Exploiting Context Analysis for Combining Multiple Entity Resolution Systems

Zhaoqi Chen
Microsoft Corporation
Redmond, USA
schen@microsoft.com

Dmitri V. Kalashnikov
Dept. of Computer Science
University of California, Irvine
dvk@ics.uci.edu

Sharad Mehrotra
Dept. of Computer Science
University of California, Irvine
sharad@ics.uci.edu*

ABSTRACT

Entity Resolution (ER) is an important real world problem that has attracted significant research interest over the past few years. It deals with determining which object descriptions co-refer in a dataset. Due to its practical significance for data mining and data analysis tasks many different ER approaches has been developed to address the ER challenge. This paper proposes a new ER Ensemble framework. The task of *ER Ensemble* is to combine the results of multiple base-level ER systems into a single solution with the goal of increasing the quality of ER. The framework proposed in this paper leverages the observation that often no single ER method always performs the best, consistently outperforming other ER techniques in terms of quality. Instead, different ER solutions perform better in different *contexts*. The framework employs two novel combining approaches, which are based on supervised learning. The two approaches learn a mapping of the clustering decisions of the base-level ER systems, together with the local context, into a combined clustering decision. The paper empirically studies the framework by applying it to different domains. The experiments demonstrate that the proposed framework achieves significantly higher disambiguation quality compared to the current state of the art solutions.

Categories and Subject Descriptors

H.2.m [**Database Management**]: Miscellaneous - Entity Resolution; H.2.m [**Database Management**]: Miscellaneous - Data Cleaning

General Terms

Algorithms, Experimentation, Performance, Reliability

Keywords

ER Ensemble, Entity Resolution, Context Analysis

*This research was supported by NSF Awards 0331707 and 0331690, and DHS Award EMW-2007-FP-02535.

1. INTRODUCTION

Entity Resolution (ER) challenges have attracted significant interest in several research communities in recent years. ER problem is primarily motivated by the need of high quality data mining and analysis, where it is important that the data being analyzed is represented accurately and interpreted properly. However, real world raw datasets are often not perfect and contain various types of issues, including errors and uncertainty, which complicate analysis. ER is one of the most common challenges that need to be addressed in preprocessing raw data. The problem is that entities in raw data are often referred to by descriptions, which are not always unique identifiers of those entities, causing ambiguity. For instance, the dataset might refer to two different John Smith's as simply 'J. Smith', or the name might be misspelled as 'John Smitx', etc, complicating analysis of such data. The goal of ER is to identify and group references that co-refer, that is, refer to the same entity. The output of an ER system is a clustering of references, where each cluster is supposed to represent one distinct entity.

Given the practical significance of the problem, a large number of diverse ER approaches have been proposed to address the entity resolution challenge. This motivates the problem of creating a single combined ER solution – an *ER ensemble*. Such a combined approach would take as input the clustering decisions produced by several base-level ER systems and combine them to produce the final higher quality clustering.

One of the most closely related and well-studied challenges is the problem of *cluster ensemble* [36] or *cluster aggregation* [16]. In that problem, the algorithm is given n initial clusterings $A^{(1)}, A^{(2)}, \ldots, A^{(n)}$. The goal is to construct a single clustering A^* that agrees with all of them as much as possible. The goal of ER ensemble problem is different. Here, the algorithm is given n ER systems $S_1, S_2, \ldots, S_n$ and the dataset $\mathcal{D}$ to process. The goal of each ER system is to group the co-referent references in $\mathcal{D}$ correctly.

Unlike cluster ensemble, ER ensemble actively uses the notion of the *correct* clustering A^+ for $\mathcal{D}$, which is also referred to as the ground truth clustering.[1] Clustering A^+ is *unknown* to the ER systems beforehand but it is always present implicitly. Each system S_i is applied to the dataset being processed to produce its clustering $A^{(i)}$. Clustering $A^{(i)}$ corresponds to what system S_i believes A^+ should be.

[1]Although there also could be the notion of the ground truth clustering A^+ in cluster ensemble as well, the difference is that there it does not play any role in constructing the final clustering A^*.

Each clustering $A^{(i)}$ thus has an implicit notion of *quality* associated with it, which reflects how different $A^{(i)}$ from A^{+} is. It is implicit because the algorithm does not know A^{+} and thus cannot measure the quality during the run time.[2] The goal of ER ensemble is to build a clustering $A^{\star}$ with the highest quality, that is, as close to A^{+} as possible.

To illustrate the difference between the two ensemble problems, consider a scenario where systems $S_1, S_2, \ldots, S_{n-1}$ always produce poor-quality clusterings, whereas S_n always produces the perfect clustering. The objective of the cluster ensemble will be to find a clustering that agrees the most with the n clusterings, including the $n-1$ poor-quality clusters. The goal of the ER ensemble will be satisfied, however, if the system would simply always output $A^{(n)}$ as its $A^{\star}$, and completely ignore the rest of the (poor) clusterings.

While there has been some work on creating ER ensembles [32,37,41], existing solutions are based on analyzing the outputs produced by the base systems. These approaches do not analyze the context features, which is the key in the solution we propose. Specifically, we realize that the disambiguation quality of ER approaches frequently depends on the *context* in which they are employed. The intuition is that a system that performs well in some context may perform poorly in another. For instance, let K be the actual number of clusters into which the references are supposed to be grouped. It is known that, in general, if K is large then a merging strategy with higher threshold may work better than that of with lower threshold. At the same time, however, the reverse might be the case for the datasets where K is small. Thus the value of K might be used as a context feature in deciding which approach is applicable best. The challenge is though that the value of K is usually unknown to the ER systems beforehand. If the context can be captured then this knowledge can be utilized to significantly improve performance of ER ensemble. We will show how it can be estimated from the outputs of the base systems.

In this paper we propose a novel ER ensemble framework. The framework employs two supervised learning approaches for mapping the decisions of the base-level ER systems together with context features into its overall clustering decision. The first approach is based on training a classifier on the decision and context features and can be viewed as an extension of the committee-based ensemble method [41]. The second approach builds on learning for each base system the effect of context on the quality of the system's output. Then this knowledge and the decision of the systems are combined to produce the final answer of the ER ensemble. The main contributions of the paper are:

- Two novel ER ensemble approaches.
- A predictive way of selecting context features that relies on estimating the unknown parameters of the data being processed.
- An extensive empirical evaluation of the proposed solution.

The rest of this paper is organized as follows. Section 2 covers related work. Section 3 defines the Entity Resolution problem and presents a motivation for a combined approach. Section 4 describes the proposed ER ensemble framework. The overall algorithm is then empirically evaluated in Section 5 and compared to several state of the art solutions, both unsupervised and supervised. Finally, we conclude in Section 6.

[2]After the algorithm completes and produces its results, the quality of the resulting clusters can be assessed using a quality metric such as the pair-wise F-measure or B-cubed. Such metrics compare the clustering to the ground truth clustering which was not known to the algorithm at the runtime.

2. RELATED WORK

This paper is related to multiple research areas as elaborated below.

2.1 Entity Resolution

ER problem has been studied in several research areas under many names such as *coreference resolution, deduplication, object uncertainty, record linkage, reference reconciliation*, etc. In the past, a wide variety of techniques have been developed for ER problem. Many of the approaches have focused on exploiting various similarity metrics. These approaches compute the similarity between each pair of references and compare it with a predefined threshold. The pairs that have similarity above the threshold are considered as co-referent and a transitive closure step is performed to get the final resolution. A number of alternate formulations of similarity metrics have been proposed in the literature. Traditional approaches exploit textual similarity between reference attributes [18]. More recent approaches consider relational similarities derived from the context of references as additional information [5, 12, 19, 22, 30, 31]. In addition, the similarity measures are no longer static: evidence can be propagated in an iterative way and one resolution decision can affect the others [10, 12]. There are some other approaches that are based on probabilistic models, including Markov Logic [35], Conditional Random Field [27], etc. We will use some of the state-of-the-art algorithms as the base-level systems in our framework.

The similarity metrics on attributes also can be learned from the data by adaptive methods [6,8,9]. The basic idea of adaptive approaches is to learn a classifier that combine similarity across multiple fields and make a match/non-match decision for each pair of records. In [32] and [37], the authors proposed active learning and employed a committee of classifiers to find the most informative (i.e., max disagreement by the committee) examples to label by user as user feedback in training data. The committee of classifiers is employed to select the input data for next-level training, while in our case the set of classifiers are used to provide the clustering output that are to be combined. There is also some recent work that proposed compositional approaches for ER problem [33, 38]. For example, [33] proposed an approach that finds a match plan based on the semantic ambiguity of the data sources that are combined. The notion of context there is used to define the semantic ambiguity of the data sources, whereas in our framework the context is used to capture the properties of the clustering results.

