Canonicalization of Database Records using Adaptive Similarity Measures

Aron Culotta
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Michael Wick
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Robert Hall
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Matthew Marzilli
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Andrew McCallum
Department of Computer Science
University of Massachusetts
Amherst, MA 01003

ABSTRACT

It is becoming increasingly common to construct databases from information automatically culled from many heterogeneous sources. For example, a research publication database can be constructed by automatically extracting titles, authors, and conference information from papers and their references. A common difficulty in consolidating data from multiple sources is that records are referenced in a variety of ways (e.g. abbreviations, aliases, and misspellings). Therefore, it can be difficult to construct a single, standard representation to present to the user. We refer to the task of constructing this representation as *canonicalization*. Despite its importance, there is very little existing work on canonicalization.

In this paper, we explore the use of edit distance measures to construct a canonical representation that is "central" in the sense that it is most similar to each of the disparate records. This approach reduces the impact of noisy records on the canonical representation. Furthermore, because the user may prefer different *styles* of canonicalization, we show how different edit distance costs can result in different forms of canonicalization. For example, reducing the cost of character deletions can result in representations that favor abbreviated forms over expanded forms (e.g. *KDD* versus *Conference on Knowledge Discovery and Data Mining*). We describe how to learn these costs from a small amount of manually annotated data using stochastic hill-climbing. Additionally, we investigate feature-based methods to learn ranking preferences over canonicalizations. We empirically evaluate our approach on a real-world publications database and show that our learning method results in a canonicalization solution that is robust to errors and easily customizable to user preferences.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management; H.2.8 [**Information Systems**]: Database Applications—*data mining*

General Terms

Algorithms

Keywords

Data mining, information extraction, data cleaning

1. INTRODUCTION

Record canonicalization is the problem of constructing a single, standard representation for a record. In many databases, canonicalization is enforced with a set of rules that place limitations or set guidelines for entering data. However, in many cases, these limitations are overlooked and it is necessary to retroactively canonicalize records. Additionally, such rules are not applicable to cases where the database contains records extracted automatically from unstructured or varying sources.

Consider a research database such as Citeseer[1] or Rexa[2] that contains information gathered from a variety of sources using automated extraction techniques. Because the data is coming from a multiple sources, it is inevitable that an attribute such as a conference name will be referenced in multiple ways. Since the data is also the result of extraction, it may also contain errors.

One might propose the reasonable solution of selecting the record that is most common in the set. However, one caveat of this approach is that often incomplete records are more

[1] www.citeseer.ist.psu.edu
[2] www.rexa.info

common than complete records. For example, this approach may canonicalizing a record as "J. Smith" when in fact the full name (John Smith) is much more desirable. Therefore, we propose a canonicalization method that selects the string or record that is most similar to all the others in the set. This way, the full name may be selected even when outnumbered by incomplete representations.

In this paper we propose a system for automatically canonicalizing records for databases that are not only derived from a large number of sources, but contain extraction errors. We desire that the algorithms produce consistent results. If a particular format is chosen as the canonical representation for one record, an analogous format is chosen for another. We demonstrate the ability to adhere to formats specified by a user's preference while alleviating the burden of communicating this choice to the system. We propose several methods for performing canonicalization based on (1) learning the parameters for string edit distance and (2) combining multiple string edit models with additional evidence by using a discriminative models (rank based maximum entropy and Margin Infused Relaxed Algorithm (MIRA)). Our results are encouraging and demonstrate that an automatic canonicalizer can be learned from relatively few training examples and be robust to the errors incurred in automatic extraction.

2. RELATED WORK

While there has not been extensive research in automatically constructing canonicalized representations, the utility of canonicalization has been explored in several papers under various guises. [8] use a discriminatively learned edit distance to perform deduplication on redundant data sets. In the task of fact extraction, [7] demonstrates the advantage of "voting", the process of selecting the fact that appears in the largest number of sources. While works such as these may not specifically refer to the problem, the underlying ideas of canonicalization are pervasive throughout many applications. In this section we review several of these applications as well as related work in the area of learning string edit distance costs.

[10]devise a system to automatically extract and consolidate information describing objects derived from multiple sources into unified database records. When a user queries this database, multiple representations of an attribute are inevitable due to naming inconsistencies across the various sources from which they were drawn. Although object deduplication is the primary goal of the this research, canonicalization arises and is addressed when the system presents results to the user. The authors address this problem by ranking the strings for each attribute based on the user's preference for the source from the particular string was extracted.

One difficulty in this approach is that if the data is extracted from a large number of sources, a non-trivial burden is placed on the users, who may not have the expertise or lack the knowledge necessary to make preference choices about each source. Additionally, the database must store source-specific meta information for each string of each attribute of each record. In general, this information may not be available, but even when available, it may be difficult to justify the large quantity of extraneous space required for the sole purpose canonicalization. Our canonicalization methods are able to be adapted to any database, regardless of whether the source of the information is available.

