Data Integration for Many Data Sources using Context-Sensitive Similarity Metrics

William W. Cohen*
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

Natalie Glance
Google, Inc
Pittsburgh, PA 15213
nglance@google.com

Charles Schafer
Google, Inc
Pittsburgh, PA 15213
cschafer@google.com

Roy Tromble
Google, Inc
Pittsburgh, PA 15213
royt@google.com

Yuk Wah Wong
Google, Inc
Pittsburgh, PA 15213
ywwong@google.com

ABSTRACT

Good similarity functions are crucial for many important subtasks in data integration, such as "soft joins" and data deduping, and one widely-used similarity function is TFIDF similarity. In this paper we describe a modification of TFIDF similarity that is more appropriate for certain datasets: namely, large data collections formed by merging together many smaller collections, each of which is (nearly) duplicate-free. Our similarity metric, called CX.IDF, shares TFIDF's most important properties: it can be computed efficiently and stored compactly; it can be "learned" using few passes over a dataset (in experiments, one or three passes are used), and is well-suited to parallelization; and finally, like TFIDF, it requires no labeled training data. In experiments, the new similarity function reduces matching errors relative to TFIDF by up to 80%, and reduces k-nearest neighbor classification error by 20% on average.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Data integration

Keywords

Information integration, record linkage, nearest neighbor classification

*This research was done while at Google.

1. INTRODUCTION

An important step in integrating heterogeneous datasets is determining a mapping between objects from one source and objects from another source—a step variously known as record linkage, matching, and deduping (among other terms) in the literature. One useful matching strategy is to use an appropriately thresholded similarity function–i.e., to consider objects as identical if they are "similar enough". One widely-used similarity function is TFIDF similarity, which arose in the field of information retrieval [19]. This similarity function scores objects as similar if they contain many important identical "terms". TFIDF similarity works well in many domains [7], and is often competitive with or superior to more expensive metrics like string edit distance [6, 9]. TFIDF is also very fast to compute—in fact, well-known indexing methods mean that terms similar to a query item x can be found in sublinear time. TFIDF can be computed with no labeled examples—so, to the extent that it is a learned metric, it is learned in an unsupervised way. The sufficient statistics for TFIDF similarity can be stored compactly—only one integer, the document frequency, need be recorded for each term. Finally, the importance weights "learned" by TFIDF can be found very quickly—with only a single pass over the database, using a process that can be easily parallelized. These factors make TFIDF useful in many practically important settings, and perhaps explains why it (and similar token-based similarity metrics) have been the focus of much recent work on "soft joins" (e.g., [2, 23]).

However, although TFIDF's efficiency and scalability make it ideal for many large-scale problems, TFIDF does not exploit all the information in all large data collections. In this paper, we consider large data collections formed by merging together many smaller collections, each of which is duplicate-free (or nearly duplicate-free). This is a very common task: for instance, consider removing duplicates from a database of product descriptions formed by merging the catalogs of a thousand individual merchants; a database of citations formed by merging the reference section of a thousand technical papers; or a database of patient records formed by merging the patient-care information from a thousand hospitals. In each case, we expect to see duplicates across the smaller collections, but not within them.

In this paper, we define a similarity metric that exploits

the statistical properties of such collections. Our metric is a variant of TFIDF, and shares TFIDF's most important properties: it is term-based, and hence can be computed efficiently; it can be stored compactly, as a single number for each term; it can be computed by a small number of passes over a dataset (in our experiments, either one or three passes are used); and it requires no labeled training data. Additionally, the form of the similarity function is unchanged from that of TFIDF: similarity is still defined as the cosine of the angle between two unit-length vectors.

Below, we first introduce our notation, present a small example dataset, define a baseline similarity metric, and discuss the limitations of this baseline. We then define a simple "context-sensitive" variant of the baseline, and discuss some variations of the context-sensitive similarity metric that are sometimes useful. We next present pseudo-code for deriving the similarity metric from a dataset, and argue that the similarity metric can, like IDF-based metrics, be computed efficiently in parallel using a map-reduce framework.

We then present experimental results with context-sensitive similarity on two types of problems: k-nearest neighbor classification, and finding duplicate objects. For the classification tasks, we show that error rates are statistically significant reduced, relative to the baseline similarity metric, on a suite of nine previously-studied tasks. On average, the improvement in error rate is about 20%. For finding duplicate bibliography entries, using a new benchmark consisting of over 100,000 bibliography entries collected from more than 400 distinct sources, we show that the number of known errors can be reduced by nearly 70% for the bibliography matching task. For a similar task involving 40 million product descriptions, we show that known errors can be reduced by more than 90%.

Finally, we discuss related work in the area of similarity metrics for matching and metric learning, and conclude.

2. A CONTEXT-SENSITIVE SIMILARITY METRIC

2.1 Notation and background

Let $D = x_1, \ldots, x_n$ be a dataset of *instances*. Each instance x_i represents some object in the world. Formally, an instance x_i consists of an *identifier* $id(x_i)$, also written id_i; a *context* $c(x_i)$, also written c_i; and a set of *features* $F(x_i)$, also written F_i.

For example x_i might correspond to a product, F_i might be the set of words in the name of the product, and c_i might be the name of the merchant offering the product.

Let V_D be the *vocabulary* associated with a dataset D, i.e., let

$$V_D \equiv \bigcup_{x_i \in D} F(x_i)$$

The elements of V_D are all features, $f_1, \ldots, f_{|V_D|}$. Any set $F_i \subseteq V_D$ can also be represented as a long, sparse vector $\mathbf{v}_i = \langle a_1, \ldots, a_{|V_D|} \rangle$, where the k-th component a_k of $\mathbf{v}_i$ is 1 if $f_k \in F_i$ and a_k is 0 if $f_k \notin F_i$. Below $\mathbf{v}(x_i)$, also written $\mathbf{v}_i$, will denote the vector representation of $F(x_i)$. We use $\|\mathbf{v}\|$ to denote the Euclidean norm of $\mathbf{v}$.

One well-known similarity function is IDF similarity [19]. Define the *document frequency of f (with respect to D)* as the number of instances that contain the feature f, i.e.,

$$\mathrm{DF}_D(f) \equiv |\{x_i \in D : f \in F(x_i)\}|$$

and define the *inverse frequency of w (with respect to D)* as

$$\mathrm{IDF}_D(f) \equiv \log(|D|/\mathrm{DF}_D(w))$$

Now define the *IDF-weighted vector for x_i* as a vector $\mathbf{w}(x_i) = \langle b_1, \ldots, b_{|V_D|} \rangle$, also written $\mathbf{w}_i$, where the k-th component b_k of $\mathbf{w}_i$ is $a_k \cdot \mathrm{IDF}_D(f_k)$. IDF similarity is defined as

$$sim_{\mathrm{IDF}}(x_i, x_j) \equiv \frac{\mathbf{w}_i \cdot \mathbf{w}_j}{\|\mathbf{w}_i\| \cdot \|\mathbf{w}_j\|}$$

This is often interpreted as the cosine of the angle between the vectors, so the term "cosine similarity" is also frequently used. (The term "TFIDF similarity" is also commonly used, where the "TF" part of the name comes from a weighting of "term" frequency within documents, a consideration that we ignore here.)

