Citation-Based Bootstrapping for Large-Scale Author Disambiguation

Michael Levin and Stefan Krawczyk
Computer Science Department, Stanford University, 353 Serra Mall, Stanford, CA 94305-9025. E-mail: melevin@google.com; stefank@cs.stanford.edu

Steven Bethard
Center for Computational Language and Education Research, University of Colorado Boulder, Boulder, CO 80309-0594. E-mail: steven.bethard@colorado.edu

Dan Jurafsky
Linguistics Department, Stanford University, 450 Serra Mall, Stanford, CA 94305. E-mail: jurafsky@stanford.edu

We present a new, two-stage, self-supervised algorithm for author disambiguation in large bibliographic databases. In the first "bootstrap" stage, a collection of high-precision features is used to bootstrap a training set with positive and negative examples of coreferring authors. A supervised feature-based classifier is then trained on the bootstrap clusters and used to cluster the authors in a larger unlabeled dataset. Our self-supervised approach shares the advantages of unsupervised approaches (no need for expensive hand labels) as well as supervised approaches (a rich set of features that can be discriminatively trained). The algorithm disambiguates 54,000,000 author instances in Thomson Reuters' Web of Knowledge with B^3 F1 of .807. We analyze parameters and features, particularly those from citation networks, which have not been deeply investigated in author disambiguation. The most important citation feature is *self-citation*, which can be approximated without expensive extraction of the full network. For the supervised stage, the minor improvement due to other citation features (increasing F1 from .748 to .767) suggests they may not be worth the trouble of extracting from databases that don't already have them. A lean feature set without expensive abstract and title features performs 130 times faster with about equal F1.

Received July 27, 2011; revised October 25, 2011 and November 28, 2011; accepted November 29, 2011

© 2012 ASIS&T • Published online 14 February 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/asi.22621

Introduction

Large bibliographic databases often fail to distinguish authors with similar names. Consider "J. Lee," attributed with over 56,000 articles in Thomson Reuters' Web of Knowledge[1], or "Kim, J.H.," with over 11,000. Clearly there is not just a single J. Lee or J.H. Kim who published all of these articles. But although distinguishing the different authors with similar names is important for any research that makes use of networks of scientific publications, the task is quite difficult. Publication databases are huge, requiring methods that can scale up to millions of articles, and methods must be capable of handling partial and conflicting metadata. Consider the example in Table 1, where "Zhao, Kun" has changed his e-mail address twice in 3 years, his first name is missing from the metadata of two of the three records, and the terminology in his articles diverges quite a bit. An author disambiguation system must nonetheless infer that these are the same person, presumably by relying on other features such as coauthorship or citation patterns.

There have been three classes of algorithms for author disambiguation.[2] Methods relying on supervised machine learning train classifiers (support vector machines, random forests, etc.) on a hand-labeled training set containing pairs of articles where similarly named authors are identified as

[1]http://www.webofknowledge.com/

[2]See Smalheiser and Torvik (2009) for a comprehensive summary of this literature.

TABLE 1. Three papers by "Zhao, Kun."

Title	Laser-induced thermoelectric voltage in normal state MgB_2 thin films
Author	Zhao, Kun
E-mail	ainiphoto@163.com
Coauthors	Lu, HB; Cheng, BL; Zhao, SQ; Yang, GZ; Wang, SF; Chen, ZH; Jin, KJ; Zhou, YL
Abstract	Laser-induced voltage has been observed in c-axis oriented MgB2 thin film at room. . .
Year	2006
Title	Characteristics of heterojunctions of amorphous $LaAlO_{2.73}$ on Si
Author	Zhao, K
E-mail	kzhao@aphy.iphy.ac.cn
Coauthors	Lu, HB; He, M; Yang, GZ; Chen, ZH; Huang, Y; Jin, KJ; Zhou, YL
Abstract	High-quality heterojunctions consisting of n-type amorphous $LaAlO_{3-\delta}$ and p-type Si. . .
Year	2005
Title	Ultraviolet photoresponse properties of SrTiO3 single crystals
Author	Zhao, K
E-mail	zhk@cup.edu.cn
Coauthors	Lu, HB; He, M; Huang, Y; Yang, GZ; Chen, ZH; Jin, KJ; Zhou, YL
Abstract	We have observed ultraviolet photosensitive properties of SrTiO3 single crystals. . .
Year	2007

Note. Our algorithm correctly clusters all of these together.

being the same or different persons (Han, Giles, Zha, Li, & Tsioutsiouliklis, 2004; Huang, Ertekin, & Giles, 2006; Kanani, McCallum, & Pal, 2007; Chen, Kalashnikov, & Mehrotra, 2007; Culotta, Kanani, Hall, Wick, & McCallum, 2007; Yang, Peng, Jiang, Lee, & Ho, 2008; Treeratpituk & Giles, 2009). Supervised algorithms require large hand-labeled training sets, especially for huge databases like MEDLINE or Web of Knowledge that contain tens of millions of articles. Such large quantities of manually annotated training data are not easily available, would be hard to make representative, and would be expensive to collect. For this reason, supervised algorithms may not be the best solution for such databases.

Unsupervised algorithms, by contrast, require no hand-labeled training data, instead defining a metric of similarity between pairs of articles and applying an unsupervised clustering algorithm such as k-means or spectral clustering (Malin, 2005; Han, Xu, Zha, & Giles, 2005; Han, Zha, & Giles, 2005; Bhattacharya & Getoor, 2006; Song, Huang, Councill, Li, & Giles, 2007; Kang et al., 2009; Fan et al., 2011; Tang, Fong, Wang, & Zhang, 2011). The similarity functions used by these clustering algorithms typically compare vectors of roughly homogeneous features, e.g., the words in the title, journal or author fields. Each resulting cluster of similar articles is assumed to correspond to a single real-world author. Unsupervised algorithms have the advantage of not needing training data, but typically explore simpler feature spaces, and often do not perform as well as supervised approaches.

In natural language processing, the common solution to these problems is a third-class: semisupervised, weakly supervised, or self-trained algorithm (Blum & Mitchell, 1998; Banko & Brill, 2001; Ng & Cardie, 2003; McClosky, Charniak, & Johnson, 2006). In these models, we are given a large unlabeled dataset and either a small hand-labeled training set (semisupervised) or a small set of high-precision patterns or features that can be used to find training examples in the unlabeled data (weakly supervised). From this labeled training set we then train a supervised classifier. Once an initial model has been trained, it can be used to tag unlabeled data. These methods have been quite successful in relation extraction (Hearst, 1998; Riloff & Jones, 1999; Chklovski & Pantel, 2004; Etzioni et al., 2005) and have advantages over unsupervised approaches: Their supervised component can learn discriminative weights for complex feature sets, and they often have better performance than fully unsupervised methods.

Some semisupervised and weakly supervised approaches have been investigated for author disambiguation, with high-precision features or additional training data being used to bootstrap initial author clusters. For example, both Bhattacharya and Getoor (2007) and Ferreira, Veloso, Goncalves, and Laender (2010) use *coauthorship* features for bootstrapping. For each author name, they cluster together articles that have that author name and one or more additional coauthor names in common. Torvik and Smalheiser (2009) use e-mail addresses and some manually curated training examples to bootstrap their clusters. All articles sharing a common e-mail address are put into the same cluster, using e-mail addresses from the article metadata and from a heuristic crawl of publisher websites. These e-mail-based clusters are then augmented with manually curated clusters from the Thomson Reuters' Highly Cited researchers database.[3]

Although these previous studies have shown that high-precision features may be useful for bootstrapping, a number of open issues remain. It is not clear how effective feature-based bootstrapping is on large-scale collections—only Torvik and Smalheiser's (2009) study was large scale, and they combined feature-based bootstrapping with additional hand-labeled training data, so it is not possible to see the effect of feature-based bootstrapping alone. It is also unknown whether existing findings about feature-based bootstrapping will hold with very large datasets—for example, contrasting with the work of Bhattacharya and Getoor (2007) and Ferreira et al. (2010) on smaller datasets, our results will show that coauthorship is not effective for bootstrapping author disambiguation on Thomson Reuters' Web of Knowledge. Moreover, we do not know much at all about the effectiveness of features other than e-mail or coauthorship for bootstrapping. Finally, because none of the previous studies used standard supervised classifiers, instead introducing alternative models at the same time as feature-based bootstrapping, we cannot tell whether performance improved because of the models or because of the bootstrapping.

[3]http://www.highlycited.com/

DOI: 10.1002/asi

Thus, we propose to investigate author disambiguation using the standard bootstrapping approach from natural language processing, where initial clusters are formed using high-precision features, and standard supervised classifiers are trained using these bootstrapped clusters. This approach has a couple of advantages. Because there is a clean separation between the unsupervised stage and the supervised stage, we can better explore which features help or hurt in each of these stages. And because the algorithm is independent of the exact features used, we can explore a wide variety of features that may be important to the task.

We explore three classes of features. These include the author and subject features widely explored in authorship identification, such as title words, journal names, author names, initials, addresses, and affiliations, subject categories, languages, and year of publication (Han et al., 2004; Han, Xu, et al., 2005; Han, Zha, et al., 2005; Huang et al., 2006; Bhattacharya & Getoor 2006; Chen, Kalashnikov, & Mehrotra, 2007; Culotta et al., 2007; Song et al., 2007; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009; Ferreira et al., 2010). The second class is a new source of information not used by most previous systems: features based on citations between articles. Important pilot studies on small name-sets have suggested that citation information might be useful. In a study on eight names, McRae-Spencer and Shadbolt (2006) show that self-citation features can provide useful information for name disambiguation. In a study on 32 names, Tang et al. (2011) show that a citation feature improves the performance of an unsupervised disambiguation algorithm. Citations have also proven quite useful for related tasks like scholarly author identification (the task of figuring out who wrote an anonymous scholarly paper); Hill and Provost (2003) found that assuming the author most often cited by a paper is the author of that paper gets the correct answer 37% of the time. Finally, we explore combinations of features to see if feature interactions prove important.