Other work has focused on learning the parameters of string edit distance with encouraging results. [11] apply string edit distances to the task of merging databases. They observe that parameters cannot be optimized individually due to the complex interaction between various edit cost weights on the outcome. Additionally they note that greedy methods are too likely to converge prematurely in local optima and that random restarts are unnecessarily expensive. Instead they propose a genetic algorithm to learn the weights of each cost and find that it stabilizes after 100 generations. In lieu of genetic approaches we propose learning the edit costs using either stochastic search, or an exhaustive search over a relatively small discrete space of possible parameter settings.

[9] learn a probability distribution over the atomic string edit operations (insertion, deletion, substitution) and define a stochastic transducer that defines the probability of a the string as either the Viterbi sequence of edit operations or the sum of all possible sequences of edit operations required to produce that string. The parameters of this generative model are learned using the expectation maximization (EM) algorithm.

[1] presents a method to learn edit distance based similarity measures of each attribute between records in order to perform deduplication. They extend the work of [9] by accommodating affine gaps. In similar fashion, the weights are learned iteratively with EM.

Similar to [8] our learning methods differ from those outlined in [9, 1] in that we are not concerned with learning a generative model. We propose several methods for learning edit distance parameter settings including stochastic and exhaustive search. Additionally, we combine the outputs of multiple parameter settings (i.e., multiple edit distance models) and other sources of evidence by learning a discriminative model over pairs of strings with rank based maximum entropy and an online MIRA.

Recently, sophisticated frameworks have been devised to handle uncertainty in databases, particularly those generated from automatically extracted records. Such systems store the top n most confident extraction results (along with corresponding probabilities or confidence measures) for each desired record leading to multiple candidate representations for each entity. [4] leverage confidence value outputs from the extraction models to improve query results on databases containing uncertainty. Fundamentally the problem is canonicalization because the system is faced with a choice when presenting multiple query results with various confidence values to the user. It is analogous to our canonicalization task except that we do not have the luxury of confidence values. While the inclusion of such values is clearly beneficial, we propose methods that achieve canonicalization in absence of such information (and often this information is strictly unavailable).

3. PROBLEM DEFINITION

Let a record R be a set of fields, $R = \{F_1 \dots F_k\}$. Let field F_i be an attribute-value pair $\langle a, v \rangle$. Table 1 shows three example records.

Systems that compile records from a variety of sources often accumulate multiple versions of the same record. The problem of detecting these different versions is called *record deduplication*. For example, Table 1 shows three records that have been predicted to be duplicates. In fact, record **(c)** is a reference to a book chapter version of the paper, whereas **(a)** and **(b)** refer to conference proceedings.

Record deduplication is a difficult problem that has been well-studied. However, in this paper we are interested in a subsequent step: how to present the user one *canonical* representation of a record with many versions.

We define the *canonicalization* problem as follows: Given a set of duplicate records $\mathbf{R} = \{R_1 \dots R_k\}$, create a *canonical* record R^* that summarizes the information in $\mathbf{R}$. We refer to the canonicalization operation as $C(\mathbf{R})$

Note that it is not always clear what the optimal canonicalized record should be. Indeed, different users may prefer different forms of canonicalization. For example, some users may prefer the abbreviated conference string *IJCAI*, while others may prefer the expanded string *International Joint Conference on Artificial Intelligence*. However, there are a few desiderata of a good canonicalization:

- **Error-free**: The canonical record should not contain errors, such as misspellings or incorrect field values. This is especially a concern when the data has been automatically extracted from noisy sources (e.g. when the source is OCR text and field assignments are automated). In these cases, there may exist *outlying* records that contain erroneous data. The canonicalizer should attempt to minimize the effect of these errors.
- **Complete**: The canonical record should contain all the accurate information contained in the duplicate records. Thus, even if all records do not contain a *date* field, the field should be included in the canonical record.
- **Representative:** The canonical record should reflect the commonality among the duplicate records. Thus, the canonical record should in some sense be *similar* to all of the duplicate records.

4. THREE CLASSES OF CANONICALIZATION SOLUTIONS

We now outline three classes of canonicalization solutions, in increasing order of ambition.

4.1 Record Selection

The *record selection* approach to canonicalization selects an existing record as its output. For example, $C(\mathbf{R})$ must select from the three records in Table 1. Record selection algorithms must ensure that the selected record contains no errors, and that is is representative of other records. Note that this approach is most prone to errors of incompleteness, since one record may not contain all the fields present in the duplicates. For example, selecting record **(a)** in Table 1 will omit the page numbers, but selecting record **(b)** will omit the full first name of the author.

4.2 Record Merging

The *record merging* approach to canonicalization constructs a canonical record by piecing together fields from different records.

While this approach can increase the *completeness* of canonicalization, it does so at the risk of introducing errors. In the worst case, an error in record deduplication may merge together records that in fact refer to different objects. Constructing one record containing fields from these non-duplicate records can result in a canonical record containing invalid information. For example, a record merging approach may return the following record:

author	Brian Milch et al.
title	BLOG: Probabilistic Models with Unknown Objects
venue	Intl. Conf. on AI
editor	**L. Getoor and B. Taskar**
pages	1352-1359

While this result is complete, it erroneously includes the editor field from record **(c)**, which is not truly a duplicate.

