Leveraging Distributed Human Computation and Consensus Partition for Entity Coreference

Saisai Gong, Wei Hu, and Yuzhong Qu

State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, PR China
`ssgong@smail.nju.edu.cn,{whu,yzqu}@nju.edu.cn`

Abstract. Entity coreference is important to Linked Data integration. User involvement is considered as a valuable source of human knowledge that helps identify coreferent entities. However, the quality of user involvement is not always satisfying, which significantly diminishes the coreference accuracy. In this paper, we propose a new approach called coCoref, which leverages distributed human computation and consensus partition for entity coreference. Consensus partition is used to aggregate all distributed user-judged coreference results and resolve their disagreements. To alleviate user involvement, ensemble learning is performed on the consensus partition to automatically identify coreferent entities that users have not judged. We integrate coCoref into an online Linked Data browsing system, so that users can participate in entity coreference with their daily Web activities. Our empirical evaluation shows that coCoref largely improves the accuracy of user-judged coreference results, and reduces user involvement by automatically identifying a large number of coreferent entities.

Keywords: #eswc2014Gong

1 Introduction

Entity coreference is to identify entities from diverse data sources that refer to the same real-world object. It is important to the reuse, integration and application of Linked Data. Many entity coreference approaches have been proposed in literature, which can be divided to two main categories: *fully-automatic* and *semi-automatic*. Although automatic methods have been continuously improved using various sophisticated algorithms, e.g., taking advantages of OWL semantics [10], computing similarities among entities [20], machine learning [11,16], they still remain far from perfect.

On the other hand, semi-automatic approaches bring user involvement into the entity coreference process and gain benefits from human knowledge. To acquire human contributions, a number of existing semi-automatic methods introduced micro-task crowdsourcing [3], some of which also dedicated to minimizing user involvement while preserving certain coreference accuracy, based on techniques like active learning [19]. In addition to use the modern crowdsourcing platform like Amazon Mechanical Turk and CrowdFlower, there are also other *distributed human computation* systems that can be used for entity coreference, e.g., [25], which hold promises for using computers and humans together to scale up the kind of tasks that only humans do well. To this end, in this paper we try to attract users to participate in entity coreference with their daily Web browsing activities. A typical scenario is that a user identifies several coreferent entities denoting the same real-world object as the current entity that she is browsing. The behind incentives for users to do so are that they like to view more data about some real-world objects across different sources in Linked Data.

For entity coreference with distributed human computation, a central problem is the quality control of users' coreference results, which draws attentions in many works [3,12]. The quality of user-judged results is not always satisfying; mistakes and outliers frequently happen due to various reasons. For example, the ambiguity of candidate entities, caused by lacking enough domain knowledge, data evolvement and so on, may lead to incorrect user judgement. Additionally, user involvement is expensive and usually slow. A user can only complete a small number of coreference tasks with limited time and energy, leading to omissions in her coreference result. Therefore, it is important to leverage all distributed user-judged results and minimize the disagreements among them.

In this paper, we propose a new approach *coCoref* to leveraging distributed human computation and consensus partition for entity coreference. coCoref improves the quality of user-judged results by aggregating users' individual results into a more robust and comprehensive consensus partition with better accuracy [22]. Furthermore, coCoref adopts the consensus partition as labelled data and proposes an ensemble learning algorithm to alleviate user involvement by automatically identifying coreferent entities that have not been judged by users. We develop coCoref as an important component in a Linked Data browsing system called SView.[1] We also believe that coCoref is applicable to various entity coreference scenarios involving users. We empirically evaluate the performance of coCoref based on real users' browsing logs from SView. We also compare coCoref with several existing systems on an OAEI test and show that coCoref automatically identifies a large amount of coreferent entities using a small portion of consensus partition.

The rest of this paper is structured as follows. Section 2 gives an overview of our approach. Section 3 introduces consensus partition. Section 4 describes ensemble learning. Our evaluation is reported in Section 5, while related work is discussed in Section 6. We conclude this paper in Section 7.

2 Overview of the Approach

We show the overview of our approach in Figure 1, where users perform coreference on a set of entities in distributed data sources. Let $\mathbf{E} = \{e_1, e_2, \ldots, e_n\}$ be the set of all entities. In this paper, an entity $e_i \in \mathbf{E}$ is denoted by a URI

