Author Disambiguation using Error-driven Machine Learning with a Ranking Loss Function

Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Abstract

Author disambiguation is the problem of determining whether records in a publications database refer to the same person. A common supervised machine learning approach is to build a classifier to predict whether a *pair* of records is coreferent, followed by a clustering step to enforce transitivity. However, this approach ignores powerful evidence obtainable by examining *sets* (rather than *pairs*) of records, such as the number of publications or co-authors an author has. In this paper we propose a representation that enables these *first-order features* over sets of records. We then propose a training algorithm well-suited to this representation that is (1) *error-driven* in that training examples are generated from incorrect predictions on the training data, and (2) *rank-based* in that the classifier induces a *ranking* over candidate predictions. We evaluate our algorithms on three author disambiguation datasets and demonstrate error reductions of up to 60% over the standard binary classification approach.

Introduction

Record deduplication is the problem of deciding whether two records in a database refer to the same object. This problem is widespread in any large-scale database, and is particularly acute when records are constructed automatically from text mining.

Author disambiguation, the problem of deduplicating author records, is a critical concern for digital publication libraries such as Citeseer, DBLP, Rexa, and Google Scholar. Author disambiguation is difficult in these domains because of abbreviations (e.g., *Y. Smith*) misspellings (e.g., *Y. Smiht*), and extraction errors (e.g., *Smith Statistical*).

Many supervised machine learning approaches to author disambiguation have been proposed. Most of these are variants of the following recipe: (1) train a binary classifier to predict whether a pair of authors are duplicates, (2) apply the classifier to each pair of ambiguous authors, (3) combine the classification predictions to cluster the records into duplicate sets.

This approach can be quite accurate, and is attractive because it builds upon existing machine learning technology (e.g., classification and clustering). However, because the core of this approach is a classifier over record pairs, nowhere are aggregate features of an author modeled explicitly. That is, by restricting the model representation to evidence over pairs of authors, we cannot leverage evidence available from examining more than two records.

For example, we would like to model the fact that authors are generally affiliated with only a few institutions, have only one or two different email addresses, and are unlikely to publish more than thirty publications in one year. None of these constraints can be captured with a pairwise classifier, which, for example, can only consider whether pairs of institutions or emails match.

In this paper we propose a representation for author disambiguation that enables these aggregate constraints. The representation can be understood as a scoring function over a *set* of authors, indicating how likely it is that all members of the set are duplicates.

While flexible, this new representation can make it difficult to estimate the model parameters from training data. We therefore propose a class of training algorithms to estimate the parameters of models adopting this representation. The method has two main characteristics essential to its performance. First, it is *error-driven* in that training examples are generated based on mistakes made by the prediction algorithm. This approach focuses training effort on the types of examples expected during prediction.

Second, it is *rank-based* in that the loss function induces a *ranking* over candidate predictions. By representing the *difference* between predictions, preferences can be expressed over partially-correct or incomplete solutions; additionally, intractable normalization constants can be avoided because the loss function is a ratio of terms.

In the following sections, we describe the representation in more detail, then present our proposed training methods. We evaluate our proposals on three real-world author deduplication datasets and demonstrate error-reductions of up to 60%.

Author	Title	Institution	Year
Y. Li	Understanding Social Networks	Stanford	2003
Y. Li	Understanding Network Protocols	Carnegie Mellon	2002
Y. Li	Virtual Network Protocols	Peking Univ.	2001

Table 1: Author disambiguation example with multiple institutions.

Author	Co-authors	Title
P. Cohen	A. Howe	How evaluation guides AI research
P. Cohen	M. Greenberg, A. Howe, ...	Trial by Fire: Understanding the design requirements ... in complex environments
P. Cohen	M. Greenberg	MU: a development environment for prospective reasoning systems

Table 2: Author disambiguation example with overlapping co-authors.

Motivating Examples

Table 1 shows three synthetic publication records that demonstrate the difficulty of author disambiguation. Each record contains equivalent author strings, and the publication titles contains similar words (*networks, understanding, protocols*). However, only the last two authors are duplicates.

Consider a binary classifier that predicts whether a pair of records are duplicates. Features may include the similarity of the author, title, and institution strings. Given a labeled dataset, the classifier may learn that authors often have the same institution, but since many authors have multiple institutions, the pairwise classifier may still predict that all of the authors in Table 1 are duplicates.

Consider instead a scoring function that considers all records simultaneously. For example, this function can compute a feature indicating that an author is affiliated with three different institutions in a three year period. Given training data in which this event is rare, it is likely that the classifier would not predict all the authors in Table 1 to be duplicates.

