Group based Self Training for E-Commerce Product Record Linkage

Wayne Xin Zhao[1,2], **Yuexin Wu**[2], **Hongfei Yan**[2] **and Xiaoming Li**[2]
[1]School of Information, Renmin University of China, China
[2]School of Electronic Engineering and Computer Science, Peking University, China
batmanfly@gmail.com, wuyuexin@gmail.com,
yhf1029@gmail.com, lxm@pku.edu.cn

Abstract

In this paper, we study the task of product record linkage across multiple e-commerce websites. We solve this task via a semi-supervised approach and adopt the self-training algorithm for learning with little labeled data. In previous self-training algorithms, the learner tries to convert the most confidently predicted unlabeled examples of each class into labeled training examples. However, they evaluate the confidence of an instance only based on the individual evidence from the instance. The correlation among data instances is rarely considered.

To address it, we develop a novel variant of the self-training algorithm by leveraging the data characteristics for the task of product record linkage. We joint consider a candidate linked pair and its corresponding correlated pairs as a group at the selection of pseudo labeled data. We propose a novel confidence evaluation method for a group of instances, and incorporate it as a re-ranking step in the self-training algorithm. We evaluate the novel self-training algorithm on two large datasets constructed based on real e-commerce Websites. We adopt several competitive methods as comparisons and perform extensive experiments. The results show that our method outperforms these baselines that do not consider data correlation.

1 Introduction

Recent years have witnessed the rapid development of online e-commerce business, e.g. Amazon and eBay, which raises the need for better storing, organizing and analyzing the large amount of product records. An important task is how to effectively link product records across multiple databases or websites. This task serves as a fundamental step for many applications. For example, it will be useful to provide entity-oriented search and product comparison analysis in eBay, where record linkage can help to unify the corresponding records (i.e. records from different sellers) given a product. Record linkage has been shown to be important in many fields, including biology (Needleman and Wunsch, 1970), database (Neiling, 2006) and text mining (Goiser and Christen, 2006; Bilenko and Mooney, 2003). In this paper, we mainly focus on the task of product record linkage for online e-commerce websites, but our method is easy to be extended to other data sources and tasks.

Early studies on record linkage were mainly based on the classical probabilistic approach developed by Fellegi and Sunter (1969), furthermore it was improved by the application of the expectation-maximization (EM) algorithm (Winkler, 1988) and the use of approximate string comparison algorithms (Christen, 2006; Winkler, 2006). The early work was not flexible to incorporate rich information. The development of machine learning techniques in the late $1990s$ provides a new approach for record linkage, and it has become the mainstream methodology for this task. The task of record linkage is usually re-casted as the record pair classification problem, i.e. whether a record pair refers to the same entity or not (Elfeky et al., 2002; Neiling, 2006; Tejada et al., 2002; Nahm et al., 2002). Supervised methods can also be used to learn distance measures for approximate string comparisons (Bilenko and Mooney, 2003;

Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers, pages 1311–1321, Dublin, Ireland, August 23-29 2014.

Cohen et al., 2003). Although supervised techniques often achieve good linkage quality, they are largely limited by the availability of the training data.

To address this problem, semi-supervised learning approaches aim to make good use of a small portion of labeled and a large amount of unlabeled data to build a better classifier (Yarowsky, 1995). Self-training is a commonly used algorithm for semi-supervised learning, where in each iteration the learner converts the most confidently predicted unlabeled examples of each class into labeled training examples. It has been successfully applied to many tasks, such as sentiment analysis (He and Zhou, 2011; Riloff et al., 2003) and object detection from images (Rosenberg et al., 2005).

In this paper, we solve the task of product record linkage via a semi-supervised approach and adopt the flexible self-training framework for learning with little labeled data. We propose a novel variant of the self-training algorithm by incorporating the correlation existing in the data instances, which is rarely studied in previous studies. To introduce our idea, we first present an illustrative example in Figure 1. There are two databases $\mathcal{D}$ and $\mathcal{D}'$, and we have three records $r_1, r_2, r_3 \in \mathcal{D}$ and another three records $r'_1, r'_2, r'_3 \in \mathcal{D}'$. Furthermore, we assume r_1 and r'_1 refer to the same product. We can see that r_1 is involved in three candidate pairs, i.e. (r_1, r'_1), (r_1, r'_2) and (r_1, r'_3). Similarly, r'_1 is involved in three candidate pairs, i.e. (r'_1, r_1), (r'_1, r_2) and (r'_1, r_3). Usually, each individual database does not contain duplicate records, once we know r_1 is linked to r'_1, we can infer the rest candidate pairs should not be linked. In other words, only if we are confident that no pair in the set $\{(r_1, r'_2), (r_1, r'_3), (r_2, r'_1), (r_3, r'_1)\}$ is not linked, r_1 is likely to be linked with r'_1.

