Adaptive Product Normalization: Using Online Learning for Record Linkage in Comparison Shopping

Mikhail Bilenko and Sugato Basu
Dept. of Computer Sciences
University of Texas at Austin
{mbilenko,sugato}@cs.utexas.edu

Mehran Sahami
Google Inc.
sahami@google.com

Abstract

The problem of record linkage focuses on determining whether two object descriptions refer to the same underlying entity. Addressing this problem effectively has many practical applications, e.g., elimination of duplicate records in databases and citation matching for scholarly articles. In this paper, we consider a new domain where the record linkage problem is manifested: Internet comparison shopping. We address the resulting linkage setting that requires learning a similarity function between record pairs from streaming data. The learned similarity function is subsequently used in clustering to determine which records are co-referent and should be linked. We present an online machine learning method for addressing this problem, where a composite similarity function based on a linear combination of basis functions is learned incrementally. We illustrate the efficacy of this approach on several real-world datasets from an Internet comparison shopping site, and show that our method is able to effectively learn various distance functions for product data with differing characteristics. We also provide experimental results that show the importance of considering multiple performance measures in record linkage evaluation.

1   Introduction

Record linkage is the problem of identifying when two (or more) *references* to an object are describing the same true entity. For example, an instance of record linkage would be identifying if two paper citations (which may be in different styles and formats) refer to the same actual paper. Addressing this problem is important in a number of domains where multiple users, organizations, or authors may describe the same item using varied textual descriptions.

Historically, one of the most examined instances of record linkage is determining if two database records for a person are referring to the same individual, which is an important data cleaning step in applications from direct marketing to survey response (e.g., the US Census). More recently, record linkage has found a number of applications in the context of several web applications, e.g., the above-mentioned task of identifying paper citations that refer to the same publication is an important problem in on-line systems for scholarly paper searches, such as *CiteSeer* and *Google Scholar*. Additionally, record linkage has been a topic of interest in natural language processing research, where it is known as the co-reference resolution or named entity disambiguation problem [21, 24].

As we show in this paper, record linkage is a key component of on-line comparison shopping systems. When many different web sites sell the same product, they provide different textual descriptions of the product (which we refer to as "offers"). Thus, a comparison shopping engine is faced with the task of determining which offers are referring to the same true underlying product. Solving this *product normalization* problem allows the shopping engine to display multiple offers for the same product to a user who is trying to determine from which vendor to purchase the product. Accurate product normalization is also critical for data mining tasks such as analysis of pricing trends.

In such a context, the number of vendors and sheer number of products (with potentially very different characteristics) make it difficult to manually craft a single similarity function that can correctly determine if two arbitrary offers refer to the same product. Moreover, for different categories of products, different similarity functions may be needed to capture the notion of equivalence for each category. Hence, an approach that allows learning similarity functions between offers from training data becomes necessary.

Furthermore, in many record linkage tasks including product normalization, records to be linked contain multiple fields (e.g., product name, manufacturer, price, etc.). Such records may either come in pre-structured form (e.g., XML or relational database records), or the fields may have been extracted from an underlying textual description [12]. While it may be difficult for a domain expert to specify a complete similarity function between two records, they are

often capable of defining similarity functions between individual record *fields*. For example, it is relatively simple to define the similarity between two prices as a function related to the inverse of the difference of the prices, or the difference between two textual product descriptions as the (well-known) cosine between their vector-space representations. Thus, an appropriate learnable similarity function for comparing records must be able to leverage multiple *basis* similarity functions that capture individual field similarities.

An important property of the product normalization domain is the fact that new data is becoming available *continuously* as product offers arrive from merchants. At the same time, a small proportion of the incoming product offers includes the Universal Product Code (UPC) attribute which uniquely identifies products. This provides a continuous source of supervision, therefore a learning approach to the linkage problem in such settings must be able to readily utilize new training data without having to retrain anew on previously seen data. *Online* learning algorithms are methods that process training examples incrementally, and employing such an algorithm provides the most flexibility in a record linkage setting where new data is arriving as a stream.

In this paper, we present an online learning approach for the record linkage problem in the product normalization setting, in which a similarity function is trained by inducing a linear combination of basis similarity functions between the fields of different offers. The weights of basis functions in the linear combination are learned from labeled training data using a version of the voted perceptron algorithm [14], which is very efficient in an online learning setting for large volumes of streaming data. This algorithm can also be deployed in batch-mode learning using standard online-to-batch conversion techniques.