4.3 Record Generation

The *record generation* approach to canonicalization is an extension of the record merging approach that may also posit field values that do not explicitly exist in any of the record duplicates.

For example, a record generation approach may return the following record:

author	Brian Milch et al.
title	BLOG: Probabilistic Models with Unknown Objects
venue	**International Joint Conference on Artificial Intelligence**
editor	
pages	1352-1359

Here, the system has generated an expanded venue value from the abbreviated form, even though this expanded form does not appear among the duplicate records. This predictive operation can be accomplished either by learning statistical patterns in the database, or by a pattern-matching approach.

While in this case record generation succeeded, in general positing field values that do not exist in any of the records can be quite dangerous and lead to unacceptable errors.

author	Brian Milch et al.
title	BLOG: Probabilistic Models with Unknown Objects
venue	IJCAI
editor	
pages	

(a)

author	B. Milch et al.
title	BLOG: Probabilistic Models with Unknown Objects
venue	Intl. Conf. on AI
editor	
pages	1352-1359

(b)

author	Brian Milch et al.
title	BLOG: Probabilistic Models with Unknown Objects
venue	
editor	L. Getoor and B. Taskar
pages	

(c)

Table 1: Three publication records predicted to be duplicates. Note that a de-duplication error has erroneously merged record (c) (a book chapter) with the other two conference papers. De-duplication errors, as well as misspellings, abbreviations, and aliases, can make canonicalization difficult.

These three solution classes motivate a number of implementations and experiments. In this paper, we describe our implementation of a record selection method and perform experiments to measure its effectiveness.

5. THREE PROPOSALS FOR RECORD SELECTION CANONICALIZATION

5.1 Edit distance with fixed costs

The motivation for our approach is to minimize the effect of pre-processing errors on canonicalization. As we have described, errors from OCR, misspellings, and incorrect deduplication can lead to poor canonicalization choices.

We make two assumptions about the behavior of pre-processing errors:

- Correct records are more common than incorrect records. That is, most records are error-free.
- Errors have high variance. For example, it is unlikely for many records to have the same exact spelling mistake.

With these assumptions in mind, we propose selecting the records that has the greatest average similarity to every other document. We define the distance between two records as the string edit distance between them.

Let $D : R_i \times R_j \mapsto \mathcal{R}$ be the edit distance between two records. Given a set of duplicate records $\mathbf{R} = \{R_1 \dots R_k\}$, we define the average edit distance of records R_i as

$$A(R_i) = \frac{\sum_{R_j \in \mathbf{R}} D(R_i, R_j)}{k} \tag{1}$$

The canonical record we return is the one with minimum average distance to every other string:

$$C^d(\mathbf{R}) = \underset{R_i \in \mathbf{R}}{\operatorname{argmin}} A(R_i)$$

We refer to $C^d(\mathbf{R})$ as the *edit distance canonicalizer*.

We now must decide on the form of D, the metric defining the distance between two strings. A natural choice is the Levenshtein distance: the number of character insertions, deletions, and replacements required to transform one string into another [5]. The recursive definition of the Levenshtein distance for strings s^n and t^m with length n and m is the following:

$$D(s^n, t^m) = \min \begin{cases} c_r(s_n, t_m) + D(s^{n-1}, t^{m-1}) \\ c_i + D(s^{n-1}, t^m) \\ c_d + D(s^n, t^{m-1}) \end{cases} \tag{2}$$

where $c_r(s_n, t_m)$ is the *replacement cost* for swapping character s_n with character t_m, c_i is the *insertion cost*, and c_d is the *deletion cost*. We can further define the replacement cost as

$$c_r(s_n, t_m) = \begin{cases} c_r^{\neq} & \text{if } s_n \neq t_m \\ c_r^{=} & \text{if } s_n = t_m \end{cases} \tag{3}$$

That is, $c_r^{\neq}$ is the cost of replacing one character with another, and $c_r^{=}$ is the cost of copying a character from one string to the next. We refer to $c_r^{\neq}$ as the *substitution cost*, and $c_r^{=}$ as the *copy cost*.

The value of the edit distance costs greatly effects the output of the system. For example, if c_i is small, then abbreviated strings will have a small distance to their expanded version. Abbreviated strings will therefore have lower values of $A(R_i)$.

Rather than requiring the user to manually tune these costs, in the next section we propose ways of learning these costs automatically given labeled examples.

5.2 Edit distance with learned costs

Suppose the user provides a labeled training set

$$\mathcal{S} = \{\langle \mathbf{R}^1, l_1 \rangle \dots \langle \mathbf{R}^n, l_n \rangle\}$$

where each set of duplicate records $\mathbf{R}^i = \{R_1 \dots R_k\}$ is annotated with labels $l_i \in \{1 \dots k\}$, indicating which of the duplicates should be selected as the canonical record (i.e., $R_{l_i} \in \mathbf{R}$ is the true canonical record). We wish to use $\mathcal{S}$ to learn the weights of D.