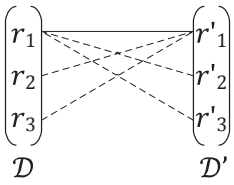


Figure 1: An illustrative example for correlation among record pairs. The real line denotes the real linkage relation and the dash line denotes the candidate linkage relation.

For the task of record linkage, the number of positive instances (i.e. linked record pairs) are usually much less than that of negative instances. We mainly consider the confidence evaluation of the candidate positive instance. By following the above idea, given a *candidate* linked pair, we treat all the correlated record pairs together as a group and evaluate the linkage confidence based on the evidence of all record pairs in this group, i.e. *group confidence evaluation*. We incorporate the group confidence evaluation into the self-training algorithm as a re-ranking step. Interestingly, once we have identified a linked pair, the rest correlated record pairs can be naturally judged as negative instances. We evaluate the novel self-training algorithm on two large datasets constructed based on real e-commerce Websites. We adopt several competitive methods as comparisons and perform extensive experiments. The results show that our method outperforms these baselines that do not consider data correlation.

2 Related Work

We have briefly described the supervised approaches for record linkage in the introduction. Now we discuss other related studies, including unsupervised clustering techniques, genetic programming based approaches and linking based on more complex constraints.

Unsupervised clustering techniques have been investigated both for improved blocking (Cohen and Richman, 2002; McCallum et al., 2000) and for automatic record pair classification (Elfeky et al., 2002). Usually, such techniques do not perform not as well as supervised approaches.

Most recently, genetic programming (GP) (Koza et al., 1999) has also been utilized to the task of record linkage. GenLink (Isele and Bizer, 2012) is a GP-based supervised learning algorithm in order to learn linkage rules from a set of existing reference links, which also suffers from the problem of lack of labeled data. Ngomo and Lyko (2013) evaluated linear and boolean classifiers against classifiers

computed by using genetic programming for the record linkage problem. Their experiments showed that both approaches did not perform well on real data.

Some other studies exploit more complex constraints that include relationships between different entity types to link all types of entities in coordination (Bhattacharya and Getoor, 2007; Dong et al., 2005; On et al., 2007). The usage of such constraints can indeed help to get better linkage results, but is in many cases domain-dependent. We try to develop an approach which can be applicable across domains.

In order to address the problem of limited labeled data, we mainly consider the semi-supervised approaches. There are rarely semi-supervised approaches specially for the record linkage problem. Some studies on improving self-training algorithms are related to our work. Self-training with editing (Li and Zhou, 2005) can help to reduce mislabeled pseudo training examples, and reserved self-training (Guan and Yang, 2013) is designed for handling imbalanced data. We have very different focus with theirs, i.e. incorporating the instance correlations into learning algorithms, which can applied to other self-training variants.

3 Problem Definition

In this section, we first introduce the preliminary related to our task. Then we formally define our studied task.

Product record. A product record r is characterized by a referred product entity e and a set of attribute values $\mathcal{V} = \{(v_i)\}_i$, where v_i denotes the value of the ith attribute in r. We use $r.e$ and $r.\mathcal{V}$ to index the product entity and attribute value set of the record r respectively. A product record corresponds to a unique product entity but a product entity can map to multiple product records across multiple databases. Attribute values are represented as strings, i.e. a sequence of characters. An attribute of a product might correspond to different descriptive text across websites.

Product record linkage. The task of product record linkage is to judge whether two product records refer to the same product entity. Given two product records r and r', we aim to judge whether $r.e$ is the same to $r'.e$. Usually, r and r' come from different product databases. Although different product databases can have different attributes for the same product and different attribute names for the same attribute, we make an assumption about the task: *candidate record pairs share the same set of attributes*. It is relatively easy to automatically identify common attributes and align attributes (Härder et al., 1999; Rundensteiner, 1999; Hassanzadeh et al., 2013), which is not our focus in this paper. We mainly study product record linkage under the same set of attributes, and this assumption makes our study more focused. If r and r' refer to the same product entity, denoted by $r \sim r'$; otherwise, we denote it by $r \not\sim r'$.

4 A General Machine Learning based Approach

Given a product type, as we mentioned above, we assume that it corresponds to a specific set of attributes, and all the product records share the same set of attributes but possibly with different descriptive text for attribute values. In this section, we further present a general supervised approach with similarity features.

4.1 Defining the similarity function

Given two product records r and r', we can obtain the similarity between their descriptive text of an attribute by using a similarity function. The major intuition is that if two records refer to the same product, they should have similar text for the same attribute, i.e. the similarity function should return a large similarity value. Let $f(\cdot, \cdot)$ denote a similarity function, which takes two text strings and returns a similarity value within the interval $[0, 1]$ for these two strings. As revealed in (Bilenko and Mooney, 2003), different attributes or fields may need different similarity functions to achieve best similarity evaluation. Thus, instead of fixing a single similarity function, we consider using the following widely used similarity functions: 1) Exact match; 2) Cosine similarity; 3) Jaccard coefficient; 4) K-Gram similarity (Kondrak, 2005); 5) Levenshtein similarity (Levenshtein, 1966); 6) Affine Gap similarity (Needleman and Wunsch, 1970).