We show empirical results with our proposed method on several real-world product normalization tasks. Notably, we show that different similarity functions are learned for different categories of products, and identify the strengths and weaknesses of the approach. We also compare three linkage rules for identifying co-referent records, one of which corresponds to a simple pairwise approach, and two others make linkage decisions between record pairs collectively. Finally, we provide some general observations regarding evaluation methodology for record linkage tasks and present an argument for utilizing multiple accuracy measures when evaluating record linkage solutions.

2 Background and Motivation

2.1 Record Linkage

The problem of identifying co-referent records in databases has been studied in the research community under various names, notably record linkage [13], the merge/purge problem [35], duplicate detection [25, 29, 4], reference/citation matching [23, 20], entity name matching and clustering [8], hardening soft databases [7], identity uncertainty [28], and robust reading [21].

The majority of solutions for record linkage treat it as a modular problem and consist of multiple stages. In the first stage, a function is selected for computing the similarity between all pairs of potentially co-referent records. This similarity function can either be hand-tuned or its parameters can be learned from training data. In the second stage, a blocking method is used to select a set of candidate record pairs to be investigated for co-reference, since it is typically prohibitively expensive to compute pairwise similarities between all pairs of records in a large database. In the final linkage stage, similarity is computed between candidate pairs, and highly similar records are identified and linked as describing the same entity. This can be achieved either via *pairwise* or via *collective* inference over individual record pairs.

Pairwise approaches classify all candidate record pairs into two categories: "matches" or "non-matches", where each candidate pair is classified independently of others. Some of the methods that employ the pairwise approach include the EM-based technique for finding optimal entity matching rules [34], the sorted neighborhood method for limiting the number of potential candidate pairs [16], and the domain-independent three-stage iterative merging algorithm for duplicate detection [25].

In contrast, collective linkage methods take a more global view instead of independently processing each candidate pair. These methods consider multiple linkage decisions in conjunction, and perform simultaneous inference to find groups of co-referent entries that map to the same underlying entity. Recently proposed algorithms that belong to this category include context-sensitive duplicate inference by iterative processing of the data [3], learning a declarative relational probability model that explicitly specifies the dependencies between related linkage decisions [28], and multi-relational inference using conditional probabilistic models to simultaneously detect matches for all related candidate pairs [27, 24]. While some of these collective approaches have been shown to be more accurate than the pairwise approach on certain domains, the simultaneous inference process makes these methods more computationally intensive.

2.2 Learning similarity functions

An important aspect of any record linkage system is the choice of the function that is used to compute similarity between records. Records in databases generally have multiple attributes of different types, each of which has an associated similarity measure. For instance, string similarity measures such as edit distance [15] or cosine similar-

ity [1] can be used to compare textual attributes like product name or paper title, while numerical functions (e.g., relative difference) can be used for real-valued attributes like price. Several recent papers have studied the problem of combining such *basis* similarity functions to obtain a composite measure of similarity between two database entries [8, 29, 32, 4]. Learning a composite similarity measure from several basis similarity functions also is related to recent research on learning composite kernels, where techniques such as boosting [10], optimizing kernel alignment [11], and semi-definite programming [19] are employed to learn an effective combination of simple base kernels from training data.

An adaptive framework for learning similarity functions is critical in the product normalization setting since the notion of offer similarity is highly domain-dependent. For example, if linkage is performed between book offers, equivalence of product names (book titles and author names) is highly indicative of co-referent book records. For electronic products, in contrast, product name similarity is insufficient to link records: offers named *"Toshiba Satellite M35X-S309 notebook"* and *"Toshiba Satellite M35X-S309 notebook battery"* have a high textual similarity but refer to different products. At the same time, for electronic items price similarity is an important indicator of offer equivalence: the notebook and the battery records have very different prices, indicating that they are not co-referent. Thus, composite similarity between product offers must be adapted to a particular domain using training examples.