2.2 Clustering Ensembles

The problem of combining multiple clusterings into a single consolidated clustering has been studied extensively under the names of *clustering ensembles* [36], *clustering combination* [14, 26] or *clustering aggregation* [16] in recent years. The final clustering is usually the one that agrees the most with the given clusterings. Many cluster ensemble approaches use a co-association matrix to accumulate the similarity between data points by all given clusterings [14, 36]. Other approaches focus on finding the correspondence between cluster labels produced by different clustering systems [26]. In many cases, the numbers of clusters in the final clustering is preset. In [16] the authors mapped the problem of clustering aggregation to the problem of correlation clustering in order to automatically identify the proper number of final clusters. Some cluster ensemble approaches focused on the generation of the base clusterings to provide diversity for ensembles [17]. It is often assumed that the clusterings are generated according to a certain distribution. For the ER ensemble problem studied in this paper, there are no such assumptions and there is no prior knowledge of how the clusters are generated. The number of clusters in the final clustering is also unknown. The goal of our work is to learn a more advanced combination function so that the algorithm is adaptive to the data being processed.

2.3 Combining classifiers

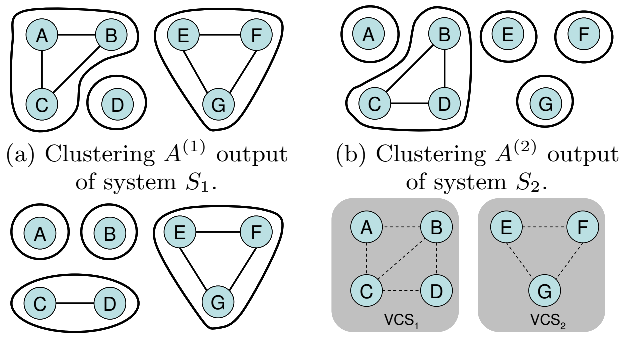
In machine learning area there is a large body of work on how to combine classifiers to achieve improved efficiency and accuracy. In [25] the authors proposed various rules to combine classifiers. Majority voting is one of the most popular rules and we will use it as one of the baseline algorithms in our experiment. Other popular approaches to combine multiple classifiers include *Bagging*, *Boosting* and *Stacking*.

Bagging and boosting are designed to combine multiple classifiers of the same type. Each classifier is trained with a different training set obtained from the original dataset using resampling techniques. The final classification is decided by voting. In Bagging (short for bootstrap aggregating), the training set is produced by sampling with replacement on original dataset and the ensemble is taken place in a parallel way. On the other hand, Boosting learns the base classifiers sequentially. The examples that are classified incorrectly by a previous classifier will be weighed higher when training the new classifier, with the hope to improve the new classifier on these difficult examples. All the base classifiers for Bagging and Boosting should be of the same type, while *Stacking*, like the solutions proposed in this paper, can combine several types of classifiers. The basic idea of *Stacking* [40] is to perform cross-validation on the base-level dataset and create a meta-level dataset from the predictions of base-level classifiers. Then a meta-level classifier is trained on the meta-level feature vectors. The base-level classifiers are therefore combined with a common classifier at the meta-level.

Zhao et al. [41] empirically combined multiple different classifiers by various combination approaches, including bagging, boosting and stacking, for the purpose of entity resolution. The base-level classifiers are state-of-the-art classifiers applied on the feature space derived from the data. The classifiers need to know the details of the database features/fields and the similarity functions as well. On the other hand, our approach take the output of existing ER systems without the need to know the details of the ER algorithms and the features of the dataset being processed. Our first approach can be viewed as a stacking approach, although the base-level classifiers are replaced by the ER systems and the prediction of base-level classifiers are replaced by our novel context features.

3. PROBLEM DEFINITION

In this section we first formally define the problem of entity resolution in Section 3.1. We then formulate the goal of the ER ensemble problem in Section 3.2. After that we explain the blocking procedure for improving the efficiency of the algorithm in Section 3.3. We conclude with a discussion of a basic solutions to the ER ensemble problem in Section 3.4.



(a) Clustering $A^{(1)}$ output of system S_1.

(b) Clustering $A^{(2)}$ output of system S_2.

(c) Ground Truth Clustering A^+.

(d) Blocking according to $A^{(1)}$ and $A^{(2)}$.

Figure 1: Example of Graph Rep. for a Toy Dataset.

3.1 Entity Resolution

In the problem of entity resolution a dataset $\mathcal{D}$ contains information about the set of objects $O = \{o_1, o_2, \ldots, o_{|O|}\}$. Objects in $\mathcal{D}$ are represented as a set of references $R = \{r_1, r_2, \ldots, r_{|R|}\}$. Each reference $r \in R$ refers to a single object $o_r \in O$. However, the ER algorithm in general does not know o_r and is not given O beforehand.

Two references r_i and r_j are said to **co-refer**, denoted as $r_i \approx r_j$, if they refer to the same object, that is, if $o_{r_i} = o_{r_j}$. The co-reference set C_r for r is the set of all references that co-refer with r, that is, $C_r = \{q \in R : q \approx r\}$. Set C_r is unknown to the algorithm beforehand and the objective of entity resolution is for each reference $r \in R$ to find its C_r. Different C_r's will partition set R into non-overlapping clusters. Thus the goal can also be formulated as clustering set R such that references in each cluster correspond to the same object and no two distinct clusters contain references to the same object. That is, each resulting cluster should accurately and completely represent an object from O.

The output of an ER system S applied to dataset $\mathcal{D}$ is a set of clusters $A = \{A_1, A_2, \ldots, A_{|A|}\}$. If two references p and q are put in the same cluster in A, this means that system S believes that p and q co-refer. Otherwise, the system decides that they do not co-refer. Hence, for each reference $r \in R$ we can also determine what system S believes is the co-reference set A_r for r: $A_r = \{q : q \in A_i \wedge r \in A_i\}$. Answer A_r is correct when its decision matches the ground truth, that is, when $A_r = C_r$ for all $r \in R$.

An ER system can make errors in clustering. To be able to compare different ER algorithms on a given dataset, several metrics have been proposed to assess the *quality* of the resulting clusterings, which essentially evaluate how different the clustering A is from the ground truth A^+.

Graphical Problem Representation. The ER problem can be visualized as a graph of references (set R) connected via co-reference edges (set E).[3] The goal for an ER algorithm is to predict for each edge $e_i \in E$, where $e_i = (p, q)$, whether it is a *positive* edge, meaning $p \approx q$, or

[3]Note at this stage it is a complete graph. We use a blocking technique to reduce the complexity as explained later.

Table 1: Notation.

Notation	Meaning
$\mathcal{D}$	The dataset being processed.
S_i	The i-th base-level ER system.
n	The number of base-level ER systems to be combined.
$A^{(i)}$	The clustering produced by S_i on $\mathcal{D}$.
A^{+}	The ground truth clustering for $\mathcal{D}$.
$E = \{e_1, \ldots, e_{\lvert E\rvert}\}$	The co-reference edge set.
e_j	The j-th co-reference edge.
$\mathbf{d}_j = \{d_{j1}, \ldots, d_{jn}\}$	The decision feature vector for e_j.
d_{ji}	The merge/not-merge decision made by S_i for edge e_j.
a_j^{+}	The ground truth label for edge e_j.
$a_j^{\star}$	The predicted label for edge e_j by the ER ensemble.
$\mathbf{f}_j = \{f_{j1}, \ldots, f_{jn}\}$	The context feature vector for e_j.
f_{ji}	The overall context feature for e_j derived from the clustering by S_i.
f_{ji}^k	The k-th type of context of e_j by S_i.
$\mathbf{v}_j = \{v_{j1}, \ldots, v_{jn}\}$	The confidence feature vector for e_j.
v_{ji}	The confidence in "merge" decision by S_i for edge e_j.

a *negative* edge, meaning $p \not\approx q$. Hence, the task of an ER algorithm can also be viewed as partitioning of the graph.

Figure 1 illustrates examples of graphical representations of various clusterings for a toy dataset with seven nodes A, B, C, D, E, F, and G. Each node corresponds to one reference. The bold circles represent clusters in which the nodes are believed to co-refer. The solid lines represent the positive edges that connect the nodes in the same cluster. All edges between the nodes from different clusters are negative edges, they are not shown in Figure 1. Figures 1(a) and 1(b) demonstrates output clusterings $A^{(1)}$ and $A^{(2)}$ produces by two different ER systems S_1 and S_2. Figure 1(c) shows the ground truth clustering A^{+}. Figure 1(d) shows the result of applying a blocking technique to reduce the number of edges to consider, which will be covered in Section 3.3.

3.2 ER Ensemble

In the setting of the ER Ensemble problem, several ER systems $S_1, S_2, \ldots, S_n$ are provided as black boxes – the internal details of their ER algorithms are not known. These systems can be applied to dataset $\mathcal{D}$ to cluster the set of references R. Each system S_i will produce its set of clusters $A^{(i)} = \{A_1^{(i)}, A_2^{(i)}, \ldots, A_{\lvert A^i\rvert}^{(i)}\}$. The goal is to be able to combine the resulting clusterings $A^{(1)}, A^{(2)}, \ldots, A^{(n)}$ into a single clustering $A^{\star}$ such that the quality of $A^{\star}$ is higher than that of $A^{(i)}$, for $i = 1, 2, \ldots, n$.

Specifically, we will consider the following instance of the problem. Each edge $e_j \in E$, where $e_j = (p, q)$, has the ground truth label a_j^{+} associated with it, where $a_j^{+} = 1$ if p and q co-refer, and $a_j^{+} = -1$, otherwise. The ground truth labeling is unknown to ER algorithms beforehand and the goal is to predict it. For edge e_j systems $S_1, S_2, \ldots, S_n$ output their decisions $\mathbf{d}_j = \{d_{j1}, d_{j2}, \ldots, d_{jn}\}$, where $d_{ji} \in \{-1, 1\}$. If S_i predicts that $a_j^{+} = 1$ (that is, predicts $p \approx q$), then $d_{ji} = 1$, otherwise $d_{ji} = -1$.