There has been a fair amount of work on methods to automatically learn edit distance costs, mostly applied to record de-duplication (See Section 2). However, we are not aware of any work that learns edit distance costs for canonicalization.

We propose two simple but effective methods to learn edit distance costs from training data: *exhaustive search* and *stochastic hill climbing*.

5.2.1 Exhaustive search

Algorithm 1 Exhaustive cost search

Input:
Training set $\mathcal{S}$
Initial costs $\mathbf{c} = \{c_i, c_d, c_r^{=}, c_r^{\neq}\}$
max, min, step
while More Costs **do**
  $\mathbf{c} \Leftarrow$ *NextCosts*($\mathbf{c}$, max, min, step)
  if $\mathcal{L}(\mathbf{c}, \mathcal{S})$ <bestLoss **then**
    bestLoss $\Leftarrow \mathcal{L}(\mathbf{c}, \mathcal{S})$
    $\mathbf{c}^* \Leftarrow \mathbf{c}$
  end if
end while

The simplest method is to exhaustively enumerate settings of each cost and maximize the canonicalization performance on the training set.

Let $\mathcal{L}(\mathbf{c}, \mathcal{S})$ be the loss function for an assignment to $\mathbf{c}$. For example, $\mathcal{L}$ may be the proportion of records in $\mathcal{S}$ for which $C^d(\mathbf{R})$ returns a non-canonical record; i.e. $C^d(\mathbf{R}^i) \neq R_{l_i}$.

We wish to optimze $\mathbf{c}$ as follows:

$$\mathbf{c}^* = \underset{\mathbf{c}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{c}, \mathcal{S}) \tag{4}$$

Since we must discretize the cost settings to perform exhaustive search, the input is the following:

- **min:** The minimum cost value
- **max:** The maximum cost value
- **step:** The amount to perturb each cost to obtain a new setting.

Search proceeds by simply cycling through each setting of $\mathbf{c}$ and returning the best setting. The details are given in Algorithm 1. The method *NextCosts* simply generates the next cost setting as determined by the step size.

5.2.2 Stochastic hill-climbing

Computing $\mathcal{L}(\mathbf{c}, \mathcal{S})$ requires computing $C^d(\mathbf{R})$ for all $\mathbf{R}^i \in \mathcal{S}$. This computational cost limits the number of settings we can enumerate using exhaustive search. Instead, we propose a simple stochastic hill-climbing algorithm to optimize Equation 4. Given an initial setting for $\mathbf{c}$, the algorithm proposes a modification to $\mathbf{c}$ and accepts the change if $\mathcal{L}(\mathbf{c}, \mathcal{S})$ decreases. This can be understood as simulated annealing without the *temperature* parameter. The details of this method are given in Algorithm 2.

The method *SampleCostElement* samples a cost uniformly from the cost vector. The method *RandomUpdate* uniformly chooses between incrementing or decrementing c by *step*.

5.3 Feature-based Ranking Models

While the adaptive edit distance approach to record selection can be simple and effective, it is limited by the small number of tunable parameters (the four costs), which limits the expressivity of the model.

Algorithm 2 Stochastic cost search

Input:
Training set $\mathcal{S}$
Initial costs $\mathbf{c} = \{c_i, c_d, c_r^{=}, c_r^{\neq}\}$
max, min, step
for $i <$*NumIterations* **do**
  $c \Leftarrow$ *SampleCostElement*($\mathbf{c}$)
  $c \Leftarrow$ *RandomUpdate*(c, step, max, min)
  $\mathbf{c} \Leftarrow$ *NextCosts*($\mathbf{c}$, max, min, step)
  if $\mathcal{L}(\mathbf{c}, \mathcal{S})$ <bestLoss **then**
    bestLoss $\Leftarrow \mathcal{L}(\mathbf{c}, \mathcal{S})$
    $\mathbf{c}^* \Leftarrow \mathbf{c}$
  end if
  $i \Leftarrow i + 1$
end for

In this section, we propose two feature-based learning approaches that enable the use of arbitrary features over the records, including the output of various edit distance measures.

Consider a set of duplicate records $\mathbf{R} = \{R_1 \dots R_k\}$. Let $F_i = \{f_1(R_i) \dots f_m(R_i)\}$ be a vector of *binary feature functions* that compute evidence indicating whether R_i should be selected as the canonical record. For example, $f_j(R_i)$ may be 1 if record R_i is the longest record in $\mathbf{R}$.

Let $\Lambda = \{\lambda_1 \dots \lambda_m\}$ be a vector of real-valued weights associated with each feature.

We can compute a score for the event that R_i is chosen as the canonical record by taking the dot product of the features and weights:

$$\tau(R_i, \Lambda) = F_i \cdot \Lambda$$

Below we describe two methods to estimate Λ from the training set $\mathcal{S}$.