An adaptive approach to product normalization can exploit two unique domain characteristics: (1) availability of limited (and noisy) training data since a small percentage of offers contains the Universal Product Codes (UPCs) that uniquely identify products, and (2) the streaming nature of the data: new product offers are continuously added to the database, calling for a similarity learning method that can utilize incoming training data incrementally. In the remainder of the paper, we describe a linkage framework suitable for this setting. Overall, the main contributions of the paper can be summarized as follows:

- We introduce a novel application domain of product normalization for comparison shopping as an instance of the record linkage problem in a data stream setting;

- We adapt an efficient online learning algorithm (averaged perceptron) for learning the similarity function for record linkage;

- We compare different types of hierarchical clustering algorithms and evaluate the accuracy of the linkage decisions they generate for the product normalization task;

- We contrast different evaluation metrics for the record linkage task and identify important issues relating to the choice of evaluation measures.

3 Approach

Our proposed approach to product normalization is a modular framework that consists of several components: an initial set of basis functions to compute similarity between fields of records to be linked, a learning algorithm for training the parameters of a composite similarity function, a method for generating linkage candidates to avoid computing similarity between all pairs of records, and, finally, the clustering-based linkage step. The following subsections describe the details of each of these components.

3.1 Basis Functions

In formulating our approach, we begin with a set of K basis functions $f_1(R_1, R_2), ..., f_K(R_1, R_2)$, defined as similarity functions between fields of records R_1 and R_2. We then learn a linear combination of these basis functions with K corresponding weights α_i and an additional threshold parameter α_0 to create a composite similarity function, f^*:

$$f^*(R_1, R_2) = \alpha_0 + \sum_{i=1}^{K} \alpha_i f_i(R_1, R_2)$$

Values provided by f^* are not constrained to be positive: the learning formulation below assumes that the threshold α_0 may take on a negative value so that for pairs of records that are not equivalent f^* returns a negative value. Once trained, f^* can be used to produce a similarity matrix S over all pairs of records. In turn, S can be used with any similarity-based clustering algorithm to identify clusters, each of which contains a set of records which presumably should be linked. Then, we can interpret each cluster as a set of records referring to the same true underlying item.

3.2 Training the Composite Similarity Function

Identifying co-referent records requires classifying every candidate pair of records as belonging to the class of co-referent pairs $\mathcal{M}$ or non-equivalent pairs $\mathcal{U}$. Given some domain Δ_R from which each record is sampled, and K similarity functions $f_k : \Delta_R \times \Delta_R \to \mathbb{R}$ that operate on pairs of records, we can produce a *pair-space vector* $\boldsymbol{x}_i \in \mathbb{R}^{K+1}$ for every pair of records (R_{i_1}, R_{i_2}): $\boldsymbol{x}_i = [1, f_1(R_{i_1}, R_{i_2}), ..., f_K(R_{i_1}, R_{i_2})]^T$. The vector includes the K values obtained from basis similarity functions concatenated with a default attribute that always has value 1, which corresponds to the threshold parameter α_0.

Any binary classifier that produces confidence scores can be employed to estimate the overall similarity of a record pair (R_{i_1}, R_{i_2}) by classifying the corresponding feature vector $\boldsymbol{x}_i$ and treating classification confidence as similarity. The classifier is typically trained using a corpus of labeled data in the form of pairs of records that are known to be either co-referent $((R_{i_1}, R_{i_2}) \in \mathcal{M})$ or non-equivalent $((R_{i_1}, R_{i_2}) \in \mathcal{U})$.

In previous work a number of classifiers have been successfully utilized for this purpose, including Naive Bayes [34], decision trees [32, 33], maximum entropy [8], and Support Vector Machines [29, 4]. These classifiers have been deployed in *batch* settings where all training data is available in advance. Since in a product normalization setting new data is arriving continuously, an *online* classifier that can be trained incrementally is required. For this reason, we employ averaged perceptron [9], a space-efficient variation of the voted perceptron algorithm proposed and analyzed by Freund and Schapire [14].

The averaged perceptron is a linear classifier: given an instance x_i, it generates a prediction of the form $\hat{y}_i = \alpha_{avg} \cdot x_i$, where α_{avg} is a vector of $(K+1)$ real weights. Weights are averaged over all weight vectors observed during the training process (as opposed to just using the final weight vector in the regular perceptron algorithm). In our approach, the weights directly correspond to the weights of basis similarity functions in the composite function described in previous section. Each x_i is a pair-space vector defined above, and we assign label -1 to class $\mathcal{U}$ of non-equivalent record pairs, and label $+1$ to class $\mathcal{M}$ of co-referent record pairs.

