Hindawi
Computational Intelligence and Neuroscience
Volume 2017, Article ID 4092135, 17 pages
https://doi.org/10.1155/2017/4092135

Research Article

Refining Automatically Extracted Knowledge Bases Using Crowdsourcing

Chunhua Li,[1] Pengpeng Zhao,[1] Victor S. Sheng,[2] Xuefeng Xian,[3] Jian Wu,[1] and Zhiming Cui[1]

[1]*School of Computer Science and Technology, Soochow University, Suzhou 215006, China*
[2]*Computer Science Department, University of Central Arkansas, Conway, AR, USA*
[3]*College of Computer Engineering, Suzhou Vocational University, Suzhou 215104, China*

Correspondence should be addressed to Pengpeng Zhao; ppzhao@suda.edu.cn

Received 23 December 2016; Revised 18 March 2017; Accepted 12 April 2017; Published 14 May 2017

Academic Editor: J. Alfredo Hernández-Pérez

Machine-constructed knowledge bases often contain noisy and inaccurate facts. There exists significant work in developing automated algorithms for knowledge base refinement. Automated approaches improve the quality of knowledge bases but are far from perfect. In this paper, we leverage crowdsourcing to improve the quality of automatically extracted knowledge bases. As human labelling is costly, an important research challenge is how we can use limited human resources to maximize the quality improvement for a knowledge base. To address this problem, we first introduce a concept of semantic constraints that can be used to detect potential errors and do inference among candidate facts. Then, based on semantic constraints, we propose rank-based and graph-based algorithms for crowdsourced knowledge refining, which judiciously select the most beneficial candidate facts to conduct crowdsourcing and prune unnecessary questions. Our experiments show that our method improves the quality of knowledge bases significantly and outperforms state-of-the-art automatic methods under a reasonable crowdsourcing cost.

1. Introduction

There are numerous information extraction projects that use a variety of techniques to extract knowledge from large text corpora and World Wide Web [1]. Example projects include YAGO [2], DBPedia [3], NELL [4], open information extraction [5], and knowledge vault [6]. These projects provide automatically constructed knowledge bases (KBs) with massive collections of entities and facts, where each entity or fact has a confidence score. However, machine-constructed knowledge bases contain noisy and unreliable facts due to the variable quality of information and the limited accuracy of extractors. Transforming these candidate facts into useful knowledge is a formidable challenge [7].

To alleviate the amount of noise in automatically extracted facts, these projects often employ ad hoc heuristics to reason about uncertainty and contradictoriness due to the large scale of the facts. There exists significant work in developing effective algorithms to perform joint probabilistic inference over candidate facts [7, 8]. Automated approaches have been improved in terms of quality but remain far from perfect. Therefore, effective methods to obtain high quality knowledge are desired. It is easy for human experts to determine whether a fact is correct or not. However, it is impossible to hire experts to correct all of them. Recently, due to the availability of Internet platforms like Amazon Mechanical Turk (MTurk), which enables the participation of human workers in a large scale, crowdsourcing has been proven to be a viable and cost-effective alternative solution. Crowdsourcing is normally used to create labelled datasets to apply machine learning algorithms and becomes an effective way to handle computer-hard tasks [9–11], such as sentiment analysis [12], image classification [13], and entity resolution [14]. The limitations of machine-based approaches and the availability of easily accessible crowdsourcing platforms inspire us to exploit crowdsourcing to improve the quality of automatically extracted knowledge bases.

In this paper, we study the problem of refining knowledge bases using crowdsourcing. Specifically, given a collection

of noisy extractions (entities and their relationships) and a budget, we can obtain a set of high quality facts from these extractions via crowdsourcing. In particular, there are two subproblems to address in this study: (1) error Detection: how can we effectively detect potential erroneous candidate facts which need to be verified by the crowd? Information extraction systems are able to extract massive collections of interrelated facts. Some facts are correct, while others are clearly incorrect and contradictory. Asking humans to verify all candidate facts is generally not feasible due to the large size of extractions. Hence, one of key challenges is to determine which subset of knowledge should be presented to the crowd for verification. (2) Knowledge Inference: how can we accurately infer consistent knowledge based on crowd feedbacks? Errors introduced from the extraction process cause inconsistencies in the knowledge base, which may contain duplicate entities and violate key ontological constraints such as subsumption, mutual exclusion, inverse, and domain and range constraints.

To address these problems, we first introduce a concept of semantic constraints, which is similar to integrity constraints in data cleaning. Then we propose rank-based and graph-based algorithms to judiciously select candidate facts to conduct crowdsourcing based on semantic constraints. Our method automatically assigns the most "beneficial" task to the crowd and infers the answers of some candidate facts based on crowd feedbacks. Experiments on NELL's knowledge base show that our method can significantly improve the quality of knowledge and outperform state-of-the-art automatic methods under a reasonable crowdsourcing cost.