In summary, we propose to apply the self-training model standardly used in natural language processing tasks like relation extraction to author disambiguation, and to investigate the citation network as a source of features for both the bootstrapping and supervised stages. We will also focus on optimizations required to scale up to the over 54,000,000 name instances in Thomson Reuters' Web of Knowledge. Our algorithm is introduced in the next section; in the following sections, we introduce the corpus, the experiments, and the results.

The Architecture of Our Self-trained Disambiguation Algorithm

As is standard in all previous work, our algorithm begins by breaking the author names into "blocks" (Fellegi & Sunter, 1969), groups of names with the same last name and first initial (e.g., "Wang, W"), as well as middle initial when applicable. Pairwise comparisons thus need only to be carried out between all pairs of articles in a block, instead of all pairs of articles in the collection, which would be intractable for even moderate sized collections.[4]

For each block, we apply our two-stage process. In stage one, the bootstrapping stage, we apply high-precision rules to find positive examples—pairs of articles that are highly likely to be written by the same author—in the unsupervised dataset. For the negative examples, we select pairs of articles that were not linked by the high-precision rules.

In stage two, the supervised training stage, we use the positive and negative examples as training data for a supervised classifier that predicts whether or not two articles have been written by the same author. The classifier predictions are then used as the similarity metric in a statistical clusterer that groups together articles written by the same author. The following sections describe each of these stages.

Stage 1: Bootstrapping Via Rule-based Clustering

The bootstrapping approach is to apply a small set of high-precision rules to a large unlabeled collection to identify pairs of articles likely written by the same author. In this section, we investigate the two features that have been proposed before for bootstrapping (coauthorship and e-mail addresses), along with some new features such as self-citation and subject area, and evaluate how well different combinations of these rules perform at bootstrapping. We also explore a new dimension of feature based rules, the use of *negative rules* that can disallow the application of positive rules in certain circumstances. The set of bootstrapping rules is described in Table 2.

All these rules are based on observations in the literature about how scientific articles are written. For example, we expect the self-citation rule to work because people are likely to cite their own work, and because even though it is common for several people to share the same name, it is unlikely that many people with the same name work in the same narrow research areas and end up citing each other. Likewise, we expect the exact coauthors rule to work because it's much more likely that three coauthors continue to work with the same fourth author than that they switch to a new fourth author with the same name. The utility of each of these rules in evaluated in the Choosing Rules for Stage 1 (Rule-Based Clustering) section.

Stage 2: Supervised Classification and Clustering

The output of the rule-based bootstrapping is a set of high-precision clusters where articles are grouped together if they are very likely to have been written by the same author. However, there may still be many such clusters for each real-world author because the rules above can only link papers together based on a small number of features. To merge the high-precision rule-based clusters into larger

[4]Blocking by last name and first initial is not perfect—authors may change names or names may be misspelled—but previous literature estimated that recall loss is less than 2% (Torvik & Smalheiser, 2009).

DOI: 10.1002/asi

TABLE 2. Positive and negative rules used in the bootstrapping stage.

Positive rules	
E-mail	E-mails match exactly.
Self-Citation	One article directly cites another within the same name block.
Exact Coauthors	Both articles have at least three authors and their last names and first/middle initials match exactly.
Coauthor_p	At least one coauthor appears on both articles.
Middle Initial (MI) w/Subject	Author shares same middle initial and subject.
Exact citation	Authors cite the same paper, i.e. bibliographic coupling, but for authors instead of articles.
Exact venue	Authors share the same publication venue.
Negative rules	
Name	Articles x and y may not be merged unless x's (first, middle initial, last) is equal to y's (first, middle, last) or x's is a subset of y's or y's is a subset of x's. For example, (William, J., Flinstone), would match (W., J., Flinstone), (William, ,Flinstone) and (W., , Flinstone).
Language	Articles in two different non-English languages may not be merged.
Coauthor_n	Articles (with multiple authors) may not be merged unless they have at least one coauthor in common.

$$\cos\begin{pmatrix}\text{“Name disam-}\\ \text{biguation in}\\ \text{author}\\ \text{citations”}\end{pmatrix}\begin{pmatrix}\text{“Author name}\\ \text{disambigua-}\\ \text{tion in}\\ \text{MEDLINE”}\end{pmatrix} = \cos\begin{pmatrix}\textit{author}:1.9 & \textit{author}:1.9\\ \textit{citations}:2.3 & \\ \textit{disambiguation}:3.7 & \textit{disambiguation}:3.7\\ \textit{in}:0.0001' & \textit{in}:0.0001\\ & \textit{medline}:4.2\\ \textit{name}:1.1 & \textit{name}:1.1\end{pmatrix} = 0.777$$

FIG. 1. Example of calculating the title feature: titles are broken into words, tf–idf is calculated for each word, and a cosine is taken over the tf–idf word vectors.

clusters that better represent real-world authors, we first train a classifier with a large number of features on the rule-based clusters, and then use this classifier as a similarity metric in an agglomerative clustering algorithm.

In the next subsection, we describe features used in training the classifier; the following sections describe the classification algorithm and how the classifier was used during the second phase clustering.

Features from previous research. Features drawn from the previous literature are shown in the first part of Table 3. Each feature is a comparison between two authors on one field value. Fields are drawn either from the article metadata or from the author metadata. Fields include author full names, first names, middle initials, and suffixes (e.g., jr.) (Huang et al., 2006; Bhattacharya & Getoor, 2006; Chen, Kalashnikov, & Mehrotra, 2007; Culotta et al., 2007; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009; Ferreira et al., 2010) and coauthorship information (Han et al., 2004; Han, Xu, et al., 2005; Han, Zha, et al., 2005; Bhattacharya & Getoor, 2006; Culotta et al., 2007; Kang et al., 2009; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009; Ferreira et al., 2010; Fan et al., 2011). We also compare authors by their addresses, affiliations, and e-mail addresses (Huang et al., 2006; Chen, Kalashnikov, & Mehrotra, 2007; Culotta et al., 2007; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009), the article title and the title of the journal it appeared in (Han et al., 2004; Han, Xu, et al., 2005; Han, Zha, et al., 2005; Chen, Kalashnikov, & Mehrotra, 2007; Culotta et al., 2007; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009; Ferreira et al., 2010), and article abstracts, keywords, subject categories, language, and year (Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009).

Comparisons are performed using the standard term frequency–inverse document frequency (tf–idf) bag-of-words scheme (Bilenko, Mooney, Cohen, Ravikumar, & Fienberg, 2005; Han, Zha, et al., 2005; Culotta et al., 2007; Treeratpituk & Giles, 2009); Figure 1 gives an example. For each metadata field, we break it into terms and form a vector of term scores. The vector has one dimension for each term in the vocabulary (the set of unique terms observed for that field across all the articles in the collection). The score for each term is calculated as the standard tf–idf score, that is, the frequency of the term in the metadata field times the term's inverse document frequency (where each instance of a field is considered to be a document).

As discussed later, we use Lucene[5] for indexing, so we use the term vectors and document frequencies exactly as they were calculated by Lucene. To get the final feature score, we use a cosine (as in Bilenko et al,. 2005 and Culotta et al., 2007) to compare the term vector for the field in one article with the term vector for the field in the other article.[6]

For most fields, we use the standard whitespace-based definition of tokens, but for three fields with special content, we define tokens differently. For middle initials, we treat each single character initial as a token. For e-mails, we treat

[5]http://lucene.apache.org

[6]We also explored other vector similarity measures; see the Other Parameters section.

DOI: 10.1002/asi

TABLE 3. The 1,080 Features (18 previous, 3 citation metadata, 24 citing and cited, and 1,035 conjunctive).

The 18 previous features	
Article metadata features	*Author metadata features*
Article Title	Author First Name
Article Abstract	Author Middle Initials
Article Keywords	Author Name Suffixes
Article Language	Author E-mail Address
Article Last Names with Initials	Author Organization
Article Addresses	Author Address
Article Corporate Affiliations	Coauthor Last Names with Initials
Article Reprint Organization	
Publication Title	
Publication Subjects	
Publication Year	
The 3 citation metadata features	
Cited Article IDs	Citing Article IDs
Cited Journal Titles	
The 24 citing and cited features	
Cited Article Title	Citing Article Title
Cited Article Abstract	Citing Article Abstract
Cited Article Keywords	Citing Article Keywords
Cited Article Language	Citing Article Language
Cited Article Last Names with Initials	Citing Article Last Names with Initials
Cited Article Addresses	Citing Article Addresses
Cited Article Corporate Affiliations	Citing Article Corporate Affiliations
Cited Article Reprint Organization	Citing Article Reprint Organization
Cited Publication Title	Citing Publication Title
Cited Publication Subjects	Citing Publication Subjects
Cited Cited Article IDs	Citing Cited Article IDs
Cited Cited Journal Titles	Citing Cited Journal Titles
The 1,035 conjunctive features	
Article Title × Author Title	Article Title × Article Abstract
Article Title × Article Keywords	Article Title × Article Language
. . . × . . .	. . . × . . .
Author E-mail Address × Cited Article Keywords	Author E-mail Address × Citing Article Keywords
Author E-mail Address × Cited Author Language	Author E-mail Address × Citing Author Language
. . . × . . .	. . . × . . .
Cited Journal Titles × Citing Publication Subjects	Cited Journal Titles × Cited Article IDs
Cited Journal Titles × Citing Article IDs	Cited Journal Titles × Cited Journal Titles

both the chunk before and after the "@" as tokens. For the publication year, we generate one token for the given year and then additional tokens for the surrounding years. Each publication year token is assigned a score using a Gaussian distribution,[7] allowing a year like 2006 to match the years 2005 and 2007 with high probability, but to match a year like 1982 with very low probability.

[7]The $\sigma = 3.408$ was empirically determined by computing the mean and variance of the year distribution of papers for an author in the training set.

Citation features. Citation features were drawn from the Web of Knowledge database, which provides for each article the list of articles that it cites in the form of unique article identifiers. Because only articles within the Web of Knowledge database are assigned identifiers, cited articles not part of Web of Knowledge will not be included in the citation network. In the full Web of Knowledge citation network, each article cites on average 22.1 articles, and 28.2% of the articles are never cited.