[1] `http://ws.nju.edu.cn/sview/`

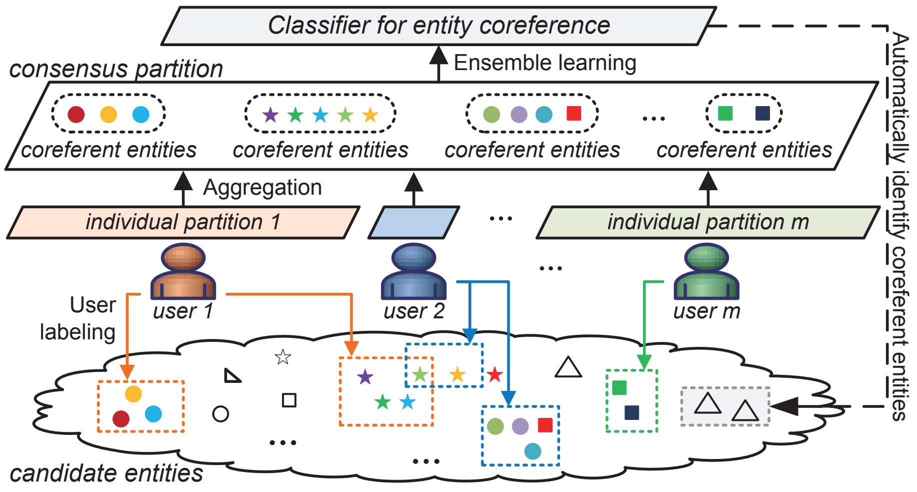

Fig. 1. Overview of coCoref

and described by a set of property-value pairs, which can be extracted by dereferencing the URI of e_i. We define the extraction of e_i's involved properties by $Prop(e_i)$, and the extraction of e_i's values w.r.t. property p_l by $Value(e_i, p_l)$.

Individual partition. Let $\mathbf{U} = \{u_1, u_2, \ldots, u_m\}$ be the set of all users participated in entity coreference on $\mathbf{E}$. For a user $u_j \in \mathbf{U}$, when she browses some entities, she may help identify some coreferent entities. However, with her limited time and energy, u_j can only view and judge a small set of entities. Let $\mathbf{X}_j \subseteq \mathbf{E}$ be the subset of entities judged by u_j. The entity coreference on $\mathbf{X}_j$ w.r.t. u_j is defined to find a partition π on $\mathbf{X}_j$, which consists of a set of pairwise disjoint nonempty subsets of $\mathbf{X}_j$. For $s_a, s_b \in \pi, a \neq b$, $s_a \cap s_b = \emptyset$, and $\bigcup_{a=1,\ldots,|\pi|} s_a = \mathbf{X}_j$. In fact, for any $s_a \in \pi$, the entities in s_a denote the same real-world object and form an *equivalence class*, where the equivalence relation holds between the entities in s_a. s_a will be updated if the user u_j links other entities to some elements in s_a (essentially, to merge two equivalence classes), or she removes some elements from s_a as she believes that they are no longer coreferent with others in s_a. Therefore, the partition π can be considered as u_j's individual partition about entity coreference on $\mathbf{X}_j$.

Consensus partition. Different users perform entity coreference on different entities and their coreference results may also have differences. For two individual partitions π_i, π_j judged by users u_i, u_j respectively, π_i, π_j can be equal, totally disjoint or have overlaps. In order to aggregate users' individual partitions and resolve their disagreements, we use consensus partition [22] to establish a more robust and comprehensive result with better overall accuracy. To formalize, let $\mathrm{T} = \{\pi_1, \pi_2, \ldots, \pi_m\}$ be a set of individual partitions. Each $\pi_j \in \mathrm{T}$ is judged by user u_j on a set of entities $\mathbf{X}_j \subseteq \mathbf{E}$. The entity coreference with distributed human computation is defined to find a partition τ on $\mathbf{X} = \bigcup_{j=1,\ldots,|\mathrm{T}|} \mathbf{X}_j$ that minimizes $\sum_{\pi_j \in \mathrm{T}} Dist(\tau, \pi_j)$, where $Dist$ is a distance function between any two partitions. We refer to the equivalence class in τ as *consensus equivalence class*.

Ensemble learning. Although many users can participate in entity coreference, there are still a large number of entities that have not been judged by any users, that is, $\mathbf{X} \subseteq \mathbf{E}$. In order to alleviate user involvement, we make use of the consensus partition as training data and build classifiers based on ensemble learning. Each base learner is trained on a random sample of consensus equivalence classes in the consensus partition; different base learners are combined to generalize a global classifier. The classifier is applied to automatically identify coreferent entities that have not been judged by enough users. For a new entity, coCoref uses the classifier to classify other coreferent entities and forms a new equivalence class. The classifier will be updated offline when more users' coreference results are collected and aggregated. Currently, coCoref does not modify users' individual partitions. The reconciliation of users' coreference results and the result from ensemble learning will be our future work.

Example 1. To help understanding, we show a running example here. Assuming that Alice browses some entity `NewYorkCity` and helps identify its coreferent entities `NY` and `TheBigApple`. She also browses another entity `Manhattan` and finds its coreferent entity `NewYorkCounty`. Therefore, Alice's individual partition is { {`NewYorkCity`, `NY`, `TheBigApple`}, {`Manhattan`, `NewYorkCounty`} }.

Similarly, Tom participates in entity coreference and delivers his individual partition { {`NewYorkCity`, `TheBigApple`, `NewYorkCounty`}, {`NY`, `Manhattan`} }. Mike forms his partition { {`NewYorkCity`, `TheBigApple`}, {`NY`, `Manhattan`} }.

coCoref aggregates the three individual partitions to build a consensus partition { {`NewYorkCity`, `TheBigApple`}, {`NY`, `Manhattan`}, {`NewYorkCounty`} }. Furthermore, coCoref trains an ensemble of classifiers on this consensus partition and uses the classifier to identify entity coreference, e.g., a new entity `Nanjing`.

3 Consensus Partition

As described in Section 2, the problem of improving the quality of user-judged coreference results is transformed to the problem of achieving a consensus partition in terms of users' individual partitions. In this section, we firstly provide a formal definition of the consensus partition problem, and then introduce an approximation algorithm for obtaining a sub-optimal solution to the problem.