One important property of the IDF similarity metric is that features are given *differential importance* in computing similarity scores. While overlapping features $f \in F_i \cap F_j$ always increase similarity, and non-overlapping features $f' \in F_i - F_j$ (or $f' \in F_j - F_i$) always decrease similarity, high-weight features are much more important to "get right" than low-weight features.

id_i	c_i	F_i
us:gh1	toysRus	guitar, hero, IIIx, controller, from, toys-R-us
us:gh2	toysRus	guitar, hero, for, gameboy, from, toys-R-us
us:b14	toysRus	lego, bionicle, kit, x14, from, *lego*
us:b23	toysRus	lego, bionicle, kit, x23, from, *lego*
us:b37	toysRus	lego, bionicle, model, x37, from, *lego*
bb:gh2	bestbuy	guitar, hero, IIIx, for, gameboy
bb:b14	bestbuy	lego, bionicle, x14, truck, kit
bb:b23	bestbuy	lego, bionicle, x23, zombie, kit
bb:b37	bestbuy	lego, bionicle, x37, watermelon, kit
cc:gh2	ccity	guitar, hero, IIIx, for, gameboy
cc:b14	ccity	lego, bionicle, x14, truck, kit
cc:b23	ccity	lego, bionicle, x23, zombie, kit
cc:b37	ccity	lego, bionicle, x37, watermelon, kit

Figure 1: A small example dataset. The elements of the sets F_i are conceptually unordered and free of duplicates, but for clarity, they are given in the order of the original English description (e.g., "guitar hero IIIx controller from toys-R-us"). Also, occurrences of a word after the first are italicized in the figure - in the actual dataset, repeated occurrences are discarded.

2.2 A small example

Figure 1 presents a small example dataset. In this dataset, the features "from", "lego" and "bionicle" would have relatively low IDF weights, as they are shared by many pairs of non-identical products. The features "IIIx", "x14", and "x37" would have higher weights, since they appear in no pairs of non-identical products.

Figure 2 shows the similarity of some pairs of instances—the pairs below the line refer to different products, and the pairs above the line refer to the same product.

Pair id_i, id_j	$sim(x_i, x_j)$	
	IDF	CX.IDF
bb:b14,cc:b14	1.00	1.00
bb:b23,cc:b23	1.00	1.00
bb:b37,cc:b37	1.00	1.00
bb:gh2,cc:gh2	1.00	1.00
bb:b23,us:b23	0.86	0.99
cc:b23,us:b23	0.86	0.99
bb:gh2,us:gh2	0.82	0.84
cc:gh2,us:gh2	0.82	0.84
cc:b14,us:b14	0.57	0.88
bb:b14,us:b14	0.57	0.88
bb:b37,us:b37	0.31	0.85
cc:b37,us:b37	0.31	0.85
us:gh1,us:gh2	0.53	0.19
bb:gh2,us:gh1	0.46	0.62
cc:gh2,us:gh1	0.46	0.62

Figure 2: Similarities of some pairs of instances in the sample dataset

2.3 A limitation of IDF similarity

In the small example, each x_i corresponds to a product, F_i is words in the name of the product, and the context c_i identifies a merchant offering the product. Neither of these similarity metrics take into account the context associated with an example; intuitively, however, they should.

The reason contexts matter is that in many cases it is possible to find contexts c such that all of the instances with a shared context are likely to be distinct. For example, suppose D is formed by merging the catalogs of many merchants, where most of the individual catalogs are substantially free of duplications. If x_i is a product, F_i is the words in the name of the product, and c_i is the merchant offering the product, then two products x_i, x_j such that $c_i = c_j$ are unlikely to be identical—as this would indicate a duplication within a single merchant, an unlikely case.

Ideally, one would like the weights that are assigned to terms to reflect this—i.e., terms should be have more weight if they are rare for many merchants, and have less weight if they are common for some mechants. However, the IDF weights do not have this property. For instance, the feature "toys-R-us" is rare (with DF=2) but appears twice in descriptions from the same merchant, while the feature "x14" is more frequent (DF=3) but appears only once in each merchant's product description. We would prefer to weight "x14" somewhat higher than "toys-R-us".

2.4 Context-dependent similarity

To capture this intuition in a principled way, let us consider two quantities. We define the probability of an *inter-context duplication for a feature f (with respect to D)*, written $\Pr^{INTER}{}_D(f)$, to be the probability that two instances x_i, x_j have *different* contexts $c_i \neq c_j$, given that x_i, x_j are a pair of instances, both containing the feature f, drawn uniformly at random but without replacement from D. Likewise, we define the probability of an *intra-context duplication for f (with respect to D)*, written $\Pr^{INTRA}{}_D(f)$, to be the probability that x_i, x_j have the *same* context c, again given that x_i, x_j are a pair of instances, containing f, drawn uniformly at random but without replacement from D.

To simplify notation below, we will drop the subscript D from $\Pr^{INTRA}$ and $\Pr^{INTER}$ (and elsewhere, when it is clear from context). We will also define some more quantities.

- Let $n \equiv |D|$ be the number of instances in the dataset.
- Let $C_D \equiv \{c(x_i) : x_i \in D\}$ be the set of contexts in the dataset
- Let $n_f \equiv \mathrm{DF}_D(f) \equiv |\{x_i \in D : f \in F_i\}|$ be the number of instances in the dataset that contain feature f.
- Let $n_{c,f} \equiv |\{x_i \in D : f \in F_i \text{ and } c_i = c\}|$ be the number of instances in the dataset that contain feature f and have context c.
- Let $D_f \equiv \{x_i \in D : f \in F_i\}$, be the instances in D that contain feature f, and let $x \sim D_f$ denote drawing x uniformly from D_f.

Then we can define $\Pr^{INTER}$ more precisely as follows:

$$\begin{aligned}
&\Pr^{INTER}(f) \equiv \\
&\quad \Pr(c_i \neq c_j | x_i \sim D_f \text{ and } x_j \sim D_f \text{ and } x_j \neq x_i) \\
&= \sum_{c \in C_D} \Pr(c_i = c | x_i \sim D_f) \cdot \Pr(c_j \neq c | x_j \sim D_f - \{x_i\}) \\
&= \sum_{c \in C_D} \frac{n_{c,f}}{n_f} \cdot \frac{n_f - n_{c,f}}{n_f - 1} \qquad (1)
\end{aligned}$$

Likewise

$$\begin{aligned}
&\Pr^{INTRA}(f) \equiv \\
&\quad \Pr(c_i = c_j | x_i \sim D_f \text{ and } x_j \sim D_f \text{ and } x_j \neq x_i) \\
&= \sum_{c \in C_D} \Pr(c_i = c | x_j \sim D_f) \cdot \Pr(c_j = c | x_j \sim D_f - \{x_i\}) \\
&= \sum_{c \in C_D} \frac{n_{c,f}}{n_f} \cdot \frac{n_{c,f} - 1}{n_f - 1} \qquad (2)
\end{aligned}$$

Equivalently, one could let $\Pr^{INTRA}(f) \equiv 1 - \Pr^{INTER}(f)$; however, Equation 2 will be useful later on when we consider smoothing. Now consider the quantity $\mathrm{CX}(f)$, defined as

$$\mathrm{CX}(f) \equiv \log \frac{\Pr^{INTER}(f)}{\Pr^{INTRA}(f)} \qquad (3)$$

If pairs containing f are likely to be within the same merchant, and unlikely to be from different merchants—as for the feature "toys-R-us" above—then $\mathrm{CX}(f)$ is small. If pairs containing f are unlikely to be within the same merchant, and likely to be from different merchants—as for the feature "x14" above—then $\mathrm{CX}(f)$ is large. Thus $\mathrm{CX}(f)$ captures the intuition discussed above.