5.3.1 Logistic Regression

The first method is to minimize the log-likelihood loss function of logistic regression, modified to reflect a ranking over records.

Let the binary random variable C_i be 1 if and only if record R_i is the true canonical record of $\mathbf{R}$. Given Λ and F, we can compute the probability of C_i as follows:

$$p(C_i|\mathbf{R}, \Lambda) = \frac{\tau(R_i, \Lambda)}{\sum_{R_j \in \mathbf{R}} \tau(R_j, \Lambda)}$$

where the score for record R_i is normalized by the scores for every other record.

We can estimate Λ from the training set $\mathcal{S}$ by minimizing the negative log-likelihood of the data given Λ:

$$\mathcal{L}(\Lambda, \mathcal{S}) = -\sum_{\mathbf{R}^i \in \mathcal{S}} \log p(C_{l_i}|\mathbf{R}, \Lambda) \tag{5}$$

Note that this is the sum of probabilities for each of the *correct* canonical records for the current setting of Λ. We also add a Gaussian prior over Λ with fixed mean and variance to mitigate over-fitting. We find the setting of Λ that minimizes Equation 5 using limited-memory BFGS, a gradient ascent method with a second-order approximation [6].

5.3.2 MIRA

MIRA (Margin Infused Relaxed Algorithm) is a relaxed, online maximum margin training algorithm [3]. It iteratively cycles through the training set and updates the parameter vector with two constraints: (1) the true canonical record must have a higher score than any other record by a given margin, and (2) the change to Λ should be minimal. This secondconstraint is to reduce fluctuations in Λ. Using the same scoring function τ as in the previous section, this optimization is solved through the following quadratic program:

$$\Lambda^{t+1} = \underset{\Lambda}{\operatorname{argmin}} ||\Lambda^t - \Lambda||^2$$

$$\text{s.t.}$$

$$\tau(R_{l_i}, \Lambda) - \tau(R_j, \Lambda) \geq 1 \quad \forall R_j \neq R_{l_i} \tag{6}$$

In this case, the MIRA parameter update is a quadratic program with constraint size equal to the number of non-canonical records in the training example. This QP can be solved efficiently using the Hildreth and D'esopo method [2]. To improve the stability of this online method, we average the parameter vectors from each update at the end of training.

6. EXPERIMENTS

6.1 Data

We collected 3,683 citations to 100 distinct papers from Rexa, an online publications search engine. These citations were automatically extracted from the headers of research papers as well as from the reference section, and record deduplication was performed automatically by Rexa. The data therefore contains misspellings, OCR errors, abbreviations, and possibly deduplication errors. To construct a labeled data set, we collect the corresponding citations to these papers from the Digital Bibliography and Library Project (DBLP)[3]. The DBLP citations are manually curated to ensure accuracy, so they provide a good source of canonical examples. In fact, as part of its pipeline, Rexa crawls the DBLP repository and performs record deduplication to merge citations together.

For these experiments, we focus on constructing the canonical representation of the conference string for each paper. This is arguably the most difficult field to canonicalize, since conferences strings appear in many different forms because of acronyms, abbreviations, and misspellings.

Using the DBLP data, we construct two versions of the dataset. In the first, the true canonicalization is the conference title acronym. This simulates the use case when the user desires abbreviated canonical forms. In the second version, the true canonicalization is the expanded conference title. This simulates the case when the user does not desire any abbreviations in the canonical form. We refer to the former version as the *acronym* dataset, and the latter as the *expanded* dataset.

Table 2 shows an example with labels from each of the datasets. We can see that the duplicate records contain a variety of abbreviated forms, as well as OCR errors in the first and fourth duplicate records (*Artifici al*,*Conferenceonarti cial*) that make canonicalization difficult.

[3]http://dblp.uni-trier.de

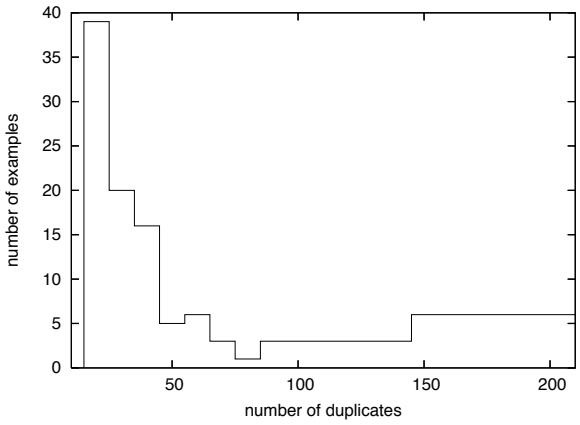

Figure 1: Distribution of number of duplicates per record.

Figure 1 shows the distribution of the number of duplicates for each record in the dataset. As we can see, most records have around 20 duplicates, but a few records have over 200 duplicates.