Voted perceptron has several properties which make it particularly appropriate for the large-scale streaming linkage task. First and foremost, is a highly efficient online learning algorithm: the hypothesis (similarity function parameters) that it generates is updated as more labeled examples become available without the need to re-train on all previously seen training data. Second, voted perceptron is a linear classifier that produces a hypothesis which is intuitive and easily interpretable by humans as relative importance of basis similarity functions, a highly attractive property for a system to be deployed and maintained on a continuous real-world task. Finally, voted perceptron is a discriminative classifier with strong theoretical performance guarantees [14]. While several previously used classifiers use linear hypotheses, online variants of decision trees have been proposed, and SVMs are also discriminative classifiers with strong theoretical guarantees, none of the previously used classifiers combine all benefits of voted perceptron.

Figure 1 shows the training algorithm for learning the parameters α_{avg}. The algorithm can be viewed as minimizing the cumulative hinge loss suffered on a stream of examples [30]. As every training record pair (R_{i_1}, R_{i_2}, y_i) with a corresponding feature vector x_i is presented to the learner, it incurs a (hinge) loss $l(x_i, y_i) = \max\{-y_i \alpha \cdot x_i, 0\}$, and the vector of weights α is updated in the direction of the gradient to reduce the loss: $\alpha = \alpha - \frac{\delta l(x_i, y_i)}{\delta \alpha}$. Intuitively, this training procedure corresponds to iterative evaluation of the prediction for every training pair, and if the prediction differs from the true label, the weights are adjusted to correct for the error. This view can lead to variations

---

Algorithm: AVERAGED PERCEPTRON TRAINING
Input: Training set of record pairs $\{(R_{i_1}, R_{i_2}, y_i)\}$, $y \in \{-1, +1\}$
      number of epochs T
      similarity functions $\mathcal{F} = \{f_i(\cdot, \cdot)\}_{i=1}^K$
Output: Weight vector $\alpha_{avg} = \{\alpha_i\}_{i=0}^K$
Algorithm:
  Initialize $\alpha_{avg} = \alpha = 0$
  Initialize $x_i = [1, f_1(R_{i_1}, R_{i_2}), ..., f_k(R_{i_1}, R_{i_2})]$ for $i = 1...M$
  For $t = 1...T$
      For $i = 1...M$
         Compute $\hat{y}_i = \text{sign}(\alpha \cdot x_i)$
         If $\hat{y}_i \neq y_i$
            $\alpha = \alpha + y_i x_i$
         $\alpha_{avg} = \alpha_{avg} + \alpha$
  $\alpha_{avg} = \frac{\alpha_{avg}}{T \cdot M}$

Figure 1. The training algorithm

of the algorithm using other loss functions, e.g. log-loss $l_{\log}(x_i, y_i) = \ln(1 + \exp(-y_i \alpha \cdot x_i))$. In our experiments, varying the loss function did not lead to a qualitative difference in final performance on the linkage task, so we only report results obtained using hinge loss.

3.3 Computing a Similarity Matrix

Once a composite similarity function f^* with weights α_{avg} has been trained to correctly predict whether a pair of records is co-referent, it can be used to compute an $m \times m$ similarity matrix $S = \{s_{ij}\}$ between all pairs of records considered for linkage, where each $s_{ij} = f^*(R_i, R_j), 1 \leq i, j \leq m$. Thus, the task of training a composite similarity function over pairs of records can be viewed as the problem of learning a matrix of similarity values where only a few values in the matrix are initially given (i.e., the labeled training pairs $((R_{i_1}, R_{i_2}), y_i)$. During training the sign of the known entries in the matrix is constrained, and they are used to estimate the remaining matrix values, each of which is a linear combination of basis similarity functions for the underlying record pair.

However, we note that for a large number of records m, computing the full similarity matrix (an $O(m^2)$ operation) may be impractical since the vast majority of record pairs are not co-referent. Therefore, it is necessary to select a subset of *candidate pairs* between which similarity is computed. Several approaches for limiting the number of candidate pairs have been proposed in the literature, such as blocking, where candidate pairs are required to share some attribute values [18], sorted neighborhood methods, where only records within a sliding window over a database sorted on several keys are considered [16], and canopies, where a computationally cheap similarity function is employed in a preliminary pass to obtain clusters of records within which linkage is performed [23]. An experimental comparison of several methods as well as several new methods based on an inverted index can be found in recent work by Baxter et al. [2]. In current work, we employ a variant of the canopies

ProductName	Brand	Price	RawDescription
Canon EOS 20D Digital SLR Body Kit (Req. Lens) USA	Canon	1499.00	Canon EOS 20d digital camera body (lens not included), BP511a battery, CG580 battery charger, USB cable, Video cable, instructions, warranty, 3 CDROM software discs, Wide strap.
Canon EOS 20d Digital Camera Body USA - Lens sold separately	Canon USA	1313.75	Canon EOS 20D is a digital, single-lens reflex, AF/AE camera with built-in flash, providing 8.2 megapixel resolution and up to 23 consecutive frames at 5fps.