To summarize, we make the following contributions:

(1) We propose a rank-based crowdsourced knowledge refining framework. We introduce a concept of semantic constraints and utilize it to detect potential contradictive facts. We present a score function taking both uncertainty and contradictoriness into consideration to select the most beneficial candidate facts for crowdsourcing.

(2) We construct a graph based on the semantic constraints and utilize the graph to ask questions and infer answers. We judiciously select candidate facts to ask in order to minimize the number of candidate facts to conduct crowdsourcing. We propose path-based and topological-sorting-based algorithms that ask multiple questions in parallel in each iteration.

(3) We develop a probability-based method to tolerate the errors introduced by the crowd and propagated through inference rules.

(4) We conduct experiments using real-world datasets on a real crowdsourcing platform. Experimental results show the effectiveness of the proposed approaches.

The rest of this paper is structured as follows. We first review related work in Section 2 and introduce basic concepts related to our work in Section 3. Then we describe our proposed approaches in Section 4. We report experimental results in Section 5 and conclude in Section 6.

2. Related Work

Information extraction techniques are widely applied in the construction of web-scale knowledge bases. In this paper, we use Never Ending Language Learner (NELL) [4] as a case study. NELL starts from a few "seed instances" of each category and relation and generates a knowledge base iteratively. It uses natural language processing and information extraction techniques to extract candidate facts from a large web corpus, using facts learned from the previous iteration as training examples. NELL has four subcomponents that extract candidate facts, namely, Pattern Learner, SEAL, Morphological Classified, and Rule Learner. NELL uses heuristics and ontological constraints to promote candidate facts into a knowledge base, assigning each promotion a confidence value.

Early work on cleaning a noisy knowledge base was considered by Cohen et al. [15]. They considered only a small subset of KB errors. Jiang et al. [8] proposed a method for cleaning knowledge bases at a broader scope using Markov Logic Networks (MLNs). This method performs joint probabilistic inference over candidate facts. To make inference and learning tractable, Jiang et al. surmounted these obstacles with a number of approximations and demonstrated the utility of joint reasoning in comparison to a baseline that considers each fact independently. More recently, Pujara et al. [7] improved the model of Jiang et al. by including multiple extractors and reasoning about coreferent entities. Furthermore, Pujara et al. used probabilistic soft logic (PSL) to avoid scalability limitation of MLNs. Dong et al. [6] employed supervised machine learning methods for fusing distinct information sources by combining noisy extractions from the web with prior knowledge derived from existing knowledge repositories. However, all of above methods are automated algorithms and do not leverage the power of crowdsourcing.

There also exist many research works that incorporate crowdsourcing into data and knowledge management, such as data cleaning [14, 16–19], record linkage [20, 21], schema matching [22–25], and knowledge acquisition [26, 27]. For example, Wang et al. [14] proposed CrowdER to solve the problem of entity resolution via crowdsourcing. Zhang et al. [16] used crowdsourcing to clean uncertain data. Chu et al. [18] proposed KATARA, a data cleaning system that utilizes the power of knowledge bases and crowdsourcing to clean tables. Demartini et al. [20] proposed ZenCrowd which uses a mixed human-machine workflow to solve the entity linking problem. Gokhale et al. [21] studied how to do hands-off crowdsourcing record linkage which requires no involvement of developers. Sarasua et al. [22] studied the problem of ontology matching using crowdsourcing. Fan et al. [23] proposed a hybrid machine-crowdsourcing system for matching web tables. Kondreddi et al. [26, 27] developed HIGGINS, a framework for human intelligence games for knowledge acquisition, to expand and complement the output of automated information extraction methods. However, so far, there has been little discussion about how to use crowdsourcing to clean a noisy knowledge base with semantic constraints.

A sport team is a group of athletes who play a sport together competitively.
(1) Is saints a sport team?
○ Yes
○ No
(2) Is padres a sport team?
○ Yes
○ No
(3) Is boilermakers a sport team?
○ Yes
○ No
(4) Is Soochow_university a sport team?
○ Yes
○ No

Box 1: An example of HITs.

3. Preliminaries

3.1. Knowledge Bases. We consider an automatically extracted knowledge base as a probabilistic knowledge base, which stores facts in a form of triple (subject, predicate, and object), for example, (Brussel, citycapitalofcountry, Belgium). Each fact t_i has a confidence score, representing the probability that the corresponding information extraction system "believes" the fact is correct. We formally define an extracted knowledge base as follows.

Definition 1. An extracted knowledge base (KB) is a 5-tuple $K = (\mathscr{E}, C, R, \Pi, L)$, where

(1) $\mathscr{E} = \{e_1, \ldots, e_{|\mathscr{E}|}\}$ is a set of entities. Each entity $e \in \mathscr{E}$ refers to a real-world object.