The first three citation features in Table 3 are extracted from the Web of Knowledge metadata. The *Cited Article IDs* feature takes the list of unique article identifiers, which indicate the articles that were cited by an article, and treats this as a text that is tokenized and used in a term-vector cosine as usual. This feature asks whether the two articles being compared cite the same articles (i.e., there is *bibliographic coupling*), which should be useful if authors tend to cite the same articles repeatedly. The *Citing Article IDs* feature works in the opposite direction, asking whether the two articles being compared were cited by the same articles (i.e., there was *co-citation;* Small, 1973), which should be useful if authors tend to be cited by the same other author, as suggested by models like author cocitation analysis (ACA; White & Griffith, 1981). The citing article IDs necessary for this feature are not stored directly as metadata, but we add them as we build the index. The last feature, *Cited Journal Titles*, could have been derived by following the links in the network, but was in fact an additional piece of metadata provided directly by the Web of Knowledge: a short name for the journal in which each cited article was found.

The next citation features were generated by traversing the citation network. We include two extensions of each of the features derived from the article metadata, shown in the third section of Table 3. In the first extension, which we call the *Cited* version of the feature, instead of collecting the feature's text from the article itself, we collect the feature's text from the articles it cites. So, for example, in Figure 2, the *Cited Article Title* feature for article 4 would concatenate the *Article Title* features from articles 5, 6, and 7, resulting in

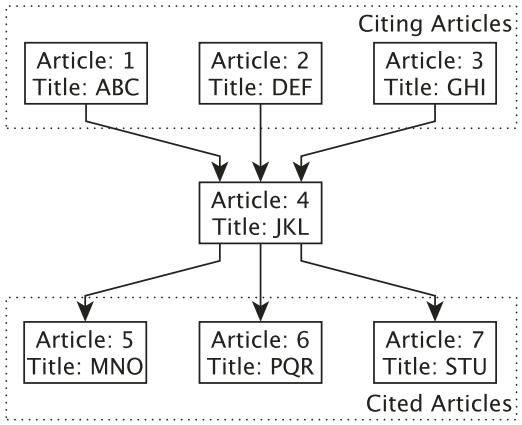

FIG. 2. An example article citation graph.

DOI: 10.1002/asi

the text *MNO PQR STU*. This text would then be tokenized and used in a term-vector cosine as usual. In the second extension, which we call the *Citing* version of the feature, the text is now collected from all of the articles which cite the article. So, in Figure 2, the text used by the *Citing Article Title* feature for article 4 would be *ABC DEF GHI*. We generate a *Cited* and *Citing* version of each of the features derived from the article metadata.[8]

Conjunctive features. In order to introduce higher-precision versions of existing features, we added product conjunctions (feature interactions) to our feature set. Given two features, say "first name" and "subject code," their product conjunction requires that both first names and subject codes must at least partially match. This produces a higher precision feature by lowering feature response in cases where only one of the two features is a strong match.

Conjunctive features were computed by taking the product of two features. More formally $f_{ab} = f_a f_b$, where feature f_a is the cosine value of feature a, and f_b is the cosine value of feature b. This implicitly gives a squared feature term if $a = b$. We produce 1,035 conjunctive features through products of all the previous work and citation features. At the bottom of Table 3 is a schematic description of this large set of conjunctive features.

Classifier-based similarity. All of the features (prior work, citation, and conjunctive) serve as clues to a supervised classifier that takes as input two articles and produces the probability that those two articles share a real-world author. To train the classifier, we need positive examples (both articles are known to be written by the same real-world author) as well as negative examples (both articles are known to be written by different real-world authors).

For the positive examples, we take all rule-based clusters that had at least two articles and sample random pairs of articles that were in the same rule-based cluster. For the negative examples, we take all the rule-based clusters that had at least two articles and sample random pairs of articles that were in different rule-based clusters.[9]

Because the rule-based clusters are high-precision—see the Choosing Rules for Stage 1 (Rule-Based Clustering) section—we can be reasonably confident that the articles in the positive examples were written by the same authors. However, because the rule-based clusters have lower recall, it is possible that negative examples may be drawing pairs of articles that are from different rule-based clusters but were actually written by the same author. Nonetheless, we found this sampling across two-or-more element clusters to be a more effective sampling strategy for collecting negative examples than sampling either random pairs of articles or random across one-or-more element clusters. Intuitively, when a cluster created by the rule clustering phase has two or more articles, this is because the rules actually fired for these articles, and the articles in this cluster share some trait that is distinct from the other clusters (e.g., in the "Smith, J." block, one cluster for "john.smith@domain1.com" and one for "jay.smith@domain2.com"). When a cluster created by the rule clustering phase has only one article, this is often because there was missing metadata, and not necessarily because this author is distinct from all of the others.

Because of hardware constraints, we were forced to limit the number of pairs sampled based on the amount of features used, sampling down to 200,000 pairs when using all features, or 10,000,000 when using fewer than 100 features. To maintain the same positive-negative ratio, we randomly discard positive and negative examples in proportion to their initial ratio.

To train a model using these positive and negative examples, we use a binary L1-regularized logistic classifier, trained using the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) algorithm (Andrew & Gao, 2007). We adopt L1-regularization because it does implicit feature selection, finding a solution where many features have zero weights (Tibshirani, 1996). The following equation shows this optimization problem, where Φ is a function that takes a training example (x_i, y_i) and produces a feature vector, and where w is the weight vector with each weight corresponding to one feature:

$$f(w) = -\sum_{i=1}^{M} \log \frac{e^{w^{\mathrm{T}}\Phi(x_i, y_i)}}{\sum_{y'} e^{w^{\mathrm{T}}\Phi(x_i, y')}} + C\sum_{j=1}^{N} |w_j|$$

The left term is the negative conditional log-likelihood of the training data, while the right term is the L1-norm of the weight vector. Thus, we optimize the likelihood of the training data, at the same time trying to minimize the absolute value of all the individual weights.

The resulting classifier, given a pair of articles, produces a probability that the articles were written by the same author.

Agglomerative clustering. The goal of our second pass of clustering is to take the initial clusters produced by the rule-based clusterer, and for each author merge together the clusters containing their papers. We utilize the standard agglomerative clustering algorithm (King, 1967), used in many unsupervised approaches to author disambiguation (Malin, 2005; Han, Xu, et al., 2005; Han, Zha, et al., 2005; Bhattacharya & Getoor, 2006; Song et al., 2007; Kang et al., 2009; Fan et al., 2011; Tang et al., 2011), which builds bigger clusters by iteratively merging the two most similar smaller clusters together.

[8]We excluded publication year from the cited/citing features because of the special handling of its term vector described above.

[9]The number of negative pairs is in general always much larger than the number of positive pairs. Because supervised classifiers with strong training skew sometimes classify everything into one class, we examined down-sampling the training data. We found, however, that all ratios of positive to negatives (e.g., 1 : 9, 1 : 90, etc.) degraded performance markedly. The best result was when we used all of the possible training data. This meant that on the large blocks, we had a positive to negative ratio of 1:1000 and greater at times.

DOI: 10.1002/asi

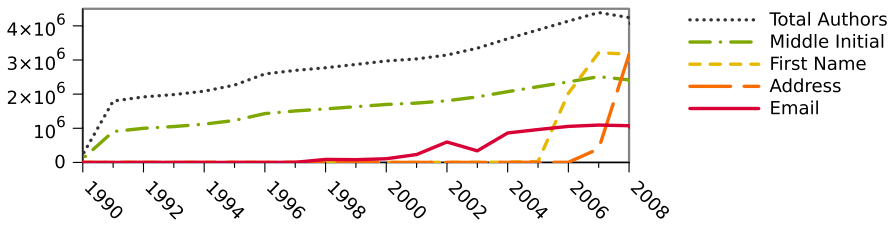

FIG. 3. The counts of select author fields from 1990 to 2008. Many of the higher precision features are based on fields only recently available. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The similarity between a pair of article clusters is computed with the logistic classifier trained above. To convert the pair of clusters into a pair of articles that the classifier can accept as input, we compute the centroid of each cluster by averaging the term vectors for each feature across all articles in the cluster. The classifier's cosine-based features are then calculated over the averaged term vectors and the classifier predicts the similarity between the clusters based on the cosine features. The two clusters identified as being most similar by the classifier (and that do not violate the negative name rule) are then merged.

The clustering process is stopped when the most similar cluster pair score drops below a threshold. Adjusting this threshold will trade precision and recall, depending on the type of errors one would prefer to avoid. We tuned our thresholds to optimize F1 score on the development set, though we did observe that higher precision could often be achieved without much loss in F1 with a raised threshold. (See Other Parameters section for details.)

Special Processing for Small Blocks

Each block was processed separately; thus a separate block-specific classifier was learned for each block. For small blocks however, there were often insufficient negative training examples. We therefore trained a separate small-block-classifier that was used for all blocks of less than 100 articles. The classifier was trained on the bootstrapped data from a collection of 100 blocks of 100 articles each.

All blocks with at least two articles were then divided into small (less than 100 articles) and large sets. The small set was clustered using rules and then the small-block-classifier. The large set was clustered with block-specific classifiers produced using the bootstrapped method described above.

Evaluation Corpus

In this section we give details of the corpus we use for our experiments, Thomson Reuters Web of Knowledge, as well as the test sets we constructed.

Thomson Reuters' Web of Knowledge

Thomson Reuters' Web of Knowledge[10] is a large collection of scientific publications, spanning fields from the social sciences, arts and humanities, to the hard and soft sciences, to medicine. It includes parts of MEDLINE[11] which is the most common large dataset used in related work (Torvik, Weeber, Swanson, & Smalheiser, 2005; Torvik & Smalheiser, 2009; Treeratpituk & Giles, 2009) making it one of the largest bibliographic dataset available for academic study. In addition to its size, the metadata available in Web of Knowledge contains a rich set of attributes, most notably citation links between articles in the dataset.