Table 2 shows an example in which co-author information is available. It is very likely that two authors with similar names that share a co-author are duplicates. However, a pairwise classifier will compute that records one and three do not share co-authors, and therefore may not predict that all of these records are duplicates. (A post-processing clustering method may merge the records together through transitivity, but only if the aggregation of the pairwise predictions is sufficiently high.)

A scoring function that considers all records simultaneously can capture the fact that records one and three each share a coauthor with record two, and are therefore all likely duplicates.

In the following section, we formalize this intuition, then describe how to estimate the parameters of such a representation.

Scoring Functions for Disambiguation

Consider a publications database D containing records $\{R_1 \dots R_n\}$. A record R_i consists of k fields $\{F_1 \dots F_k\}$, where each field is an attribute-value pair $F_j = \langle \textit{attribute, value} \rangle$. *Author disambiguation* is the problem of partitioning $\{R_1 \dots R_n\}$ into m sets $\{\mathcal{A}_1 \dots \mathcal{A}_m\}$, $m \leq n$, where $\mathcal{A}_l = \{R_j \dots R_k\}$ contains all the publications authored by the lth person. We refer to a partitioning of D as $T(D) = \{\mathcal{A}_1 \dots \mathcal{A}_m\}$, which we will abbreviate as T.

Given some partitioning T, we wish to learn a scoring function $S : T \mapsto \mathcal{R}$ such that higher values of $S(T)$ correspond to more accurate partitionings. Author disambiguation is then the problem of searching for the highest scoring partitioning:

$$T^* = \operatorname*{argmax}_T S(T)$$

The structure of S determines its representational power. There is a trade-off between the types of evidence we can use to compute $S(T)$ and the difficulty of estimating the parameters of $S(T)$. Below, we describe a *pairwise scoring function*, which decomposes $S(T)$ into a sum of scores for record pairs, and a *clusterwise scoring function*, which decomposes $S(T)$ into a sum of scores for record clusters.

Let T be decomposed into p (possibly overlapping) substructures $\{t_1 \dots t_p\}$, where $t_i \subset T$ indicates that t_i is a substructure of T. For example, a partitioning T may be decomposed into a set of record pairs R_i, R_j.

Let $f : t \to \mathcal{R}^k$ be a *substructure feature function* that summarizes t with k real-valued features[1].

Let $s : f(t) \times \Lambda \to \mathcal{R}$ be a *substructure scoring function* that maps features of substructure t to a real value, where $\Lambda \in \mathcal{R}^k$ is a set of real-valued parameters of s. For example, a linear substructure scoring function simply returns the inner product $\langle \Lambda, f(t) \rangle$.

Let $S_f : s(f(t_1), \Lambda) \times \ldots \times s(f(t_n), \Lambda) \to \mathcal{R}$ be a *factored scoring function* that combines a set of substructure scores into a global score for T. In the simplest case, S_f may simply be the sum of substructure scores.

Below we describe two scoring functions resulting from different choices for the substructures t.

[1]Binary features are common as well: $f : t \to \{0, 1\}^k$

Pairwise Scoring Function

Given a partitioning T, let t^{ij} represent a pair of records R_i, R_j. We define the *pairwise scoring function* as

$$S_p(T, \Lambda) = \sum_{ij} s(f(t^{ij}), \Lambda)$$

Thus, $S_p(T)$ is a sum of scores for each pair of records. Each component score $s(f(t^{ij}), \Lambda)$ indicates the preference for the prediction that records R_i and R_j co-refer.

This is analogous to approaches adopted recently using binary classifiers to perform disambiguation (Torvik *et al.* 2005; Huang, Ertekin, & Giles 2006).

Clusterwise Scoring Function

Given a partitioning T, let $t^{\mathbf{k}}$ represent a set of records $\{R_i \dots R_j\}$ (e.g., $t^{\mathbf{k}}$ is a block of the partition). We define the *clusterwise scoring function* as the sum of scores for each cluster:

$$S_c(T, \Lambda) = \sum_{\mathbf{k}} s(f(t^{\mathbf{k}}), \Lambda)$$

where each component score $s(f(t^{\mathbf{k}}, \Lambda))$ indicates the preference for the prediction that all the elements $\{R_i \dots R_j\}$ co-refer.