3.1 Formalization

The goal of computing consensus partition is to aggregate individual partitions such that the disagreements among them are minimized. Disagreements can be caused by mistakes, omissions and so on. There are two reasons that consensus partition can improve the quality of individual partitions. Firstly, by minimizing disagreements in the consensus partition, mistakes and outliers made by a few users can be filtered out because there is no agreement on how they should be aggregated [7]. In other words, consensus partition is more robust to mistakes and outliers. Secondly, the consensus partition is naturally more comprehensive than any individuals since it comprises more coreferent entities from individual partitions to avoid omissions by individual users.

Next, we provide a formalization of consensus partition. The disagreements between two partitions can be measured with a distance function, and various distance functions have been proposed [22]. In this paper, we use the *symmetric difference distance* [9] for our purpose. Specifically, let T be a set of individual partitions, each individual partition π_j is generated from user u_j on a subset of entities $\mathbf{X}_j \subseteq \mathbf{E}$. Computing a consensus partition in our approach is to find a partition τ on $\mathbf{X} = \bigcup_{j=1,\ldots,|\mathrm{T}|} \mathbf{X}_j$ that minimizes the following distance:

$$\begin{aligned} Dist_{\mathrm{T}}(\tau) &= \sum_{\pi_j \in \mathrm{T}} Dist(\tau, \pi_j) \\ &= \sum_{\pi_j \in \mathrm{T}} \sum_{v<w} (\delta_\tau(e_v, e_w)\psi_{\pi_j}(e_v, e_w) + (1 - \delta_\tau(e_v, e_w))\delta_{\pi_j}(e_v, e_w)), \end{aligned} \quad (1)$$

where, for two entities e_v, e_w and a partition π,

$$\delta_\pi(e_v, e_w) = \begin{cases} 1, & \text{if } \exists s_l \in \pi, e_v \in s_l, e_w \in s_l \\ 0, & \text{otherwise} \end{cases}, \quad (2)$$

$$\psi_\pi(e_v, e_w) = \begin{cases} 1, & \text{if } \exists s_a, s_b \in \pi, a \neq b, e_v \in s_a, e_w \in s_b \\ 0, & \text{otherwise} \end{cases}. \quad (3)$$

Let N_{vw} be the number of partitions of which e_v, e_w are in different equivalence classes, i.e., $N_{vw} = |\{\pi \in \mathrm{T} \mid \exists s_a, s_b \in \pi, a \neq b, e_v \in s_a, e_w \in s_b\}|$, and M_{vw} be the number of partitions of which e_v, e_w are in the same equivalence class, i.e., $M_{vw} = |\{\pi \in \mathrm{T} \mid \exists s_a \in \pi, e_v \in s_a, e_w \in s_a\}|$. We rewrite Eq. (1) as:

$$\begin{aligned} Dist_{\mathrm{T}}(\tau) &= \sum_{v<w} (\delta_\tau(e_v, e_w) \sum_{\pi_j \in \mathrm{T}} \psi_{\pi_j}(e_v, e_w) + (1 - \delta_\tau(e_v, e_w)) \sum_{\pi_j \in \mathrm{T}} \delta_{\pi_j}(e_v, e_w)) \\ &= \sum_{v<w} (\delta_\tau(e_v, e_w) N_{vw} + (1 - \delta_\tau(e_v, e_w)) M_{vw}) \\ &= \sum_{v<w} M_{vw} - \sum_{v<w} \delta_\tau(e_v, e_w)(M_{vw} - N_{vw}) . \end{aligned} \quad (4)$$

Note that $\sum_{v<w} M_{vw}$ is independent to τ, so minimizing $Dist_{\mathrm{T}}(\tau)$ is equivalent to maximize $\sum_{v<w} \delta_\tau(e_v, e_w)(M_{vw} - N_{vw})$. Let $Q_{vw} = M_{vw} + N_{vw}$, where $0 \leq Q_{vw} \leq |\mathrm{T}|$. We have $\sum_{v<w} \delta_\tau(e_v, e_w)(M_{vw} - N_{vw}) = 2\sum_{v<w} \delta_\tau(e_v, e_w)\phi_{vw}$, where $\phi_{vw} = M_{vw} - \frac{Q_{vw}}{2}$.

3.2 An Approximation Algorithm for Consensus Partition

Computing consensus partition to maximize $\sum_{v<w} \delta_\tau(e_v, e_w)(M_{vw} - N_{vw})$ has been proven to be a NP-complete problem [9]. Due to the hardness of the prob-

lem, it is intractable to exactly solve it on a large scale, e.g., entity coreference in Linked Data. Various approximations or heuristics with/without performance guarantees have been proposed to give a sub-optimal solution to the problem. In our approach, we use an approximation algorithm called CC-Pivot [1], because CC-Pivot is usually more efficient and applicable for large-scale data than others [9]. The details of CC-Pivot applied in our method is shown in Algorithm 1, where ϕ'_{vw} in Line 4 is defined as follows:

$$\phi'_{vw} = \begin{cases} \phi_{vw}, & Q_{vw} \geq \theta \\ -\frac{Q_{vw}}{2}, & \text{otherwise} \end{cases}, \tag{5}$$

where Q_{vw} and ϕ_{vw} are defined in Section 3.1. θ is a threshold that is used to see whether two given entities are judged by enough users.