This work focuses on a subset of Web of Knowledge for which we purchased an academic license from Thomson Reuters. It includes all bibliographic and citation information as well as abstracts from the SCIE, SSCI, and AHCI databases from 1990 to 2008. We first discarded nonarticle documents (reviews, meetings, patents), leaving 14 million articles. These articles cover 253 subjects (primarily focusing on the hard sciences) and 54 million author instances, with the bulk of the data coming from the years 1990 to 2008.

Disambiguating authors in this dataset is challenging because of its size and its lack of metadata for the earlier years. Figure 3 shows the counts of different metadata from 1990 to 2008. First names only really become available in 2005, and addresses in 2007. Although e-mail addresses start showing up around 2000, even by 2008 only about one in four authors have e-mail address metadata.

From 54 million author instances blocking produces roughly 3.2 million blocks, whose size follows a power law distribution as seen in Figure 4. The largest block in the dataset was "Lee, J." with 56,462 articles. Figure 4 shows what the cumulative distribution frequency of the Web of Knowledge dataset when divided into blocks (note that just over 95% of the blocks are less than 100 articles in size). So the vast majority are small blocks; however, there are still 5,634 blocks with 1,000 articles or more, with 130 blocks larger than 10,000 articles in size.

The entire Web of Knowledge data set was indexed into two Lucene[12] indices with term stemming and stop word removal, one for article and one for author data. This was to enable rapid retrieval, and to leverage tf–idf information efficiently provided by Lucene.

The dataset contains a small percentage (0.16%) of articles in which two authors have the same last name and first initial. Without distinguishing metadata for each author,

[10]http://www.webofknowledge.com/

[11]http://www.nlm.nih.gov/pubs/factsheets/medline.html

[12]http://lucene.apache.org/

DOI: 10.1002/asi

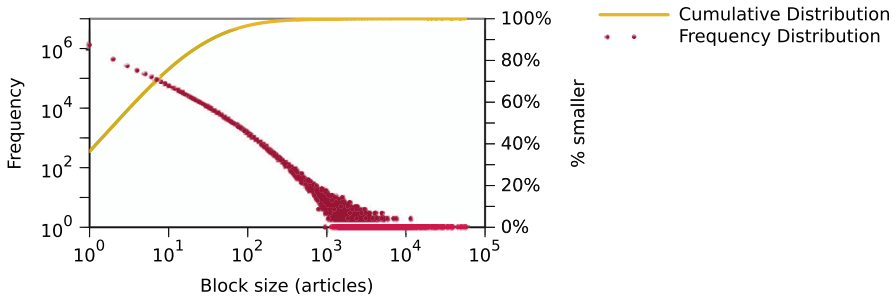

FIG. 4. The size of blocks follows a power distribution—there are many more small blocks than there are large ones. The vast majority of blocks in Web of Knowledge are below 100 in size. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

it is impossible to differentiate these authors; they were therefore removed from our data.

Creating a Test Set by Soliciting Publications

One of the main difficulties in evaluating author disambiguation methods is the lack of a large and accurately labeled corpus. We chose not to use Thomson Reuters' Highly Cited authors set, because this dataset is strongly biased toward authors with very large numbers of publications and because the publication lists were often incomplete. Instead, we collected our own evaluation dataset by using the e-mail addresses in the dataset to produce a stratified sample of e-mail addresses and solicited each author in the sample to submit a list of publications. Although our dataset has its own biases as well, we felt the advantages of stratified sampling and the completeness of using author-supplied publication lists made this effort worthwhile. The following describes the sampling process, the collection process, and parsing of the submitted data into a gold standard dataset.

Sampling process. After grouping authors into last name and first initial blocks, we created a list of e-mail addresses ordered by frequency. E-mails were selected only for blocks that contained more than one e-mail address, as these were potentially ambiguous names. The created list was divided into a set of 100 bins by frequency, and a stratified sample of 2,000 e-mails was selected. The top bin contained names such as "Zhang, W.," "Wang, W.," and "Li, J." The bottom bins contained much less ambiguous names such as "Fuxe, K.," "Lebwohl, M.," and "Watanabe, O."

Collection process. The selected e-mails were sent out over a period of 1 week to conform with spam protection restrictions. Each e-mail politely requested the recipient's list of publications and linked to a web form that allowed them to submit a link to their list of publications online, upload a curriculum vitae (CV), or cut and paste a list of publications. Despite approximately 25% of the e-mails bouncing back, we received 237 responses with a 15% response rate on e-mails that actually got through. The majority of respondents pasted their publication list into the text box, leaving only 25% of the data to be manually processed.

Matching publications to web of knowledge. For automated evaluation, the submitted publications needed to be matched to internal Web of Knowledge document IDs. The formatting of each of the submitted responses varied making this a nontrivial task. Rather than rely on manual matching, we used the following fully automated process:

1. Split the publication text into chunks potentially containing article titles by scanning for each of three patterns in turn:
   (a) Publication numbering: "1.", "1)", . . .
   (b) Year with delimiter: "2008.", "(2008)", . . .
   (c) Year alone: any four-digit number from 1970–2010
2. Each individual split will still contain a large number of nontitle words, so we match chunks with the following noise-tolerant process:
   (a) Use the Porter stemmer algorithm to stem all words and remove stopwords from the text chunks.
   (b) For each text chunk, construct bigram shingles over its contents and insert each shingle as a phrase in a boolean OR query.
   (c) Query the Lucene article index for all articles that list the desired author and match some of the query shingles in the title to get a fixed-size list of potential matches. We found we needed to fetch no more than 10 to get accurate results.
   (d) Normalize the returned titles in the same fashion as the text chunks.
   (e) Spell-correct any word that is one edit-distance away to account for typos and data-entry errors.
   (f) Compare each result R (composed of bigram shingles r) with the query Q (and its shingles q), and rescore with

$$score(R) = \frac{1}{\sum_{q \in Q} |q|} \sum_{r \in R \cap Q} |r|.$$

   (g) Finally, select the highest scoring match or none if there are no results with a score greater than a threshold of 0.67.

DOI: 10.1002/asi

Using this process we were able to match 15,335 articles out of 36,132 extracted text chunks (42.44%). During error analysis on our training data, we found a number of false positives. More recent articles sometimes appeared in the Web of Knowledge database but were not listed on the authors submitted publication lists. We therefore expanded our gold sets with all articles matching an e-mail address with an author already in the set. This increased the size of our dataset to 15,750 articles. On average 8.8% of authors' articles in the Web of Knowledge database did not appear in the submitted publications list. The mean number of matched publications per author was 67 with a median of 21.

The author blocks of these matched publications were then ordered by the average rank of the e-mails contained, ordering the authors by numbers of publications. These blocks were then sampled and split into stratified sets for training (3,951 articles), development (4,718 articles), and testing (7,081 articles), keeping all the articles for each author block together in the same set.

The training set was not used in training any classifiers. It was used only for feature engineering. The development set was used for evaluating feature sets and rule-based clustering evaluation. We tuned our approach on this set. The test set was used only for evaluating our final system. We chose to have a larger test set because our algorithm does not require the training data. This larger set provides a very good proxy for unseen data, as we only ever used it for final evaluation, with no other analysis performed on it.

Experiments: Rules, Features, and Parameters

In this section, we describe our experiments for determining which rules to use in the first bootstrapping stage, i.e., which features to use in the second supervised classification stage, and our studies on other algorithm details, such as the type of classifier or the clustering parameters. All experiments in this section are evaluated on the development set using the F1 and B^3 metrics described in the following subsection.

Evaluation Metrics

We employ two standard clustering metrics for evaluation, both of which are defined in terms of a precision, P, a recall R and their harmonic mean, F: pairwise F1 (Vilain, Burger, Aberdeen, Connolly, & Hirschman, 1995), and B^3 (Bagga & Baldwin, 1998). The pairwise F1 metric looks at all the pairs of articles in the reference clustering and asks for each pair if the two articles show up in the same or different clusters in the system clustering. Because it is based on pairs, it over-penalizes merging or separating gold clusters quadratically in the size of the cluster. Given C_{gold}, the reference clusters, and C_{system}, the clusters predicted by the system, pairwise F1 is defined as:

$$\text{pairs}(C) = \bigcup_{c \in C} \{\{a_1, a_2\} : a_1 \in c \wedge a_2 \in c \wedge a_1 \neq a_2\}$$

$$P(C_{\text{gold}}, C_{\text{system}}) = \frac{|\text{pairs}(C_{\text{gold}}) \cap \text{pairs}(C_{\text{system}})|}{|\text{pairs}(C_{\text{system}})|}$$

$$R(C_{\text{gold}}, C_{\text{system}}) = \frac{|\text{pairs}(C_{\text{gold}}) \cap \text{pairs}(C_{\text{system}})|}{|\text{pairs}(C_{\text{gold}})|}$$

$$F(C_{\text{gold}}, C_{\text{system}}) = \frac{2 \cdot P(C_{\text{gold}}, C_{\text{system}}) \cdot R(C_{\text{gold}}, C_{\text{system}})}{P(C_{\text{gold}}, C_{\text{system}}) + R(C_{\text{gold}}, C_{\text{system}})}$$

B^3 looks at each article author and asks what the intersection is between the cluster for that article author in the reference clustering and the cluster for that article author in the system clustering. Given the set of articles, A, and the reference and system clusters, C_{gold} and C_{system}, the B^3 F is defined as:

$$\text{cluster}(C, a) = c : c \in C \wedge a \in c$$

$$P(A, C_{\text{gold}}, C_{\text{system}}) = \frac{1}{|A|} \sum_{a \in A} \frac{|\text{cluster}(C_{\text{gold}}, a) \cap \text{cluster}(C_{\text{system}}, a)|}{|\text{cluster}(C_{\text{system}}, a)|}$$

$$R(A, C_{\text{gold}}, C_{\text{system}}) = \frac{1}{|A|} \sum_{a \in A} \frac{|\text{cluster}(C_{\text{gold}}, a) \cap \text{cluster}(C_{\text{system}}, a)|}{|\text{cluster}(C_{\text{gold}}, a)|}$$

$$F(A, C_{\text{gold}}, C_{\text{system}}) = \frac{2 \cdot P(A, C_{\text{gold}}, C_{\text{system}}) \cdot R(A, C_{\text{gold}}, C_{\text{system}})}{P(A, C_{\text{gold}}, C_{\text{system}}) + R(A, C_{\text{gold}}, C_{\text{system}})}$$

We rely primarily on pairwise F1 for our analysis because it is more standard, though we do report B^3 on the final test set.