Table 1. Sample records from the *DigitalCameras* dataset

method where we limit candidate pairs to those having a minimum token overlap in one of the attributes. Similarity is not computed for records that do not share a canopy, and they are not allowed to merge in the clustering process.

3.4 Linkage via Clustering

Armed with a similarity matrix, we face the subsequent problem of identifying groups of equivalent records. As described in Section 2, two primary approaches have been previously used for the actual task of linking co-referent records: (1) pairwise linkage followed by transitive closure, and (2) collective approaches where linkage decisions between multiple pairs are made in conjunction. In this work, we compare the linkage performance of three variants of the Hierarchical Agglomerative Clustering (HAC) algorithm, of which one, single-link HAC, is a pairwise approach, and the other two (group-average HAC and complete-link HAC) are bottom-up greedy collective approaches [17].

Given a combined similarity function trained as described above, hierarchical agglomerative clustering initially places every record in its own singleton cluster. Then, at every step, two clusters which are most similar according to a chosen linkage rule are merged. With single-link linkage, similarity between clusters is defined as the *highest* similarity of any individual cluster members; with group-average linkage, similarity between clusters is defined as the *average* similarity of all pairs of objects from the two clusters; and with complete-link linkage, similarity between clusters is defined as the *lowest* similarity of any individual cluster members. Thus, group-average and complete-link HAC variants are greedy but efficient collective linkage techniques since the merge decision for every record pair depends on similarities of other pairs from the clusters to which the records under consideration belong.

4 Experimental Methodology

We have evaluated our approach on three datasets sampled from the FROOGLE comparison shopping site: *DigitalCameras* (4823 product offers), *Camcorders* (4531 product offers), and *Luggage* (6912 product offers). Each of the datasets was created by selecting a corresponding category in the product hierarchy and sampling product offers that contain the UPC (Universal Product Code) field, which uniquely identifies any commercial product. While

less than 10% of all data includes the UPC values, they provide the "golden standard" labels for evaluating linkage accuracy. In our experiments, the UPC values were only used for evaluation purposes, while in an actual fielded product normalization system they can be used as a highly useful attribute for computing overall similarity (although our results below indicate that UPC values alone should not be used as a single linkage criterion due to the presence of noise and certain domain artifacts discussed below).

Every product offer contains several fields, which include textual attributes such as `ProductName` and `Brand`, as well as numerical attributes such as `Price`, and categorical features such as `ProductCategory`. Table 1 presents an example of co-referent records from the *DigitalCameras* dataset.

We have used several base similarity functions f_k. For all string attributes, cosine similarity was used [1], along with a variant of string edit distance [15] on extracted model strings for electronic products. For numeric attributes, relative difference $f_{diff}(a,b) = \frac{|a-b|}{\max\{a,b\}}$ was used. For categorical attributes, the inverse of path length in the FROOGLE category hierarchy was used.

In our experiments, every dataset was split into two folds for each trial. Every record was randomly assigned to one of the two folds, and cross-validation was performed with one of the folds used for training, in which the overall similarity function was trained as described in Section 3.2. The two folds were then merged, and clustering was done on the entire dataset. Performance statistics were collected separately for the pairs on the training and testing folds, as well as the entire dataset. This methodology corresponds to the "consulting" linkage evaluation methodology [5], where the test fold contains records corresponding both to entities seen in training data as well as entities for which records can only be found in the test set.