One simple way to extend the IDF metric to additionally incorporate this new context-dependent notion of importance is simply to use the product of $\mathrm{IDF}(f)$ and $\mathrm{CX}(f)$ to weight features. Let the *context-dependent IDF-weighted vector for x_i* be a vector $\mathbf{z}(x_i) = \langle b'_1, \ldots, b'_{|V_D|} \rangle$, also written $\mathbf{z}_i$, where the k-th component b'_k of $\mathbf{z}_i$ is $a_k \cdot \mathrm{IDF}(f_k) \cdot \mathrm{CX}(f_k)$. Now the context-dependent IDF similarity can be defined as

$$sim_{CX.IDF}(x_i, x_j) \equiv \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \cdot \|\mathbf{z}_j\|}$$

A small note: $\mathrm{CX}(f)$ is undefined if $n_f = 1$, or if $\Pr^{INTRA}(f) = 0$. For the former case, we define $\mathrm{CX}(f) \equiv 1$ if $n_f = 1$, which

leaves the IDF weight for f unchanged; we conjecture that this makes little difference in practice, since features that occur only once in the dataset are not very useful for similarity comparisons anyway. The latter case can be avoided by smoothing the estimate for $\Pr^{INTRA}$ away from zero, an issue which is discussed below.

2.5 Some refinements of the metric

The definitions of $\Pr^{INTRA}$ and $\Pr^{INTER}$ above are maximum likelihood estimates based on the data. If there is prior knowledge of the importance of a feature f, this can be incorporated by smoothing the estimate toward the expected. We smooth estimates using a Dirichlet prior, encoded as two values p_0, m_0, representing a prior probability p_0 and a "strength" (expressed in number of equivalent examples) m_0 respectively. Given observations supporting the estimate p_1 based on m_1 examples, the Dirichlet-smoothed estimate is

$$\hat{p} = p_1 \cdot \frac{m_1}{m_1 + m_0} + p_0 \cdot \frac{m_0}{m_1 + m_0} = \frac{p_1 m_1 + p_0 m_0}{m_1 + m_0}$$

In some applications, there are multiple types of features, rather than a single "type" of feature. For instance, there might be tokens taken from a short name of a product, but also tokens taken from a longer description; a simple way of handling this would be to create two features f^n_w and f^d_w for each word w, where f^n_w (respectively f^d_w) represents word w appearing in the name (respectively description) of a product. Assume that features in $V(D)$ can be grouped into disjoint sets $V_1, \ldots, V_m$, corresponding to different "types" of features. If one believes that there are different types of features have different average levels of informativeness, then an empirical Bayesian method can be used to smooth feature values in each subset V_ℓ together.

To do this, the $\Pr^{INTRA}$ values for features f are computed with a default prior over each subset of features $f \in V_\ell$, and the mean μ_ℓ and standard deviation σ_ℓ of these values are computed. From this, a prior value p_ℓ is computed for features in V_ℓ, as well as a prior strength m_ℓ, which are defined as

$$p_\ell \equiv \frac{p_0 m_0 + p'_\ell m'_\ell}{m_0 + m'_\ell} \quad (4)$$

$$m_\ell \equiv m_0 + m'_\ell \quad (5)$$

where p_0, m_0 are a user-defined universal prior, $p'_\ell = \mu_\ell$, and $m'_\ell \equiv \mu_\ell(1-\mu_\ell)/\sigma^2_\ell$. This choice for p'_ℓ, m'_ℓ has a nice mathematical property: with these values, the expected value and standard deviation of the prior distribution, as a Beta distribution, is equivalent to the observed mean and standard deviation of the values of the features in V_ℓ. After the priors are set, values for $\Pr^{INTRA}$ can be re-computed using the revised priors (as well as values for $\Pr^{INTER}$, and CX).

We also find it useful to bound the similarity function away from unity; this is useful since often, objects with identical features still have some chance of being distinct. To avoid getting extreme values of similarity one can adjust the metric by conceptually adding to every instance x_i one additional feature with weight γ that never appears anywhere else in the dataset. Larger values of γ lead to smaller maximal similarities.

Equivalently, we can unroll and then modify the definitions of inner product and $\|\mathbf{v}\|$. Let $z_{i,f}$ be the component of $\mathbf{z}_i$ that corresponds to feature f, and similarly for $z_{j,f}$. Then we let

$$sim_{CX.IDF}(x_i, x_j) \equiv \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \cdot \|\mathbf{z}_j\|} = \frac{\sum_f z_{i,f} \cdot z_{j,k}}{\sqrt{\sum_f z^2_{i,f}}\sqrt{\sum_f z^2_{j,f}}}$$

$$\approx \frac{\sum_f z_{i,f} \cdot z_{j,k}}{\sqrt{\gamma^2 + \sum_f z^2_{i,f}}\sqrt{\gamma^2 + \sum_f z^2_{j,f}}}$$

Using nonzero value of γ in the last line will bound $sim_{CX.IDF}(x_i, x_j)$ away from 1.0.

In the experiments below, we will use two types of smoothing.

- In the first variant, $\Pr^{INTRA}$ and $\Pr^{INTER}$ are computed as in Equations 1 and 2, save that in estimating $\Pr(c_i = c|\cdot)$ and $\Pr(c_i \neq c|\cdot)$, we use Dirichlet smoothing with $p_0 = 0.5$ and $m_0 = 1$ (i.e., a Laplace correction). This minimal amount of smoothing is necessary to keep $\mathrm{CX}(f)$ from taking extreme values: below we will call this "Laplace-corrected" CX.IDF, or simply CX.IDF.

- In the second variant, we compute Laplace-corrected CX.IDF in a first pass, and then use the approach of Figure 4 as a second pass; here, we again use $p_0 = 0.5$ and $m_0 = 1$ as the "universal prior" of Equations 4 and 5. Below we call this "CX.IDF with empirical priors", or simply "smoothed CX.IDF".

Either of these types of smoothing can be combined with a non-zero value of γ: we used two values of γ, $\gamma = 10$ and $\gamma = 0$, leading to four variants of CX.IDF.

Preliminary experiments suggested that for most purposes, the Laplace smoothing is comparable to the empirical prior method, but that empirical priors are preferable when (a) there are many distinct types of features, and some types tend to be highly informative, while some tend to be less informative, or (b) most of the features are extremely informative, with a few frequent exceptions. As an example of the former case, consider matching product names with two types of features, one type derived from a long textual "marketing text" field, and one type derived from a shorter "part number" field. An example of the latter case might be hard identifiers that contain a few frequent noisy cases (e.g., part numbers like "000000000000", "N/A", or "UNK").

2.6 Computing context-dependent similarity

One important property of the context-dependent IDF similarity metric is that it can be computed quite inexpensively, even for large datasets. Figures 3 and 4 show one algorithm for this, described using "map-reduce" framework [12]. Programs implemented using this framework can be easily parallelized, so that a single large task can be run on many smaller computers in parallel quite efficiently.