The task of ER ensemble is for each $e_j \in E$ to provide a mapping $\mathbf{d}_j \to a_j^{\star}$. Here, $a_j^{\star} \in \{-1, 1\}$ is the prediction of the combined algorithm for edge e_j, where $a_j^{\star} = 1$ if the overall algorithm believes $a_j^{+} = 1$, and $a_j^{\star} = -1$ otherwise. Table 1 summarizes some of the notation used.

Table 2: Examples of the decision features for toy dataset. Here, d_{j1} is the decision by S_1 and d_{j2} is the decision by S_2.

Edge	Decision Features	Truth	$g(e_j) = w_1 d_{j1} + w_2 d_{j2}$	
	(d_{j1}, d_{j2})	a_j^{+}	$w_1 \geq w_2$	$w_1 < w_2$
A-B	$(+1, -1)$	$\mathbf{-1}$	$+1$	-1
A-C	$(+1, -1)$	$\mathbf{-1}$	$+1$	-1
A-D	$(-1, -1)$	$\mathbf{-1}$	-1	-1
B-C	$(+1, +1)$	$\mathbf{-1}$	$+1$	$+1$
B-D	$(-1, +1)$	$\mathbf{-1}$	-1	$+1$
C-D	$(-1, +1)$	$\mathbf{+1}$	-1	$+1$
E-F	$(+1, -1)$	$\mathbf{+1}$	$+1$	-1
E-G	$(+1, -1)$	$\mathbf{+1}$	$+1$	-1
F-G	$(+1, -1)$	$\mathbf{+1}$	$+1$	-1
Accuracy	–	–	56%	44%

Example. Table 2 presents the decision features produced by ER systems S_1 and S_2 and the ground truth labels of all the edges in the toy dataset in Figure 1. For example, in $A^{(1)}$ produced by S_1, nodes A, B, and C are clustered in one group. Therefore, the decisions d_{j1} for edges $A - B$, $A - C$ and $B - C$ are all $+1$. The decisions d_{j1} for edges $A - D$, $B - D$, $C - D$ are all -1 since node D is in a separate cluster. We will discuss the $g(e_j)$, w_1, and w_2 mentioned in Table 2 later in Section 3.4.

3.3 Using Blocking to Improve Efficiency

To address the scalability issues, we employ the blocking process in our approach. The standard blocking uses canopy-like techniques [18, 28]. The general idea is to first utilize some simple and fast (but crude) techniques to identify quickly the pairs of references that have a chance to co-refer and those that are highly unlikely to co-refer. Only pairs of references that might co-refer are then analyzed further. This avoids creating $O(\lvert R\rvert^2)$ edges needed to connect each pair of references. The connected subgraphs, that result from applying blocking, form blocks of potential co-references. Such a block is known as Virtual Connected Subgraph or *VCS* [8]. All nodes in the same VCS have certain similarity so that any two of them *might* co-refer. Any nodes in different VCS's are assumed to have no chance to co-refer. The clustering task can be logically viewed as that of partitioning the VCS's.

One natural way to apply blocking in our context, which can be enhanced further by other blocking techniques, is to consider decisions $d_{j1}, d_{j2}, \ldots, d_{jn}$ made for each edge e_j by base systems $S_1, S_2, \ldots, S_n$. If at least one decision d_{ji} is positive, then the two nodes that correspond to edge e_j are put into the same VCS and the edge will be represented by a feature vector and considered for further processing. Otherwise, the nodes will be put into separate VCS's since none of the base systems believes they could co-refer. For the toy scenario in Figure 1, the blocking process will divide the seven nodes into two VCS's, demonstrated in Figure 1(d).

Blocking is important to improve not only the efficiency but also the quality of our algorithms. The concept of VCS will be used when we create context features that are utilized by ER ensemble, which will be elaborated in Section 4.1.

3.4 Naive Solution: Voting

A basic approach for combining multiple systems is majority voting. Voting has been widely used for the problem of classification. To decide whether a given edge $e_j \in E$

is positive or negative, a voting solution would count the predictions $d_{ji} \in \{-1, +1\}$ made by each base-level system S_i. If $\sum_i d_{ji} \geq 0$ then more base level systems voted e_j to be positive and edge e_j would be considered positive in the final decision.[4] Otherwise, if $\sum_i d_{ji} < 0$ then edge e_j will be considered negative.

A more advanced voting scheme is weighted voting (WV). WV associates with each base level system S_i weight w_i that represents how good decisions of S_i are in general. For making its final edge prediction, weighted voting analyzes $\sum_i d_{ji} w_i$ and compares it with zero. Thus, the majority voting can be viewed as an instance of the weighted voting where $w_i = 1$ for each base system S_i.

Let us continue to use the toy dataset in Figure 1 to illustrate the limitation of the WV solution. Suppose that for any edge e_j we employ a standard WV function $g(e_j) = w_1 d_{j1} + w_2 d_{j2}$, with weights w_1 and w_2, where $w_1 + w_2 = 1$. WV will predict $a_j^* = 1$ if $g(e_j) \geq 0$, and $a_j^* = -1$ otherwise. Now, if we set $w_1 \geq w_2$, then the combined prediction a_j^* for edge e_j will always be the same as d_{1j} by S_1. On the other hand, if we set $w_1 < w_2$, then the a_j^* will always be the same as d_{2j} by S_2. The last two columns of Table 2 demonstrate the resulting edge labels in these two situations. If we define the accuracy of an ER system as the number of edges that are correctly labeled, we can see that the accuracy of S_1 is 56% (5 out of 9 edges are labeled correctly), and the accuracy of S_2 is 44% (4 out of 9 edges are labeled correctly). Therefore, no matter how we choose w_1 and w_2, the best accuracy we can get from combining the two base systems by Weighted Voting is 56%.

The problem with the voting framework, including weighted voting (WV), is that it is a *static* non-adaptive scheme with respect to the context. In the next section we define context features used by the proposed solution and show how they can be employed to improve the quality of entity resolution. We will illustrate on the above example the effect of using such context features on the quality of the outcome.

4. CONTEXT BASED FRAMEWORK

In this section we present the proposed context based ER ensemble framework. Its goal is to combine clustering results of multiple ER systems $S_1, S_2, \ldots, S_n$. The framework is aware of neither the internal details of each individual system S_i nor on which principles the clusters are generated by S_i. The proposed solution generates the final clustering $A^\star$ based on the output $A^{(i)}$ produced by each system S_i as well as the context. In this section we start by defining context features in Section 4.1. After that we overview the overall approach in Section 4.2. The solution utilizes a meta level classification covered in Section 4.3. Section 4.4 then explains how the results of meta classifications are utilized to create the final clustering.

4.1 Context Features

Intuitively, deciding which features to use for making edge decisions should be guided by several principles, including:

- *Effectiveness.* The features should be effective for capturing which entity resolution system works well in the given context.
- *Generality.* For the approach to be generic, the features should also be fairly generic. Thus, they should not rely on types of information that is present in only one or a few datasets/domains.

[4]Ties are broken by assuming e_j to be positive.

Finding features that satisfy these criteria turned out to be a nontrivial task. Since we desire a general ER ensemble solution that treats the base-level systems as black boxes, we can only use the features that are derived from the output of those systems, which are the clusters of nodes/references in the graphical representation. While there can be many features, we have observed that they can be frequently classified as either global (VCS-level) features (e.g., the number of clusters) or local (cluster-level) features (e.g., the node fanout). An interesting property of the global and local features we use is that, unlike the commonly used features, they are *predictive.* They are based on predicting or estimating the ground truth values of some unknown parameters.

Number of Clusters. Section 1 has already motivated using the number of clusters K as a context feature. The first feature we use is derived from the expected number of clusters per VCS as well as the number of clusters generated by each base-level system. First, for each edge $e_j \in E$ its VCS is identified. Let K_i be the number of clusters generated by system S_i for that VCS. Observe that the true number of clusters K^+ in the VCS is not known beforehand to the algorithm and thus cannot be employed as a feature. However, K^+ can be estimated by various methods. Specifically, we use regression[5] to estimate the true number of clusters in the VCS based on the values of $K_1, K_2, \ldots, K_n$.

Let $K^\star$ be the estimated number of clusters. For each edge $e_j \in E$ the first context feature vector $\mathbf{f}_j^1 = \{f_{j1}^1, f_{j2}^1, \ldots, f_{jn}^1\}$ is defined as $f_{ji}^1 = 1 - \frac{|K_i - K^\star|}{\Delta_i^1}$. The parameter Δ_i^1 is a normalization factor used to normalize the values of f_{ji}^1 to $[0, 1]$ interval. The closer the K_i is to $K^\star$, the more confidence we can place in the answer of system S_i and the larger f_{ji}^1 is. Given so defined features, when making the decision on edge e_j the classifier can learn to give preference to those systems S_i's that have higher values of f_{ji}^1.