(2) $C = \{c_1, \ldots, c_{|C|}\}$ is a set of categories (or types). Each category $c \in C$ is a subset of $\mathscr{E}$. Each entity $e \in \mathscr{E}$ belongs to one or more categories.

(3) $R = \{r_1, \ldots, r_{|R|}\}$ is a set of predicates. Each $r \in R$ defines a binary relation between one or more pairs of types. For example, the *wasBornIn* predicate specifies a binary relation between *Person* and *Place*. We call the types of subject and object domain and range of the predicate, respectively.

(4) $\Pi = \{(t_1, p_1), \ldots, (t_{|\Pi|}, p_{|\Pi|})\}$ is a set of weighted facts. For each $(t, p) \in \Pi$, t is a triple (x, r, y) representing a fact that relation r holds between x and y, where $x, y \in \mathscr{E}, r \in R$; $p \in \mathbb{R}$ is a weight indicating the probability that an information extraction system believes the corresponding fact is correct, that is, a confidence score.

(5) $L = \{F_1, \ldots, F_{|L|}\}$ is a set of ontological relations. It defines concept hierarchy and semantic relationships between categories and relations.

The definition of C implies a concept hierarchy: for any $c_i, c_j \in C$, c_i is subclass of c_j if and only if $c_i \subseteq c_j$. Typing provides semantic context for extracted entities and is commonly adopted by the information extraction systems, so we make it an integral part of the definition. We use $Cat(x, c)$ to denote that x is an entity of category c and use $Rel(x, y, r)$ to denote that relation r holds between entities x and y.

An automatically extracted knowledge base could be very large and noisy. For example, the knowledge vault [6] has 1.6B triples, of which 324M have a confidence of 0.7 or higher, and 271M have a confidence of 0.9 or higher. NELL so far has acquired a knowledge base with over 80M confidence-weighted facts, 2M of which have a confidence of 0.9 or higher. The overall estimated precision of NELL's promoted facts across first 66 iterations is 74%.

3.2. Crowdsourcing. There exist a number of crowdsourcing platforms, such as MTurk and CrowdFlower. In such platforms, we can ask human "workers" to complete microtasks. For example, we may ask them to answer questions like "Is Italy a country?" Each microtask is referred as a human intelligent task (HIT). After having completed a HIT, a worker is rewarded with a certain amount of money based on the difficulty of the HIT. That is, invoking the crowd for knowledge cleaning comes with a monetary cost. In addition, a human worker may not always produce a correct answer for a HIT. To mitigate such human errors, we assign each HIT to multiple workers and then take a majority vote. However, even when majority votes are used, we may still get incorrect answers from the crowd. As a consequence, it is crucial to take human errors into account when designing a crowd-based algorithm.

Given a set of candidate facts to be sent to the crowd, we need to combine them into HITs. For each fact, the crowd needs to verify whether the fact is correct or not. We have four questions as one HIT, where each question contains a candidate fact requiring workers to verify its correctness. Box 1 shows an example of questions we generate as an HIT for MTurk. A brief description of the HIT is shown at the top. To assist workers to understand the fact, we provide a description of the related category or relation and use a human format for each fact.

4. Methodology

Our method takes an automatically extracted knowledge base as input and identifies a set of true facts from noisy

extractions through crowdsourcing. We first introduce the concept of semantic constraints that can be used to detect potential erroneous facts and do inference among candidate facts. And then we propose a score function to measure the usefulness of candidate facts, in order to conduct crowdsourcing. In Section 4.3, we will explain how to leverage semantic constraints as inference rules to prune unnecessary questions. Finally, we will discuss our error-tolerant techniques.

4.1. Semantic Constraints. Integrity constraints are effective tools used in data cleaning. This section introduces a similar concept called semantic constraints that can be used to clean noisy knowledge bases. These constraints can be learned from training data or derived from ontological constraints. The ontological constraints can be seen as axioms or rules in first-order logic. For example, we can represent an ontological constraint (every *Athlete* is a *Perso*n) with a rule, $Athlete(x) \Rightarrow Person(x)$. Similarly, since a *City* is not a *Person*, we can have a following rule, $\text{City}(x) \Rightarrow \neg Person(x)$.

We derive semantic constraints according to ten types of ontological relations used in NELL: subsumption among categories and relations (e.g., every bird is an animal); mutually exclusive categories and relations (e.g., no person is a location); inversion (for mirrored relations like TeamHasPlayer and PlaysForTeam); the type of the domain and range of each predicate (e.g., the mayor of a city must be a person); the functionality of relations (e.g., a person has only one birth date); antisymmetric (e.g., if person a writes book b, then b cannot write a); and antireflexive (e.g., a company cannot produce itself).