Briefly, the dataset D is converted first to pairs (f, c), where f is a feature appearing in context c, and then this output is sorted and grouped into pairs of the form

$$(f, \langle c_1, \ldots, c_n \rangle)$$

where f is a feature and $\langle c_1, \ldots, c_n \rangle$ is a sorted list of all contexts that co-occur with that feature. This can in turn be converted to a list of the form

$$(f, \langle (c_1, n_{c_1,f}) \ldots, (c_n, n_{c_n,f}) \rangle)$$

1. *Weight the features.*
   - *(Map step 1).* Let the list *MapOut* be an empty list.
   - For each $x_i \in D$ and each $f \in F_i$,
     - Append the pair (f, c_i) to *MapOut*.
   - Sort *MapOut* lexicographically.
   - In the sorted *MapOut*, collect sequences of adjacent pairs $(f, c_1), \ldots, (f, c_n)$ with the same feature f and replace them with a single entry $(f, \langle c_1, \ldots, c_n \rangle)$. Call the revised list *ReduceIn*.
   - *(Reduce step 1).* Let the list *ReduceOut* be empty, and let the list *CXVals* be empty.
   - For each pair $(f, \langle c_1, \ldots, c_n \rangle)$ in *ReduceIn*:
     - From the list $\langle c_1, \ldots, c_n \rangle$, compute n_f and $n_{c,f}$ for every c in the list.
     - Compute $\mathrm{CX}(f)$ using Equations 1, 2, and 3.
     - Compute $\mathrm{IDF}(f) = \log(n/n_f)$
     - Append the pair $(f, \mathrm{CX}(f) \cdot \mathrm{IDF}(f))$ to *ReduceOut*
     - Append the pair $(f, \mathrm{CX}(f))$ to *CXVals*

Figure 3: Computing context-dependent IDF similarity

where the pair $(c_i, n_{c_i,f})$ contains a context plus the number of times it was duplicated. Below we will call such a list a *context histogram for feature* f.

Note that none of the intermediate representations of the dataset are larger than the original dataset, and for any particular feature f, the context histogram will be relatively small—it is bounded in size by the number of different contexts. Importantly, the weight $\mathrm{CX}(f) \cdot \mathrm{IDF}(f)$ for feature f depends only on the context histogram; this is why it is easy to parallelize the computation of the weights.

Figure 4 presents similar pseudo-code for smoothing with empirically-derived priors.

3. EXPERIMENTAL RESULTS

3.1 Classification of short strings

The first set of experiments we will discuss will be for the task of K-nearest neighbor (k-nn) classification. Classification is perhaps the best-studied task that uses similarity metrics–in fact, a data integration system that supports similarity joins can be straightforwardly used for k-nn classification, using a reduction proposed by Cohen and Hirsh [8]. Hence, a plausible way to evaluate a similarity metric is to use it to classify instances qualitatively like those that must be matched. For instance, Cohen and Hirsh evaluated some variations of IDF similarity using a k-nn classifier (for $k = 30$) on a suite of nine problem where the instances were short descriptions, such as book titles, web page titles, game titles, or names of different bird species. These problems are summarized in Table 1.

While none of these datasets have a natural notion of "context", an alternative version of CX.IDF can be used for classification. We let each *class label* be represented by a different context, and then modify each vector by *dividing* the

1. *Compute priors for the features.*
   - Clear all counters a_ℓ, b_ℓ.
   - For each pair $(f, CX(f))$ in the list *CXVals* produced by the method of Figure 3:
     - Let ℓ be the "type" of feature f, (i.e., the set V_ℓ such that $f \in V_\ell$).
     - Increment a_ℓ by $CX(f)$.
     - Increment b_ℓ by $CX(f)^2$.
     - Increment c_ℓ by 1.
   - For each "feature type" ℓ,
     - Let $p'_\ell = a_\ell / c_\ell$ be the observed average score of features of type ℓ.
     - Let $\sigma_\ell = \sqrt{b_\ell / c_\ell - (p'_\ell)^2}$ be the observed variance of this average.
     - Let $m'_\ell = (p'_\ell(1 - p'_\ell)/\sigma^2)$
     - Let $m_\ell = m_0 + m'_\ell$
     - Let $p_\ell = (p'_\ell m'_\ell + p_0 m_0)/(m_0 + m'_\ell)$
2. *Re-weight the features.* Re-execute *Map step 1* and *Reduce step 1* of the method of Figure 3, but in computing $\mathrm{CX}(f)$, use Equations 3 and Dirichlet-smoothed versions of Equations 1,and 2, where the prior is m_ℓ and p_ℓ above (such that ℓ is the type of feature f).

Figure 4: Computing context-dependent IDF similarity with empirical priors

IDF weights of each feature by the CX weight, rather than multiplying it by the CX weight. Recall that the motivation of the original CX.IDF formula was to increase the importance of features that are distributed across many "contexts", and decrease the importance of features that are concentrated in a single "context". For classification we would like to do the reverse. We call this variant IDF/CX.

We performed the same experiments conducted by Cohen and Hirsh with our system. As in Cohen and Hirsh, if an example has no neighbors (i.e., no training cases have non-zero similarity) then the most frequent class is predicted. The results are shown in Table 2.

Depending on the variant used, the IDF/CX weighting scheme increases error very slightly on two or three of the nine problems, but decreases error (by up to 40%) on other problems. On average, relative to the baseline IDF metric with $\gamma = 0$, the error rate is decreased by 14% for IDF/CX with $\gamma = 0$, and decreased by nearly 20% for the other three IDF/CX variants. This reduction is statistically significant in each case (at 95% confidence or above, using a paired t-test across the nine problems.)[1]

For these problems we looked at two values of γ: $\gamma = 10$ and $\gamma = 0$. (No additional tuning of γ was done, and the default non-zero weight of $\gamma = 10$ was established on separate datasets.) A value of $\gamma = 10$ statistically significantly improves over $\gamma = 0$ for unsmoothed IDF/CX, but not for

[1]The results use our re-implementation of the Cohen and Hirsh results as a baseline, but the average performance of our baseline is nearly identical to the results published by Cohen and Hirsh, and the average reduction in error for the smoothed versions of IDF/CX are statistically significantly improved over the published Cohen and Hirsh results as well.

problem	#train	#test	#classes	#terms	text-valued field	label
memos	334	10cv	11	1014	document title	category
cdroms	798	10cv	6	1133	CDRom game name	category
birdcom	914	10cv	22	674	common name of bird	phylogenic order
birdsci	914	10cv	22	1738	common + scientific name of bird	phylogenic order
hcoarse	1875	600	126	2098	company name	industry (coarse grain)
hfine	1875	600	228	2098	company name	industry (fine grain)
books	3501	1800	63	7019	book title	subject heading
species	3119	1600	6	7231	animal name	phylum
netvet	3596	2000	14	5460	URL title	category

Table 1: Description of benchmark problems

Task	Error	IDF		IDF/CX		smoothed IDF/CX	
	Rate	$\gamma = 0$	$\gamma = 10$	$\gamma = 0$	$\gamma = 10$	$\gamma = 0$	$\gamma = 10$
memos	0.36	1.00	1.01	0.81	0.81	0.80	0.82
cdroms	0.55	1.00	0.99	1.03	0.97	0.98	0.99
birdcom	0.18	1.00	0.98	0.69	0.60	0.63	0.62
birdsci	0.12	1.00	0.96	0.63	0.60	0.65	0.63
hcoarse	0.70	1.00	0.96	1.09	1.01	1.00	1.01
hfine	0.80	1.00	0.98	1.05	1.00	1.00	1.02
books	0.41	1.00	0.99	0.92	0.92	0.92	0.90
species	0.07	1.00	1.00	0.79	0.61	0.58	0.55
netvet	0.30	1.00	1.05	0.74	0.74	0.76	0.73
average	0.38	1.00	0.99	$*0.861$	$*0.807$	$*0.813$	$*0.808$

Table 2: IDF/CX on nine k-nn classification problems with short descriptions. In the last six columns, performance is expressed as a fraction of baseline error rate. Starred averages are statistically significantly below 1.0.

the IDF metric or the smoothed IDF/CX metric. The differences between the smoothed and unsmoothed IDF/CX variants are not statistically significant.