Node Fanout. The second context feature type is defined at the edge level. It exploits the nodes adjacent to the edges. Let N_{vi} for any node $v \in V$ be the number of positive edges incident to it according to clustering $A^{(i)}$ produced by system S_i. Let $N_v^\star$ be the estimated true number of positive edges for v computed by employing regression on $\{N_{v1}, N_{v2}, \ldots, N_{vn}\}$. Then, for each edge $e_j \in E$ its context feature vector $\mathbf{f}_j^2 = \{f_{j1}^2, f_{j2}^2, \ldots, f_{jn}^2\}$ is defined as $f_{ji}^2 = 1 - \frac{1}{2} \sum_{v \in e_j} \frac{|N_{vi} - N_v^*|}{\Delta_i^2}$. Here Δ_i^2 is a normalization factor employed for scaling values of f_{ji}^2 to $[0, 1]$ interval, and $v \in e_j$ refers to the two end nodes of edge e_j.

The overall context feature vector $\mathbf{f}_j$ is computed as a linear combination of the two feature vectors $\mathbf{f}_j = \alpha_1 \mathbf{f}_j^1 + \alpha_2 \mathbf{f}_j^2$. The framework allows for adding more types of contexts if needed. In case of m types of context features, the overall feature vector is computed as

$$f_{ji} = \sum_{k=1}^{m} \alpha_k f_{ji}^k, \qquad (1)$$

where $\sum_k \alpha_k = 1$.

[5]Regression refers to a standard mathematical apparatus for learning a function from data. In this case we learn a function $K = g(K_1, \ldots, K_n)$. We use a third party regression algorithm in our framework.

Table 3: Example of Context Features. $f_{ji} = \frac{f^1_{ji}+f^2_{ji}}{2}$.

Edge	Truth	S_1			S_2			
	a^+_j	f^1_{j1}	f^2_{j1}	f_{j1}	f^1_{j2}	f^2_{j2}	f_{j2}	a^*_j
A-B	**−1**	0.67	0.33	0.5	0.67	0.67	0.67	−1
A-C	**−1**	0.67	0.5	0.59	0.67	0.83	0.75	−1
A-D	**−1**	0.67	0.5	0.59	0.67	0.83	0.75	−1
B-C	**−1**	0.67	0.5	0.59	0.67	0.5	0.59	−1
B-D	**−1**	0.67	0.5	0.59	0.67	0.5	0.59	−1
C-D	**+1**	0.67	0.67	0.67	0.67	0.67	0.67	+1
E-F	**+1**	1	1	1	0	0.33	0.17	+1
E-G	**+1**	1	1	1	0	0.33	0.17	+1
F-G	**+1**	1	1	1	0	0.33	0.17	+1

Notice that the values of $K^\star$ and $N^\star$ learned from the classifiers might not always be accurate predictions of K^+ and N^+. The intuition is that by aggregating the contexts derived from such features (which are perhaps different from the exact values) and the decisions of multiple base-level algorithms, the ER ensemble will be able to tolerate errors in the predicted $K^\star$ and $N^\star$ and will be able to achieve a better quality clustering solution. In general, adding more context features can increase the robustness and accuracy of the system, as long as they satisfy the above criteria and are derived directly from the output of base-level ER systems.

Example. Let us illustrate the concepts introduced in this section with the help of the running example from Figure 1. Recall that the figure shows a scenario where the goal is to cluster seven elements based on the output of two base-level ER systems S_1 and S_2. Table 3 demonstrates values for the context features as well as the ground truth labels for the corresponding edges.

Let us demonstrate the context feature generation using edge $A - B$ as an example. For simplicity of calculations assume that using regression the algorithm accurately predicted $K^\star = K^+ = 3$ for VCS$_1$ and $K^\star = K^+ = 1$ for VCS$_2$. Observe that the numbers of clusters K_1 and K_2 for VCS$_1$ are both 2, and for VCS$_2$ the number of clusters K_1 is 1 and K_2 is 3. Thus, for edge $A - B$ in VCS$_1$, the VCS-level context feature $f^1_{ji} = 1 - \frac{|2-3|}{3} = 0.67$, given that $\Delta^1_i = 3$. The value of Δ^1_i is computed as the maximum possible number of clusters for VCS$_1$ (which is the same as the number of nodes in this VCS) minus the minimum possible number of clusters in VCS$_1$ (which is 1).

Now let us compute the edge-level context feature vectors. Observe that for node A we have $N_1 = 2$, $N_2 = 0$, and $N^\star = 0$. For node B we have $N_1 = 2$, $N_2 = 2$, and $N^\star = 0$. Therefore, for edge $e_j = A - B$ we have $f^2_{j1} = 1 - \frac{1}{2}\left(\frac{|2-0|}{3} + \frac{|2-0|}{3}\right) = 0.33$ and $f^2_{j2} = 1 - \frac{1}{2}\left(\frac{|0-0|}{3} + \frac{|2-0|}{3}\right) = 0.67$, given that $\Delta^2_i = 3$, which is the number of nodes in the VCS$_1$ minus 1, that is the maximum value N_1 could have when all nodes are grouped in one cluster. If we choose $\alpha_1 = \alpha_2 = \frac{1}{2}$ in Eq. (1), we have $f_{j1} = \frac{0.67+0.33}{2} = 0.5$ and $f_{j2} = \frac{0.67+0.67}{2} = 0.67$. The context feature vectors of all the other edges are computed in a similar fashion. Their values are shown in Table 3.

4.2 Overview of the Approach

In this section we present an overview of all the necessary components of the proposed solution. The overall framework is demonstrated in Figure 2. In the proposed supervised learning approach, the process of combining multiple ER systems can be divided into Training and Testing (Application) parts. The steps of these two parts are outlined below:

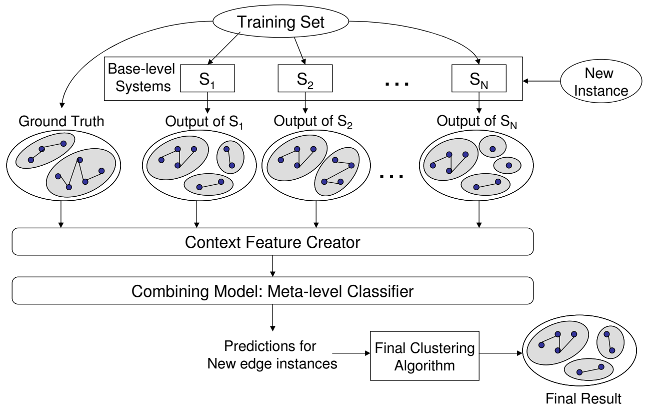


Figure 2: The ER Ensemble Framework. The ground truth clustering is used only for training.

- **Training:**
  1. Apply base-level ER systems $S_1, S_2, \ldots, S_n$ to the training dataset $\mathcal{D}_{train}$. For each edge e_j, record the output of the base-level systems $S_1, S_2, \ldots, S_n$ to generate the decision feature $\mathbf{d}_j$ (Section 3.2).
  2. Apply blocking preprocess to reduce the number of edges in E to be considered (Section 3.3).
  3. For each edge $e_j \in E$ determine its context $\mathbf{f}_j$ to be used as the input of the combining model (Section 4.1).
  4. Use both the decision feature vector $\mathbf{d}_j$ and context feature vector $\mathbf{f}_j$ to train the combining model given the ground truth labels a^+_j are known for the training dataset $\mathcal{D}_{train}$ since it is fully labeled (Section 4.3).
- **Apply to the dataset being processed:**
  1. Same as for training, except for now applying $S_1, S_2, \ldots, S_n$ to the new dataset being processed $\mathcal{D}$.
  2. Same as for training.
  3. Same as for training.
  4. Apply trained meta-level combining model to the meta-level feature vector $\mathbf{d}_j \times \mathbf{f}_j$ to predict the class label $a^\star_j$ for each edge e_j.
  5. Create the resulting clustering from the predictions of meta-level classifier for the edges (Section 4.4).

In the subsequent sections we will explain these steps in more details.

4.3 Meta-level Classification

In this section we present two algorithms for meta-level classification. Both algorithms utilize the context features defined in the previous section. The first one takes advantage of the richer feature space by learning a mapping from the decision and context features into edge label predictions by training a classifier (Section 4.3.1). The second approach analyzes each individual base-level ER system S_i and trains a prediction model to estimate the goodness of the performance of S_i under various contexts. It then uses these prediction models to adjust decisions made by the base systems and combines them into the overall prediction for each edge (Section 4.3.2).

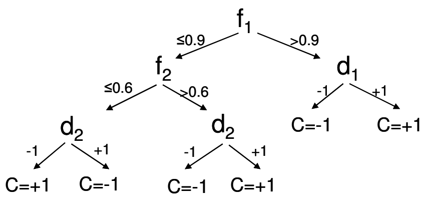


Figure 3: Example Decision Tree for Toy Dataset.