We perform 5-fold cross validation on the data, with each split containing 80 training examples and 20 testing examples. To evaluate performance, we consider two measures:

- **Mean Reciprocal Rank (MRR):** The average rank among the duplicates given to the true canonical record. This measure is commonly used in information retrieval to evaluate search results.
- **Accuracy (Acc):** The proportion of predicted canonical records that are truly canonical. This can also be understood as MRR at rank 1.

6.2 Systems

We evaluate eight different systems:

- **Edit-Distance, Fixed Costs (ED-F)** - Levenshtein string distance canonicalizer with the default costs $c_i = 1, c_d = 1, c_r^{\neq} = 1, c_r^{=} = 0$. (See Section 5.1.)
- **Edit-Distance, Exhaustive Cost Search (ED-E)** - Levenshtein string distance canonicalizer with costs set by exhaustive search (See Section 5.2.1.) We set $max = 1.0$, $min = -1.0$, and $step = 0.5$, resulting in 81 different settings.
- **Edit-Distance, Stochastic Cost Search (ED-S)** - Levenshtein string distance canonicalizer with costs set by stochastic search (See Section 5.2.2.) We set $step = 0.2$ and perform 20 iterations.

acronym canonical record
In AAAI

expanded canonical record
In proceedings of the Ninth National Conference on Artificial Intelligence

duplicate records
In proceedings of the Ninth National Conference on *Artifici al* Intelligence
proc the 9th National Conf on AI
in proc AAAI
in proceedings of the Ninth National *Conferenceonarti cial* Intelligence

Table 2: An example of a set of conference strings to be canonicalized annotated with two versions of preferred canonicalization: acronym and expanded forms.

- **Logistic Regression (LR)** - Feature-based ranking model with exponential loss function. (See Section 5.3.1.)
- **MIRA (M)** - Feature-based ranking model with large-margin loss function. (See Section 5.3.2.)
- **Shortest (S)** - A baseline method that ranks records in increasing order of length.
- **Longest (L)** - A baseline method that ranks records in decreasing order of length;
- **Most Common (C)** - Assign each record a count equal to the number of exact string duplicates contained in the record set. Rank records in decreasing order of count.

The features for the feature-based canonicalizers are as follows:

- **Edit-distance features:** We construct four different edit-distance canonicalizers with four different setting of the Levenshtein costs. For each cost setting, we compute $C^d(\mathbf{R})$ as in Equation 1. The binary features for a record indicate if it is the first, second, or third highest ranked record according to $C^d(\mathbf{R})$ (for example, one feature is *ranked-second-by-canonicalizer-three*). These features allow the classifier to serve as a "meta-canonicalizer" by aggregating the output of many different canonicalizers.
- **Text features:** We compute several features that examine the properties of the strings themselves.
  - *Acronyms:* This feature is true if the record contains a token on a list of known acronyms (e.g., *ICML*).
  - *Abbreviations:* This feature is true if the record contains a token on a list of known abbreviations (e.g., *conf* for *conference* and *proc* for *proceedings*).
  - *Relative length:* We compute the character length of each record and create features that indicate if a record is the first, second, or third longest or shortest record.

canonicalizer	MRR	Acc	Time (s)
LR	**.708** (.017)	**.6** (.015)	75 (5)
M	.661 (.017)	.55 (.035)	80 (4)
ED-S	.597 (.053)	.5 (.061)	278 (26)
ED-E	.578 (.053)	.5 (.061)	925 (65)
C	.551 (.023)	.53 (.043)	.05 (.02)
ED-F	.438 (.034)	.37 (.04)	6 (1)
L	.426 (.033)	.28 (.03)	.06 (.05)
S	.087 (.007)	0 (0)	.07 (.06)

Table 3: Mean reciprocal rank, accuracy, and running time on *expanded* dataset. The numbers in parentheses are the standard error over five cross-validation trials.

canonicalizer	MRR	Acc	Time (s)
M	**.94** (.014)	**.92** (.012)	103 (5)
LR	.935 (.02)	.92 (.025)	63 (5)
ED-E	.868 (.027)	.82 (.04)	866 (99)
ED-S	.865 (.028)	.82 (.04)	254(33)
S	.767 (.038)	.64 (.043)	.02 (.004)
C	.126 (.033)	.06 (.024)	.05 (.03)
ED-F	.059 (.004)	0 (0)	6 (2)
L	.049 (.003)	0 (0)	.011 (.001)

Table 4: Mean reciprocal rank, accuracy, and running time on *acronym* dataset. The numbers in parentheses are the standard error over five cross-validation trials.

6.3 Results

Tables 3 and 4 display results for the eight different methods on the *acronym* and *expanded* datasets. We can see that for the *expanded* data logistic regression (**LR**) outperforms the fixed cost edit-distance (**ED-F**) by 27% MRR, and further outperforms the stochastic search edit distance (**ED-S**) by 11% MRR.