To summarize, the IDF/CX weighting scheme improves significantly over the baseline IDF, and improves more when smoothed appropriately. Furthermore, smoothing by using the simple Laplace correction and $\gamma = 10$ performs about as well as the more expensive empirically Bayesian smoothing.

3.2 Matching bibliography entries

In order to perform a more direct evaluation of CX.IDF's utility for matching object descriptions, we generated data for a representative matching task that included contexts. We performed a Google search for publically-indexed files with the extension ".bib" that contained the phrase "machine learning", and downloaded 418 such files that were valid LaTeX bibliography files, which collectively contained 115,940 bibliography entries. The context of a bibliography entry was the file from which it was extracted, and each bibliography entry was described by features f^{ℓ}_{w} where w is a word (a space-separated string with case folded but without stemming) and ℓ is a field of the bibliography entry (one of "title"; "booktitle" or "journal"; "author"; "editor"; "keywords"; "cite" or "key"; "volume"; "month"; "year"; "pages"; "number"; "series"; "note"; "address"; "howpublished" or "type"; "publisher", "organization", "school", or "institution"; 'or abstract".) Below, these bibliography-entry fields will be used to establish groups for empirical Bayesian smoothing: specifically, we will define distinct set of features $V_{\ell} \equiv \{f^{\ell}_{w_1}, \ldots, f^{\ell}_{w_n}\}$ for the words contained in with different fields. There are a total of 387,383 distinct features of these 17 types.

To evaluate the correctness of matching, it is useful to consider matchable items that have "hard identifiers"—strings that can act as clear and unambiguous identifiers. To this end, we identified all bibliography entries which were (a) not identified as parts of a book and (b) contained a field labeled "url" or "doi" with a value that is a URL containing one of substrings "doi", "pdf", or "ps". The vast majority of these fields are either ACM digital library document identifiers, or else on-line versions of papers; hence when such fields exist, they can be used as "hard identifiers". However, only a small minority of the papers (2,922) have such identifiers. We define a pair of bibliography entries x, y to be *correct* if both x and y have hard identifiers and $id(x) = id(y)$, *incorrect* if both x and y have hard identifiers and $id(x) \neq id(y)$, and *uncertain* otherwise.

To evaluate a similarity metric, we followed a methodology similar to that used in previous papers on similarity joins [7]. First, a fixed set of pairs x, y are generated. Second, this set of pairs is sorted by their similarity. Finally, the *interpolated average precision* is plotted at each position in the list. Non-interpolated average precision at rank r is the number of correct pairs at rank r or above divided by the number of correct or incorrect pairs at rank r or above. (Hence, uncertain pairs are simply ignored in computing non-interpolated average precision.) Interpolated average precision at rank r is the maximum value of non-interpolated average precision at any rank $r' > r$.

The results are shown in Figure 5. "Smoothed CX.IDF" is CX.IDF with smoothing using empirical Bayes, where the feature types are defined by the field from which a token was taken, and a value of $\gamma = 10$ was used throughout. We ranked a large set of candidate pairs (approximately 300,000, selected using the method described in Section 3.4, with

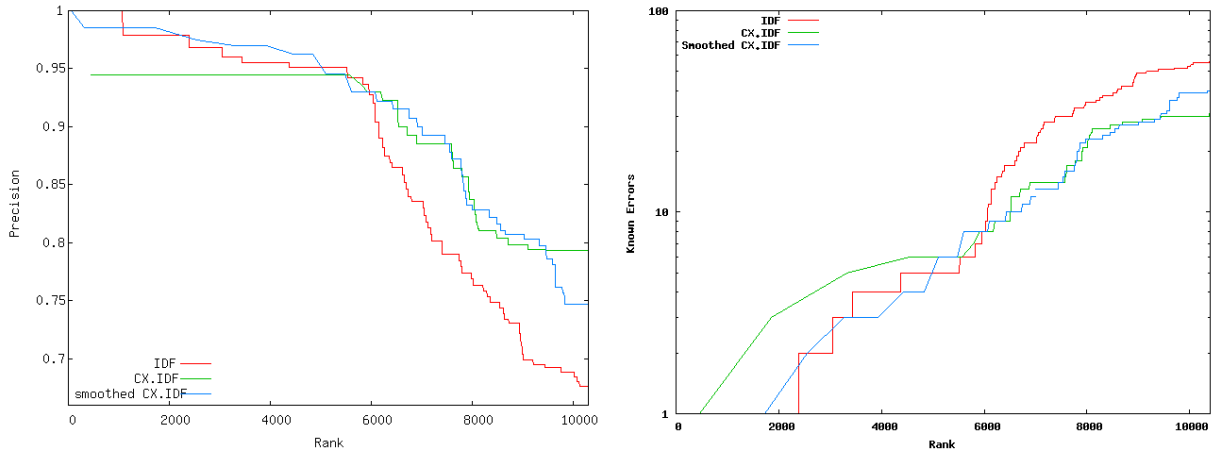

Figure 5: Matching bibliography entries from 418 sources: interpolated average precision using IDF and two varieties of CX.IDF. Left: precision versus rank. Right: number of known errors versus rank.

thresholds of CX $\geq$ 1.5 and DF $\leq$ 1000.) On the left of the figure, we display precision for the first 10,000 ranks. Smoothed CX.IDF outperforms the IDF metric over most of this range—although there is a narrow interval (around ranks 5000-6000) where IDF performs best. The first 10,000 ranks is the range most interesting for a system that requires high-precision matches.[2]

The right-hand side plot Figure 5 summarizes the same data in a different way. Here, we plot the number of known matching errors (i.e., errors detectable using the hard identifiers) against rank—for instance, the graph shows that in the first 7000 pairs listed, the IDF ranking contains 22 known errors and the smoothed CX.IDF ranking contains about 12 known errors. At the same position in the ranking, the smoothed CX.IDF ranking contains only 108 pairs known to be correct, and the IDF ranking contains only 111 known-correct pairs.

Because these numbers are relatively small, precision estimates based on them will necessarily have high variance. We thus also report, in Table 3, the total number of known errors for each similarity method for the remainder of the ranking. In this range, the "unsmoothed" CX.IDF performs best, reducing the number of errors made by the baseline IDF ranking by nearly 70% for some recall levels. Smoothed CX.IDF also reduces errors substantially.

Rank	IDF	CX.IDF		Smooth CX.IDF	
$\times$ 1000	Errors	Errors	Δ	Errors	Δ
10	53	30	-43.40	39	-26.42
20	141	54	-61.70	87	-38.30
30	210	77	-63.33	155	-26.19
40	291	103	-64.60	236	-18.90
50	401	129	-67.83	308	-23.19
75	631	192	-69.57	449	-28.84
100	882	274	-68.93	582	-34.01
200	1583	527	-66.71	1018	-35.69
300	1945	788	-59.49	1293	-33.52
500	2464	1308	-46.92	1814	-26.38
1000	3207	2390	-25.48	2724	-15.06
2500	4664	4530	-2.87	4508	-3.34
3000	5245	5245	0.00	5245	0.00

Table 3: Number of known errors as a function of rank position for three similarity metrics for bibliography entries.