4.2 The learning framework

Based on these similarity functions, we propose a general learning framework for product record linkage by using similarity values of different fields as features.

Given a product type, we assume that there are A attributes and K similarity functions. For two records r and r', we can obtain a similarity feature vector $\mathbf{x} = [x_{a,k}]_{i=1,k=1}^{A,K}$, which is indexed by an attribute and a similarity function: $x_{a,k}$ denotes the similarity of the ath attribute between r and r' by using the kth similarity function. Furthermore, each feature vector $\mathbf{x}$ will correspond to a unique binary label y which indicates that r and r' refer to the same product entity. Given a set of record pairs and their linkage labels $\{(\mathbf{x}, y)\}$, we can learn a classifier which is able to predict the linkage label given the similarity feature vector of two records. To this end, we have reformulated the task of product record linkage as a binary classification problem. Any classifiers can be used for this task. In what follows, we will use *instances* and *candidate pairs* alternatively.

5 Group based Self-Training

In the above, we have presented a supervised learning approach for product record linkage. The approach is easy to apply in practice, however, the performance is largely limited by the availability of training data. For our current task, i.e. product record linkage, the generation of labeled data becomes even much harder: there are usually many product types and it is infeasible to create a large amount of labeled data for each type. Although it is difficult to obtain labeled data, we can easily obtain sufficient unlabeled data. Thus, in this paper, we study the task of product record linkage in a semi-supervised setting by leveraging both the learning ability of the classifiers and the usefulness of the large amount of unlabeled data. We propose a novel group based self-training algorithm for product record linkage. Before introducing our method, we first introduce the general self-training algorithm.

5.1 The general self-training algorithm

Self-training is a semi-supervised learning algorithm. It starts training on labeled data only, after each iteration, the most confidently predicted unlabeled samples would be incorporated as new labeled data, i.e. pseudo labeled data, decided by confidence scores from the classifier. After several iterations, it is expected to get a better classifier trained with both labeled data and pseudo labeled data. The general procedure of self-training algorithm is summarized in Algorithm 1.

Algorithm 1: The general procedure of the self-training algorithm.

```
Input: labeled dataset L, unlabeled dataset U, the classifier C.
U' ← S randomly selected examples from U, S is usually set to 0.5 × |U|;
repeat
    Training the classifier: Use L to train C, and label the examples in U';
    Selecting pseudo labeled data: Select T most confidently classified examples from U' and add them to L;
    Filling unlabeled data: Refill U' with examples from U, to keep U' at a constant size of S examples.
until I iterations or U = ∅;
return The extended labeled dataset L and the trained classifier C.
```

We can see that self-training is a wrapper algorithm by taking a classifier as the learning component, and it has three major steps in an iteration: 1) training classifier; 2) selecting pseudo labeled data; and 3) filling unlabeled data. Among the three steps, the most important step is the pseudo labeled data selection. Previously, the most commonly used method is to select the top confident instances of the classifier, and it is easy to see that the performance of self-training relies on the learning ability of the embedded classifier.

5.2 Group confidence evaluation

Recall that each instance is a pair of product records (r, r') and their label indicates whether they should be linked or not. Let $P^L(r, r')$ denote the confidence that r and r' refer to the same product entity (linked

confidence), and $P^N(r, r')$ denote the confidence that r and r' refer to different product entities (non-linked confidence). $P^L(r, r')$ and $P^N(r, r')$ can be estimated by the confidence scores from the classifier. In the task of product record linkage, there are usually more negative instances, i.e. the number of non-linked pairs is much more than that of linked pairs. Thus, we mainly study the confidence of a candidate positive instance. The standard self-training algorithm selects top ranked positive instances according to the confidence scores estimated by the classifier, i.e. we select the instances with large linked confidence $P^L(\cdot, \cdot)$. However, when applied to product record linkage, it ignores important characteristics underlying the data, which will be potentially helpful to the task.