From these results, we can conclude that the feature-based canonicalizers consistently outperform the edit-distance canonicalizers. The difference between the two feature-based canonicalizers is small: logistic regression outperforms MIRA (**M**) by nearly 5% MRR on the *expanded* data, but MIRA outperforms logistic regression by .5% MRR on the *acronym* data. Similarly, the difference between the two cost learning methods is small (**ED-S** versus **ED-E**).

Furthermore, we can conclude that cost-learning greatly im-

features	MRR	Acc
edit-distance	.496 (.027)	.29 (.033)
edit-distance + text	.708 (.017)	.6 (.015)

Table 5: Mean reciprocal rank and accuracy on the *expanded* dataset. The numbers in parentheses are the standard error over five cross-validation trials.

proves the performance of the edit-distance canonicalizers, increasing MRR by nearly 16% on the *expanded* data and by 80% on the *acronym* data. The pronounced difference on the *acronym* data can be attributed to the fact that the default setting used in **ED-F** has unit cost for inserting characters. This gives acronym records a large distance from non-acronym records, making it unlikely they will have the lowest average distance. However, the cost learning methods can discover settings that do not penalize insertions, thereby reducing the average edit-distance of acronyms.

None of the simpler baseline methods perform consistently well across the two datasets. Simply choosing the shortest or longest record is significantly worse than using one of the more complex record selection algorithms we propose. Similarly, choosing the most common record

6.4 Impact of features

We investigate the impact of features on the performance of the feature-based canonicalizers. Table 5 displays performance with and without the textual features described in Section 6.2. These results show that using edit-distance features alone outperforms the fixed-cost edit-distance canonicalizer **ED-F** by nearly 6% (.496 versus .438 from Table 3).

6.5 Learning rates

In a real-world application, it may be difficult to obtain labeled data from the user. We therefore perform experiments to evaluate how many labeled examples are needed to obtain accurate results. Figures 2 and 3 plot performance as the proportion of training data used increases. As we can see, using only 10% of the data (8 examples), performance is already quickly approaching its maximum.

6.6 Robustness to noise

We perform additional experiments to measure how robust the methods are to the introduction of non-canonical records. For each training example $\mathbf{R}$, we introduce records as follows:

- Select an incorrect record uniformly at random $R_i \in \mathbf{R}$ s.t. $R_i \neq R_{l_i}$.
- Add n duplicates of of R_i to $\mathbf{R}$.

Figures 4 and 5 show results as n varies from 0 to 20. We compare the four learning methods, as well as the *Most Common* baseline (**C**). These figures show that the feature-based methods are quite robust to noise, as their accuracy drops only slightly as n increases. The *Most Common* baseline degrades significantly, which is unsurprising since as n increases it is very likely that it chooses the incorrect record. The exhaustive cost-learning method also appears relatively robust; however the stochastic cost-learning method degrades significantly.

6.7 Scalability

In Table 3 and 4 we report the wall-clock running time of each method. Note that this includes the time to train each method. The logistic regression requires about one minute to train and evaluate on 80 training examples and 20 testing examples. Note that the long running times of the cost-learning methods is high because for each setting of costs, the average edit-distance for each record must be recomputed to calculate the loss function. This is in contrast to the feature-based methods, which uses fixed costs for the edit-distance features.

For databases containing many records with many duplicates, the computation of the average edit-distance may become burdensome. The edit distance computation has time complexity $O(n^2)$ where $|\mathbf{R}| = n$. $A(R_i)$ requires iterating over all records, and we must compute this n times. Since the edit distance canonicalizer is used as input to the feature-based canonicalizers, these have time complexity $\Omega(n^2)$. Thus, to canonicalize m records will require $\Omega(n^2 m)$ time.

We can alleviate the quadratic dependence on n by pruning elements of $\mathbf{R}^i$ that are unlikely to be chosen as the canonical record. We propose the following method to prune records for the feature-based canonicalizers:

- Build a feature-based canonicalizer C' that only uses text features, not edit-distance features. Thus, this canonicalizer does not require the n^2 computation to compute edit-distance.
- Score each element of $\mathbf{R}$ using C'.
- Remove all R_i with scores less than threshold δ.

This method therefore prunes records from consideration prior to computing the edit-distance. We leave empirical evaluation of this approximation for future work.

7. CONCLUSION AND FUTURE WORK

In this paper, we have introduced the canonicalization problem and proposed three broad classes of solutions. We have implemented one class of solution and empirically evaluated it on manually annotated data. These experiments show that it is possible to build a system to accurately learn canonicalization preferences with only a few examples. In future research, we plan to consider record merging and record generation approaches to canonicalization, as well as joint models that perform deduplication and canonicalization together.