We use the following notations: *Sub* and *RSub* for subclass relationships; *Mut* and *RMut* for mutual exclusion relationships; *Dom* and *Ran* for domain and range relationships; Inv for inversion; *Fun* for functionality, *AntiSym* for antisymmetric, and *AntiRef* for antireflexive.

There are two types of semantic constraints according to the label transitive relation between candidate facts: *contradictive relation* and *positive relation*. The derived semantic constraints are shown as follows.

Contradictive Relation. In semantic constraints of this type, if a candidate fact is correct, we can infer that another candidate fact must be incorrect. Violations of contradictive relations indicate potential errors.

$$Mut(c1, c2) \wedge Cat(x, c1) \Longrightarrow \neg Cat(x, c2) \quad (1)$$

$$RMut(r1, r2) \wedge Rel(x, y, r1) \Longrightarrow \neg Rel(x, y, r2) \quad (2)$$

$$Mut(c1, c2) \wedge Dom(r, c1) \wedge Cat(x, c2) \Longrightarrow \neg Rel(x, y, r) \quad (3)$$

$$Mut(c1, c2) \wedge Dom(r, c1) \wedge Rel(x, y, r) \Longrightarrow \neg Cat(x, c2) \quad (4)$$

$$Mut(c1, c2) \wedge Ran(r, c1) \wedge Cat(x, c2) \Longrightarrow \neg Rel(y, x, r) \quad (5)$$

$$Mut(c1, c2) \wedge Ran(r, c1) \wedge Rel(y, x, r) \Longrightarrow \neg Cat(x, c2) \quad (6)$$

$$Fun(r) \wedge Rel(x, y, r) \Longrightarrow \neg Rel(x, z, r) \quad (7)$$

$$AntiRef(r) \Longrightarrow \neg Rel(x, x, r) \quad (8)$$

$$AntiSym(r) \wedge Rel(x, y, r) \Longrightarrow \neg Rel(y, x, r) \quad (9)$$

Positive Relation. In semantic constraints of this type, if a candidate fact is correct, we can infer another candidate fact is also correct.

$$Sub(c1, c2) \wedge Cat(x, c1) \Longrightarrow Cat(x, c2) \quad (10)$$

$$RSub(r1, r2) \wedge Rel(x, y, r1) \Longrightarrow Rel(x, y, r2) \quad (11)$$

$$Inv(r1, r2) \wedge Rel(x, y, r1) \Longrightarrow Rel(y, x, r2) \quad (12)$$

$$Dom(r, c) \wedge Rel(x, y, r) \Longrightarrow Cat(x, c) \quad (13)$$

$$Ran(r, c) \wedge Rel(x, y, r) \Longrightarrow Cat(y, c) \quad (14)$$

Given semantic constraints and a set of candidate facts, we will generate a set of ground rules. A ground rule is a rule containing only candidate facts and no variables. We first instantiate a formula of semantic constraint using ontological relations and candidate facts in the knowledge base. Then, we omit the part of instantiated ontological relations since they are deemed to be true and obtain ground rules containing only candidate facts. For example, (a) in Box 2 are sample ontological relations defined in NELL. Considering candidate facts (b) in Box 2, corresponding ground rules generated according to semantics constraints are shown (c) in Box 2.

While contradictive semantic constraints can be used to detect potential erroneous facts, both positive constraints and contradictive constraints can be used to do inference among candidate facts.

4.2. Ranking Facts Based on Benefit. In this section, we propose a rank-based method for knowledge refining. We would like to select the most beneficial candidate facts to conduct crowdsourcing under a given budget for a knowledge base. It is obvious that we prefer to choose the facts that the corresponding information extraction system is most uncertain about. In addition, the facts that violate semantic constraints the most are of high risk and important ones to the knowledge base. It is beneficial to verify them via crowdsourcing. In this paper, we will use the contradictoriness to estimate the risk and importance of candidate facts. In summary, we will first evaluate the benefit of candidate facts in terms of improving the quality of the knowledge base by taking both uncertainty and contradictoriness into consideration. Then we will rank them according to their evaluation scores and choose top k facts to conduct crowdsourcing.

Uncertainty Score. A threshold τ is often applied to probabilistic extractions. Facts with a high confidence can be assumed to be correct, while facts with a confidence less than threshold τ are deemed to be most likely incorrect. The most uncertain