4.3.1 *Context-Extended Classification*

This section presents the first proposed meta-level classification approach. Recall that to generate features from the training set, the algorithm uses regression to compute $K^\star$ from $\{K_1, K_2, \ldots, K_n\}$ and another regression to compute $N_v^\star$ from $\{N_{v1}, N_{v2}, \ldots, N_{vn}\}$. The algorithm then computes f_{ji} according to Eq. (1). At this point it is possible to learn a mapping from features $\mathbf{d}_j \times \mathbf{f}_j$ (or, alternatively $\mathbf{d}_j \times \mathbf{f}_j^1 \times \mathbf{f}_j^2$) into edge predictions $a_j^\star$ by training a classifier using the ground truth labels a_j^+. For instance, Figure 3 presents a sample decision tree classifier learned by utilizing the meta-level features from Table 3 (for the $\mathbf{d}_j \times \mathbf{f}_j^1 \times \mathbf{f}_j^2$ case) as the training data. Given such a decision tree, the predictions a^* for all the edges in the toy dataset are presented in Table 3, which are exactly the same as the ground truth labels.

The proposed approach however employs a more effective and accurate strategy by reducing the dimensionality of feature space. Specifically, the approach computes values v_{ji}:

$$v_{ji} = \begin{cases} f_{ji} & \text{if } d_{ji} = +1; \\ 1 - f_{ji} & \text{if } d_{ji} = -1. \end{cases} \tag{2}$$

Observe that while f_{ji} ($0 \le f_{ji} \le 1$) serves as a confidence in decision $d_{ji} \in \{-1, +1\}$ by system S_i for edge e_j, the value of v_{ij} serves as a confidence in decision $+1$ by S_i for e_j. Namely, if decision $d_{ji} = +1$ then $v_{ji} = f_{ji}$ and thus v_{ji} is a confidence in $+1$ decision. If decision $d_{ji} = -1$ then the confidence in -1 decision is f_{ji} while the confidence in $+1$ decision is $1 - f_{ji}$ which equals to v_{ji} by its definition. The value of v_{ij} can be viewed as a confidence in the "merge" decision by S_i for e_j. If it is high, the likely correct decision for e_j is "merge" and if it is low – "do not merge".

Instead of using $2n$-dimensional $\mathbf{d}_j \times \mathbf{f}_j$ vector to train a classifier, the proposed approach employs n-dimensional vector $\mathbf{v}_j$, where $\mathbf{v}_j = (v_{j1}, v_{j2}, \ldots, v_{jn})$. Values of a_j^+ serve as the class labels for the classifier. Recall that for edge $e_j = (p, q)$ the ground truth label a_j^+ can take on two values of -1 or $+1$ depending on whether p and q co-refer. After the algorithm has the meta-level feature vectors $\mathbf{v}_j$ created for all $e_j \in E$, it can learn a meta-level classifier that is able to predict the class of meta vectors for new edge instances.

Notice that we could use different types of classifiers for this purpose, such as Logistic, Naive Bayes, SVM, Decision Trees, etc [39]. Often the output of such classifiers includes the predictions of class values as well as the confidences associated with the predictions. When such confidences are available, the algorithm can transform them into edge weights and apply Correlation Clustering to get a more effective solution as will be discussed in Section 4.4.

4.3.2 *Context-Weighted Classification*

The key idea of the second meta-level classification approach is to learn for each base-level system S_i a model $\mathcal{M}_i$ that would predict how accurate S_i tends to be in a given context. Recall from Section 4.1 that for each edge e_j the framework computes the decision feature d_{ji} by system S_i and the context features $(f_{ji}^1, f_{ji}^2, \ldots, f_{ji}^m)$, where $m = 2$ in that section. In training data, the ground truth label a_j^+ that tells whether this edge is positive or negative is also available. Now, by comparing d_{ji} and a_j^+, we can determine whether the system S_i made a correct or wrong decision for edge e_j for the given context features $(f_{ji}^1, f_{ji}^2, \ldots, f_{ji}^m)$. Specifically, we can define c_{ji} as

$$c_{ji} = \begin{cases} +1 \text{ (correct)} & \text{if } d_{ji} = a_j^+; \\ -1 \text{ (wrong)} & \text{if } d_{ji} \neq a_j^+. \end{cases} \tag{3}$$

To create model $\mathcal{M}_i$ for S_i the algorithm trains a classifier on $(f_{ji}^1, f_{ji}^2, \ldots, f_{ji}^m)$ as data points and c_{ji} as class labels. Most modern classifiers such as SVM, provide a measure of confidence in the classification decision p_{ji}, in addition to the decision itself (c_{ji}^*). These confidences are employed by the algorithm as will be explained shortly. Figure 4 illustrates the context feature spaces for the toy dataset created by S_1 and S_2. Each points in the space corresponds to one edge, and the "+" and "−" denote the correct or wrong decision by the system.

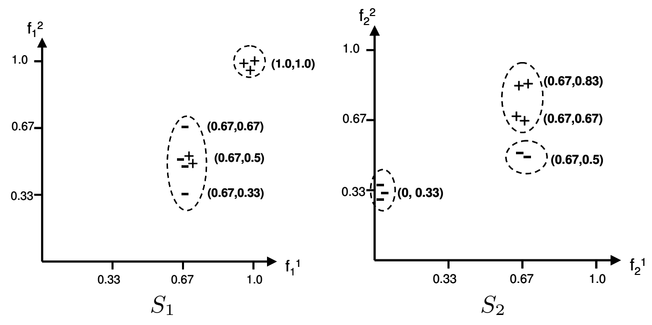


Figure 4: Example of a classifier learned on the context feature space. Here, '+' corresponds to an correct decision and '−' to wrong decision.

Once model $\mathcal{M}_i$ is trained for each base-level system S_i, the algorithm can apply it for making its edge prediction a_j^* on new data. The process can be summarized as follows:

1. Apply each S_i to the new data. Compute the decision feature $d_{ji} \in \{-1, 1\}$ and the context features $(f_{ji}^1, f_{ji}^2, \ldots, f_{ji}^m)$ for each edge e_j.

2. For each edge e_j apply $\mathcal{M}_i$ on features $(f_{ji}^1, f_{ji}^2, \ldots, f_{ji}^m)$ to get $c_{ji}^* \in \{-1, 1\}$ and p_{ji}. Here, c_{ji}^* indicates whether the decision d_{ji} is likely to be correct or wrong, and p_{ji} is the confidence associated with this assessment by the classifier. If $c_{ji}^* = -1$, this means with confidence p_{ji} decision d_{ji} is wrong, and then the decision is adjusted to the reverse one, which is more probable according to $\mathcal{M}_i$: $d_{ji} = -d_{ji}$.

3. Combine the predictions by all the base-level systems to create a new meta-level vector: $\mathbf{v_j} = (v_{j1}, \ldots, v_{jn})$, where $v_{ji} = d_{ji} p_{ji}$ is the weighted adjusted decision of base-level system S_i on edge e_j. Learn a meta-level classifier to map $\mathbf{v_j} \to a_j^*$ the same way as discussed in Section 4.3.1.

Example. Let us again use the toy dataset to illustrate the algorithm. Assume the dotted circles in Figure 4 represent the classifiers learned for this toy dataset. Let us use the edge $A - B$ as an example. The context feature vector (f_1^1, f_1^2) for system S_1 is $(0.67, 0.33)$ as shown in Table 3. The classifier will predict it as "$-$" with probability $\frac{2}{3}$, since 4 out of 6 edges in that circle area predicted "$-$". That is, with probability $\frac{2}{3}$, S_1 made a wrong decision on edge $A - B$, and therefore, the decision d_{j1} of S_1 on this edge needs to be adjusted: $d_{j1} = -d_{j1} = -1$. The context feature vector (f_2^1, f_2^2) for S_2 is $(0.67, 0.67)$, for which the classifier will make the prediction to be "$+$" with probability 1. This means that with probability 1 system S_2 made a correct decision on edge $A - B$ and therefore $d_2 = -1$. If we use weighted combination to predict the final edge class, the combined weight on edge $A - B$ will be: $-1 \times \frac{2}{3} + (-1) \times 1 = -\frac{5}{3} < 0$. Therefore the final decision of the ensemble algorithm on edge $A - B$ will be that it is a negative edge, i.e., A and B do not co-refer. By using the same procedure for the rest of the edges in this example we will get the final prediction exactly the same as the ground truth. Recall from Section 3.4 that this was not possible when using a weighted voting solution since it does not utilize the context.

4.4 Creating Final Clusters

Once the algorithm gets the class label predictions a_j^* for each edge e_j (along with the associated confidence p_j) using the meta-level classifier, it then can create the final clusters by applying one of the generic clustering algorithms, such as agglomerative clustering and correlation clustering. The proposed framework employs Correlation Clustering (CC) which generates the partitioning of data points that optimizes a score function [3]. CC was specifically designed for the situation where there are conflicts in edge labeling.[6] It takes as input a graph where nodes correspond to references and edges between two nodes u and v are labeled, in the simplest case, with $+1$ if there is evidence that u and v co-refer, and with -1 otherwise. CC finds a clustering that has the most agreement with the edge labeling, therefore limiting the conflicts as much as possible.

Since optimal partitioning is an NP-hard problem, many approximation algorithms have been proposed in the past. The framework employs a variation of the approximation algorithm similar to that of in [27]. Interestingly, by setting certain parameters accordingly, correlation clustering can be made to behave like agglomerative clustering, that is correlation clustering is a more generic scheme [20].