8. 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,

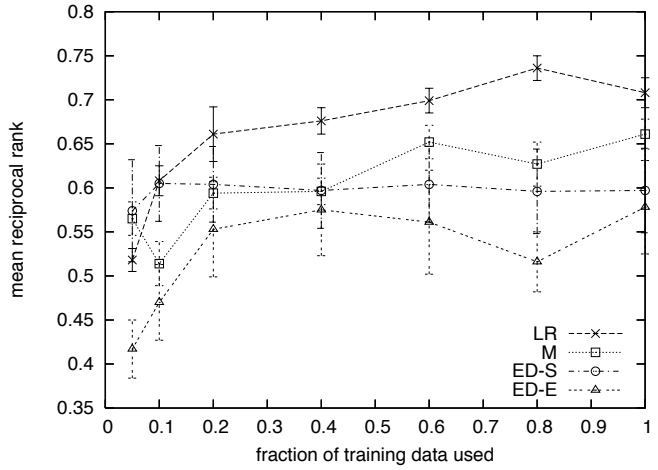

Figure 2: Learning curves for the *expanded* dataset.

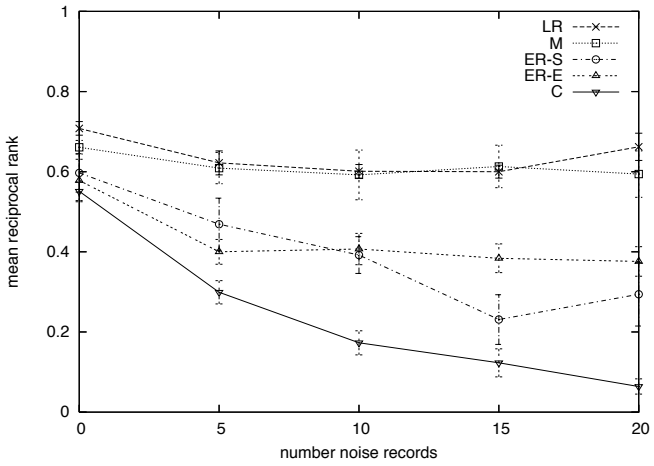

Figure 4: Noise experiments for the *expanded* dataset.

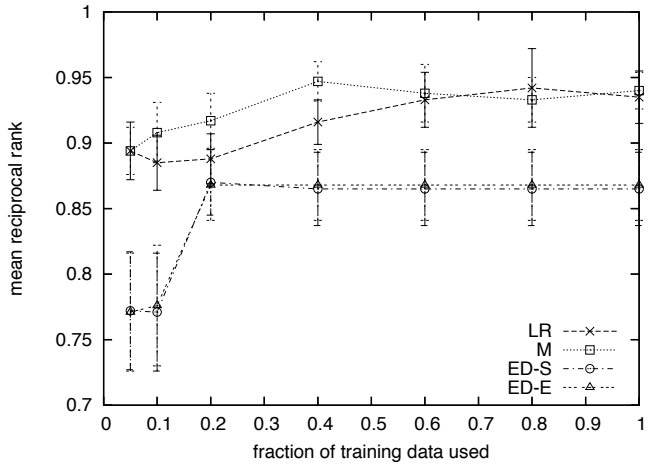

Figure 3: Learning curves for the *acronym* dataset.

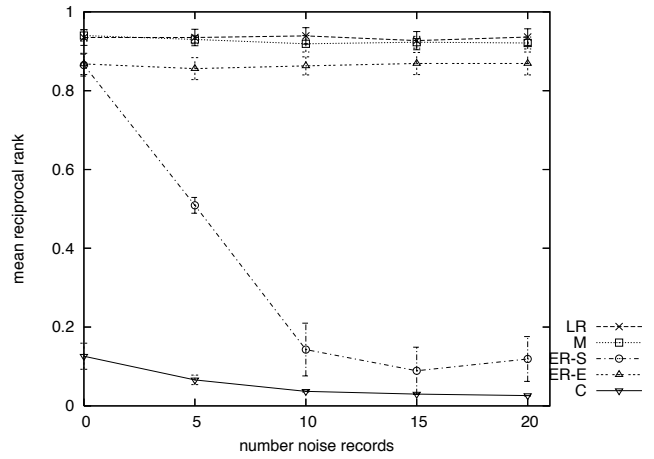

Figure 5: Noise experiments for the *acronym* dataset.

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

9. ADDITIONAL AUTHORS

10. REFERENCES

[1] M. Bilenko and R. J. Mooney. Learning to combine trained distance metrics for duplicate detection in databases. Technical Report AI-02-296, University of Texas at Austin, 2002.

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

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

[4] R. Gupta and S. Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, 2006.

[5] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSR*, 163(4):845–848, 1965.

[6] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.

[7] G. Mann and D. Yarowsky. Multi-field information extraction and cross-document fusion. In *ACL*, 2005.

[8] A. McCallum, K. Ballare, and F. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *Conference on Uncertainty in AI*, 2005.

[9] E. S. Ristad and P. N. Yianilos. Learning string edit

distance. Technical Report CS-TR-532-96, Princeton University, 1997.

[10] S. Tejada, C. A. Knoblock, and S. Minton. Learning object identification rules for information integration. *Information Systems*, 26(8):607–633, 2001.

[11] J. J. Zhu and L. H. Unger. String edit analysis for merging databases. In *KDD*, 2000.