Algorithm 1: CC-Pivot [1]

```
  Input: entity set $\mathbf{X}$, individual partition set T
  Output: consensus partition $\tau$ on $\mathbf{X}$
1 Choose a pivot entity $e_v \in \mathbf{X}$ uniformly at random;
2 Let $C \leftarrow \{e_v\}$, $\mathbf{X}' \leftarrow \emptyset$;
3 foreach $e_w \in \mathbf{X}, w \neq v$ do
4     if $\phi'_{vw} > 0$ then
5         $C \leftarrow C \cup \{e_w\}$;
6     else
7         $\mathbf{X}' \leftarrow \mathbf{X}' \cup \{e_w\}$;
8 return $\tau \leftarrow \{C\} \cup$ CC-Pivot($\mathbf{X}'$, T);
```

Algorithm 1 repeatedly chooses a pivot entity e_v uniformly at random from the unpartitioned entity set. Then, the algorithm generates an equivalence class containing e_v and every entity e_w holding $\phi'_{vw} > 0$. The recursion continues on the rest entities until all entities are checked. Algorithm 1 is a 3-approximation algorithm with time complexity $\mathcal{O}(|\tau| \cdot |\mathbf{X}| \cdot |\mathrm{T}|)$ [1], where τ denotes the final consensus partition. By using this approximation algorithm, more robust and comprehensive coreference results can be achieved efficiently.

Example 2. We show the running process of Algorithm 1 on Example 1. Assume $\theta = 2$. Initially, `NewYorkCity` is chosen as the pivot entity. The algorithm finds $\phi'_{\texttt{NewYorkCity,NY}} = -0.5$, $\phi'_{\texttt{NewYorkCity,Manhattan}} = -1.5$, $\phi'_{\texttt{NewYorkCity,NewYorkCounty}} = 0$ and $\phi'_{\texttt{NewYorkCity,TheBigApple}} = 1.5$. Only $\phi'_{\texttt{NewYorkCity,TheBigApple}} > 0$, so `NewYorkCity` and `TheBigApple` are put together and form an equivalence class { `NewYorkCity`, `TheBigApple` }. Algorithm 1 continues for the remaining three entities. It selects `NY` as the pivot entity and finds $\phi'_{\texttt{NY,Manhattan}} = 0.5$ and $\phi'_{\texttt{NY,NewYorkCounty}} = -1$, so it puts `NY` and `Manhattan` together and forms a new equivalence class { `NY`, `Manhattan` }. Now, only `NewYorkCounty` is left, which forms the third equivalence class { `NewYorkCounty` }. The final consensus partition is { {`NewYorkCity`, `TheBigApple`}, {`NY`, `Manhattan`}, {`NewYorkCounty`} }.

4 Ensemble Learning

User involvement is expensive and slow. Consequently, there are still a lot of coreferent candidates that have not been judged by users. In this section, we will build classifiers based on the consensus partition using ensemble learning and use them to automatically identify coreferent entities.

4.1 Training Data

A classifier decides whether a candidate entity is coreferent with the given one. So the training examples used in our method are entity pairs. We leverage the consensus partition to generate training examples. Specifically, all entities in the same consensus equivalence class pairwise compose positive examples, while entities across different consensus equivalence classes form negative examples. We keep the sizes of positive and negative examples at the same order of magnitude.

We follow the assumption that coreferent entities often have similar descriptions [20]. Thus, we use the similarities between property-values in entity pairs as learning features. Let $\mathbf{P}$ be all properties associated with the entities in the training set. For two properties $p_i, p_j \in \mathbf{P}$ w.r.t. a pair of entities (e_v, e_w) in the training set, $VSIM_{p_i,p_j}(e_v, e_w)$ is defined as follows:

$$VSIM_{p_i,p_j}(e_v, e_w) = \max_{(o,o') \in VP_{p_i,p_j}(e_v,e_w)} sim(o, o'), \tag{6}$$

$$\begin{aligned} VP_{p_i,p_j}(e_v, e_w) = &\{(o, o') \mid o \in Value(e_v, p_i), o' \in Value(e_w, p_j)\} \\ &\bigcup \{(o, o') \mid o \in Value(e_v, p_j), o' \in Value(e_w, p_i)\}, \end{aligned} \tag{7}$$

where $0 \leq VSIM_{p_i,p_j}(\cdot, \cdot) \leq 1$ and $sim(\cdot, \cdot)$ computes the value similarities:

– If both values are entities (URIs), the similarity equals 1 if their URIs are identical or they are in the same consensus equivalence class; otherwise 0.
– If both values are numerics like `xsd:double` or `xsd:integer`, their similarity equals 1 if their difference is less than a threshold (0.1); otherwise 0.
– If both values are boolean, their similarity equals 1 iff they are equal.
– For other cases, we normalize and split the value strings, and compute their Jaccard similarity.