5. EXPERIMENTAL EVALUATION

In this section we empirically study the effectiveness of our approaches by comparing it with other combination approaches on datasets taken from two different domains. We first present our experimental setup including datasets and evaluation metrics. We show that using the proposed framework for ER ensemble can improve the overall clustering results compared to those of the individual base-level algorithms and the state-of-the-art clustering ensemble techniques. We also show that the proposed solution is more stable with the changes of base-level algorithms than traditional cluster aggregation algorithms.

[6]E.g., edges (u, v) and (v, w) are labeled $+1$ whereas edge (u, w) is labeled -1.

5.1 Experimental Setup

5.1.1 Dataset

To show the domain-independence of our ER ensemble approach, we test our algorithm on datasets taken from two domains: the Web and Publication domains. For the Publication domain we use the RealPub dataset from [7,8], which contains 11,682 *publications*, 14,590 *authors*, 3084 *departments* and 1494 *organizations*. The goal is to resolve the *author* entities. For the Web domain we test the proposed framework on the dataset created by Bekkerman and McCallum in WWW'05 [4]. The dataset contains webpages for 12 different person names. The dataset has been created by querying the web using the Google search engine with different person names. The top 100 returned webpages of the Web search were gathered and labeled manually for each person to obtain the "ground truth" of the data. We conduct further manual examination and correct some errors in the original labeling. Also, in the original WWW'05 dataset, all pages labeled as "others" were treated as one namesake for each query. We carefully examined these pages and relabeled many of them to refer to the proper namesake. We perform leave-one-out cross-validation on the 12 person names to evaluate the performance of our proposed framework on WePS task, i.e., clustering web pages that are about the same namesakes. The reason we choose dataset from the Web domain is that the ambiguity of the references is usually high in those cases. For example, in RealPub, the most common uncertainty for a VCS is 2-4 (i.e., for each VCS there are two to four namesakes), while for WWW datasets, for example, the uncertainty ranges from 2 to 61 and the average is around 30.

5.1.2 Evaluation Metrics

For ER problem, the most adopted evaluation metrics in the literature is $F1$-measure, which is the harmonic mean of *Precision* and *Recall*. Recently, researchers argued $F1$-measure is not adequate for the task of ER and proposed to use *Purity* and *Inverse Purity* and their harmonic mean F_P to evaluate the quality of clusterings [2,8]. For the web domain, including in WePS-1 and WePS-2 challenges [1], *B-cubed* has been used as another measure in addition to F_P [21,29]. We will employ all the three measures for assessing the quality of clusterings.

5.1.3 Baseline Methods

We will choose the base-level algorithm that shows the highest-quality results on the whole dataset as one of the baselines. We refer to this baseline as `BestBase` algorithm. Another baseline we use is a Majority Voting (`MajorVot`) approach. For each pair of webpages, the decision of whether or not they should be merged is made by the majority of the base-level algorithms. Once we have the decision for each pair, we make the final clustering decision by applying transitive closure to the data. This approach is essentially the same as the state-of-the-art clustering ensemble approach proposed by Fred and Jain in [14]. A similar baseline we use is Weighted Voting (`WtVot`) scheme, in which the weight of each base-level system is learned from the training data.[7]

[7]In `WtVot`, the weight w_i is learned from training data for

We also implement the clustering aggregation algorithms `BestClustering`, `BALLS`, and `Agglomerative` algorithms presented in [16] as other baselines to compare with. We refer the algorithm derived from [41] as `StandERE` (Standard ER Ensemble), which combines multiple base-level systems by creating meta-level feature vectors from decision features only. We cannot apply the algorithms from [41] directly because the base-level systems we are combining are assumed to be provided by the third party and we do not know the details of how the clustering is conducted. On the other hand, in [41] the base-level systems are state-of-the-art classifiers that classify the reference pairs (i.e., make match/no-match decisions) with known features (e.g., person names, city, phone number, etc.) from the data itself. We refer to our ER ensemble approaches as `ExtendedERE` (Extended ER Ensemble) (Section 4.3.1) and `WeightedERE` (Weighted ER Ensemble) (Section 4.3.2).

5.1.4 *Classifiers*

We use classifiers implemented in Weka suite [15] for the purpose of generating context features as well as training our meta-level learning algorithm. The classifiers we explored for context feature generation are those designed for numeric class type, including *SMOreg*, *LinearRegression*, *DecisionTable*, etc. The classifiers we explored for meta-level learning are those for nominal class type, including: *JR48*, *SMO*, *Logistic*, etc. These classifiers have been studied and evaluated as meta-level classifiers in the literature for classification problems [34]. We experimented with most of the classifiers for empirical evaluation. In the results we report next, we use *SMOreg* (a support vector regression model implemented in Weka) for the context feature learning and *Logistic* (a logistic regression model in Weka) for the meta-level classifier. We choose them because they in general yield better results. There are also some classifiers that produce close results but the difference are not statistically significant and they need longer training/running time (e.g., *LogitBoost*, *Bagging*, *AdaBoost*, etc. [15]).

5.2 Experiments on Web Domain

This section evaluates the proposed ensemble solution on the Web domain. It compares it with other ensemble techniques and studies the effect of the selection of the base-level algorithms on the results of these ensemble approaches.

5.2.1 *Base-level Algorithms*

The strength of our ensemble approach is best manifested when different base-level algorithms have best performance on different part of the data. The learning approach employed by the proposed algorithm will decide the final solution for the combination of base-level algorithms, based on the context of the data.

To illustrate this, we first conducted a simple experiment. In it, we manually introduced the "ideal" base-level system – an artificial system that knows the ground truth clustering (which is impossible in real life) and always outputs it as its result. In this case no matter what the context of the data is, the ideal system always gets the best result. When we add this ideal system to the set of the base-level algorithms, as expected, the clustering result of our ensemble algorithm is always the same as that of the ideal system. This simple experiment shows that if one base-level algorithm consistently performs the best, the proposed approach will catch this property and will also get the best results. In the rest of this section we will show that, when, on the other hand, none of individual algorithms performs the best on all parts of the dataset, our algorithm combines the strengths of multiple base-level algorithms and delivers superior final result.

Table 4: Sample Set of Base-level Algorithms.

Algorithm	Descriptions
Algo 1	eTF/IDF, Agglom. Algo., threshold= 0.03
Algo 2	eTF/IDF, Agglom. Algo., threshold= 0.05
Algo 3	eTF/IDF, Agglom. Algo., threshold= 0.10
Algo 4	eTF/IDF, Agglom. Algo., threshold= 0.12
Algo 5	eTF/IDF, Agglom. Algo., threshold= 0.16
Algo 6	CS + eTF/IDF, CC Algo., neg_w= 0.0 [23]
Algo 7	CS + eTF/IDF, CC Algo., neg_w= -0.5 [23]
Algo 8	CS + eTF/IDF, CC Algo., neg_w= -1.0 [23]
Algo 9	Iterative Context-based Agglom. Algo. [5]
Algo 10	Clustering With Rich Features [13]

For the base-level systems, we mainly choose the algorithms that are known to perform well for entity resolution on Web domain, proposed in [20]. We use variations of those algorithms with different parameter settings and also some other ER algorithms [5, 13] to be included as the base-level algorithms in our proposed framework. Table 4 presents a summary of the set of base-level algorithms we use in the following experiments. The main difference among these algorithms are (a) the similarity metrics chosen to compare pairs of references and (b) the clustering algorithms that generate the final clusters based on similarities.

Algo 1-5 are Agglomerative Vector Space clustering algorithms with eTF/IDF as similarity metrics and with different merging threshold. eTF/IDF is the enhanced TF/IDF method that involves preprocessing of the web pages to be processed. The thresholds are chosen so that these algorithms perform differently on different persons. For some persons for the 12 queried names, the algorithms with lower thresholds perform better, and for other persons, the situation is reverse. Algo 6-8 are variations of the algorithm proposed in [23]. We apply an off-the-shelf information extraction tool (i.e., GATE [11]) to extract entities (i.e., Persons, Locations, and Organizations) from the web pages and create an entity-relation graph on the data. The similarities are computed by combining the connection strengths (CS) derived from the graph and eTF/IDF between web pages. An approximation of Correlation Clustering (CC) algorithm [3] is then applied to generate the final clustering. More details of this algorithm can be found in [23]. Algo 9 and 10 are implementations of algorithms presented in [5] and [13] respectively. The algorithm in [5] was initially proposed for publication domain and not intended for Web domain. We have done the corresponding transformation of the modules of the original algorithm to make it applicable to the WePS task. Notice that in the paper that initially used the WWW'05 dataset [4], the algorithm proposed is supposed to solve a related but different problem, which is to find all webpages that are about a particular namesake.

which ground truth is known. If edge e_j in the training data is positive, an equation is created: $\sum_i d_{ji} w_i > 0 - s_j$. If e_j is negative, the equation for it is $\sum_i d_{ji} w_i < 0 + s_j$. Here s_j is a non-negative slack variable which is similar to those used in SVM. By creating all equations for all the edges in the training data and by requesting that the overall slack be minimized, the task reduces to solving a linear program which produces the desired values of weights w_i's.