Choosing Rules for Stage 1 (Rule-Based Clustering)

Over 100 rule combinations were evaluated on the development set. When combining rules, using multiple positive rules (e.g., E-mail and Self-citation) will result in more merges since both rules are applied, while adding negative rules (e.g., Name or Language) will result in fewer merges because some merges will be refused. We determined performance of each rule combination by averaging the pairwise F1 cluster evaluation metric on each block. Table 4 summarizes the performance of select rule combinations.[13] The first seven entries in Table 4 show us the performance of each of the positive rules by themselves.

Because we augmented our gold datasets with e-mail matches manually, the e-mail feature yields an artificial 100% precision. For this reason, we exclude it from the

[13]We omit results for the negative Coauthor$_n$ rule because it proved to be too restrictive resulting in low recall.

DOI: 10.1002/asi

TABLE 4. Select rule-based clustering results on the development data with 95% confidence intervals.

Rule		Pairwise evaluation		
		Precision	Recall	F1
E-mail	Mean	1.000 ± 0.000	0.081 ± 0.026	0.145 ± 0.038
	Median	1.000	0.044	0.092
Self-Citation	Mean	0.880 ± 0.034	0.462 ± 0.077	0.569 ± 0.070
	Media	0.926	0.500	0.605
MI with Subject	Mean	0.564 ± 0.102	0.064 ± 0.054	0.192 ± 0.073
	Median	0.597 ±	0.000 ±	0.027 ±
Exact Citation	Mean	0.297 ± 0.093	0.736 ± 0.083	0.285 ± 0.088
	Median	0.051	0.334	0.093
Exact Venue	Mean	0.475 ± 0.087	0.081 ± 0.024	0.122 ± 0.031
	Median	0.485	0.052	0.090
Coauthor$_p$	Mean	0.226 ± 0.086	0.684 ± 0.080	0.206 ± 0.073
	Median	0.026	0.799	0.051
Exact Coauthors	Mean	0.367 ± 0.116	0.002 ± 0.002	0.004 ± 0.003
	Median	0.000	0.000	0.000
E-mail	Mean	1.000 ± 0.000	0.087 ± 0.038	0.159 ± 0.048
With Language and Name	Median	1.000	0.038	0.087
Self-Citation	Mean	0.882 ± 0.037	0.376 ± 0.076	0.492 ± 0.072
With Language and Name	Median	0.949	0.313	0.492
MI with Subject	Mean	0.616 ± 0.100	0.053 ± 0.044	0.058 ± 0.045
With Language and Name	Median	0.760	0.000	0.000
Coauthor$_p$	Mean	0.441 ± 0.085	0.452 ± 0.071	0.384 ± 0.071
With Language and Name	Median	0.425	0.469	0.344
E-mail & Self-Citation	Mean	0.867 ± 0.050	0.499 ± 0.075	0.578 ± 0.067
	Median	0.943	0.500	0.625
E-mail & Self-Citation	Mean	0.890 ± 0.037	0.493 ± 0.074	0.594 ± 0.066
With Language and Name	Median	0.954	0.500	0.638
E-mail & Self-Citation	Mean	0.718 ± 0.091	0.523 ± 0.079	0.483 ± 0.078
& MI Subject	Median	0.904	0.532	0.480
E-mail & Self-Citation	Mean	0.823 ± 0.056	0.505 ± 0.075	0.571 ± 0.065
& MI Subject with Language and Name	Median	0.907	0.527	0.624

discussion of single features below. However the precision of this feature will always be very high because e-mail addresses are rarely reused. Its low recall is indicative of the fact that e-mail metadata is missing from much of the data.

The Self-Citation feature was the most precise of all the single features, achieving a mean precision of 0.880, with half of the clusters having a precision of 0.926 (the median) or higher. The variability across clusters was also the lowest of all the single features, with a 95% confidence interval of ± 0.034. Self-Citation also achieved a reasonable recall, 0.462, indicating that authors quite frequently cite their previous work.

Most other features were substantially poorer in terms of precision (and therefore F1). The next highest precision came from Middle Initial with Subject, at 0.564, followed by looking for common venues (Exact Venue) at 0.475. Both of these features had recalls lower than 0.060. Even the addition of negative rules like Language and Name did not raise the precision much (only to 0.616 for Middle Initial with Subject), and hurt the recall even more. Thus, matching by common subject areas or common venues is often inaccurate and fails to find most of the necessary matches.

The coauthorship features had some of the lowest precisions for single features, with matching a single coauthor (Coauthor$_p$) achieving only 0.226, and matching three coauthors (Exact Coauthors) achieving only 0.367, though Coauthor$_p$ had the highest median recall (0.799). Both of these features also had extremely low median precisions (less than 0.03) with large 95% confidence intervals (more than ± 0.080), suggesting that they were merging correctly in only a few clusters, while in most clusters they were producing many bad merges. These results are unlike those of Ferreira et al. (2010), which found good performance on DBLP and BDBComp datasets when matching on only a single coauthor. This may be a result of the way these datasets were collected. The original DBLP dataset was not manually checked; instead the top 11 largest blocks by first initial and last name were taken, and gold-standard author clusters were assumed wherever a full name was available, matched across articles, and the full name had more than five citations. However, another explanation for these differences might be the average number of authors per paper. Ferreira et al. (2010) do not report the average number of authors, but Bhattacharya and Getoor (2007) found that that their own coauthorship-based disambiguation system's

DOI: 10.1002/asi

performance peaked when there were two coauthors per paper, achieving F1 of 0.994 and 0.985 on Citeseer and arXiv datasets where articles averaged 1.9 authors per paper, but only F1 of 0.819 on a BioBase dataset where articles averaged 5.3 authors per paper. In the Web of Knowledge, there are on average 3.78 authors per paper, suggesting that Web of Knowledge is more like their difficult BioBase dataset where coauthorship was not as helpful.

Looking at combinations of features, the best rule-based clustering performance was from combining E-mail with Self-Citation and the negative Name and Language rules, achieving precision of 0.890, recall of 0.493 and F1 of 0.594. Half of the resulting clusters had precision of 0.954 (the median) or higher. This performance already puts it into the state-of-the-art category for unsupervised methods. Thus, Self-Citation is the backbone of the rule-based clusterer, and because it yields high-precision clusters that are large enough to provide quality training data, it enables us to bootstrap the rest of our author disambiguation system.

Choosing Features for Stage 2 (Supervised Classification)

We next examine which features were most important for training our classifiers on the rule-based clusters. With exhaustive feature selection infeasible, we looked at several simple feature selection strategies and evaluated their performance on the development set of 3,951 articles. We considered the following feature sets:

All Features: The full set of features described previously.
Previous work: The subset of features that were used in previous literature.
By Weight: A selected set of features, based on the weights learned for features by the classifier. Specifically, we looked at the median weight for each feature across the classifiers that were learned one for each block. If the median value was >0.1 or <–0.1, we included the feature in this feature set. Intuitively, this collected the features that contributed something substantial to the classification function at least half the time.
By Weight − Citation: The By Weight features, minus all the citation features.
By Weight – Citation + 1: The By Weight – Citation features, but with the single additional citation feature that most improved performance: Cited Journal Titles.
Speed: A selected set of features designed to speed up classification. Computation time of tf–idf features was proportional to the number of terms, so the abstract, title, and especially their cited/citing variants were of greatest concern. These features alone accounted for well over 75% of feature computation time. We selected the features that most frequently had nonzero weights and whose computation time fell within our speed criteria, resulting in 57 features. Table 5 shows some of these features and the percent of blocks in which they appeared. Computing the speed-optimized feature set is 130 times faster than computing all features.
Speed – Citation: The Speed-Optimized feature set, minus all the citation features.

TABLE 5. Top 10 speed-optimized features from feature selection (excluding conjunctions) with % blocks that had the feature weighted nonzero.

100%	Citing Keywords	97%	Cited Subject Cat.	95%	Initials
100%	Cited Keywords	95%	E-mail	91%	Addresses
98%	Citing Subject Cat.	95%	Language		
98%	Addresses	93%	Cited Journal Titles		

The above-mentioned feature sets were evaluated on the development set using our best rule-based clustering approach, and had their agglomerative clustering threshold tuned to give maximum F1 scores. These results are presented in Table 6.

As compared with previous work, the full feature set achieved higher precision (0.780 vs. 0.766) at the expense of lower recall (0.740 vs. 0.773). Selecting features By

TABLE 6. Feature set evaluation on the development dataset with 95% confidence intervals.

			Pairwise evaluation		
Feature set	Cite?	#	Precision	Recall	F1
All Features, mean	Yes	1,080	0.780 ± 0.066	0.740 ± 0.070	0.716 ± 0.063
All Features, median			0.895	0.860	0.820
Previous Work, mean	No	18	0.766 ± 0.060	0.773 ± 0.070	0.733 ± 0.059
Previous Work, median			0.850	0.898	0.815
By Weight, mean	Yes	73	0.777 ± 0.063	0.778 ± 0.067	0.746 ± 0.059
By Weight, median			0.870	0.892	0.826
By Weight – Citation, mean	No	27	0.750 ± 0.059	0.813 ± 0.065	0.748 ± 0.055
By Weight – Citation, median			0.798	0.939	0.821
By Weight – Citation + 1, mean	Yes	28	0.775 ± 0.058	0.822 ± 0.068	0.767 ± 0.060
By Weight – Citation + 1, median			0.859	0.955	0.857
Speed, mean	Yes	57	0.772 ± 0.059	0.764 ± 0.069	0.732 ± 0.057
Speed, median			0.860	0.891	0.802
Speed – Citation, mean	No	22	0.763 ± 0.058	0.750 ± 0.069	0.724 ± 0.058
Speed – Citation, median			0.841	0.882	0.810

Note. The Cite? column indicates whether or not the feature set contains citation features. The # column indicates the number of features in the feature set.