(a) Ontological Constrains
Domain(ceoof, ceo)
Ran(ceoof, company)
Sub(ceo, person)
RSub(ceoof, topmemberoforganization)
RSub(topmemberoforganization, worksfor)
RSub(topmemberoforganization, personleadsorganization)
Rsub(worksfor, personbelongstoorganization)
RMut(topmemberoforganization, organizationleadbyperson)
(b) Candidate Facts
$t1$ Rel(x, y, ceoof)
$t2$ Cat(x, ceo)
$t3$ Cat(y, company)
$t4$ Rel(x, y, topmemberoforganization)
$t5$ Rel(x, y, organizationleadbyperson)
$t6$ Rel(x, y, worksfor)
$t7$ Cat(x, person)
$t8$ Rel(x, y, personbelongstoorganization)
(c) Ground Rules
$t1 \Rightarrow t2$
$t1 \Rightarrow t3$
$t1 \Rightarrow t4$
$t2 \Rightarrow t7$
$t4 \Rightarrow t6$
$t4 \Rightarrow t7$
$t6 \Rightarrow t7$
$t6 \Rightarrow t8$
$t8 \Rightarrow t7$
$t5 \Rightarrow]t6$
$t6 \Rightarrow]t5$

Box 2: Sample semantic constraints in NELL.

facts are those whose probability are closest to threshold τ. We use $conf_m(t_i)$ to denote the machine-based probability estimation of a fact t_i being correct. Therefore, we model the uncertainty of a fact t_i as follows.

$$Uncertainty\left(t_i\right) = 1 - \left|conf_m\left(t_i\right) - \tau\right|. \tag{15}$$

The information extraction systems commonly provide a confidence score for each candidate fact, that is, the weight p in the knowledge base definition. We adopt the weight p_i as the machine-based probability estimation of a fact t_i being correct.

$$conf_m\left(t_i\right) = p_i. \tag{16}$$

Information extraction systems usually use many different extraction techniques to generate candidates. For example, NELL produces separate extractions from lexical, structural, and morphological patterns. If the patterns used to extract each candidate fact are provided, this extra information can help us better estimate the probability. We can use a simple logistic regression model learned from training data to predict the probability of each candidate fact being correct [28, 29]. The features are whether each pattern cooccurs with the candidate fact, and the coefficients reflect the reliability of patterns.

Contradictoriness Score. Based on contradictive semantic constraints introduced in Section 4.1, we can detect inconsistency, errors, and conflicts among candidate facts as the violations of these constraints. The more facts a fact is contradictory with, the more likely it is a potential error. We define the contradictoriness score of a fact t_i as follows.

$$Contradictoriness\left(t_i\right) = 1 + \sum_j n_j\left(t_i\right), \tag{17}$$

where $n_j(t_i)$ is the number of violated ground rules of F_j when t_i appears in F_j. The addition of 1 ensures the contradictoriness scores are greater than zero.

For example, considering a semantic constraint (rule) F_1 (defined in Section 4.1), supposing there are two ground atoms $Mut(country, bird)$ and $Mut(city, bird)$, and three facts $Cat(Italy, country)$, $Cat(Italy, city)$, and $Cat(Italy, bird)$, then the fact $Cat(Italy, bird)$ violates two ground rules of F_1. Hence, $n_1(Cat(Italy, country)) = 2$.

Combining the above two factors, we use the following function to rank candidate facts.

$$\begin{aligned} Score\left(t_i\right) &= Uncertainty\left(t_i\right) \\ &\quad * Contradictoriness\left(t_i\right). \end{aligned} \tag{18}$$

Based on ranking scores, we select a batch of candidate facts to conduct crowdsourcing at a time. Algorithm 1

Input: A set of extracted candidate facts K, semantic constraints SCs, a budge B, a threshold τ
Output: Refined knowledge base K'
(1) Initialize $confidences$ with machine based estimations
(2) Calculate $Uncertainty$ scores using Eq. (15)
(3) Calculate $Contradictoriness$ scores using Eq. (17)
(4) Calculate $Scores$ using Eq. (18)
(5) Rank candidate facts by scores and select top B instances $\triangle_B$ from K to conduct crowdsourcing
(6) $K_U \leftarrow K \setminus \triangle_B$
(7) $\triangle_p \leftarrow$ facts with a confirm from the crowd
(8) $K' \leftarrow \triangle_p \cup \{t_i \mid t_i \in K_U, conf(t_i) \geq \tau\}$

Algorithm 1: Rank-based knowledge refining.

describes the overall procedure for refining a knowledge base. Given an extracted knowledge base and a set of semantic constraints (SCs), it first initializes confidences of candidate facts being correct with machine-based estimations and calculates scores using (15)–(18). Then, it selects B candidate facts $\triangle_B$ from the knowledge base (K) to conduct crowdsourcing where B is a budget allowed for improving the knowledge base.

4.3. Leveraging Semantic Constraints Pruning Unnecessary Questions. In this section, we discuss how to utilize semantic constraints as inference rules to reduce the crowdsourcing cost. The rank-based method discussed above simply selects top B candidate facts to conduct crowdsourcing at a time. However, by leveraging semantic constraints, we can infer the correctness of a candidate fact from other facts without acquiring the intelligence from the crowd. Thus, we can effectively use the budget for crowdsourcing. For example, if $child(x, y)$ is correct, we do not need to crowdsource the candidate fact $parent(y, x)$ since it can be inferred to be correct based on the inversion constraint (12). If $countrycapital(x, y)$ is correct, we can infer any $countrycapital(x, z)$ is incorrect based on the functionality constraint (7).