Let us examine the illustrative example in Figure 1. Recall that r_1 and r'_1 refer to the same product, i.e. $r_1 \sim r'_1$. We can see that r_1 is involved in three candidate pairs, i.e. (r_1, r'_1), (r_1, r'_2) and (r_1, r'_3). Similarly, r'_1 is involved in three candidate pairs, i.e. (r'_1, r_1), (r'_1, r_2) and (r'_1, r_3). We totally have a set of five candidate pairs, i.e. $\{(r_1, r'_1), (r_1, r'_2), (r_1, r'_3), (r_2, r'_1), (r_3, r'_1)\}$. Here we follow the assumption of the one-to-one mapping, i.e. given two databases, a product record can link to at most one record in the other database. By leveraging the correlation among candidate pairs, with $r_1 \sim r'_1$, we can infer the rest four candidate pairs must not be linked, i.e. $r_1 \not\sim r'_2, r_1 \not\sim r'_3, r_2 \not\sim r'_1, r_3 \not\sim r'_1$. Next, we formally characterize the above idea and present the algorithm. Given two databases $\mathcal{D}$ and $\mathcal{D}'$, let $\mathcal{C} \subset \mathcal{D} \times \mathcal{D}'$ denote the candidate pair set where two product records in a pair come from $\mathcal{D}$ and $\mathcal{D}'$ respectively. Consider a candidate pair $(r, r') \in \mathcal{C}$, where $r \in \mathcal{D}, r' \in \mathcal{D}'$. We consider the following two sets: $\mathcal{S}_r = \{(r, b)|(r, b) \in \mathcal{C}, b \in \mathcal{D}' \text{ and } b \neq r'\}$ and $\mathcal{S}_{r'} = \{(a, r')|(a, r') \in \mathcal{C}, a \in \mathcal{D} \text{ and } a \neq r\}$. Intuitively, if we know $r \sim r'$, then all the pairs in both $\mathcal{S}_r$ and $\mathcal{S}_{r'}$ must not be linked. Thus, we define **the conflicting set** of pair (r, r') as $\mathcal{S}_{cfl}^{r,r'} = \mathcal{S}_r \cup \mathcal{S}_{r'}$.

With the definition of the conflicting set, let us reconsider the pseudo labeled data selection. The straightforward way is to evaluate each instance with their linked confidence $P^L()$ from the classifier. However, it oversimplifies the data dependence and does not make use of the correlated characteristics. Consider an instance, which is a record pair (r, r'), we can have the following two properties:

- If $r \sim r'$, then $\forall (a, b) \in \mathcal{S}_{cfl}^{r,r'}$, we have $a \not\sim b$;
- If $\exists (a, b) \in \mathcal{S}_{cfl}^{r,r'}$ and $a \sim b$, then we have $r \not\sim r'$.

The above properties suggest that it should be helpful to consider the correlation among instances when evaluating the confidence of a positive instance, i.e. a candidate linked record pair. Intuitively, if two records refer to the same product entity, they should have large linked confidence and their conflicting pairs should have large non-linked confidence. We propose to use the following method to evaluate the linkage confidence between r and r'

$$\text{Conf}(r, r') = P^L(r, r') \Big(\prod_{(a,b) \in \mathcal{S}_{cfl}^{r,r'}} P^N(a, b) \Big)^{1/M}, \tag{1}$$

where $M = |\mathcal{S}_{cfl}^{r,r'}|$, $P^L(\cdot, \cdot)$ and $P^N(\cdot, \cdot)$ are positive and negative confidence scores estimated by the classifier respectively. Note that we take the geometric mean of the non-linked confidence of these conflicting pairs, which is to reduce the affect of large outlier values and the varying size of the conflict sets. We treat a candidate linked pair and all the candidate pairs in its conflicting set as a group. The group confidence evaluation consists of two intuitions: 1) the confidence that two records should be linked; 2) the confidence that any pair of records in the conflicting set must not be linked. We have taken these two aspects into a unified evaluation score.

5.3 The proposed self-training algorithm

In this part, we present the novel self-training algorithm based on the group confidence evaluation. We have the similar steps with the general self-training algorithm in Algorithm 1. The major focus is to modify the step of *pseudo labeled data selection*. As mentioned above, we mainly consider the confidence evaluation of positive instances. Our method for pseudo labeled data selection is three-step process:

- Select top T' most confidently classified positive examples by the classifier;
- Rerank these T' examples by the group confidence scores defined in Equation 1;
- Select top T examples from the reranked T' examples ($T \leq T'$) as pseudo positive instances and their corresponding conflicting instances in the conflicting sets as pseudo negative instances.

We select positive instances not only based on the instance itself but also their corresponding conflicting instances: if we have high confidence about a positive instance, then the confidence of their conflicting instances being negative should be high, too. Next, we present the detailed group based self-training algorithm in Algorithm 2.

Algorithm 2: The procedure of the group based self-training algorithm.

Input: labeled dataset $\mathcal{L}$, unlabeled dataset $\mathcal{U}$, the classifier $\mathcal{C}$.
$\mathcal{U}' \leftarrow S$ randomly selected examples from $\mathcal{U}$;
repeat
  Training the classifier: Use $\mathcal{L}$ to train $\mathcal{C}$, and label the examples in $\mathcal{U}'$;
  Selecting pseudo labeled data selection:
  - Select T' most confident positive examples from $\mathcal{U}'$ and add them to $\mathcal{L}$;
  - Calculate the group confidence scores for the T' examples according to Equation 1.
  - Rerank these T' examples by their group confidence scores and add top T examples to $\mathcal{L}$ as the pseudo positive instances.
  - For each of the T examples, add their conflicting instances to $\mathcal{L}$ into as the pseudo negative instances.