3.3 Matching product descriptions

As a larger-scale matching problem, we took a set of descriptions of consumer products and performed a similar experiment. Consumer product description matching is both commercially important and technically difficult (e.g., previous work has shown only moderately accurate performance for clustering products, even using similarity functions trained using supervised learning and labeled product-description pairs [5].) The sample included approximately 40 million product descriptions from several hundred distinct sources, and included descriptions of products offered for sale in June 2009 by Google Shopping; product descriptions extracted from web sites; and hand-constructed product catalogs. More than half of these product descriptions included some sort of universally-recognized "hard" identifier (e.g., ISBN number or a UPC code). The set of products is extemely diverse: about two-thirds of the hard identifiers are distinct.

As before, we generated a list of weakly similar pairs, and ranked these pairs by each of three similarity functions: IDF, CX.IDF, and smoothed CX.IDF. In all cases, a value of $\gamma = 10$ was used. There were approximately 60 million pairs in the generated list. The features here were more carefully engineered than in the bibliography-matching problem above, but all feature-tuning was done on a much earlier version of the dataset.

The results are shown in Figures 6 and 7. Overall performance on this task is much higher, but qualitatively the results are similar. At high levels of recall, CX.IDF far outperforms IDF in terms of precision, and smoothed CX.IDF has precision between IDF and "unsmoothed" CX.IDF. We

[2]Since the total number of correct pairs is unknown, recall cannot be determined, although clearly recall is linearly related to rank. If one extrapolates the rate of duplications of bibliography entries with hard identifiers to the entire set, one would expect about 5000 true duplications.

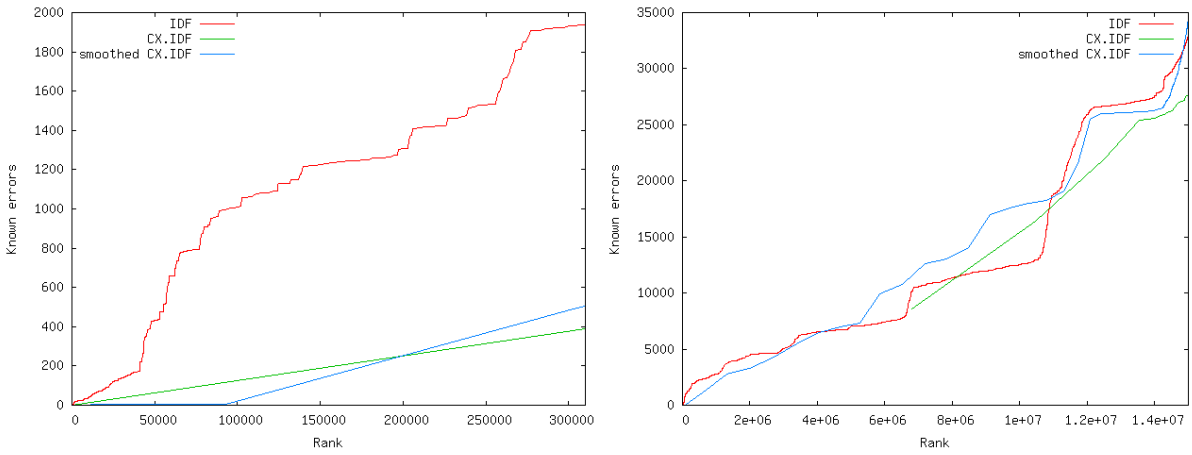

Figure 7: Matching product descriptions from several hundred sources. Left, number of errors versus ranked 300,000 pairings. Right, number of errors versus rank for the top-ranked 14 million pairings.

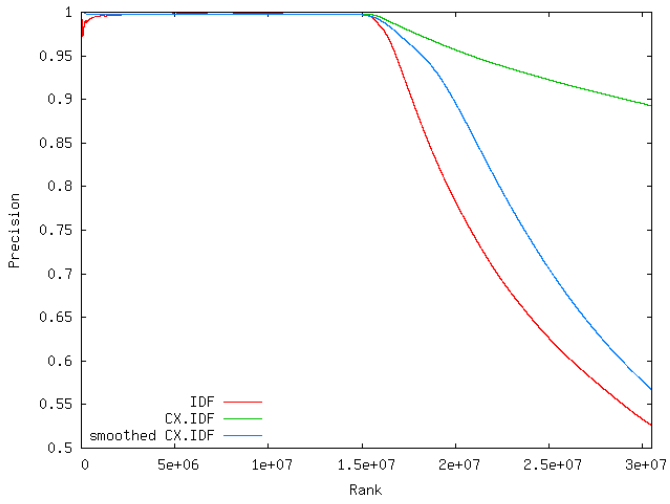

Figure 6: Matching product descriptions from several hundred sources: precision plotted against rank, using IDF and two varieties of CX.IDF, over the first 30 million pairings.

used $\gamma = 10$ on all of these experiments.

Performance at lower precision levels is harder to see, since all three algorithms show quite good performance; also, due to rounding errors, "unsmoothed" CX.IDF gives the same score to the top 6.8 million (1.7%) of the candidate pairs, making 8,579 errors on these top-ranked pairs for a precision of 99.8%.

To visualize performance here, we again look at the number of errors for each method. Figure 7 shows errors for the first 300,000 pairs on the left, and errors for the first 14 million pairs on the right. Again, at lower recall levels, smoothed CX.IDF performs best, followed closely by unsmoothed CX.IDF. At higher recall levels, unsmoothed CX.IDF generally performs best—with the exception of ranks between about 8 and 11 million, where ordinary IDF has the lowest error rate. Table 4 gives the number of known errors made for certain ranks, including parts of the high-

1. Let F_{canopy} be a subset of features.
2. Let C be the empty set.
3. For each $f \in F_{\text{canopy}}$,
   (a) Let $X_f = \{x_1, \ldots, x_{\text{DF}(f)}\}$ be the set of all instances that have f as a feature (i.e., the inverted index for f).
   (b) For each pair (x_i, x_j) such that $x_i \in X_f$, $x_j \in X_f$, and $x_i \neq x_j$,
      - Add (x_i, x_j) to C.

Table 5: A simple algorithm for generating pairs x_i, x_j of likely-to-be-similar instances.

precision range (ranks 5,000 through 650,000); the middle range, where IDF performs best (ranks 2,000,000 through 10,000,000); and the high-recall range (ranks 10,000,000 through 30,000,000). As in Table 3, we also show the percentage improvement over baseline.

To summarize, in both matching experiments, there is a small part of the recall-precision curve in which IDF is competitive with the CX.IDF variants. However, over most of the curve, one or both of CX.IDF variants greatly outperform the IDF baseline.

3.4 Generating plausible candidate pairs

Another important use of IDF scores is for selecting candidate pairs for which similarity will be computed. A number of well-known heuristics for finding candidate pairs involve finding features f with high IDF, and then proposing pairs of items x_i, x_j that share feature f. These pairs can be conveniently found by using an inverted index for f. Many fast similarity-join methods use this trick explicitly (e.g., [2, 7]), while others use it indirectly via information-retrieval engines (which often use "shortcut" retrieval methods based on high-IDF terms [21]) (e.g., [14]).