Table 5: **Comparison of Multiple Aggregation Algorithms with Different Number of Base-Level ER Systems.**

	5 Base-level Systems			10 Base-level Systems			20 Base-level Systems		
Method	$\mathbf{F_P}$	**B-Cubed**	**F1**	$\mathbf{F_P}$	**B-Cubed**	**F1**	$\mathbf{F_P}$	**B-Cubed**	**F1**
BestBase	0.856	0.817	0.709	0.872	0.818	0.715	0.872	0.818	0.715
MajorVot	0.846	0.793	0.668	0.872	0.830	0.726	0.857	0.802	0.682
WtVot	0.850	0.807	0.697	0.867	0.817	0.733	0.861	0.811	0.726
BestClust	0.822	0.770	0.599	0.859	0.815	0.682	0.840	0.783	0.657
BALLS	0.846	0.794	0.674	0.878	0.844	0.763	0.864	0.815	0.719
Agglomer	0.850	0.797	0.676	0.861	0.817	0.714	0.852	0.797	0.685
StandERE	0.856	0.811	0.714	0.872	0.828	0.741	0.867	0.825	0.744
ExtendedERE	**0.902**	**0.877**	**0.775**	**0.908**	**0.880**	**0.798**	**0.910**	**0.882**	**0.801**
WeightedERE	**0.929**	**0.902**	**0.864**	**0.930**	**0.911**	**0.872**	**0.935**	**0.919**	**0.890**

Table 6: **Results of Combining 20 base-level ER systems in terms of F_P Measure.**

Name	BestBase	MajorVot	WtVot	BestClust	BALLS	Agglomer	ExtERE	BestIndiv	WtERE
Cheyer	0.951	0.910	0.957	0.906	0.922	0.904	0.995	0.995 (Alg 0)	0.995
Cohen	0.867	0.868	0.914	0.865	0.874	0.881	0.882	0.931 (Alg 14)	0.919
Hardt	0.783	0.828	0.788	0.606	0.818	0.737	0.900	0.898 (Alg 0)	0.943
Israel	0.847	0.886	0.868	0.857	0.880	0.875	0.891	0.875 (Alg 10)	0.934
Kaelbling	0.971	0.908	0.971	0.901	0.934	0.908	0.994	0.994 (Alg 0)	0.994
Mark	0.851	0.806	0.813	0.795	0.802	0.795	0.834	0.866 (Alg 10)	0.929
McCallum	0.945	0.968	0.946	0.968	0.968	0.968	0.946	0.968 (Alg 6)	0.973
Mitchell	0.876	0.913	0.933	0.902	0.908	0.913	0.902	0.906 (Alg 9)	0.927
Mulford	0.824	0.861	0.858	0.854	0.871	0.871	0.867	0.865 (Alg 14)	0.870
Ng	0.895	0.853	0.897	0.866	0.873	0.873	0.919	0.896 (Alg 10)	0.916
Pereira	0.795	0.790	0.795	0.843	0.816	0.784	0.788	0.852 (Alg 7)	0.902
Voss	0.863	0.695	0.596	0.719	0.704	0.714	0.821	0.863 (Alg 19)	0.898
Mean	**0.872**	**0.857**	**0.862**	**0.840**	**0.864**	**0.852**	**0.895**	**0.909**	**0.935**

Therefore, we cannot include it as a base-level algorithm in our experiments.

5.2.2 *Comparison of various combination algorithms*

In Table 5 we show the results of comparing various combination algorithms using F_P, B-Cubed, and $F1$ measures. We also study the effect of different number of base-level ER systems on the ensemble results. For the test with 5 base-level systems, we choose eTF/IDF+Agglomerative Algorithms with threshold 0.1, 0.2, and Algo6, Algo9, Algo10 from Table 4. For the test with 10 base-level systems, we use all algorithms as presented in Table 4. For the test with 20 base-level systems, we add more algorithms by varying the parameter settings of base algorithms from Table 4. To be more specific, we choose 14 eTF/IDF+Agglomerative Algorithms with threshold varied between 0.03 and 0.3, and Algo6 – Algo10 from Table 4, plus one new Web ER algorithm published most recently [24]. The thresholds for eTF/IDF+Agglomerative Algorithms are chosen such that the algorithms with different thresholds perform differently on different parts of the dataset.

In Table 5 we can see that for all three measures, the `WeightedERE` algorithm produces the best performance on combining multiple ER systems. Also the results of `ExtendedERE` and `WeightedERE` show consistent improvement as the number of base system increases, whereas the results of other combining approaches sometimes fluctuate. This illustrates the robustness of the proposed approaches. The vulnerability of the other combination algorithms is caused by the way the combining is carried out. As we mentioned before, all the existing clustering ensemble algorithms (including those baseline algorithms we implemented to compare with, except `BestBase` which is not an ensemble algorithm) look for the clustering that agrees the most with the existing base clusterings. In that case, the ensemble result highly relies on the features of the set of base-level algorithms.

For example in Table 5, the result of `MajorVot` for 10 base-level systems is slightly better than that of `BestBase`, which is the best single base-level algorithm on this dataset. This is because there are not many "bad" base algorithms that can deteriorate the voting result. However, for 20 base-level systems, the `MajorVot` performs worse than the `BestBase` because many new base algorithms are introduced and many of them do not perform well on the dataset. Thus, in order to find a clustering that agrees with as many base clusterings as possible, the voting result will not be good by traditional combination algorithms. Similarly, when there are fewer base algorithms, the change to any one of them will have great impact to the overall result. That is why the ensemble results of 5 base-level systems are worse as well, as shown in Table 5. On the other hand, `ExtendedERE` and `WeightedERE` are more dynamic approaches that can adapt to the dataset being processed and are less vulnerable to the choices of base systems than `MajorVot` and cluster ensemble techniques.

The reason why the proposed framework outperforms others on overall result is that none of the base-level systems performs *consistently* better than the others on all the data. This point is illustrated in Table 6, which breaks down the dataset to consider 12 person names separately, and shows the results in terms of F_P on individual names when combining 20 base-level systems. Table 6 has `BestIndiv` column that shows which base-level algorithm performed the best on a given person name. As we can see the algorithms in this column are different.

Recall that the `BestBase` algorithm chooses one single base-level system that outperforms the others on overall dataset. We can see from Table 6 that the `BestBase` does not always perform the best on all names. This provides the opportunity for improvement and that is why the new ensemble solutions work well.

Our framework takes advantage of learning the best combination of base-level systems given certain context from the

Table 7: Comparing different algorithms with 5 and 10 base-level ER systems on RealPub dataset.

Method	5 Base Algo.		10 Base Algo.	
	$\mathbf{F_P}$	**B-Cubed**	$\mathbf{F_P}$	**B-Cubed**
BestBase	0.855	0.821	0.876	0.847
MajorVot	0.836	0.799	0.872	0.842
WtVot	0.857	0.824	0.877	0.850
BestClust	0.836	0.799	0.845	0.808
BALLS	0.835	0.797	0.849	0.814
Agglomer	0.836	0.799	0.855	0.821
StandERE	0.858	0.828	0.873	0.843
ExtendedERE	**0.869**	**0.836**	**0.880**	**0.849**
WeightedERE	**0.880**	**0.854**	**0.908**	**0.889**

training data. Although for individual names, the `ExtendedERE` and `WeightedERE` do not always generate the best result, the overall performance on the whole dataset is superior. We have used the standard 1-tailed paired t-test, with $\alpha = 0.05$, to measure the statistical significance of our results when compared to other approaches in Table 6. The results for both `ExtendedERE` and `WeightedERE` have been found to be significantly better.

5.3 Experiments on Publication Domain

We also test our framework on RealPub dataset. Since some of the base-level systems we used are designed specifically for Web application and do not apply to publication domain, we cannot use the same set of base-level systems that we use on Web domain. In [8] the authors developed an adaptive algorithm that is proved to work well on this dataset. We choose the adaptive algorithms with various parameter settings along with some baseline algorithms presented in [8] as base-level algorithms. We refer to the variations of the relationship-based algorithms as `RelER` and `RelAA` [8] when we describe the base-level algorithms.

Table 7 demonstrates the comparison results of multiple approaches when combining 5 and 10 different base-level systems. For the test with 5 base-level systems, we choose 1 context-based algorithm (`Context`) and 4 relationship-based algorithms (3 RelER with different thresholds and 1 RelAA) as presented in [8]. For the test with 10 base-level systems, we use 1 `Context`, 5 RelER and 4 RelAA with different thresholds. From Table 7 we can see that both `ExtendedERE` and `WeightedERE` algorithms get higher quality results than all the other cluster ensemble algorithms. `WeightedERE` tends to outperform `ExtendedERE` as it employs a more direct link between the context and performance of a given base system. That is, the `WeightedERE` algorithm learns the performance of each individual base-level system separately and adjust the decision of each system accordingly in the context feature space. The adjusted decisions are added up to the total weight of each edge.

We also study the effect of different combinations of base-level systems by exploring how the choice of the base system impact the ensemble results. Table 8 presents the results for two different combinations of 5 base-level systems. Combination 1 consists of 3 RelER algorithms with threshold 0.05, 0.01, 0.005, and 1 RelAA algorithm with threshold 0.1. Combination 2 consists of 3 RelER algorithms with threshold 0.0005, 0.0001, 0.00005, and 2 RelAA algorithms with threshold 0.01 and 0.001. We see marginal improvement for both combinations by `ExtendedERE` and significant improvement by `WeightedERE`. The improvement for Combination 2 is more than that of Combination 1 because the base-level

Table 8: Comparing different combination of 5 base-level ER systems on RealPub dataset.