Learning Scoring Functions

Given some training database D^T for which the true author disambiguation is known, we wish to estimate the parameters of S to maximize expected disambiguation performance on new, unseen databases. Below, we outline approaches to estimate Λ for pairwise and clusterwise scoring functions.

Pairwise Classification Training

A standard approach to train a pairwise scoring function is to generate a training set consisting of all pairs of authors (if this is impractical, one can prune the set to only those pairs that share a minimal amount of surface similarity). A classifier is estimated from this data to predict the binary label *SameAuthor*.

Once the classifier is created, we can set each substructure score $s(f(t^{ij}))$ as follows: Let $p_1 = P(SameAuthor = 1 | R_i, R_j)$. Then the score is

$$s(f(t^{ij})) \propto \begin{cases} p_1 & \text{if } R_i, R_j \in T \\ 1 - p & \text{otherwise} \end{cases}$$

Thus, if R_i, R_j are placed in the same partition in T, then the score is proportional to the classifier output for the positive label; else, the score is proportional to output for the negative label.

Clusterwise Classification Training

The pairwise classification scheme forces each coreference decision to be made independently of all others. Instead, the clusterwise classification scheme implements a form of the clusterwise scoring function described earlier. A binary classifier is built that predicts whether all members of a set of author records $\{R_i \cdots R_j\}$ refer to the same person. The scoring function is then constructed in a manner analogous to the pairwise scheme, with the exception that the probability p_1 is conditional on an arbitrarily large set of mentions, and $s \propto p_1$ only if all members of the set fall in the same block of T.

Algorithm 1 Error-driven Training Algorithm

```
Input:
  Training set D
  Initial parameters Λ^0
  Prediction algorithm A
while Not Converged do
  for all ⟨X, T*(X)⟩ ∈ D do
    T(X) ⇐ A(X, Λ^t)
    D_e ⇐ CreateExamplesFromErrors(T(X), T*(X))
    Λ^{t+1} ⇐ UpdateParameters(D_e, Λ^t)
  end for
end while
```

Error-driven Online Training

We employ a sampling scheme that selects training examples based on errors that occur during inference on the labeled training data. For example, if inference is performed with agglomerative clustering, the first time that two non-coreferent clusters are merged, the features that describe that merge decision are used to update the parameters.

Let $\mathcal{A}$ be a prediction algorithm that computes a sequence of predictions, i.e., $\mathcal{A} : X \times \Lambda \rightarrow T^0(X) \times \ldots \times T^r(X)$, where $T^r(X)$ is the final prediction of the algorithm. For example, $\mathcal{A}$ could be a clustering algorithm. Algorithm 1 gives high-level pseudo-code of the description of the error-driven framework..

At each iteration, we enumerate over the training examples in the original training set. For each example, we run $\mathcal{A}$ with the current parameter vector Λ^t to generate $\mathbf{T}(X)$, a sequence of predicted structures for X.

In general, the function *CreateExamplesFromErrors* can select an arbitrary number of errors contained in $\mathbf{T}(X)$. In this paper, we select only the *first* mistake in $\mathbf{T}(X)$. When the prediction algorithm is computationally intensive, this greatly increases efficiency, since inference is terminated as soon as an error is made.

Given $\mathcal{D}_e$, the parameters Λ^{t+1} are set based on the errors made using Λ^t. In the following section, we describe the nature of $\mathcal{D}_e$ in more detail, and present a ranking-based method to calculate Λ^t.

Learning To Rank

An important consequence of using a search-based clustering algorithm is that the scoring function is used to *compare* a set of possible modifications to the current prediction. Given a clustering T^i, let $\mathcal{N}(T^i)$ be the set of predictions in the *neighborhood* of T^i. $T^{i+1} \in \mathcal{N}(T^i)$ if the prediction algorithm can construct T^{i+1} from T^i

in one iteration. For example, at each iteration of the greedy agglomerative system, $\mathcal{N}(T^i)$ is the set of clusterings resulting from all possible merges a pair of clusters in T^i. We desire a training method that will encourage the inference procedure to choose the best possible neighbor at each iteration.

Let $\hat{N}(T) \in \mathcal{N}(T)$ be the neighbor of T that has the maximum *predicted* global score, i.e., $\hat{N}(T) = \text{argmax}_{T' \in \mathcal{N}(T)} S_g(T')$. Let $S^* : T \rightarrow \mathcal{R}$ be a global scoring function that returns the *true* score for an object, for example the accuracy of prediction T.