4.3.1. Graph-Based Algorithm. To leverage semantic constraints, we model the selected candidate facts (under a given budget) for crowdsourcing as a graph based on ground inference rules and try to infer the correctness of some candidate facts using the graph model.

Definition 2 (graph model). Given a set of candidate facts, we build a directed graph $G = (V, \mathscr{E})$, where each vertex in V is a candidate fact, $\mathscr{E} = \mathscr{E}_p \cup \mathscr{E}_c$, where $\mathscr{E}_p$ represents all positive relations and $\mathscr{E}_c$ represents all contradictive relations. Given two candidate facts t_i and t_j, if $t_i \Rightarrow t_j$, there is a directed edge $e \in \mathscr{E}_p$ from t_i to t_j to represent this positive relation; if $t_i \Rightarrow \neg t_j$, there is a directed edge in $\mathscr{E}_c$ from t_i to t_j to represent this contradictive relation.

Figure 1 shows the graph for candidate facts in Box 2. We use $G_p = (V, \mathscr{E}_p)$ to denote the subgraph containing only edges in $\mathscr{E}_p$ and $G_c = (V, \mathscr{E}_c)$ to denote the subgraph containing only edges in $\mathscr{E}_c$.

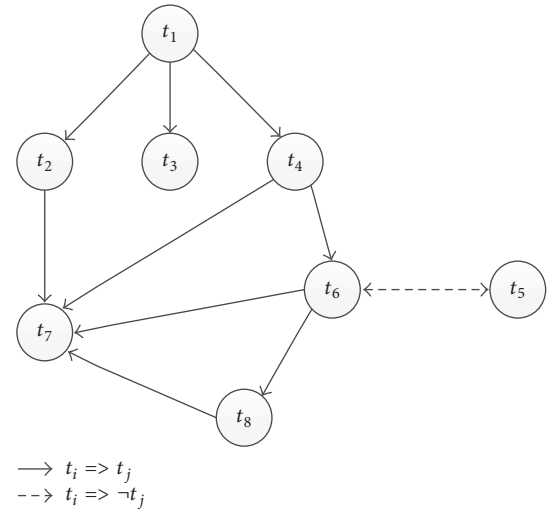

Figure 1: A sample of graph model.

Graph Coloring. Each vertex in G has two possibilities: (1) the candidate fact is correct and we color it Green; (2) the candidate fact is incorrect and we color it Red. Initially, each vertex is uncolored. Our goal is to utilize the crowd to color all vertices.

A straightforward method is to take the candidate fact on each vertex as a question and ask workers to answer the question, that is, whether the candidate fact is correct. If a worker thinks that the candidate fact is correct, the worker returns Yes and No otherwise. Based on the workers' results, we get a voted answer on each vertex. If majority of workers vote Yes, we color it Green; otherwise we color it Red. Next, we interchangeably use vertex, fact, and question if the context is clear.

This method is rather expensive as there are many vertices in the graph. To address this issue, we propose an effective coloring framework to reduce the number of questions. Algorithm 2 shows the pseudocode. It first constructs a graph based on ground inference rules (line 1-2). Then it selects an uncolored vertex t_i and asks workers to answer Yes or No for the vertex (line 4). If majority of workers vote Yes, we not only color t_i Green, but also color all of its descendants in $G_p(desc_p(t_i))$ Green and color all of their children in

Input: A set of facts $K = \{t_1, t_2, \ldots, t_n\}$, semantic constraints SCs
Output: All vertices are colored as Green or Red
(1) Generate ground inference rules using semantic constraints
(2) Construct $G = (V, \mathscr{E})$ based on ground inference rules
(3) **while** there exist uncolored vertices in V **do**
(4) Select an uncolored vertex t_i to conduct crowdsourcing;
(5) **if** majority workers vote Yes **then**
(6) color t_i and $desc_p(t_i)$ Green;
(7) color $child_c(t_i)$ and $child_c(desc_p(t_i))$ Red;
(8) color $ance_p(child_c(t_i))$ and $ance_p(child_c(desc_p(t_i)))$ Red;
(9) **else**
(10) color t_i and $ance_p(t_i)$ Red;
(11) **end if**
(12) **end while**
(13) **return** colored V;

Algorithm 2: Graph coloring.