Method	Combination 1		Combination 2	
	$\mathbf{F_P}$	**B-Cubed**	$\mathbf{F_P}$	**B-Cubed**
Base Algo. 1	0.618	0.600	0.871	0.840
Base Algo. 2	0.816	0.777	0.876	0.847
Base Algo. 3	0.851	0.817	0.875	0.845
Base Algo. 4	0.855	0.821	0.875	0.845
Base Algo. 5	0.837	0.799	0.874	0.844
StandardERE	0.858	0.828	0.873	0.843
ExtendedERE	**0.869**	**0.836**	**0.879**	**0.848**
WeightedERE	**0.880**	**0.854**	**0.911**	**0.893**

systems in Combination 2 perform more consistently. For Combination 1, the accuracy in terms of F_P vary between 0.618 and 0.855 for base-level systems. For Combination 2, the base-level systems perform mostly well and their overall performance is very close, although different algorithm may work better in different part of the data.

5.4 Discussion

Section 4.1 has discussed two different types of context features, one is at the VCS level and the other one is at the edge level. Section 4.3 has presented several different ways of combining them to be used by different algorithms. For `ExtendedERE`, the features are combined as a linear combination according to Eq. (1). A natural question is that how to decide the α's, i.e., how the two types of features will affect the ER ensemble result. We found it usually depends on the dataset being processed. For example, in WWW'05 dataset, there are fewer VCS's but for each VCS there are more edges. In this case, the learning of the estimated number of clusters may not be accurate and it will be better to rely more on the edge level context. That is, $\alpha_1 < \alpha_2$ in Eq. (1). On the other hand, for RealPub dataset, the number of VCS's is large while the number of edges per VCS is small. So better results can be obtained by giving higher weight to the VCS level context features, i.e., $\alpha_1 > \alpha_2$ in Eq. (1).[8] This problem does not exist for the `WeightedERE` algorithm defined in Section 4.3.2 since all the context features are employed in the multi-dimensional context feature space and the importance of each dimension will be learned by the prediction model automatically.

Efficiency. The running time of the ER ensemble algorithm consists of several parts: running base-level ER systems and loading the decisions by these systems on the edges, running the two regression classifiers to derive the context features, applying meta-level classification to predict the edge classes, and creating final clusters. Notice the base-level ER systems can be run in parallel. The running time of the loading and building of the meta-level data model is linear to the number of base-level algorithms and the number of edges to be processed. The running time of applying classifiers depends mostly on the classifiers chosen to create context features and train the meta-level classifier. The training part of the classifiers can be done off line and different classifiers perform very differently. For example, for a training set with 20 base-level systems and 45K edge instances, the time it takes to learn the model are: 53.66 sec for *J48*, 4.55 sec for *NaiveBayes*, 0.11 sec for *IB1*, etc. Once

[8]In our experiments, we chose α_1 as 0.2 for the former dataset, and 0.8 for the later one.

the model has been built, it is usually very fast to classify the new data. For 5K dataset, it takes less than 1 sec and for 50K dataset, it takes less than 5 sec for most classifiers.

Effects of Blocking. Blocking is important to improve the efficiency of our algorithms. For the Web domain experiments we did on WWW'05 dataset, the difference with or without blocking in terms of the model size and the running time is of one order of magnitude, and for RealPub dataset, the difference is of two orders of magnitude. It also has great impact on the quality of ER ensemble since the VCS level context features rely on the blocking.

6. CONCLUSION

This paper develops a novel ER ensemble framework for combining multiple base-level entity resolution systems. We proposed the Context-Extended and Context-Weighted Ensemble approaches for meta-level classification that enable the framework to adapt to the dataset being processed based on the local context. Through an extensive empirical study we demonstrated the advantage of the proposed solution over the other state of the art techniques. Specifically, the study demonstrates the superiority of the Context-Weighted Ensemble approach. As future work we plan to explore principally different ways to create context features and a different classification scheme that trades quality for efficiency.

7. REFERENCES

[1] J. Artiles, J. Gonzalo, and S. Sekine. The semeval-2007 weps evaluation: Establishing a benchmark for the web people search task. In *SemEval*, 2007.

[2] J. Artiles, J. Gonzalo, and F. Verdejo. A testbed for people searching strategies in the www. In *SIGIR*, 2005.

[3] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *IEEE Sympos. on Foundations of Computer Science*, 2002.

[4] R. Bekkerman and A. McCallum. Disambiguating web appearances of people in a social network. In *WWW*, 2005.

[5] I. Bhattacharya and L. Getoor. Iterative record linkage for cleaning and integration. In *DMKD*, 2004.

[6] M. Bilenko and R. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *KDD*, 2003.

[7] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting relationships for object consolidation. In *IQIS Workshop at ACM SIGMOD Conference*, June 17 2005.

[8] Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Adaptive graphical approach to entity resolution. In *JCDL*, 2007.

[9] W. W. Cohen and J. Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *SIGKDD*, 2002.

[10] A. Culotta and A. McCallum. Joint deduplication of multiple record types in relational data. In *CIKM*, 2005.

[11] H. Cunningham, D. Maynard, K. Bontcheva, and Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *ACL'02*.

[12] X. Dong, A. Halevy, and J. Madhavan. Reference reconcil -iation in complex information spaces. In *SIGMOD*, 2005.

[13] E. Elmacioglu, Y. F. Tan, S. Yan, M.-Y. Kan, and D. Lee. PSNUS: Web people name disambiguation by simple clustering with rich features. In *SemEval*, 2007.

[14] A. L. N. Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(6):835–850, 2005.

[15] S. Garner. Weka: The waikato environment for knowledge analysis. In *New Zealand Comput. Sci. Res. Conf.*, 1995.

[16] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *ICDE*, 2005.

[17] S. T. Hadjitodorov and L. I. Kuncheva. Selecting diversifying heuristics for cluster ensembles. In *Multiple Classifier Systems*, 2007.

[18] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.

[19] D. Kalashnikov, S. Mehrotra, and Z. Chen. Exploiting relationships for domain-independent data cleaning. In *SIAM Data Mining*, 2005.

[20] D. V. Kalashnikov, Z. Chen, S. Mehrotra, and R. Nuray. Web people search via connection analysis. *IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE)*, 20(11), Nov. 2008.

[21] D. V. Kalashnikov, Z. Chen, R. Nuray-Turan, S. Mehrotra, and Z. Zhang. WEST: Modern technologies for Web People Search. In *ICDE*, 2009.

[22] D. V. Kalashnikov and S. Mehrotra. Domain-independent data cleaning via analysis of entity-relationship graph. *ACM Transactions on Database Systems (ACM TODS)*, 31(2):716–767, June 2006.

[23] D. V. Kalashnikov, S. Mehrotra, Z. Chen, R. Nuray-Turan, and N. Ashish. Disambiguation algorithm for people search on the web. In *ICDE*, 2007.

[24] D. V. Kalashnikov, R. Nuray-Turan, and S. Mehrotra. Towards breaking the quality curse. A web-querying approach to Web People Search. In *SIGIR*, 2008.

[25] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, 1998.

[26] B. Long, Z. M. Zhang, and P. S. Yu. Combining multiple clusterings by soft correspondence. In *ICDM*, 2005.

[27] A. McCallum and B. Wellner. Conditional models of identity uncertainty with application to noun coreference. In *NIPS*, 2004.

[28] A. K. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD*, 2000.

[29] R. Nuray-Turan, Z. Chen, D. V. Kalashnikov, and S. Mehrotra. Exploiting Web querying for Web People Search in WePS2. In *2nd Web People Search Evaluation Workshop (WePS 2009), 18th WWW Conference*, 2009.

[30] R. Nuray-Turan, D. V. Kalashnikov, and S. Mehrotra. Self-tuning in graph-based reference disambiguation. In *DASFAA*, 2007.

[31] B.-W. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. In *ICDM*, 2006.

[32] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[33] W. Shen, P. DeRose, L. Vu, A. Doan, and R. Ramakrishnan. Source-aware entity matching: A compositional approach. In *ICDE*, 2007.

[34] G. Sigletos, G. Paliouras, C. D. Spyropoulos, and M. Hatzopoulos. Combining information extraction systems using voting and stacked generalization. *Journal of Machine Learning Research*, 6:1751–1782, 2005.

[35] P. Singla and P. Domingos. Entity resolution with markov logic. In *ICDM*, 2006.

[36] A. Strehl and J. Ghosh. Cluster ensembles: A knowledge reuse framework for combining partitionings. In *Journal of Machine Learning Research*, 2002.

[37] S. Tejada, C. A. Knoblock, and S. Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *SIGKDD*, 2002.

[38] A. Thor and E. Rahm. Moma - a mapping-based object matching system. In *CIDR*, 2007.

[39] I. H. Witten and E. Frank. Data mining: Practical machine learning tools and techniques. Morgan Kaufmann, 2005.

[40] D. Wolpert. Stacked generalization. *Neural Networks*, 1992.

[41] H. Zhao and S. Ram. Entity identification for heterogeneous database integration–a multiple classifier system approach and empirical evaluation. *Inf. Syst.*, 30(2):119–132, 2005.