DOI: 10.1002/asi

TABLE 7. Other vector similarity measure evaluation with max F1 scores on the training dataset using previous features.

	Pairwise Evaluation		
Measure	Precision	Recall	F1
Cosine	0.766	0.773	0.732
Jaccard	0.755	0.714	0.686
Jensen-Shannon	0.826	0.651	0.682

Weight resulted in the benefits of both of these approaches (precision 0.777 and recall 0.778). Removing the citation features from By Weight resulted in lower precision (0.750 vs. 0.777) but higher recall (0.813 vs. 0.778). But adding back in just the single Cited Journal Titles feature restored the original precision (to 0.775) while maintaining the high recall (0.822). The speed optimized feature set performed similar to previous work and slightly below our feature selected versions. Removing the citation features from the speed-optimized feature set degraded performance, indicating that citation features help a little.

We used the Speed feature set for disambiguating the Web of Knowledge.

Other Parameters

Comparing different vector similarity measures. We tested two alternatives (Jaccard and Jensen-Shannon) to the cosine method for comparing feature vectors, using the previous work feature set on a stratified sample from the training data. Table 7 shows the maximum F1 scores achieved for each measure after tuning the agglomerative clustering threshold. The Jensen-Shannon measure had the highest precision but lowest recall and overall lowest F1. Jaccard had a slightly higher F1 than Jensen-Shannon but cosine had the best overall F1, with the highest recall and higher precision than Jaccard. Because our objective is to create large, high-precision clusters, cosine is a better choice than either Jaccard or Jensen-Shannon. This is also consistent with previous work that found better results with the cosine similarity measure (On, Lee, Kang, & Mitra, 2005).

Cosine also has the additional advantage that it is faster to compute than Jaccard or Jensen-Shannon, because the term vectors for each article can be normalized once ahead of time, and the cosine is then just a simple dot product over these term vectors. Jaccard and Jensen-Shannon do not have a simple equivalent to this prenormalization, and thus require expensive division operations on every pairwise comparison instead of just once per article.

Clustering parameters. Our best feature set exhibits a fairly stable set of agglomerative clustering thresholds that allow switching between recall and precision without much of a drop in F1. Figure 5 shows this ability to have an F1 not far off from the max of 0.767, while still having a precision of just under 0.840.

We also explored alternative similarity calculations. Previous work relied on max-link clustering—measuring similarity based only on the closest pair of author instances—while we used centroid clustering—measuring similarity by averaging the feature values of all articles in a cluster, and comparing these centroids for pairwise similarity. Table 8 shows a comparison of max-link and centroid clustering. Our results show that centroid clustering is more precise, resulting in a higher overall F1.

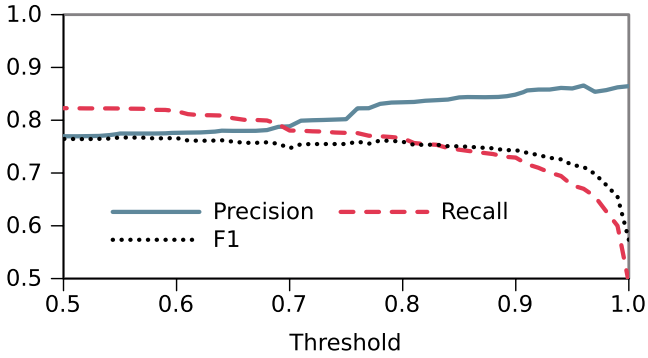

FIG. 5. Precision, recall, and F1 by agglomerative clustering threshold using the best feature set. Our figures and tables show averages of the P/R/F1 values across dev-set blocks (hence, F1 is not guaranteed to lie between P and R). [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE 8. Performance of tuned max-link and centroid based clustering using speed-optimized features on the development dataset, with the same rule-based clustering.

Clusters	Precision	Recall	F1
Max-Link	0.723	0.803	0.721
Centroid	0.772	0.764	0.732

Alternative classifier: random forests. We also benchmarked an alternative classifier, random forests (Breiman, 2001), which had previously been applied to this task (Treeratpituk & Giles, 2009). We used an optimized Weka-based implementation[14] in place of the logistic classifier, using the previous work feature set and holding other parameters constant. The random forests were tuned on a representative stratified sample of the training set, optimizing for out of bag error. The best performing settings were reached with 300 trees. The number of features to be randomly selected from in building a tree followed standard practice of $log_2(M) + 1$ where M is the total number of features (Breiman, 2001; Treeratpituk & Giles, 2009). In comparison to running with the logistic classifier, random forests were slower and memory requirements were larger. To train the random forest model under the same time and memory constraints we had for the logistic classifier, we were forced to reduce the training set to only 500,000 examples. The logistic classifier scales much better and is able to handle 10,000,000 training examples under the same constraints.

[14]http://code.google.com/p/fast-random-forest/

DOI: 10.1002/asi

TABLE 9. Performance of random forests and our unsupervised algorithm on the development dataset as compared with our rule-based and logistic classifier approaches, using previous features.

Classifier	Precision	Recall	F1
Rule-based	0.890	0.493	0.594
Logistic classifier	0.766	0.773	0.733
Random forests	0.864	0.491	0.573
Unsupervised agglomerative cluster	0.774	0.549	0.586

Table 9 shows that random forests achieved higher precision, but much lower recall, and thus a lower F1 score than the logistic classifier. The random forest results are very similar to the rule-based clustering results, suggesting that the random forest model is overfitting and just memorizing the training data. This may be an effect of the smaller number of training examples it had access to as a result of its memory and speed concerns.

Test Set Evaluation

The previous section showed that our system performs well on the development data and explored a number of properties of the algorithm. For the rule-based stage 1, we showed Self-Citation was the single best rule, and combining E-mail with Self-Citation and the negative Name and Language rules gave best overall performance. For the supervised stage 2, a number of feature sets offered improved F1 over prior work features, including the 27-feature By Weight − Citation feature set, and a 28-feature set that included only one citation feature, Cited Journal. Finally, we showed that our clustering was relatively robust to the agglomerative clustering threshold, and showed higher F1 on the development data for centroid clustering as opposed to max-link clustering, for logistic regression as opposed to random forests.

In this section, we compare the development-set performance of our algorithm against an unsupervised baseline, and then report our performance on the held-out test set.

An Unsupervised Baseline: Agglomerative Clustering

Do weakly supervised algorithms like ours perform better than fully unsupervised algorithms? Our intuition was that they would, because the second, supervised stage of our algorithm can learn to discriminatively weigh features in a way that is not possible for fully unsupervised algorithms. To test this prediction, and provide this important baseline for our algorithm, we implemented an unsupervised agglomerative clusterer.

The starting point was the output of the rule-based stage. Where the supervised approach takes vectors of feature cosine scores and uses the learned weights in a weighted sum of the feature cosines, in our unsupervised approach, we simply take the average of the feature cosines. This is essentially the same as weighting all features equally.

We then evaluated the clusterings at each agglomerative clustering similarity threshold in increments of 0.01 until there was only one cluster. The precision, recall, and F1 for each agglomerative clustering threshold were then averaged and the best overall F1 result was chosen. This approach produces a very conservative baseline; as with a real system, one would not know the best agglomerative clustering threshold to use. Table 9 shows that our unsupervised agglomerative clusterer with perfect hindsight produced a better F1 score than random forests, but was worse than both the rule-based and the logistic classifier.

Test Set Results

Previous sections have summarized our results on the development test set. In this section, we now confirm the performance of our algorithm on the test dataset. Table 10 shows that our bootstrapped approach accurately disambiguates Web of Knowledge authors, and that our new features show significant improvement over features used in previous work. Surprisingly, the speed-optimized features scored a higher mean F1 on the test set than the feature set that performed best on the development set, suggesting that in optimizing our feature set for fast computation, we have not sacrificed clustering accuracy.

Analysis

We performed a number of analyses on our system.

Statistics of Resulting Dataset

Our rule-based system disambiguated the 52,332,037 author instances from the Web of Knowledge into 25,327,334 clusters (unique authors). Our full, self-trained disambiguation algorithm merged these as appropriate to produce 13,873,529 authors. Given that there were 14,365,790 articles in the Web of Knowledge, this means that our system predicts that, on average, each author in the Web of Knowledge has published slightly more than one paper there.

The author sets inferred by both the rule-based disambiguation system and the self-trained disambiguation system are available by writing to the last author of this article. These datasets may be distributed only to individuals or institutions that have first acquired the Web of Knowledge dataset from Thomson Reuters.

Performance on Large Versus Small Blocks

Does the system perform well on small as well as large blocks? Table 12 shows that performance with our best feature set is similar all the way up to blocks of size 10,000, though it degrades somewhat for the largest blocks. Using the fastest feature set, performance degrades some even at blocks larger than 1,000 and more-so for the largest blocks. Of course, the largest blocks are generally the most ambiguous, so we expect them to be harder.

DOI: 10.1002/asi

TABLE 10. Final results on the test set.

Method		Pairwise F1 evaluation			B^3 Evaluation		
		Precision	Recall	F1	Precision	Recall	F1
Rule-based only	Mean	0.900	0.536	0.639	0.939	0.559	0.666
	Median	0.959	0.527	0.676	0.971	0.538	0.676
Bootstrapped, features from previous work	Mean	0.771	0.784	0.728	0.794	0.789	0.751
	Median	0.890	0.892	0.816	0.887	0.895	0.822
Bootstrapped, best features	Mean	0.783	0.823	0.766	0.808	0.822	0.781
	Median	0.883	0.928	0.870	0.889	0.909	0.864
Bootstrapped, speed- optimized features	Mean	0.832	0.788	0.784	0.868	0.797	0.807
	Median	0.887	0.889	0.859	0.926	0.889	0.864

TABLE 11. The 57 speed-optimized features from the system with the highest test set performance.