G_c ($child_c(t_i)$ and $child_c(desc_p(t_i))$) Red and their ancestors in G_p ($ance_p(child_c(t_i))$ and $ance_p(child_c(desc_p(t_i)))$) Red (line 6–8). In other words, for $t_i \Rightarrow t_j$, we can infer that t_j is also correct; for $t_i \Rightarrow \neg t_j$, we can infer t_j is incorrect. If majority of workers vote No, we not only color t_i Red, but also color all of its ancestors in G_p ($ance_p(t_i)$) Red (line 10). In other words, for $t_j \Rightarrow t_i$, we infer that t_j is also incorrect. If all the vertices have been colored, the algorithm terminates. Otherwise, it selects an uncolored vertex and repeats the above steps (line 3–12).

Obviously, this method can reduce the crowdsourcing cost as we can avoid asking questions for many unnecessary vertices. For example, considering the constructed graph in Figure 1, a naive method is to conduct crowdsourcing for all eight facts. However, if we first conduct crowdsourcing for t_6, as majority of workers vote Yes, we can color t_6 and their descendants t_7 and t_8 Green and color t_5 Red without conducting crowdsourcing for its descendants. Then if we continue to conduct crowdsourcing for t_4, as majority of workers vote No, we can color t_4 and its ancestor t_1 Red.

An important problem in the algorithm is to select the minimum number of vertices to conduct crowdsourcing, so that all vertices in the graph are colored. We will first formulate the question selection problem and then propose a path-based algorithm and a topological-sorting-based algorithm that select multiple vertices in each iteration to solve the problem.

4.3.2. Optimal Vertex Selection. As we know, we have the basic coloring strategy: if a vertex is Green, then all of its descendants in G_p are Green but all of its children in G_c and their ancestors in G_p are Red; if a vertex is Red, then all of its ancestors in G_p are Red. We will discuss how to support the case that the two conditions do not hold in Section 4.4.

Definition 3 (optimal graph coloring). Given a graph, the optimal graph coloring problem aims to select the minimum number of vertices as questions to color all the vertices using the coloring strategy.

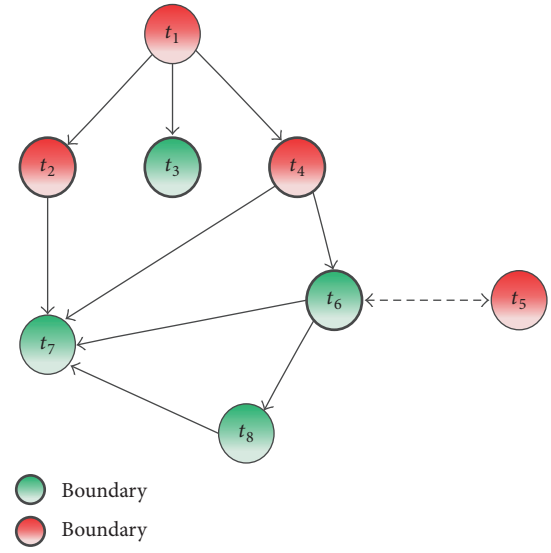

Figure 2: A sample of boundary vertex.

For example, in Figure 2, if we sequentially select vertices t_1, t_2, t_3, t_4, t_5, t_7, and t_6, we should ask seven questions. The optimal crowdsourced vertices are t_2, t_3, t_4, and t_6 (highlighted by bold cycles), because the colors of these vertices cannot be inferred based on the colors of other vertices. Next we will study how to identify the optimal vertices. Before that, we introduce a concept for making our explanation easily.

Definition 4 (boundary vertex). A vertex is a boundary vertex if its color cannot be inferred based on other vertices' colors. There are four cases in G_p: (1) all of its parents have different colors with the vertex; (2) all of its children have different colors with the vertex; (3) it has no parent and its color is Green; or (4) it has no children and its color is Red. In addition, there are two cases in G_c: (1) all of its parents are Red; (2) it has no parents (in-edge).

Input: $G = (V, \mathscr{E}_p \cup \mathscr{E}_c)$
Output: All vertices in G are colored as Green or Red
(1) **while** there exist uncolored vertices in V **do**
(2) Compute disjoint paths in G_p using maximal matching;
(3) **if** there exist paths with length > 1 **then**
(4) $v \leftarrow$ select the optimal vertex of the longest path
(5) **else**
(6) $v \leftarrow$ select an optimal vertex according to G_c
(7) **end if**
(8) Crowdsourcing v and color G;
(9) Remove the colored vertices;
(10) **end while**
(11) **return** colored V

Algorithm 3: Question selection: SinglePath.

For example, t_6 is a boundary vertex in G_p as its parent t_4 has a different color. t_8 is not a boundary vertex as its parent t_6 has the same color and t_8's color can be inferred based on t_6's color. t_6 is also a boundary vertex in G_c as its parent t_5 is Red.