Given this notation, we can now fully describe the method *CreateExamplesFromErrors* in Algorithm 1. An *error* occurs when there exists a structure $N^*(T)$ such that $S^*(N^*(T)) > S^*(\hat{N}(T))$. That is, the best predicted structure $\hat{N}(T)$ has a lower *true* score than another candidate structure $N^*(T)$.

A training example $\langle \hat{N}(T), N^*(T) \rangle$ is generated[2], and a loss function is computed to adjust the parameters to encourage $N^*(T)$ to have a higher predicted score than $\hat{N}(T)$. Below we describe two such loss functions.

Ranking Perceptron The perceptron update is

$$\Lambda^{t+1} = \Lambda^t + y \cdot F(T)$$

where $y = 1$ if $T = N^*(T)$, and $y = -1$ otherwise.

This is the standard perceptron update (Freund & Schapire 1999), but in this context it results in a ranking update. The update compares a pair of $F(T)$ vectors, one for $\hat{N}(T)$ and one for $N^*(T)$. Thus, the update actually operates on the *difference* between these two vectors. Note that for robustness we average the parameters from each iteration at the end of training.

Ranking MIRA We use a variant of MIRA (Margin Infused Relaxed Algorithm), a relaxed, online maximum margin training algorithm (Crammer & Singer 2003). We updates the parameter vector with three constraints: (1) the better neighbor must have a higher score by a given margin, (2) the change to Λ should be minimal, and (3) the inferior neighbor must have a score below a user-defined threshold τ (0.5 in our experiments). The second constraint is to reduce fluctuations in Λ. This optimization is solved through the following quadratic program:

$$\Lambda^{t+1} = \underset{\Lambda}{\text{argmin}} \, ||\Lambda^t - \Lambda||^2 \text{ s.t.}$$

$$S(N^*(T), \Lambda) - S(\hat{N}(T), \Lambda) \geq 1$$

$$S(\hat{N}, \Lambda) < \tau$$

The quadratic program of MIRA is a norm-minimization that is efficiently solved by the method of Hildreth (Censor & Zenios 1997). As in perceptron, we average the parameters from each iteration.

[2]While in general $\mathcal{D}_e$ can contain the entire neighborhood $\mathcal{N}(T)$, in this paper we restrict $\mathcal{D}_e$ to contain only two structures, the incorrectly predicted neighbor and the neighbor that should have been selected.

Experiments

Data

We used three datasets for evaluation:

- **Penn:** 2021 citations, 139 unique authors
- **Rexa:** 1459 citations, 289 unique authors
- **DBLP:** 566 citations, 76 unique authors

Each dataset contains multiple sets of citations authored by people with the same last name and first initial. We split the data into training and testing sets such that all authors with the same first initial and last name are in the same split.

The features used for our experiments are as follows. We use the first and middle names of the author in question and the number of overlapping co-authors. We determine the rarity of the last name of the author in question using US census data. We use several different similarity measures on the title of the two citations such as the cosine similarity between the words, string edit distance, TF-IDF measure and the number of overlapping bigrams and trigrams. We also look for similarity in author emails, institution affiliation and the venue of publication whenever available. In addition to these, we also use the following first-order features over these pairwise features: For real-valued features, we compute their minimum, maximum and average values; for binary-valued features, we calculate the proportion of pairs for which they are true, and also compute existential and universal operators (e.g., "there exist a pair of authors with mismatching middle initials").

Results

Table 3 summarizes the various systems compared in our experiments. The goal is to determine the effectiveness of the clusterwise scoring functions, error-driven example generation, and rank-based training. In all experiments, prediction is performed with greedy agglomerative clustering.

We evaluate performance using three popular measures: **Pairwise**, the precision and recall for each pairwise decision; **MUC** (Vilain *et al.* 1995), a metric commonly used in noun coreference resolution; and **B-Cubed** (Amit & Baldwin 1998).

Tables 4, 5, and 6 present the performance on the three different datasets. Note that the accuracy varies significantly between datasets because each has quite different characteristics (e.g., different distributions of papers per unique author) and different available attributes (e.g., institutions, emails, etc.).

The first observation is that the simplest method of estimating the parameters of the clusterwise score performs quite poorly. $C/U/L$ trains the clusterwise score by uniformly sampling sets of authors and training a binary classifier to indicate whether all authors are duplicates. This performs consistently worse than $P/A/L$, which is the standard pairwise classifier.