For a pair of entities (e_v, e_w) in the training set, its feature vector of d-dimension is denoted by $\mathbf{F} = [f_1, f_2, \ldots, f_d]'$, where $f_l = VSIM_{p_i,p_j}(e_v, e_w)$, $1 \leq l \leq d$, $p_i, p_j \in \mathbf{P}$ and $d \leq \frac{|\mathbf{P}|(|\mathbf{P}|-1)}{2}$. In our approach, we use the subset of property pairs $\{(p_i, p_j) \mid \exists (e_v, e_w) \in \mathbf{X} \times \mathbf{X}, VSIM_{p_i,p_j}(e_v, e_w) > 0\}$ to construct the feature vector. More sophisticated strategies may be developed to select a better subset of property pairs for learning, but it is out of scope of this paper.

4.2 Ensemble Learning Model

Ensemble learning is a popular learning paradigm, which employs multiple base learners and combines their predictions. The predictive performance of an ensemble is usually much better than that of base learners. As aforementioned, the size of training set is relatively small because users can only accomplish a small number of coreference tasks. The reason for using ensemble learning in our approach is that, even the training examples are insufficient or the features used for training are not strong enough, ensemble learning may still find a good classifier [4], which is very suitable to our application scenario. Various methods for constructing ensembles have been developed using different base learning algorithms, manipulating input features and so on. We choose manipulating training examples to train base learners as this kind of methods is more suitable when the number of training examples is relatively small [4]. We choose decision tree as our base learner. Bagging and Boosting are two common ways to manipulate training examples and Bagging may be more robust to noises in the training examples than Boosting [17]. Because there are still a small amount of mistakes or outliers in the consensus partition, we choose Bagging and build base learners on random samples of training data.

Certain property pairs with their values are important for identifying coreferent entities. Unfortunately, when the training set is not representative, some potentially important property pairs cannot be characterized by the training data. As a result, they may not be chosen to split nodes in the construction of decision trees. To address this problem, we combine our base learners using Random Forests [2]. Random Forests is an ensemble classifier that combines a collection of decision trees. When splitting each node of a decision tree, Random Forests firstly randomly selects a subset of variables (property pairs in our context) and then leverages the most important variable in the subset based on information gain to split the node. In this way, the potentially important pairs that are not characterized by the training data can also be used to identify coreferent entities. Furthermore, Random Forests can handle training examples with thousands or even tens of thousands of features. This is also very suitable to our approach as $\mathbf{P}$ can be very large. For implementation, we use Weka 3 to realize our algorithm and adopt the default value setting for the parameter of the number of features when splitting a tree node. We set the parameter of the number of trees to 30 in our experiments.

Example 3. In Example 1, two positive examples from the consensus partition are `(NewYorkCity, TheBigApple)` and `(NY, Manhattan)`. Other entity pairs are negative examples such as `(NewYorkCity, Manhattan)` and `(NewYorkCity, NewYorkCounty)`. Using Random Forests, an ensemble of three decision trees is learnt based on the training data. A decision tree uses different property pairs to find coreferent entities , e.g., `(homepage, homepage)` and `(geometry,geometry)`. By using the classifier, we will identify `Nanking` coreferent with `Nanjing`.

5 Evaluation

We integrated coCoref in an online Linked Data browsing system called SView. When a user browses an entity using SView, she can check candidate coreferent entities provided by SView if she wants to browse more relevant data. The candidates are founded by SView using `owl:sameAs` links and web services like sameas.org[2]. The user just needs to accept or reject some candidates. All the accepted ones are then added into the equivalence class of the current entity being browsed in the user's individual partition, and their data will be integrated with that of the current entity immediately for browsing. In addition, the user can also remove some previous accepted entities from the equivalence class listed by SView if she believes they are no longer coreferent with others.

In this section, we will firstly present the evaluation on real users' browsing logs from SView. Then, we will report the experimental results on the OAEI New-York Times (NYT) test. All the experiments were carried out on an 2.5GHz Intel Core i5 CPU, Windows 7 and 1GB Java virtual memory.

5.1 Test on SView Dataset

Dataset. The target of this experiment is to evaluate how coCoref improves the quality of user-judged results using consensus partition. We collected 36 registered users' individual partitions in SView from Oct. 2013 to Dec. 2013 and used them in this test. This dataset contains 1,489 entities from 76 distributed data sources in terms of their URI namespaces. In average, a user viewed 41 entities. An entity was viewed by 2.6 users in average. To identify which entities are truly coreferent, we invited three master students in our group with good experience on entity coreference to manually build a reference partition for the 1,489 entities. For this reference partition, an entity is coreferent with 1.6 other entities in average (the maximal size is 51), and 40.3% (704 in 1,489) entities are coreferent with at least one other. This means that the entities in the SView dataset have diverse numbers of coreferent entities.

Experiment setup. Using the reference partition as golden standard, we evaluated the consensus partition and individual partitions generated from the 36 users in terms of the following five measures: Precision, Recall, F-measure, Rand Index and Normalized Mutual Information (NMI). These measures are well-known criteria showing how well a partition matches the golden standard. For a partition π, $S(\pi)$ counts the total number of entity pairs in the same equivalence class:

$$S(\pi) = \{(e_v, e_w) \mid \exists s_a \in \pi, e_v \in s_a, e_w \in s_a, v < w\}. \quad (8)$$

[2] `http://sameas.org/`

Let π_{ref} denote the golden standard partition. The Precision, Recall and F-measure for a partition π w.r.t. π_{ref} are calculated as follows:

$$\text{Precision} = \frac{S(\pi) \bigcap S(\pi_{ref})}{S(\pi)}, \quad \text{Recall} = \frac{S(\pi) \bigcap S(\pi_{ref})}{S(\pi_{ref})},$$
$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{9}$$

Rand Index penalizes both false positive and false negative decisions in partition, while NMI can be information-theoretically interpreted. Their values are both rational numbers in range $[0, 1]$; a higher value indicates a better partition. Their detailed calculation methods can be found in [23].