We note that in an operational on-line environment, it is not necessary to re-cluster all previous data whenever a new record becomes available. Rather, for efficiency, every new record can be assigned to the existing cluster closest to it as long as the record's similarity to that cluster is above some predetermined threshold; otherwise the record is to be placed in a new singleton cluster. While we make note of this point, the operational set-up of the system is not the core focus of this paper, and thus our experimental results
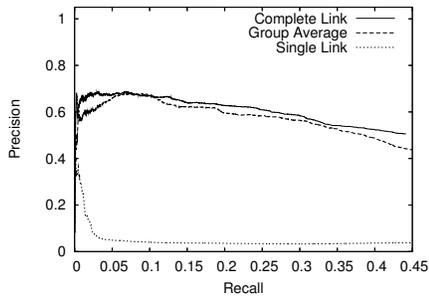
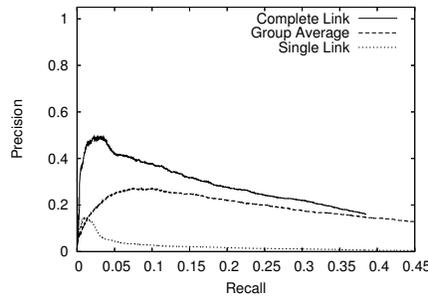**Figure 2. Test set precision-recall curve for** *DigitalCamera*

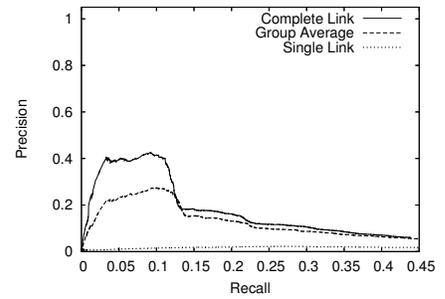**Figure 3. Test set precision-recall curve for** *Camcorders*

Figure 4. Test-set precision-recall curve for *Luggage*

focus on batch clustering of all available data to accurately evaluate the efficacy of the learning approach and each clustering method. The problem of making merge decisions efficiently for incoming records has been recently studied in the context of data cleaning in data warehouses [6], and fielded systems should combine that method with our approach for online learning of similarity functions.

In the clustering process, precision, recall and F-measure defined over pairs of duplicates were computed after each merge step in the clustering process. Precision is the fraction of identified co-referent pairs that are correct, recall is the fraction of co-referent pairs that were identified, and F-measure is the harmonic mean of precision and recall. While traditional use of precision-recall curves involves interpolating several precision points at standard recall levels to the highest value seen for any lower recall, we have found that such interpolation may grossly misrepresent the dynamics of the clustering process. Therefore, we report non-interpolated (observed) precision values, averaged over 10 trials of 2 folds each.

5 Results and Discussion

Figures 2-4 show precision-recall curves on the 3 datasets. Different points on the curves were obtained by successively merging the two most similar clusters during the clustering process, and measuring the resulting precision and recall on records from the test set. Going along a precision-recall curve from left to right, more records are grouped into the same cluster as clusters are successively merged – this typically results in higher recall for co-referent records but decreases the precision if non-equivalent records with low similarity are put into the same cluster.

In the observed precision-recall plots, there is an initial drop followed by an immediate rise of precision at low recall values. To take a closer look at the results, we zoom in on the early part of the precision-recall curve in Figure 5,
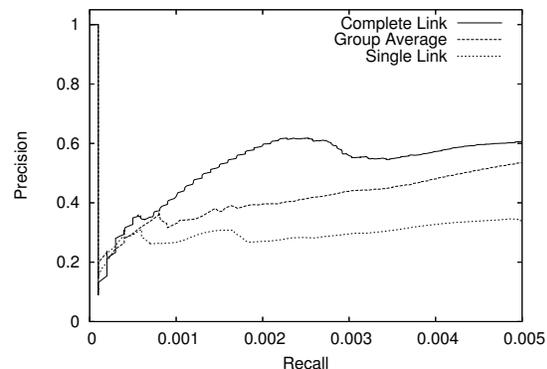

Figure 5. Early part of precision-recall curve for *DigitalCamera*

which shows a sharp drop in precision soon after the beginning. Analysis of the experiment traces reveals that the observed decrease in precision is due to erroneous UPC identifiers provided by the merchants on some of the product offers, leading to label noise. Because a number of the erroneously labeled offers are highly similar to correctly labeled co-referent offers, they were merged at very early stages, resulting in the observed precision decrease. Inspection of the clustering traces revealed that these initial co-reference predictions were actually correct but were marked as mistakes due to the erroneous UPCs. As more clusters were merged, subsequent correct linkages of offers with proper UPC identifiers cancelled out this effect, bringing the precision back up. This recovery of precision with increasing recall due to cluster merging continued till more heterogeneous clusters started to get merged. This resulted in some non-coreferent offers being put into the same cluster, making the clusters impure and decreasing the overall precision, as can be seen in Figure 2.