  Filling unlabeled data: Refill $\mathcal{U}'$ with examples from $\mathcal{U}$, to keep $\mathcal{U}'$ at a constant size of S examples.
until *I iterations or $\mathcal{U} = \emptyset$*;
return *The extended labeled dataset $\mathcal{L}$ and the trained classifier $\mathcal{C}$.*

On one hand, our group based self-training algorithm naturally exploits the correlation among data instances and evaluate the confidence scores in a broader view, which avoids the decision conflicts caused by the data dependence. On the other hand, we focus on evaluating the confidence of being a positive instance, which further reduces the bias from imbalanced data distribution. Thus, it is expected to achieve better performance in the task of product record linkage.

Most classifiers can provide the estimated confidence scores $P^L()$ (i.e. for a positive instance) and $P^N()$ (i.e. for a negative instance): Maximum-Entropy models output the conditional probabilities of an instance for each class (Berger et al., 1996); the Decision Tree C4.5 algorithm is also able to compute the probability distribution over different classes for each instance (Quinlan, 1993).

6 Experiments

6.1 Construction of the test collection

We test our method on two real e-commerce datasets respectively from Jingdong[1] and eTao[2]. Jingdong is the largest B2C e-commerce company and eTao is one of the largest product search portals in China. Due to the extremely large product databases, it is infeasible to generate training data on each product type for these two product databases. We consider two popular kinds of products: laptop and camera. These two kinds of products cover a considerable amount of brands and models, especially suitable for the test of record linkage. Both Jindong and eTao have set up specific categories for these two kinds of products respectively, thus we can easily crawl the product records under the corresponding category label. To generate linked record datasets, we first manually align attributes (i.e. fields) for these two kinds between Jindong and eTao. We summarize the numbers of aligned fields and some example fields in Table 1. Not all the records contain the information for all the fields, we set the value of the empty field to a "*NULL*" string.

[1] http://www.jd.com
[2] http://www.etao.com

We adopt a blocking approach (Baxter et al., 2003) to automatically generate a set of candidate pairs, i.e. a record in Jindong is to be linked with a record in eTao. This approach consider all pairwise links between Jindong records and eTao records for the same kind of product. If there exists at least one common word in the field of *brand* or *model* between a record pair, we consider it to be a candidate pair. The automatic method generates 20,094 candidate pairs and 12,157 candidate pairs respectively for *LAPTOP* and *CAMERA*. Then we invite professional workers from an e-commerce company to link records across these two product databases. Instead of examining all the candidate pairs, the labeling process adopts a product-oriented way to generate the gold standard. Given a product record of a database, the annotator first identifies the product entity that the record refers to, then she looks for the corresponding record in another database. In the annotation process, Web access is available all the time. Annotators can make use of the search engines of Jindong and eTao to accelerate the product lookup. A linked record pair is treated as a positive instance. Finally, we identify 501 linkable products (i.e. 501 positive instances) in LAPTOP dataset, and 478 linkable products (i.e. 478 positive instances) in CAMERA dataset. All the other candidate pairs are automatically labeled as *negative*. We present the the data statistics in Table 1.

Dataset	# positive instances	# negative instances	# fields	Example fields
LAPTOP	501	19593	10	OS, screen size, CPU type, ram size
CAMERA	478	11679	11	lens type, sensor type, focal length, aperture size

Table 1: Basic statistics of datasets.

6.2 Experimental setup

For each kind of product, we divide the dataset into two parts, i.e. a training set and a test set. In order to examine different methods in a semi-supervised setting, we keep a small amount of instances in the training set, and we assume all the methods can use of the data (without labels) in the test set. There are more negative instances, we mainly consider the amount of positive instances, and the number of positive instances is called as *the number of seeds*. We randomly generate the training set with the given number of seeds. Once we add one positive instance into the training set, we add all the its conflicting instances into the training set. This is to reduce the correlation between training instances and testing instances for a fair comparison. In later experiments, given the seed number, we will generate ten random training sets and take the average of ten runs as the final performance. In later experiments, we do not explicitly report the number of negative instances unless needed.

We adopt three widely used evaluation metrics for the classification task: Precision, Recall and the F-measure [3].