We next consider improvements to the training algorithm. The first enhancement is to perform error-driven

Component	Name	Description
Score Representation	Pairwise (P)	See Section **Pairwise Scoring Function**.
	Clusterwise (C)	See Section **Clusterwise Scoring Function**.
Training Example Generation	Error-Driven (E)	See Section **Error-driven Online Training**.
	Uniform (U)	Examples are sampled u.a.r. from search trees.
	All-Pairs(A)	All positive and negative pairs
Loss Function	Ranking MIRA (M_r)	See Section **Ranking MIRA**.
	Non-Ranking Mira (M_c)	MIRA trained for classification, not ranking.
	Ranking Perceptron (P_r)	See Section **Ranking Perceptron**.
	Non-Ranking Perceptron (P_c)	Perceptron for classification, not ranking.
	Logistic Regression(L)	Standard Logistic Regression.

Table 3: Description of the various system components used in the experiments.

	Pairwise			**B-Cubed**			**MUC**		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
$C/E/M_r$	36.0	96.0	22.1	48.8	97.9	32.5	79.8	99.2	66.8
$C/E/M_c$	24.4	99.2	13.9	35.8	98.7	21.8	71.2	98.3	55.8
$C/E/P_r$	52.0	77.9	39.0	63.8	84.1	51.4	88.6	94.5	83.4
$C/E/P_c$	40.3	99.9	25.3	52.6	99.6	35.7	81.5	99.6	68.9
$C/U/L$	36.2	74.8	23.9	46.1	80.6	32.2	80.4	89.6	73.0
$P/A/L$	44.9	96.8	29.3	56.0	95.0	39.7	87.2	96.4	79.5

Table 4: Author Coreference results on the Penn data. See Table 3 for the definitions of each system.

training. By comparing $C/U/L$ with $C/E/P_c$ (the error-driven perceptron classifier) and $C/E/M_c$ (the error-driven MIRA classifier), we can see that performing error-driven training often improves performance. For example, in Table 6, we see pairwise F1 increase from 82.4 for $C/U/L$ to 93.1 for $C/E/P_c$ and to 91.9 for $C/E/M_c$. However, this improvement is not consistent across all datasets. Indeed, simply using error-driven training is not enough to ensure accurate performance for the clusterwise score.

The second enhancement is to perform a rank-based parameter update. With this additional enhancement, the clusterwise score consistently outperforms the pairwise score. For example, in Table 5, $C/E/P_r$ obtains nearly a 60% reduction in pairwise F1 error over the pairwise scorer $P/A/L$. Similarly, in Table 5, $C/E/M_r$ obtains a 35% reduction in pairwise F1 error $P/A/L$. While perceptron does well on most of the datasets, it performs poorly on the DBLP data ($C/E/P_R$, Table 6). Because the perceptron update does not constrain the incorrect prediction to be below the classification threshold, the resulting clustering algorithm can over-merge authors. MIRA's additional constraint ($S(\hat{N}, \Lambda) < \tau$) addresses this issue.

In conclusion, these results indicate that simply increasing representational power by using a clusterwise scoring function may not result in improved performance unless appropriate parameter estimation methods are used. The experiments on these three datasets suggest that error-driven, rank-based estimation is an effective method to train a clusterwise scoring function.

Related Work

There has been a considerable interest in the problem of author disambiguation (Etzioni *et al.* 2004; Dong *et al.* 2004; Han *et al.* 2004; Torvik *et al.* 2005; Kanani, McCallum, & Pal 2007); most approaches perform pairwise classification followed by clustering. Han, Zha, & Giles (2005) use spectral clustering to partition the data. More recently, Huang, Ertekin, & Giles (2006) use SVM to learn similarity metric, along with a version of the DBScan clustering algorithm. Unfortunately, we are unable to perform a fair comparison with their method as the data is not yet publicly unavailable. On *et al.* (2005) present a comparative study using coauthor and string similarity features. Bhattacharya & Getoor (2006) show suprisingly good results using unsupervised learning.

There has also been a recent interest in training methods that enable the use of global scoring functions. Perhaps the most related is "learning as search optimization" (LaSO) (Daumé III & Marcu 2005). Like the current paper, LaSO is also an error-driven training method that integrates prediction and training. However, whereas we explicitly use a ranking-based loss function, LaSO uses a binary classification loss function that labels each candidate structure as *correct* or *incorrect*. Thus, each LaSO training example contains *all* candidate predictions, whereas our training examples contain only the highest scoring incorrect prediction and the highest scoring correct prediction. Our experiments show the advantages of this ranking-based loss function. Additionally, we provide an empirical study to quantify the effects of different example generation and loss function decisions.