Simple features	
Language	Cited Keywords
Last Names with Initials	Citing Keywords
Addresses	Cited Language
Reprint Organization	Citing Language
Subjects	Cited Last Names with Initials
Year	Cited Subjects
Middle Initials	Citing Subjects
E-mail Address	Cited Cited Journal Titles
Cited Journal Titles	Citing Cited Journal Titles
Conjunctive features	
Language × Cited Keywords	Reprint Organization × Middle Initials
Language × Cited Language	Reprint Organization × Reprint Organization
Language × Cited Last Names with Initials	Subjects × Middle Initials
Language × Cited Subjects	Middle Initials × Middle Initials
Language × Cited Cited Journal Titles	Cited Journal Titles × Middle Initials
Language × Cited Keywords	Cited Keywords × Cited Language
Language × Citing Language	Cited Keywords × Middle Initials
Language × Citing Subjects	Cited Language × Cited Cited Journal Titles
Language × Citing Cited Journal Titles	Cited Language × Cited Language
Language × Middle Initials	Cited Language × Year
Language × Cited Journal Titles	Cited Last Names with Initials × Cited Language
Language × Language	Cited Subjects × Cited Language
Language × Reprint Organization	Cited Subjects × Middle Initials
Language × Year	Cited Cited Journal Titles × Middle Initials
Last Names with Initials × Middle Initials	Citing Language × Cited Language
Last Names with Initials × Language	Citing Language × Citing Language
Addresses × Cited Language	Citing Subjects × Cited Subjects
Addresses × Middle Initials	Citing Subjects × Middle Initials
Addresses × Language	Citing Cited Journal Titles × Cited Cited Journal Titles
	Citing Cited Journal Titles × Middle Initials

TABLE 12. Performance by block size on the development set with 95% confidence intervals.

Block size	Blocks	Example	F1	F1 speed
0–1,000	13	Rief, W	0.782 ± 0.058	0.778 ± 0.059
1,000–10,000	36	Lewis, M	0.783 ± 0.068	0.738 ± 0.069
10,000–100,000	13	Zhang, J	0.710 ± 0.060	0.668 ± 0.057

Note. For each bin, the number of blocks in that bin and and an example name from the bin are given.

TABLE 13. Pretrained and individually bootstrapped classifier performance on the development dataset with 95% confidence intervals.

	Method	Precision	Recall	F1
Blocks of size < 400	Pretrained	0.777 ± 0.162	0.923 ± 0.088	0.838 ± 0.133
	Bootstrapped	0.790 ± 0.148	0.968 ± 0.050	0.858 ± 0.108
All blocks	Pretrained	0.651 ± 0.077	0.813 ± 0.063	0.668 ± 0.068
	Bootstrapped	0.810 ± 0.044	0.763 ± 0.050	0.760 ± 0.053

Recall that we trained a separate small-block classifier to use for clustering small blocks of less than 100 articles, because the rule-based methods often failed to produce enough negative training examples for these smaller blocks. The classifier was trained on the bootstrapped data generated for 100 blocks of 100 articles each. Table 13 shows that on smaller blocks, the classifier performs on par with individually bootstrapped classifiers. Overall, however, classifier performance on all blocks decreases without per block training data.

Scalability

Scaling to process over 54 million author instances provided a number of unique challenges. The major bottlenecks in processing came from generating feature vector data and training the classifier, so feature selection proved crucial.

The processing time of our total algorithm was almost linear with respect to the number of articles within the block, with a very small quadratic term (see Figure 6). Because

DOI: 10.1002/asi

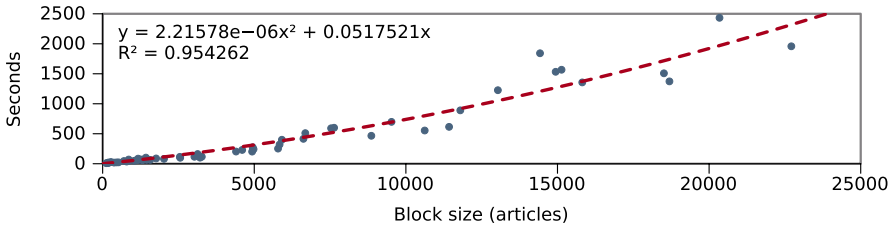

FIG. 6. Distribution of bootstrap method runtime with speed-optimized features by block size. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

TABLE 14. Approximate processing time excluding delays due to unrelated processor/memory load on the system and interruptions.

Set	Blocks	Articles	Processing time
Small	1,979,754	25,519,302	23 hours
Large	83,730	27,479,693	1 day, 4 hours
Total	2,063,484	52,998,995	2 days, 3 hours

each name block can be processed independently of the others, the process is very easily parallelized. The author disambiguation system with speed-optimized features was run on two 8-core 2.4 Ghz machines with 72 GB of RAM each. We were able to disambiguate the entire Web of Knowledge dataset using our system in 5 days and 20 hours. Table 14 gives some additional timing details.

Error Analysis

We analyzed output clusters to understand the errors and whether they were introduced by rule clustering or by the agglomerative clusterer.

What we got almost right. We examined a random sample of the clusters that received high precision above 0.85. In this sample, each article that was marked incorrect by our automatic metrics (i.e., each precision error) did in fact correctly belong to that author. We found that these articles were not included by the authors in their list of publications, generally because they tended to be a different version of an article already in the publication list. (See Table 15 for examples.)

What we got wrong—false positives. False positives were generally articles that looked very similar textually and had missing metadata (especially first names) that would have discriminated them otherwise. Table 16 shows an example. We find that these similar articles were clustered together by the agglomerative clusterer, and not the rule clusterer stage; without more metadata only having the negative name rule for the agglomerative clusterer was not sufficient.

For the clusters that had low precision (below 0.75), we found that the errors were introduced during the rule clustering stage. This generally capped our precision, as it tainted the cluster. This problem is the hardest to solve, as it would mean investigating whether the precision on the feature rule-matching stage can be made even higher.

What we got wrong—false negatives. Not merging articles for one author was generally caused by either lack of metadata, offering the classifier too little information when the titles and subjects differ, or large differences in the dimensionality of the feature vectors between very small and very large clusters.

TABLE 15. Example of articles written by authors but incorrectly omitted in the author-supplied gold standard that lead to lowered precision.

Two different versions of the same article		
(1)	Title	Problems, methods and specialization
	Author	Jackson, M.
	Abstract	Software engineering is not a discipline; it is an aspiration, as yet unachieved. Many approaches have been proposed, including reusable components. . .
	Publication	Software Engineering Journal
	Year	1994
	. . .	
(2)	Title	Problems, methods, and specialization
	Author	Jackson, M.
	Abstract	The large aspiration to place the whole of software development alongside the established branches as one more branch of engineering is misconceived. . . .
	Publication	IEEE Software
	Year	1994
	. . .	
One article in gold, the other selectively omitted		
(3)	Title	Distributed feature composition: A virtual architecture for telecommunications services
	Author	Jackson, M.
	Coauthor(s)	Zave, P.
	Publication	IEEE Transactions on Software Engineering
	Year	1998
	. . .	
(4)	Title	A component-based approach to telecommunication software
	Author	Jackson, M.
	Coauthor(s)	Zave, P.
	Publication	IEEE Software
	Year	1998
	. . .	

DOI: 10.1002/asi

TABLE 16. Examples of false positives; all similar articles on nanotubes.

(1)	Title	Low-temperature synthesis and photocatalytic properties of ZnO nanotubes by thermal oxidation of Zn nanowires
	Author	Li, H.
	Coauthor(s)	Lu, H.B., Shuai, M., Li, J.C., Fu, Q., Fhu, M., Tian, Y., Zhu, B.P., Liao, L.
	Abstract	A low-cost and catalyst-free two-step approach has been developed to produce ZnO nanotubes (ZNTs) by simple thermal oxidation of Zn nanowires . . .
	Year	2008
	. . .	
(2)	Title	A carbon-nanotube field-emission display with simple electron-beam trajectory control
	Author	Li, H. (Hao)
	Coauthor(s)	Johnson, M.R., Coll, B.F., Howard, E., Dean, K.A., Dworsky, L., Marshbanks, L.
	Abstract	A unique gated cathode structure for a carbon-nanotube-based field-emission display has been designed and built. This structure optimizes the electron-. . .
	Year	2007
	. . .	
(3)	Title	The structures and electrical transport properties of germanium nanowires encapsulated in carbon nanotubes
	Author	Li, H.
	Coauthor(s)	Zhang, X.Q., Liew, K.M.
	Abstract	The structures of Ge nanowires are studied by means of geometry optimization method in this paper. As the radii of carbon nanotubes. . .
	Year	2007
	. . .	

Conclusions

In this work, we have presented an accurate, scalable model for author disambiguation that clusters publications into their real-world authors. We introduced a simple self-citation clustering rule and demonstrated that, in combination with other simple rules, we can generate high-precision initial clusters without any manually annotated training data. These rule-based clusters then allowed us to bootstrap a statistical clusterer, first by training a supervised classifier on the rule-based clusters, and then by applying the learned classifier as the similarity metric in an agglomerative clustering algorithm. We introduced some new features for the similarity metric based on the citation network and feature conjunctions, and this improved the quality of our clusterings over the features from prior work. To evaluate our models, we collected and aligned the publications of over 200 real-world authors to Thomson Reuters' Web of Knowledge, and showed that our model could cluster these publications with a precision of 0.832 and a recall of 0.788. Finally, we showed that with some simple feature selection and no loss in model performance, this model scaled up to cluster more than 54,000,000 author instances in Thomson Reuters' Web of Knowledge.