We compare the following methods for the task of product record linkage:

- *Supervised Classifier* (**SC**): the standard supervised classifier, which does not consider the unlabeled data at all.
- *Traditional Self-Training* (**t-ST**): the traditional self-training method in Algorithm 1 which adds an equal amount of samples of each class in pseudo labeled selection at each iteration.
- *Proportional Self-Training* (**p-ST**): the traditional self-training method in Algorithm 1 but add samples according to the class distribution at each iteration.
- *Simple Group Based Self-Training* (**s-ST**): a simplified version of our approach without the group confidence valuation, which directly selects samples of high confidence scores estimated from the classifier together with their conflicting pairs as negative samples at each iteration.
- *Group Based Self-Training* (**g-ST**): the proposed group based self-training algorithm in Algorithm 2, which uses the group confidence evaluation method to select pseudo positive instances.

[3] http:/en.wikipedia.org/wiki/Precision_and_recall

Recall all the methods rely on the wrapped classifier. We select two classic but very different classifiers: the Maximum Entropy model (***MaxEnt***) and the Decision Tree $C4.5$ (***Tree***). We implement these two classifiers using the machine learning toolkit Weka[4]. We use the six similarity functions to obtain similarity values between two records on each field as features. All the self-training based methods run ten iterations and at each iteration they add the same number of positive instances, i.e. 30. Different methods select pseudo *negative instances* differently. *t-ST* does not consider the correlation between data instances, and it adds top 30 confident negative instances. *p-ST* adds top $30 \times \frac{\#\text{negative instances}}{\#\text{positive instances}}$ confident negative instances. Both *p-ST* and *g-ST* take all the conflicting instances of the selected pseudo positive instances as the negative instances. We present the average numbers of pseudo negative instances at an iteration in Table 2. As will be revealed later, although *p-ST* adds more negative instances, *g-ST* performs much better than *p-ST*, which indicates simply adding more negative instances might not lead to better performance. We do not perform specific preprocessing steps to make the data balanced (e.g. under-sampling or over-sampling), and we find the data distribution does not significantly affect the performance of the classifiers on our dataset.

Dataset	t-ST	p-ST	s-ST	g-ST
LAPTOP	30	950	845	854
CAMERA	30	655	569	584

Table 2: Average numbers of pseudo negative instances selected at each iteration.

6.3 Results and analysis

Overall performance comparison. To test the performance under weak supervision, we first set the seed number to 30, which nearly takes up a proportion of 5% of the labeled data. We present the results of different methods in Table 3 and Table 4. We first examine the performance of the baselines. We can see that semi-supervised learning is very effective to improve over the the supervised classifier when the amount of training data is small. It is interesting to see that *s-ST* performs best among all the baselines. Recall that the major difference between *s-ST* and other baselines is that it select the conflicting pairs of the pseudo positive instances as the negative instances. It indicates that it is important to consider the correlation among the data instances. In addition, Decision Tree seems to be more competitive than Maximum Entropy Model for product record linkage. Then we take our group based self-training algorithm into comparison. In terms of F1 measure, we can see that it is consistently better than all the baselines on two datasets respectively by using two different classifiers. It is worth looking into the performance comparison on precision and recall. We can see that (1) *s-ST* and *g-ST* yield better results in terms of precision while the other baselines yield better results in terms of recall; (2) our method *g-ST* largely improves over the best baseline *s-ST*. It is not surprising to have these observations since that our group evaluation method is more careful at the selection of pseudo positive instance: it considers the evidence from the conflicting instances.

Methods	MaxEnt			Decision Tree		
	P	R	F1	P	R	F1
SC	0.246	0.910	0.382	0.301	**0.931**	0.454
t-ST	0.264	**0.925**	0.411	0.328	0.921	0.484
p-ST	0.350	0.831	0.487	0.412	0.887	0.539
s-ST	**0.979**	0.632	0.767	0.909	0.754	0.823
g-ST	0.936	0.742	**0.826**	**0.912**	0.843	**0.876**

Table 3: Results on LAPTOP dataset.