We wished to explore the effectiveness of context-sensitive similarity metrics for generating candidate pairs. In order to do this, we took the bibliography data, and generated the

Rank × 1000	IDF Errors	CX.IDF Errors	Δ	Smooth CX.IDF Errors	Δ
5	23	–	–	4	-82.61
20	84	–	–	7	-91.67
100	1,009	–	–	482	-52.23
650	2,363	–	–	1,376	-41.77
2,000	4,487	–	–	4,325	-3.61
5,000	7,047	8,579	+21.74	7,555	+7.21
10,000	12,573	16,430	+30.68	17,969	+42.92
10,000	3,348,616	665,592	-80.12	1,437,196	-57.08
25,000	7,440,369	1,238,221	-83.36	5,214,735	-29.91
30,000	11,316,451	1,841,019	-83.73	9,226,685	-18.47

Table 4: Number of known errors as a function of rank position for three similarity metrics for product descriptions.

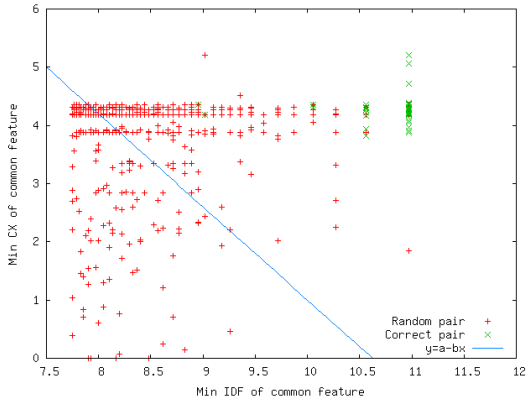

Figure 8: Pairs of bibliography entries x_i, x_j, showing the minimum IDF value of any common feature $f \in F_i \cap F_j$ (on the x axis) and the minimum CX value of any common feature $f \in F_i \cap F_j$ (on the y axis).

set of all pairs x_i, x_j that shared the same hard identifier: there were 131 of these pairs. We will henceforth call this the *known-correct* pair set. (Of course, since only a subset of entries have hard identifiers, this is only a subset of the full set of correct pairs.)

We also generated a set of candidate pairs using the method outlines in Table 5: we picked a set of features F_{canopy}, created inverted indices X_f for all features $f \in F_{\text{canopy}}$, and then used these inverted indices to generate all pairs (x_i, x_j) that contained at least one common feature $f \in F_{\text{canopy}}$. This technique is quadratic in the size of the largest inverted index, so to make this computationally feasible, we let F_{canopy} be the set of all features with $\text{DF}(f) < 1000$. Equivalently, this set could be defined as all features f with $\text{IDF}(f) > \theta$, where θ is (on this dataset) around 7.8 for this dataset. This set contains about 4.4 million pairs, and will be called the *canopy* pair set.

From the canopy set, we sampled a much smaller set of 447 pairs, which we will call the *random* pair set, for the purpose of visualization. In Figure 8, we plot each pair (x_i, x_j) in *random* using a red "+", and also each pair in *known-correct* using a green "×". The position of each pair is chosen so that its x-axis position is the *smallest* IDF value of any common feature $f \in F_i \cap F_j$, and the y-axis position is the *smallest* CX value of any common feature $f \in F_i \cap F_j$.

From inspection of the figure, it is easy to see that the *known-correct* pairs cluster in the top right, while the *random* pairs are more widely dispersed. The pairs in the top right have a common feature that has high IDF, and also common feature with high CX. This suggests that one could restrict the canopy set by using a more restrictive set of features F_{canopy}: specifically, one could either (a) impose a higher threshold on $\text{IDF}(f)$ or (b) combine a threshold on $\text{IDF}(f)$ with a threshold on $\text{CX}(f)$. The blue line in Figure 8 suggests one such combination, namely a linear combination of IDF and CX.

More specifically, the graph shows that most known-correct pairs (all but four) have an x axis position of 10 or greater. Thus, a set of candidate pairs generated by taking all pairs sharing a common feature f with IDF$\geq$ 10 would be expected to include most of the correct, and hence have fairly high recall. Likewise, we can see that relatively few of the pairs in *random* set are to the right of the line $x = 10$ (only 24 of 447). Thus, increasing the threshold from $\theta = 7.8$ to $\theta = 10$ would significantly decrease the size of the canopy (by a factor of around 20), while maintaining high recall. In Figure 8 thus helps to justify the commonly-used heuristic of using high-IDF features to generate canopies.

Looking now at the y-axis position, we see that most (all but eight) of the *known-correct* pairs have an y position of more than 4. This suggests imposing a secondary threshold on $\text{CX}(f)$ for features in F_{canopy}—i.e., defining

$$F_{\text{canopy}} = \{f : \text{IDF}(f) > \theta_1 \text{ and } \text{CX}(f) > \theta_2\}$$

The *canopy* set of 4.4 million pairs described above corresponds to $\theta_1 \approx 7.8$ (or equivalent, $\text{DF}(f) < 1000$) and $\theta_2 > -\infty$. Increasing θ_2 to 3.8 reduces the number of *canopy* by more than 23%, without changing the coverage of the *gold* pairs at all. Alternatively, increasing θ_2 to 4.0 reduces the size of the *canopy* set by more than 40%, and reduces recall on the gold set by only 5%.

While in some practical cases a 40% improvement in speed is worth a modest loss in recall, in this paper we do not use aggressive thresholds on CX (or IDF) in generating candidate pairs. In the matching experiments of Sections 3.3 and 3.2, we used thresholds $\text{DF}(f) < 1000$ and $\text{CX}(f) > 1.5$, which are in each case well below the thresholds associated with any *known-correct* pairs.

4. RELATED WORK

Context-sensitive similarity, as defined here, is closely related to the problem of clustering with *instance-level constraints* [22]. In particular, for the CX.IDF weighting scheme, each context c could be viewed as a set of (soft) *cannot-link constraints* that hold between all pairs x_i and x_j such that $c_i = c_j$. Likewise, for the IDF/CX weighting scheme, each context c could be viewed as a set of soft *must-link* constraints. A number of techniques have been proposed for incorporating such constraints in clustering (e.g., see [1] for an overview of recent research.) One particularly related piece of prior work is the technique of Oyama and Tanaka [17], which, like this paper, focuses on the effect of cannot-link constraints only. Oyama and Tanaka show that if only cannot-link constraints are available, then a distance metric learned using convex quadratic programming methods can be used to improve object identification performance. The work described here differs primarily in its emphasis on very scalable techniques; we also demonstrate that our techniques are applicable to a wide range of information-integration related tasks.

Shen et al [20] introduce a technique called "source-aware matching", which has some points of similarity with context-sensitive similarity. In source-aware matching, instances from many sources are clustered, using different matching methods (e.g., different similarity functions, or different thresholds for the same function) to match instances extracted from different sources. Source-aware matching requires user input for each source (solicited via active learning) to determine parameters of the source-specific matching functions. Shen et al also describe methods for finding a "match plan" that orchestrates sequence of individual source-aware matching operations used to build an overall clustering of the data. In contrast, we use unsupervised methods to learn a single similarity function that can be applied to any pair instances, for any source. To some extent the two techniques are complementary, however: for instance, one could use a "match plan" to orchestrate a series of intra-source merges using a single context-sensitive similarity function.