Besides the noisy UPC values, we observed other domain artifacts that lowered the precision numbers obtained

in our experiments. For example, differently colored variants of the same product often have different UPC labels. Their linkage therefore penalizes observed precision values, although these offers are co-referent for comparison shopping purposes. Nevertheless, compared to a human expert-constructed similarity function computed on the same attributes (which we considered as a baseline), the performance of the learned weights was significantly higher, improving the accuracy of actual fielded linkage systems.

In all the experiments, complete-link HAC outperformed group-average HAC, which in turn performed better than single-link HAC. These results conclusively show that for the product normalization task it is recommended to use complete-link linkage either in batch or online settings. This can be explained by the fact that complete-link linkage avoids elongated clusters created by the single-link rule due to the chaining effect. Chaining occurs as offers are added to a cluster because of high similarity to a single cluster member, leading to long "chains" of linked offers, while offers at the extremes of the chain have low similarity, leading to error accumulation. In contrast, complete-link linkage yields tight, well-separated clusters, while the nature of clusters obtained by group-average linkage is typically in between single-link and complete-link. Overall, the clusters obtained by collective linkage methods are best suited for the product normalization task.

While our online approach to learning the similarity function is highly scalable, computational efficiency of the clustering algorithm used for linkage is a big concern. Even though HAC is an $O(n^2 \log n)$ algorithm [22], it can be efficiently implemented in practice. If batch clustering is desired, all the three variants of HAC can be parallelized effectively, using different parallel computation models and data structures [26]. It is also easy to combine blocking techniques (e.g., canopy creation) for data-preprocessing with hierarchical clustering, further increasing its efficiency.

Learning the similarity function from training data gave different weights for the various attributes of the 3 datasets. For example, the weights corresponding to similarity between `Brand` and the `ModelStrings` attributes in the *Luggage* dataset were higher than that for *Camcorders* or *DigitalCameras*, which can be explained by the fact that these attributes tend to identify luggage products uniquely more often than electronic items. These results show the importance of learning individual similarity functions for different product categories in the product normalization setting.

An important component of the hierarchical cluster merging stage is deciding when to stop the clusters from being merged further, so that the remaining clusters can be reported as groups of co-referent records. While one approach is to choose the number of clusters which gives the peak F-Measure value in the precision-recall curve, it is advisable to look at other cluster evaluation measures be-
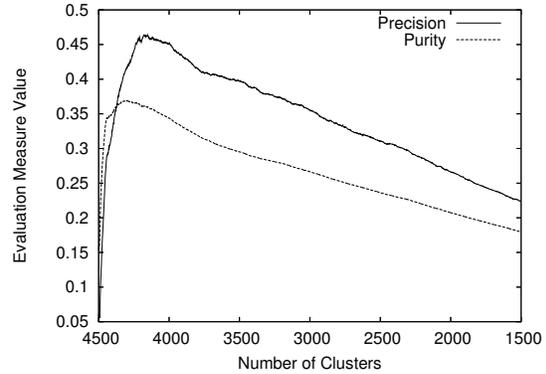

Figure 6. Purity and Precision values Vs Number of clusters for *Camcorders*

fore making this choice. Another popular cluster evaluation measure used by clustering practitioners is average cluster purity [31]. It measures the average proportion of the dominant class label in a cluster – a higher cluster purity measure typically implies purer clusters. Figure 6 shows how the pairwise precision and purity values change with the number of clusters for the *Camcorders* dataset, as clusters are successively merged in the complete-link HAC algorithm. The purity measure peaks around 4300 clusters, while precision has a peak value at 4200 clusters. Depending on whether having homogeneous clusters or pairwise accuracy is a higher priority, one of these measures may be preferable. Thus, it is necessary to keep several metrics in mind when deciding at which level to stop merging to trade off between correct pairwise linkage and overall cluster purity, one of which may be preferable in a real-world linkage system.

6 Conclusions

The record linkage problem is ubiquitous in modern large-scale databases, and in this paper we have introduced a scalable, adaptive approach for product normalization, an instance of linkage critical in online comparison shopping systems. Our method relies on an online learning algorithm for training a combined similarity function, permitting it to be deployed in settings where data is arriving continuously. Our experimental results show the efficacy of the proposed approach. We have found that linkage methods which consider similarity of multiple pairs of records jointly lead to significantly higher performance than a pairwise approach where each linkage decision is made in isolation. Additionally, we have observed that previously employed clustering and linkage evaluation metrics have non-overlapping maxima, suggesting that for real-world tuning of record linkage systems several evaluation measures should be considered.