	Pairwise			B-Cubed			MUC		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
$C/E/M_r$	74.1	86.3	65.0	78.0	94.6	66.4	82.7	98.0	71. 5
$C/E/M_c$	39.4	98.1	24.7	59.3	96.6	42.8	72.7	96.7	58.2
$C/E/P_r$	86.4	87.4	85.5	81.8	81.6	82.0	88.8	87.4	90.2
$C/E/P_c$	49.5	87.2	34.5	65.8	94.6	50.4	78.3	96.2	66.0
$C/U/L$	45.0	87.3	30.3	67.2	86.9	54.8	82.0	87.8	76.4
$P/A/L$	66.2	72.4	61.0	75.7	75.5	76.0	88.6	85.3	92.2

Table 5: Author Coreference results on the Rexa data. See Table 3 for the definitions of each system.

	Pairwise			B-Cubed			MUC		
	F1	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**
$C/E/M_r$	92.2	94.2	90.2	89.0	94.4	84.2	93.5	98.5	89.0
$C/E/M_c$	91.9	90.6	93.2	87.6	94.8	81.5	90.7	98.5	84.1
$C/E/P_r$	45.3	29.4	99.4	72.9	57.8	98.8	94.2	89.3	99.6
$C/E/P_c$	93.1	91.0	95.3	90.6	92.0	89.3	94.3	97.2	91.6
$C/U/L$	82.4	95.7	72.3	83.2	93.0	75.3	93.1	96.2	90.3
$P/A/L$	88.0	84.6	91.7	86.1	84.6	87.8	93.0	93.0	93.0

Table 6: Author Coreference results on the DBLP data. See Table 3 for the definitions of each system.

Conclusions and Future Work

We have proposed a more flexible representation for author disambiguation models and described parameter estimation methods tailored for this new representation. We have performed empirical analysis of these methods on three real-world datasets, and the experiments support our claims that error-driven, rank-based training of the new representation can improve accuracy. In future work, we plan to investigate more sophisticated prediction algorithms that alleviate the greediness of local search, and also consider representations using features over entire clusterings.

Acknowledgements

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract #NBCHD030010, in part by U.S. Government contract #NBCH040171 through a subcontract with BBNT Solutions LLC, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, in part by Microsoft Live Labs, and in part by the Defense Advanced Research Projects Agency (DARPA) under contract #HR0011-06-C-0023.0

References

Amit, B., and Baldwin, B. 1998. Algorithms for scoring coreference chains. In *Proceedings of MUC7*.

Bhattacharya, I., and Getoor, L. 2006. A latent dirichlet model for unsupervised entity resolution. In *SDM*.

Censor, Y., and Zenios, S. 1997. *Parallel optimization: theory, algorithms, and applications*. Oxford University Press.

Crammer, K., and Singer, Y. 2003. Ultraconservative online algorithms for multiclass problems. *JMLR* 3:951–991.

Daumé III, H., and Marcu, D. 2005. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*.

Dong, X.; Halevy, A. Y.; Nemes, E.; Sigurdsson, S. B.; and Domingos, P. 2004. Semex: Toward on-the-fly personal information integration. In *IIWEB*.

Etzioni, O.; Cafarella, M.; Downey, D.; Kok, S.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.; and Yates, A. 2004. Web-scale information extraction in KnowItAll. In *WWW*. ACM.

Freund, Y., and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. *Machine Learning* 37(3):277–296.

Han, H.; Giles, L.; Zha, H.; Li, C.; and Tsioutsiouliklis, K. 2004. Two supervised learning approaches for name disambiguation in author citations. In *JCDL*, 296–305. ACM Press.

Han, H.; Zha, H.; and Giles, L. 2005. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL*.

Huang, J.; Ertekin, S.; and Giles, C. L. 2006. Efficient name disambiguation for large-scale databases. In *PKDD*, 536–544.

Kanani, P.; McCallum, A.; and Pal, C. 2007. Improving author coreference by resource-bounded information gathering from the web. In *Proceedings of IJCAI*.

On, B.-W.; Lee, D.; Kang, J.; and Mitra, P. 2005. Comparative study of name disambiguation problem using a scalable blocking-based framework. In *JCDL*, 344–353. New York, NY, USA: ACM Press.

Torvik, V. I.; Weeber, M.; Swanson, D. R.; and 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.

Vilain, M.; Burger, J.; Aberdeen, J.; Connolly, D.; and Hirschman, L. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of MUC6*, 45–52.