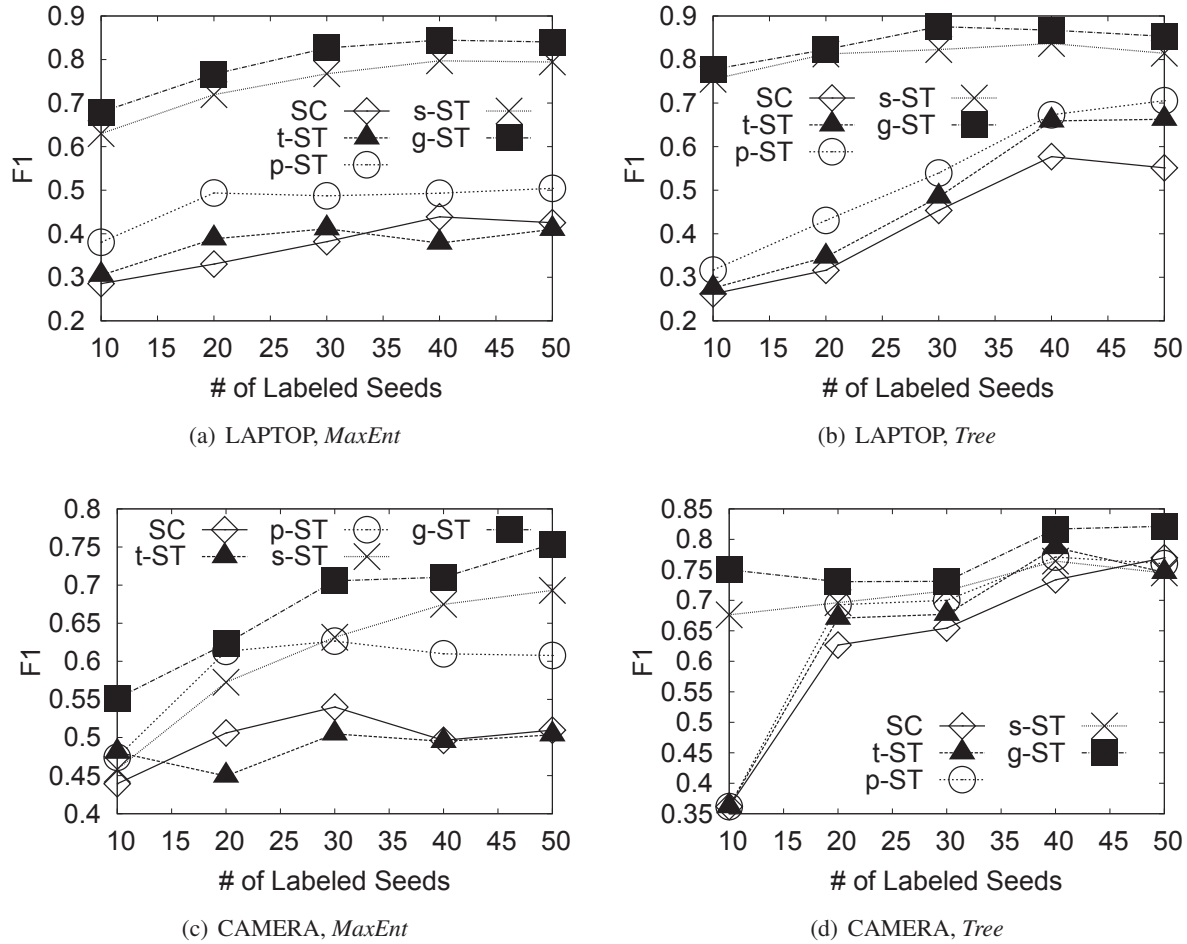
Parameter tuning. In the above, we have shown the results of different methods with 30 positive instances. The number of seeds is particularly important for self-training algorithms, and we want to ex-

[4]http://www.cs.waikato.ac.nz/ml/weka

Methods	MaxEnt			Decision Tree		
	P	R	F1	P	R	F1
SC	0.387	0.891	0.540	0.493	**0.965**	0.652
t-ST	0.352	**0.892**	0.504	0.537	0.963	0.677
p-ST	0.501	0.871	0.626	0.573	0.942	0.700
s-ST	**0.931**	0.479	0.632	0.962	0.570	0.716
g-ST	0.917	0.574	**0.706**	**0.965**	0.588	**0.731**

Table 4: Results on CAMERA dataset.

amine how it affects the performance of these methods. By varying the number of seeds from 10 to 50 with a step of 10, we present the F1 results in Figure 2 on two datasets by using two classifiers. We can see that our method is consistently better than baselines with the varying of the seed number. Especially, our method still works well when there is little labeled data, i.e. $\#seeds = 10$. With a weaker classifier, i.e. *MaxEnt*, our method yields more improvement than that with *Tree*. Besides the seed number, there are another two factors which potentially affect the performance: (1) the iteration number and (2) the number of pseudo positive instances selected at each iteration. We also examine the tuning results of these two parameters and find our method is consistently better than *s-ST* with the varying of these two factors. These results show that our method is very effective and it is of high stability and practicability.



(a) LAPTOP, *MaxEnt*

(b) LAPTOP, *Tree*

(c) CAMERA, *MaxEnt*

(d) CAMERA, *Tree*

Figure 2: Performance comparison with varying seed numbers (i.e. # of positive instances).

7 Conclusion

In this paper, we develop a novel variant of the self-training algorithm by leveraging the data characteristic for the task of product record linkage. We joint consider a candidate linked pair and its corresponding correlated pairs as a group, at the selection of pseudo labeled data. We propose a confidence evaluation method for a group of instances, and incorporate it as a re-ranking step in the self-training algorithm. We evaluate the novel self-training algorithm on two large datasets constructed based on real e-commerce Websites. We adopt several competitive methods as comparisons and perform extensive experiments. The results show that our method outperforms these baselines that do not consider data correlation. We also carefully examine the affects of various parameters, and the tuning results indicate the stability and robustness of our method.