In order to determine the threshold θ used for computing consensus partition (see Eq. (5) in Section 3.2), we evaluated different values for θ and computed a consensus partition using Algorithm 1 on each value. The Precision and Recall of the resulting partitions w.r.t. various θ are shown in Fig. 2. According to the figure, we set $\theta = 3$ for our experiment since it led to very high precision while keeping good recall (we prefer the precision larger than 0.8). This setting distinguishes whether each entity pair is judged by at least three users. We ran Algorithm 1 ten times and computed the average values of the five measures on the consensus partition, respectively. The average running time was 1.02 seconds.

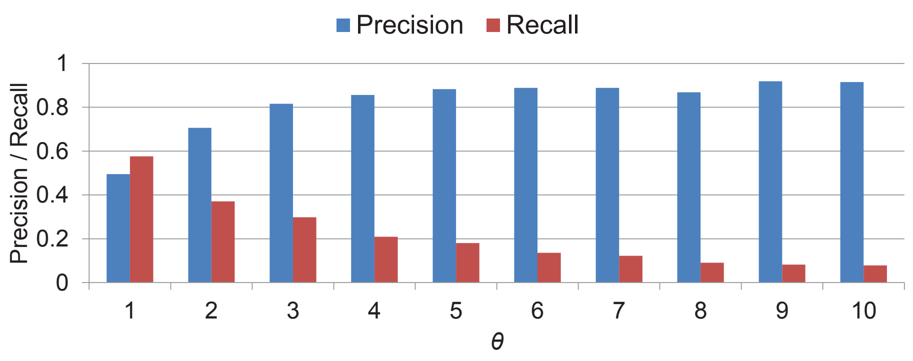

Fig. 2. Precision and Recall versus different thresholds

We used the average values of the five measures on the 36 users' individual partitions as baseline. We compared coCoref with Best-of-K [9], which is a 2-approximation algorithm to compute consensus partition. The idea of Best-of-K is to select the best individual partition as consensus partition. To this end, we selected the individual partitions with highest value of F-measure, Rand Index and NMI respectively and compared the three results of Best-of-K with that of coCoref. The comparison results are listed in Table 1.

Table 1. Performance comparison on the SView dataset

	Precision	Recall	F-measure	Rand Index	NMI
Baseline	0.665	0.071	0.120	0.012	0.105
Best-of-K with highest F-measure	0.773	0.201	0.319	0.021	0.208
Best-of-K with highest Rand Index	**0.863**	0.186	0.306	0.090	0.439
Best-of-K with highest NMI	0.708	0.107	0.186	0.088	0.446
coCoref	0.807	**0.297**	**0.434**	**0.997**	**0.951**

Result analysis. From Table 1, we can observe that the consensus partition computed by coCoref has very high score of Rand Index and NMI as compared with others. Thus, coCoref's consensus partition matched the golden standard better than others. coCoref also has the highest Recall and F-measure. All of these means the coreference results from coCoref's consensus partition are more comprehensive since it aggregated more coreferent entities. From the value of Precision on baseline, we can see the average accuracy of the 36 users' coreference results is low. There were not a few mistakes and outliers in many users' coreference results. Compared with the Precision of baseline, the Precision of coCoref's consensus partition improved very largely by 21%. Furthermore, though many users' individual coreference results are noisy, coCoref's consensus partition still achieves close precision 0.807 to the best user's result (0.863). This indicates that coCoref is robust and can largely improve the accuracy of coreference results.

5.2 Test on OAEI NYT

Dataset. The target of this experiment is to evaluate the effectiveness of consensus partition and ensemble learning algorithm on a large scale. We leveraged the results of several tools participated in the OAEI NYT test to conduct this experiment. The NYT test is to rebuild the linkages between the New-York Times dataset and three external large-scale datasets: DBpedia, Freebase and Geonames on the domains of locations, organizations and people. The tools that we used were ObjectCoref [11], Zhishi.links [16] and Knofuss [15]. Zhishi.links and Knofuss offered the download links of their coreference results in their papers. We used the three tools' results to simulate the coreference results of three users, which contain 33,914 entities in all. We set $\theta = 2$ in this experiment because there are only three systems. Besides, the NYT test offers the golden standard that can be used to evaluate the performance of coreference algorithms. The golden standard is provided in random segments for cross-validation of learning systems that use training data.

Experiment setup. We firstly built a consensus partition for the total 33,914 entities based on the results of the three tools. The original coreference results of each tool are entity pairs. To form a partition, we assumed that transitivity holds on each dataset, i.e., if a tool identifies two coreferent entity pairs (e_i, e_j)

and (e_j, e_k), then (e_i, e_k) is assumed to be also coreferent. Based on this assumption, we clustered coreferent entities for each tool's dataset and constructed its partition. With the partitions for the three tools' results, we applied coCoref to build consensus partition. The running time for computing consensus partition was 211.5 seconds. The F-measure of the consensus partition compared with the coreference results of ObjectCoref, Zhishi.links and Knofuss that we collected are listed in Table 2. As shown in the table, we can find that the consensus partition of coCoref generally performed better than ObjectCoref, Zhishi.links and Knofuss in terms of F-measure, which is in accordance with the results in the previous experiment.