5. CONCLUDING REMARKS

Deduping, or removing duplicated objects, is an important task in heterogeneous data integration. Here we have proposed a novel modification of IDF similarity that is designed explicitly for large data collections formed by merging together many smaller collections, each of which is duplicate-free (or nearly so). The new metric, CX.IDF, takes into account the "context" from which each description was extracted; however, it accounts for context without adding additional parameters to the similarity function, thus retaining many of the desirable properties of the IDF metric. Like IDF, CX.IDF can be computed efficiently and stored compactly; like IDF, CX.IDF can be "learned" using a single pass over a dataset, and can be implemented easily in a map-reduce framework, allowing simple parallel implementations, and a variant of CX.IDF that supports smoothing with empirically derived priors can be computed nearly as efficiently (with three map-reduce passed over the dataset). Finally, CX.IDF requires no labeled training data—the only additional information that is used is the source (or "context") of each object.

As validation of the metric, we performed k-nn experiments with a suite of nine previously-studied classification tasks, and CX.IDF was shown to reduce error rates by 14-19% relative to an IDF baseline. We also performed experiments in which candidate duplicated pairs were scored and ranked by similarity, and then various prefixes of the ranked lists were evaluated by estimating precision (or simply the number of known errors) for pairs with known "hard" identifiers. In one domain, bibliography entries, approximately 3 million candidate pairs involving over 100,000 bibliography-entry objects were scored using IDF and two varieties of CX.IDF: a highly efficient one-pass version that uses simple *ad hoc* smoothing techniques, and a three-pass version that uses empirically tuned priors. In the second domain, product descriptions, 60 million pairs involving 40 million products were scored and ranked with the same metrics.

The results were qualitatively similar in both domains. The CX.IDF methods greatly outperform the baseline IDF method for high recall levels. In the product domain, at ranks 20-30 million, the empirically-smoothed CX.IDF variant makes 18-57% fewer errors, and the one-pass CX.IDF version makes more than 80% fewer errors. In the bibliography domain, at ranks 50-200 thousand, the empirically-smoothed CX.IDF variant makes 23-35% fewer errors than IDF, and the one-pass CX.IDF version makes more than 65% fewer errors.

At the high-precision end of the ranking, all methods perform extremely well for the product domain: however, the empirically-smoothed CX.IDF version clearly outperforms the baseline, making 41-91% fewer errors for ranks 5-650 thousand. In the smaller bibliography domain, there is no clear difference between the two techniques—they perform comparably for the first 10,000 ranks. In both domains, there is an intermediate range in which the baseline IDF is comparable to, or even better than, either CX.IDF variant: however, this range is relatively narrow (as can be seen in the right-hand plots of Figure 5 and 7).

The results of this paper have evaluated context-sensitive similarity functions in a limited way—via the ordering they induce of pairs of instances. Further work will be needed on how and when context-sensitive similarity functions interact with methods that cluster objects based on multiple types of similarity evidence—while some such approaches are by design similarity-function insensitive (e.g., [3]) others appear to be more sensitive to the function used (e.g., [13]), and still other approaches require probabilistic properties that CX.IDF does not possess (e.g., [18, 4]). Future work will also be needed to evaluate how context-sensitive similarity interacts with systems that directly query a database of similarity-produced clusters (e.g., [15]), or directly query collections of instances using similarity-based queries (e.g., [7]).

The results in this paper have also considered context-sensitive similarity methods of a particular narrow type (CX.IDF and some close variants). In particular, we have considered only similarity functions based using an unordered set of atomic features (sometimes called "term-based" methods [9], since the features are usually tokens.) It is not clear how to extend our approach to context-sensitivity to edit-distance like methods (e.g., [6]) which retain some information about feature ordering. This is an important topic for further research, since edit-distance based methods, while usually more expensive to compute, are sometimes more accurate than feature-based methods

(e.g., [14, 10, 11]); it has also been shown than using ordering information can in some cases speed up "similarity join" computations [23]. Recently, Moreau et al [16] extended the SoftTFIDF similarity function [9] and formalized a family of "robust" hybrid similarity functions that combine aspects of edit-distance similarities and feature-based similarities; this may be an appropriate starting point for extending context-sensitive similarity functions.

6. REFERENCES

[1] Sugato Basu, Ian Davidson, and Kiri Wagstaff, editors. *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC Press, 2008.

[2] R. J Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, page 131âĂŞ140, 2007.

[3] O. Benjelloun, H. Garcia-Molina, D. Menestrina, Q. Su, S. E Whang, and J. Widom. Swoosh: A generic approach to entity resolution. *The VLDB Journal*, 18(1):255âĂŞ276, 2009.

[4] I. Bhattacharya and L. Getoor. A latent Dirichlet model for unsupervised entity resolution. In *SIAM International Conference on Data Mining*, pages 47–58, 2006.

[5] Mikhail Bilenko, Sugato Basu, and Mehran Sahami. Adaptive product normalization: Using online learning for record linkage in comparison shopping. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 58–65, Washington, DC, USA, 2005. IEEE Computer Society.

[6] Mikhail Bilenko, William W. Cohen, Stephen Fienberg, Raymond J. Mooney, and Pradeep Ravikumar. Adaptive name-matching in information integration. *IEEE Intelligent Systems*, 18(5):16–23, September/October 2003.

[7] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, July 2000.

[8] William W. Cohen and Haym Hirsh. Joins that generalize: Text categorization using WHIRL. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 169–173, New York, NY, 1998.

[9] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web (IIWeb-03)*, 2003.

[10] William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, 2002.

[11] R. da Silva, R. Stasiu, V. M Orengo, and C. A Heuser. Measuring quality of similarity functions in approximate data matching. *Journal of Informetrics*, 1(1):35âĂŞ46, 2007.

[12] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI'04: Sixth Symposium on Operating System Design and Implementation*, 2004.

[13] Hui Han, Hongyuan Zha, and C. Lee Giles. Name disambiguation in author citations using a k-way spectral clustering method. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 334–343, New York, NY, USA, 2005. ACM.

[14] Andrew McCallum, Kamal Nigam, and Lyle H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Knowledge Discovery and Data Mining*, pages 169–178, 2000.

[15] D. Mimno, A. McCallum, and G. Miklau. Probabilistic representations for integrating unreliable data sources. In *Information integration on the web (IIWeb), papers from the 2007 AAAI Workshop*, page 74âĂŞ79, 2007.

[16] E. Moreau, F. Yvon, and O. Cappe. Robust similarity measures for named entities matching. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008.

[17] Satoshi Oyama and Katsumi Tanaka. Distance metric learning from cannot-be-linked example pairs, with application to object identification. In *Constrained Clustering: Advances in Algorithms, Theory, and Applications*. Chapman & Hall/CRC Press, 2008.

[18] Hanna Pasula, Bhaskara Marthi, Brian Milch, Stuart Russell, and Ilya Shpitser. Identity uncertainty and citation matching. In *Advances in Neural Processing Systems 15*, Vancouver, British Columbia, 2002. MIT Press.

[19] Gerard Salton, editor. *Automatic Text Processing*. Addison Welsley, Reading, Massachusetts, 1989.

[20] W. Shen, P. DeRose, L. Vu, A. H Doan, and R. Ramakrishnan. Source-aware entity matching: A compositional approach. In *IEEE 23rd International Conference on Data Engineering, 2007. ICDE 2007*, page 196âĂŞ205, 2007.

[21] Howard Turtle and James Flood. Query evaluation: strategies and optimizations. *Information processing and management*, 31(6):831–850, November 1995.

[22] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1103–1110, 2000.

[23] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 131–140, New York, NY, USA, 2008. ACM.