The major contribution and novelty of this paper is the novel group confidence evaluation to model the correlation existing in data. Although we develop the idea in the setting of self-training algorithms, it will be promising to be applied in other learning algorithms, i.e. active learning.

Acknowledgements

We thank the anonymous reviewers for his/her thorough review and highly appreciate the comments. This work was partially supported by the National Key Basic Research Program (973 Program) of China under grant No. 2014CB340403, 2014CB340405 and NSFC Grant 61272340. Xin Zhao was supported by MSRA PhD fellowship. Xin Zhao and Yuexin Wu contributed equally to this work and should be considered as joint first authors. Xin Zhao is the corresponding author.

References

Rohan Baxter, Peter Christen, and Tim Churches. 2003. A comparison of fast blocking methods for record linkage. In *ACM SIGKDD*, volume 3, pages 25–27. Citeseer.

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Indrajit Bhattacharya and Lise Getoor. 2007. Collective entity resolution in relational data. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):5.

Mikhail Bilenko and Raymond J Mooney. 2003. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48. ACM.

Peter Christen. 2006. A comparison of personal name matching: Techniques and practical issues. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 290–294. IEEE.

William W Cohen and Jacob Richman. 2002. 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*, pages 475–480. ACM.

William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg, et al. 2003. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 2003, pages 73–78.

Xin Dong, Alon Halevy, and Jayant Madhavan. 2005. Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 85–96. ACM.

Mohamed G Elfeky, Vassilios S Verykios, and Ahmed K Elmagarmid. 2002. Tailor: A record linkage toolbox. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 17–28. IEEE.

Ivan P Fellegi and Alan B Sunter. 1969. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210.

Karl Goiser and Peter Christen. 2006. Towards automated record linkage. In *Proceedings of the fifth Australasian conference on Data mining and analystics-Volume 61*, pages 23–31. Australian Computer Society, Inc.

Zhiguang Liu Xishuang Dong Yi Guan and Jinfeng Yang. 2013. Reserved self-training: A semi-supervised sentiment classification method for chinese microblogs.

Theo Härder, Günter Sauter, and Joachim Thomas. 1999. The intrinsic problems of structural heterogeneity and an approach to their solution. *The VLDB Journal*, 8(1):25–43.

Oktie Hassanzadeh, Ken Q Pu, Soheil Hassas Yeganeh, Renée J Miller, Lucian Popa, Mauricio A Hernández, and Howard Ho. 2013. Discovering linkage points over web data. *Proceedings of the VLDB Endowment*, 6(6):445–456.

Yulan He and Deyu Zhou. 2011. Self-training from labeled features for sentiment analysis. *Information Processing & Management*, 47(4):606–616.

Robert Isele and Christian Bizer. 2012. Learning expressive linkage rules using genetic programming. *Proceedings of the VLDB Endowment*, 5(11):1638–1649.

Grzegorz Kondrak. 2005. N-gram similarity and distance. In *String Processing and Information Retrieval*, pages 115–126. Springer.

John R Koza, Forrest H Bennett III, and Oscar Stiffelman. 1999. *Genetic programming as a Darwinian invention machine*. Springer.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. In *Soviet physics doklady*, volume 10, page 707.

Ming Li and Zhi-Hua Zhou. 2005. Setred: Self-training with editing. In *Advances in Knowledge Discovery and Data Mining*, pages 611–621. Springer.

Andrew McCallum, Kamal Nigam, and Lyle H Ungar. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

Un Yong Nahm, Mikhail Bilenko, and Raymond J Mooney. 2002. Two approaches to handling noisy variation in text mining. In *Proceedings of the ICML-2002 workshop on text learning (TextML2002)*, pages 18–27. Citeseer.

Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453.

Mattis Neiling. 2006. Identification of real-world objects in multiple databases. In *From Data and Information Analysis to Knowledge Engineering*, pages 63–74. Springer.

Axel-Cyrille Ngonga Ngomo and Klaus Lyko. 2013. Unsupervised learning of link specifications: Deterministic vs. non-deterministic. *Ontology Matching*, page 25.

Byung-Won On, Nick Koudas, Dongwon Lee, and Divesh Srivastava. 2007. Group linkage. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 496–505. IEEE.

John Ross Quinlan. 1993. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. 2003. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models.

Elke Rundensteiner. 1999. Special issue on data transformation. *IEEE Techn. Bull. Data Engineering*, 22(1).

Sheila Tejada, Craig A Knoblock, and Steven Minton. 2002. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 350–359. ACM.

William E Winkler. 1988. Using the em algorithm for weight computation in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research Methods, American Statistical Association*, volume 667, page 671.

William E Winkler. 2006. Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.