7 Acknowledgments

References

[1] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[2] R. Baxter, P. Christen, and T. Churches. A comparison of fast blocking methods for record linkage. In *Proc. of the KDD-2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pp. 25–27, 2003.

[3] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *Proc. of the SIGMOD-2004 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD-04)*, pp. 11–18, 2004.

[4] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proc. of ACM SIGKDD 2003*, pp. 39–48, 2003.

[5] M. Bilenko and R. J. Mooney. On evaluation and training-set construction for duplicate detection. In *Proc. of the SIGKDD 2003 Workshop on Data Cleaning, Record Linkage, and Object Consolidation*, pp. 7–12, 2003.

[6] S. Chaudhuri, K. Ganjam, V. Ganti, and R. Motwani. Robust and efficient fuzzy match for online data cleaning. In *Proc. of ACM SIGMOD 2003*, pp. 313–324, 2003.

[7] W. W. Cohen, H. Kautz, and D. McAllester. Hardening soft information sources. In *Proc. of ACM SIGKDD 2000*, pp. 255–259, 2000.

[8] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proc. of ACM SIGKDD 2002*, pp. 475–480, 2002.

[9] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of EMNLP-2002*, pp. 1–8, 2002.

[10] K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *NIPS 15*, pp. 537–544, 2003.

[11] N. Cristianini, A. Elisseeff, J. Shawe-Taylor, and J. Kandola. On kernel target alignment. In *NIPS 14*, pp. 367–373, 2002.

[12] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the World-Wide Web. In *Proc. of Agents-1997*, pp. 39–48, 1997.

[13] I. P. Fellegi and A. B. Sunter. A theory for record linkage. *J. of the American Statistical Association*, 64(328):1183–1210, Dec. 1969.

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

[15] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, 1997.

[16] M. A. Hernández and S. J. Stolfo. The merge/purge problem for large databases. In *Proc. of ACM SIGMOD-1995*, pp. 127–138, 1995.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[18] M. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *J. of the American Statistical Association*, 84:414–420, 1989.

[19] G. R. G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *J. of Machine Learning Research*, 5:27–72, 2004.

[20] S. Lawrence, K. Bollacker, and C. L. Giles. Autonomous citation matching. In *Proc. of Agents-1999*, pp. 392–393, 1999.

[21] X. Li, P. Morie, and D. Roth. Robust reading: Identification and tracing of ambiguous names. In *Proc. of NAACL-2004*, pp. 17–24, 2004.

[22] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[23] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of ACM SIGKDD-2000*, pp. 169–178, 2000.

[24] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS 17*, pp. 905–912, 2005.

[25] A. E. Monge and C. P. Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proc. of SIGMOD 1997 Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 23–29, 1997.

[26] C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21:1313–1325, 1995.

[27] Parag and P. Domingos. Object Identification with Attribute-Mediated Dependences. In *Proc. of PKDD-2005*, 2005.

[28] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity uncertainty and citation matching. In *NIPS 15*, pp. 1401–1408, 2003.

[29] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proc. ACM SIGKDD-2002*, pp. 269–278, 2002.

[30] S. Shalev-Shwartz, Y. Singer, and A. Y. Ng. Online and batch learning of pseudo-metrics. In *Proc. of ICML-2004*, pp. 743–750, 2004.

[31] A. Strehl, J. Ghosh, and R. J. Mooney. Impact of similarity measures on web-page clustering. In *AAAI-2000 Workshop on Artificial Intelligence for Web Search*, pp. 58–64, 2000.

[32] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proc. of ACM SIGKDD-2002*, pp. 350–359, 2002.

[33] V. S. Verykios, G. V. Moustakides, and M. G. Elfeky. A Bayesian decision model for cost optimal record matching. *The VLDB Journal*, 12(1):28–40, 2003.

[34] W. E. Winkler. Using the EM algorithm for weight computation in the Fellegi-Sunter model of record linkage. *American Statistical Association, Proc. of the Section on Survey Research Methods*, pp. 667–671, 1988.

[35] W. E. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Census Bureau, Washington, DC, 1999.