Table 2. F-measure comparison on the OAEI NYT test

	Consensus partition	ObjectCoref	Zhishi.links	Knofuss
NYT–DBpedia loc.	**0.948**	0.859	0.910	0.891
NYT–DBpedia org.	**0.939**	0.882	0.900	0.916
NYT–DBpedia peop.	**0.985**	0.958	0.970	0.960
NYT–Freebase loc.	**0.951**	0.938	0.882	0.913
NYT–Freebase org.	**0.959**	0.901	0.870	0.889
NYT–Freebase peop.	**0.988**	0.973	0.926	0.942
NYT–Geonames loc.	0.937	**0.938**	0.910	0.878

We can obtain positive training examples from the consensus partition, which are entity pairs within the same equivalence classes. We divided these positive examples into 10 folds according to the division of the NYT golden standard. The training examples out of the golden standard, which were false positive coreference results of the consensus partition, were randomly assigned to the 10 folds as positive examples. Therefore, the number of positive examples is different from that of the golden standard. For each fold, we randomly generated a set of negative training examples holding the similar size of positives. Table 3 shows the statistical data of the training set. We adopted 10-fold cross validation, each time we learnt on each fold of training data, and validated the classifier's Precision and Recall on the combination of the remaining 9 folds. Then, we averaged the Precision and Recall, and computed the average F-Measure. The learning results compared with those of the consensus partition are shown in Fig. 3.

Result analysis. From Fig. 3 and Table 2, we observe that our ensemble learning algorithm achieved higher F-measure than ObjectCoref, Zhishi.links and Knofuss, and is comparable to (sometimes better than) the consensus partition. This indicates that, by using only a small subset (10%) of training data from the consensus partition, our ensemble learning algorithm successfully found a similar size of correct coreferent entities as consensus partition. Many coreferent entities can be automatically identified by ensemble learning, therefore user involvement

Table 3. Statistics of training data

	Locations	Organizations	People
Positive examples in NYT–DBpedia	1,788	1,851	4,902
Negative examples in NYT–DBpedia	2,752	2,730	1,803
Positive examples in NYT–Freebase	1,773	2,812	4,868
Negative examples in NYT–Freebase	1,986	3,937	9,864
Positive examples in NYT–Geonames	1,692		
Negative examples in NYT–Geonames	3,688		

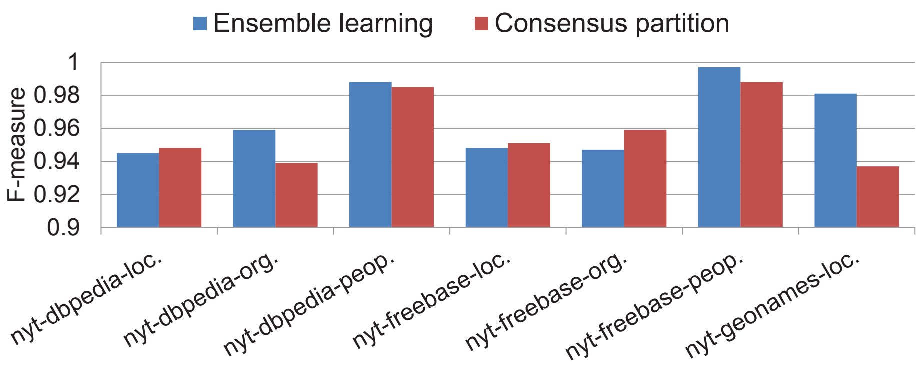

Fig. 3. Performance of ensemble learning

can be significantly reduced in real-world scenarios. Also, we can observe that the original F-measure of the consensus partition is already very good.

6 Related Work

In the Semantic Web area, traditional works address entity coreference mainly from two directions: fully-automatic and semi-automatic. One kind of automatic methods is by equivalence reasoning in terms of standard OWL semantics, e.g., `owl:sameAs` [8] and inverse functional properties [10]; the other is by similarity computation, with the assumption that instances denote the same object if they share similar property-values [14,20]. Recent works also used machine learning techniques to learn complex similarity combination rules [11,16]. We refer the reader to the report of the OAEI instance matching track for more details [6].

While the automatic methods can suggest coreferent entities, for many applications they must still be manually verified by humans. Entity coreference is recognized as the AI-complete problem, which is hard to be solved by computers but easy for humans [25]. To leverage user involvement, the works in [3,18] used crowdsourcing platforms for entity coreference, which differ from our deployment. Sig.ma [21] developed a Web-based user interface to view Linked Data, which allowed users to give judgement by filtering property-values and data sources. iamResearcher [25] addressed the scientific publication author identity coreference problem for integrating distributed bibliographic datasets. Currently they do not present any mechanism to learn from user-judged results.

To resolve disagreements among user-judged results, dedicated algorithms were proposed to estimate the quality of users, allowing for the rejection and blocking of the unreliable users [3,12]. After that, different user-judged results can be aggregated using machine learning or other customizable methods [5]. Different from them, our approach uses consensus partition to minimize the disagreements among all users, because even reliable users can make mistakes. Furthermore, various theoretical results on consensus partition are provided including the performance guarantee [9].

