE* platform, on which our experiments are based. To the best of our knowledge, data of this size and coverage is the first to be available in author disambiguation research. By releasing the data publicly, we hope to provide the basis for further research on author disambiguation and related topics.

References

1. L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
2. L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux. API design for machine learning software: experiences from the scikit-learn project. *CoRR*, abs/1309.0238, 2013.

3. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008.
4. I. P. Fellegi and A. B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64:1183–1210, 1969.
5. A. A. Ferreira, M. A. Gonçalves, and A. H. Laender. A brief survey of automatic methods for author name disambiguation. *Acm Sigmod Record*, 41(2):15–26, 2012.
6. A. A. Ferreira, A. Veloso, M. A. Gonçalves, and A. H. Laender. Effective self-training author name disambiguation in scholarly digital libraries. In *Proceedings of the 10th annual joint conference on Digital libraries*, pages 39–48. ACM, 2010.
7. J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.
8. A. Gentil-Beccot, S. Mele, A. Holtkamp, H. B. O'Connell, and T. C. Brooks. Information resources in high-energy physics: Surveying the present landscape and charting the future course. *Journal of the American Society for Information Science and Technology*, 60(1):150–160, 2009.
9. I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine learning*, 46(1-3):389–422, 2002.
10. H. Han, L. Giles, H. Zha, C. Li, and K. Tsioutsiouliklis. Two supervised learning approaches for name disambiguation in author citations. In *Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on*, pages 296–305. IEEE, 2004.
11. J. Huang, S. Ertekin, and C. L. Giles. Efficient name disambiguation for large-scale databases. In *Knowledge Discovery in Databases: PKDD 2006*, pages 536–544. Springer, 2006.
12. I.-S. Kang, S.-H. Na, S. Lee, H. Jung, P. Kim, W.-K. Sung, and J.-H. Lee. On co-authorship for author disambiguation. *Information Processing & Management*, 45(1):84–97, 2009.
13. M. Khabsa, P. Treeratpituk, and C. L. Giles. Large scale author name disambiguation in digital libraries. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 41–42. IEEE, 2014.
14. D. Lange and F. Naumann. Frequency-aware similarity measures: why arnold schwarzenegger is always a duplicate. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 243–248. ACM, 2011.
15. M. Levin, S. Krawczyk, S. Bethard, and D. Jurafsky. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology*, 63(5):1030–1047, 2012.
16. W. Liu, R. Islamaj Doğan, S. Kim, D. C. Comeau, W. Kim, L. Yeganova, Z. Lu, and W. J. Wilbur. Author name disambiguation for pubmed. *Journal of the Association for Information Science and Technology*, 65(4):765–781, 2014.
17. G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, pages 431–439, 2013.
18. B. Malin. Unsupervised name disambiguation via social network similarity. In *Workshop on link analysis, counterterrorism, and security*, volume 1401, pages 93–102, 2005.
19. D. M. McRae-Spencer and N. R. Shadbolt. Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, pages 53–54. ACM, 2006.
20. M. E. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences*, 98(2):404–409, 2001.

21. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
22. L. Philips. The double metaphone search algorithm. *C/C++ Users J.*, 18(6):38–43, June 2000.
23. S. Ruggles, M. Sobek, C. A. Fitch, P. K. Hall, and C. Ronnander. *Integrated public use microdata series*. Historical Census Projects, Department of History, University of Minnesota, 2008.
24. N. R. Smalheiser and V. I. Torvik. Author name disambiguation. *Annual review of information science and technology*, 43(1):1–43, 2009.
25. Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 342–351. ACM, 2007.
26. A. Strotmann and D. Zhao. Author name disambiguation: What difference does it make in author-based citation analysis? *Journal of the American Society for Information Science and Technology*, 63(9):1820–1833, 2012.
27. R. L. Taft. Name search techniques. Technical Report Special Report No. 1, New York State Identification and Intelligence System, Albany, NY, February 1970.
28. The National Archives. The soundex indexing system, May 2007.
29. V. I. Torvik and N. R. Smalheiser. Author name disambiguation in medline. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(3):11, 2009.
30. 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. ACM, 2009.
31. P. Treeratpituk and C. L. Giles. Name-ethnicity classification and ethnicity-sensitive name matching. In *AAAI*. Citeseer, 2012.
32. J. H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.