The success of our algorithm, drawing on and extending earlier work making partial use of semisupervised methods, points to the importance of bootstrapping as a strategy in solving tasks like author disambiguation on extremely large databases. On databases of this size, hand-labeling a training set for supervised learning is prohibitively expensive. But unsupervised algorithms that don't require hand-labeled training data often have trouble taking advantage of large sets of rich features, as shown in our unsupervised agglomerative clustering experiment. Unsupervised algorithms fail in this respect because they have no mechanism for learning to weight features discriminatively. Our self-supervised approach thus shares the advantages of unsupervised approaches (no need for expensive hand labels) as well as supervised approaches (a rich set of features that can be discriminatively trained).

Our results also shed light on the use of citation features in author disambiguation, suggesting that citation features can provide important information, but not in every aspect of processing. We show that *self-citation* is the best single bootstrapping feature, and performs very strongly when combined with features based on e-mails, author names, and languages. But for the supervised stage, other citation features offer only minor improvements, increasing F1 only from .748 to .767, suggesting that they might not be worth the trouble of extracting from databases that don't already have them.

The fact that self-citation is the most important feature immediately suggests applications to disambiguation tasks in domains with full text but without the full citation network, because self-citation can be approximated by features based on the occurrences of the author name in the text.

Finally, our results have something to say about other useful features. Table 5 showed the 10 top speed-optimized features. Some of these are the traditional features (e-mail, addresses, language, initials) that have been long studied. Others, however, use the citation network. The most useful of these network features seem to be the *keywords* and *subject categories* from the citing and cited papers. At least in this experiment, keywords and subject categories completely made up for the absence of expensive features based on full vocabularies, like title and abstract. This suggests an important cue for situations where a citation network is available: the use of properties of the cited and citing papers.

DOI: 10.1002/asi

This result extends the standard use of co-citation and bibliographic coupling in bibliometrics. We can see the importance of these features in Table 11, which shows the 57 features in the set that performed best on the final test set. Many of these are feature conjunctions, suggesting the further importance of considering interactions between features in future work.

Of course any experiment on a single corpus can only be preliminary, even a large corpus like Thomson Reuters' Web of Knowledge. It will be important to test our algorithms and insights on other datasets with other rich constraints. And the real test of our work will be the application of the disambiguated authors to better address deep scientific questions, like the role that individual authors play in the production of scientific knowledge.

Acknowledgments

This research was generously supported by NSF award 0835614. Many thanks to Dan McFarland and Chris Manning for advice throughout the project, to Konstantinos Katsiapis, Xiaolin Shi and Sonal Gupta and the other members of the MIMIR research group for help along the way, and to the anonymous reviewers for extraordinarily constructive feedback.

References

Andrew, G., & Gao, J. (2007). Scalable training of 11-regularized log-linear models. In Proceedings of the 24th International Conference on Machine Learning (ICML '07) (pp. 33–40). New York: ACM Press.

Bagga, A., & Baldwin, B. (1998). Algorithms for scoring coreference chains. Recall, 5(1), 2.

Banko, M., & Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (pp. 26–33). Stroudsburg, PA: Association for Computational Linguistics.

Bhattacharya, I., & Getoor, L. (2006). A latent dirichlet model for unsupervised entity resolution. In Proceedings of SIAM International Conference on Data Mining (pp. 47–58). New York: ACM Press.

Bhattacharya, I., & Getoor, L. (2007). Collective entity resolution in relational data. ACM Transactions on Knowledge Discovery from Data, 1(1), 5.

Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., & Fienberg, S. (2005). Adaptive name matching in information integration. IEEE Intelligent Systems, 18(5), 16–23.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In Proceedings of the 11th Annual Conference on Computational Learning Theory (pp. 92–100). New York: ACM Press.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5–32.

Chen, Z., Kalashnikov, D.V., & Mehrotra, S. (2007). Adaptive graphical approach to entity resolution. In Proceedings of the Seventh ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '07) (pp. 204–213). New York: ACM Press.

Chklovski, T., & Pantel, P. (2004). Verbocean: Mining the web for fine-grained semantic verb relations. In D. Lin & D. Wu (Eds.), Proceedings of Empirical Methods on Natural Language Processing (EMNLP '04) (pp. 33–40). Stroudsburg, PA: Association for Computational Linguistics.

Culotta, A., Kanani, P., Hall, R., Wick, M., & McCallum, A. (2007). Author disambiguation using error-driven machine learning with a ranking loss function. In Proceedings of the Sixth International Workshop on Information Integration on the Web (IIWeb-07), Vancouver, Canada: AAAI Press.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked T., Soderland, S., . . . Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. Artificial Intelligence, 165(1), 91–134.

Fan, X., Wang, J., Pu, X., Zhou, L., & Lv, B. (2011). On graph-based name disambiguation. Journal of Data and Information Quality, 2(10), 1–10, 23.

Fellegi, I.P., & Sunter, A.B. (1969). A theory for record linkage. Journal of the American Statistical Association, 64(328), 1183–1210.

Ferreira, A., Veloso, A., Goncalves, M.A., & Laender, A.H.F. (2010). Effective self-training author name disambiguation in scholarly digital libraries. In Proceedings of the 2011 Joint International Conference on Digital Libraries (JCDL '10). New York: ACM Press.

Han, H., Giles, L., Zha, H., Li, C., & Tsioutsiouliklis, K. (2004). Two supervised learning approaches for name disambiguation in author citations. In Proceedings of the Fourth ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '04) (pp. 296–305). New York: ACM Press.

Han, H., Xu, W., Zha, H., & Giles, C.L. (2005). A hierarchical naive bayes mixture model for name disambiguation in author citations. In Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05) (pp. 1065–1069). New York: ACM Press.

Han, H, Zha, H., & Giles, C.L. (2005). Name disambiguation in author citations using a k-way spectral clustering method. In Proceedings of the Fifth ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05) (pp. 334–343). New York: ACM Press.

Hearst, M.A. (1998). Automated discovery of WordNet relations (pp. 131–151). Cambridge, MA: The MIT Press.

Hill, S., & Provost, F. (2003). The myth of the double-blind review?: Author identification using only citations. ACM SIGKDD Explorations Newsletter, 5(2),179–184.

Huang, J., Ertekin, S., & Giles, C.L. (2006). Efficient name disambiguation for large-scale databases. Lecture Notes in Computer Science, 4213, 536.

Kanani, P., McCallum, A., & Pal, C. (2007). Improving author coreference by resource-bounded information gathering from the web. In Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI '07) (pp. 429–434). San Francisco, CA: Morgan Kaufmann Publishers.

Kang, I.S., Na, S.H., Lee, S., Jung, H., Kim, P., Sung, W.K., & Lee, J.H. (2009). On co-authorship for author disambiguation. Information Processing & Management, 45(1), 84–97.

King, B. (1967). Step-wise clustering procedures. Journal of the American Statistical Association, 62(317), 86–101.

Malin, B. (2005). Unsupervised name disambiguation via social network similarity. In Proceedings of the SIAM SDM Workshop on Link Analysis, Counterterrorism and Security. New York: ACM Press.

McClosky, D., Charniak, E., & Johnson, M. (2006). Effective self-training for parsing. In Proceedings of the Human Language Technology Conference of the NAACL, Main Conference (pp. 152–159). Stroudsburg, PA: Association for Computational Linguistics.

McRae-Spencer, D.M., & Shadbolt, N.R. (2006). Also by the same author: Aktiveauthor, a citation graph approach to name disambiguation. In Proceedings of the Sixth ACM/IEEE-CS Joint Conference on Digital Libraries (pp. 53–54). New York: ACM Press

Ng, V., & Cardie, C. (2003). Weakly supervised natural language learning without redundant views. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (pp. 94–101). Stroudsburg, PA: Association for Computational Linguistics.

On, B-W, Lee, D., Kang, J., & Mitra, P. (2005). Comparative study of name disambiguation problem using a scalable blocking-based framework. In Proceedings of the Fifth ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05) (pp. 344–353). New York: ACM Press.

Riloff, E., & Jones, R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In Proceedings of the National Conference on Artificial Intelligence (pp. 474–479). Cambridge, MA: AAAI Press / The MIT Press.

Smalheiser, N.R., & Torvik, V.I. (2009). Author name disambiguation. Annual Review of Information Science and Technology, 43, 287–313.

Small, H. (1973). Co-citation in the scientific literature: A new measure of the relationship between two documents. Journal of the American Society for Information Science, 24(4), 265–269.

Song, Y., Huang, J., Councill, I.G., Li, L., & Giles, C.L. (2007). Efficient topic-based unsupervised name disambiguation. In Proceedings of the Seventh ACM/IEEE-CS Joint Conference on Digital libraries (JCDL '07) (pp. 342–351). New York: ACM Press.

Tang, J., Fong, A.C., Wang, B., & Zhang, J. (2011). A unified probabilistic framework for name disambiguation in digital library. doi. ieeecomputersociety.org/10.1109/TKDE.2011.13

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological), 58(1), 267–288.

Torvik, V.I., & Smalheiser, N.R. (2009). Author name disambiguation in medline. ACM Transactions on Knowledge Discovery from Data, 3(3), 1–29.

Torvik, V.I., Weeber, M., Swanson, D.R., & Smalheiser, N.R. (2005). A probabilistic similarity metric for medline records: a model for author name disambiguation. Journal of the American Society for Information Science and Technology, 56(2), 140–158.

Treeratpituk, P., & Giles, C.L. (2009). Disambiguating authors in academic publications using random forests. In Proceedings of the Ninth ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '09) (pp. 39–48). New York: ACM Press.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., & Hirschman, L. (1995). A model-theoretic coreference scoring scheme. In Proceedings of the Sixth conference on Message understanding (MUC6 '95) (pp. 45–52). Stroudsburg, PA: Association for Computational Linguistics.

White, H.D., & Griffith, B.C. (1981). Author co-citation: A literature measure of intellectual structure. Journal of the American Society for Information Science, 32, 163–171.

Yang, K-H., Peng, H-T., Jiang, J-Y., Lee, H-M., & Ho, J-M. (2008). Author name disambiguation for citations using topic and web correlation. In Proceedings of the 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL '08) (pp. 185–196). Berlin, Heidelberg: Springer-Verlag.

DOI: 10.1002/asi