Some methods also focused on how to make the best use of human contributions, e.g., by machine learning to minimize user involvement while preserving certain coreference accuracy [13,24], which implicitly assumed that a user can certainly give truth about coreference result, and there is only one single right answer for each pair of coreferent entities, but both of the two assumptions do not conform to the real world. In this paper, we ground our learning algorithm on the consensus partition, which can accommodate inconsistencies and errors. Moreover, we integrate the entity coreference tasks in users' browsing activities, which gives users incentives for active participation.

7 Conclusions and Future Work

Distributed human computation for entity coreference is important to improve its accuracy, however, user involvement is often expensive, slow and error-prone.

In this paper, we proposed coCoref to leverage distributed human computation and consensus partition for entity coreference. The main contributions of this paper are as follows:

- All distributed users' individual partitions are collected and aggregated to build a consensus partition, which increases the scale of coreferent entities and resolves the disagreements simultaneously.
- Ensemble learning is performed on the consensus partition to alleviate user involvement, which automatically identifies a large number of coreferent entities that users have not judged.
- We integrated coCoref with the Web browsing activities, which is different from many approaches that use crowdsourcing platforms. We also conducted experiments to demonstrate the good accuracy of consensus partition and the considerable reduction of user involvement.

In future work, we look forward to studying other sophisticated methods to consensus partition and ensemble learning with the consideration of user preference.

Acknowledgements This work is supported in part by the National Natural Science Foundation of China under Grant Nos. 61370019 and 61170068, and in part by the Natural Science Foundation of Jiangsu Province under Grant Nos. BK2011189 and BK2012723.

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. *Journal of the ACM*, 55(5):23, 2008
2. Breiman, L.: Random forests. *Machine learning*, 45(1):5–32, 2001
3. Demartini, G., Difallah, D., Cudré-Mauroux, P.: ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: *WWW*, pp. 469–478, 2012
4. Dietterich, T.G.: Ensemble methods in machine learning. In: *Multiple Classifier Systems*, pp. 1–15, 2000
5. Do, H.H., Rahm, E.: COMA: A system for flexible combination of schema matching approaches. In: *VLDB*, pp. 610–621, 2002
6. Ferrara, A., Nikolov, A., Noessner, J., Scharffe, F.: Evaluation of instance matching tools: The experience of OAEI. *Journal of Web Semantics*, 21:49–60, 2013
7. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1(1):4, 2007
8. Glaser, H., Jaffri, A., Millard, I.: Managing co-reference on the semantic web. In: *WWW Workshop on LDOW*, 2009
9. Goder, A., Filkov, V.: Consensus clustering algorithms: Comparison and refinement. In: *ALENEX*, pp. 109–117, 2008
10. Hogan, A., Zimmermann, A., Umbrich, J., Polleres, A., Decker, S.: Scalable and distributed methods for entity matching, consolidation and disambiguation over linked data corpora. *Journal of Web Semantics*, 10:76–110, 2012

11. Hu, W., Chen, J., Qu, Y.: A self-training approach for resolving object coreference on the semantic web. In: *WWW*, pp. 87–96, 2011
12. Ipeirotis, P., Provost, F., Wang, J.: Quality management on Amazon Mechanical Turk. In: *ACM SIGKDD workshop on Human Computation*, pp. 64–67, 2010
13. Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. *Journal of Web Semantics*, 23:2–15, 2013
14. Li, J., Tang, J., Li, Y., Luo, Q.: RiMOM: A dynamic multi strategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1218–1232, 2009
15. Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: *ESWC*, pp. 119–133, 2012
16. Niu, X., Rong, S., Wang, H., Yu, Y.: An effective rule miner for instance matching in a web of data. In: *CIKM*, pp. 1085–1094, 2012
17. Rokach, L.: Pattern classification using ensemble methods. *World Scientific*, 2010
18. Sarasua, C., Simperl, E., Noy, N.: CrowdMap: Crowdsourcing ontology alignment with microtasks. In: *ISWC*, pp. 525–541, 2012
19. Settles, B.: Active learning literature survey. *University of Wisconsin–Madison*, 2010
20. Song, D., Heflin, J.: Automatically generating data linkages using a domain-independent candidate selection approach. In: *ISWC*, pp. 649–664, 2011
21. Tummarello, G., Cyganiak, R., Catasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.ma: Live views on the web of data. *Journal of Web Semantics*, 8(4):355–364, 2010
22. Vega-Pons, S., Ruiz-Shulcloper, J.: A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337–372, 2011
23. Wagner, S., Wagner, D.: Comparing clusterings: an overview. Universität Karlsruhe, Fakultät für Informatik, 2007
24. Wang, J., Kraska, T., Franklin, M., Feng, J.: CrowdER: Crowdsourcing entity resolution. In: *VLDB*, pp. 1483–1494, 2012
25. Yang, Y., Singh, P., Yao, J., Au Yeung, C.-M., Zareian, A., Wang, X., Cai, Z., Salvadores, M., Gibbins, N., Hall, W., Shadbolt, N.: Distributed human computation framework for linked data co-reference resolution. In: *ESWC*, pp. 32–46, 2011