Here we use V_B to denote all the boundary vertices in the graph and $V_B(G_p)$ and $V_B(G_c)$ to denote the boundary vertices in G_p and G_c, respectively. There are overlaps between $V_B(G_p)$ and $V_B(G_c)$, so $V_B = V_B(G_p) \cap V_B(G_c)$. Ideally, all vertices in V_B should be checked, because their colors cannot be inferred. Thus, the number of vertices checked using any algorithm should not be smaller than the number of boundary vertices in V_B. However, since we do not know the ground truths of all vertices in the graph, we cannot identify the boundary vertices in advance. To address this problem, we propose effective algorithms to identify the boundary vertices in G_p with a theoretical guarantee and use a greedy algorithm to identify the boundary vertices in G_c, respectively. Meanwhile, we note that there are more boundary vertices in G_c because of the limited influence of a vertex in G_c than those in G_p. Hence, we consider firstly the boundary vertices in G_p.

Optimal Vertex Selection in G_p. Given a path in G_p, we can use a binary search method to select the boundary vertices. We initially crowdsource the mid-vertex on the path. Based on the result of the mid-vertex, we determine the next step. There are two situations: (1) If the mid-vertex is colored Green, its descendants' colors can be inferred but its ancestors' colors can not be inferred. Thus, we can crowdsource the next mid-vertex between the current vertex and the source vertex of the path. (2) If the mid-vertex is colored Red, its ancestors' colors can be inferred but its descendants' colors can not be inferred. Therefore, we can crowdsource the next mid-vertex between the current vertex and the destination vertex of the path. Iteratively, we can find all the boundary vertices. For the path P with $|P|$ vertices, the number of crowdsourcing vertices is $O(\log |P|)$.

Optimal Vertex Selection in G_c. We greedily select the vertices with no in-edge in G_c and the vertices with the largest confidence value in each contradictive group (i.e., connected subgraph), since only Green vertices can be used to infer colors of its children.

4.3.3. Path-Based Algorithm. We can divide the graph G_p into a set of disjoint paths (i.e., any two paths have no common vertices). Then we can use the binary search method described above to determine the vertices for crowdsourcing. As the maximum length of a path is $|V|$, the number of crowdsourcing vertices is $O(\beta \log |V|)$, where β is the number of disjoint paths. If $\beta = 1$, we need to crowdsource $\log |V|$ vertices.

Finding β Disjoint Paths. In order to find the disjoint paths, we transform the graph G_p into a bipartite graph $G_p^b = ((V_1^b, V_2^b), \mathscr{E}_p^b)$, where $V_1^b = V_2^b = V$ and there is an edge between $v_1 \in V_1^b$ and $v_2 \in V_2^b$ if there is an edge $(v_1, v_2) \in \mathscr{E}_p$. We find maximal matching in G_p^b, which is a maximal set of edges in G_p^b where any two edges do not share a common vertex in V_1^b and V_2^b. That is, for any two edges (v, v'), (u, u') in the matching, $v \neq u$ and $v' \neq u'$. Obviously, any two edges in the matching sharing the same vertex in V must be on the same path. Based on this idea, we utilize the maximal matching to find the β disjoint paths as follows.

Let M denote the maximal matchings, Y_1 denote the set of the first vertices in M and Y_2 denote the set of the second vertices in M. Then $V_2^b - Y_2$ is the set of vertices that have no in-edges, and we can take them as the first vertices of paths. For each such a vertex v, if it has an edge (v, v'), we take v' as the second vertex in a path. Then we check whether v' has an edge (v', v''). Iteratively, we can find the path starting at v. The paths computed using maximal matching satisfy disjoint, complete, and minimal paths [30].

Then we propose a serial path-based vertex-selection algorithm. The pseudocode is shown in Algorithm 3. It computes disjoint paths in G_p using maximal matching and selects the optimal vertex of the longest path to conduct crowdsourcing. When there is no path with length greater than 1, it selects an optimal vertex according to G_c.

Input: $G = (V, \mathscr{E}_p \cup \mathscr{E}_c)$
Output: All vertices in G are colored as Green or Red
(1) **while** there exist uncolored vertices in V **do**
(2) Compute β disjoint paths in G_p using maximal matching;
(3) **if** there exist paths with length > 1 **then**
(4) $N \leftarrow$ select an optimal vertex of each path with lengh > 1;
(5) **else**
(6) $N \leftarrow$ select optimal vertices according to G_c;
(7) **end if**
(8) Crowdsourcing N in parallel and color G;
(9) Remove the colored vertices;
(10) **end while**
(11) **return** colored V

Algorithm 4: Question selection: Multi-Path.

Input: $G = (V, \mathscr{E}_p \cup \mathscr{E}_c)$
Output: All vertices in G are colored as Green or Red
(1) **while** there exist uncolored vertices in V **do**
(2) Do a topological sorting on the uncolored vertices in G_P and obtain $|L|$ sets, $L_1, L_2, \ldots, L_{|L|}$;
(3) **if** $|L| > 1$ **then**
(4) $N \leftarrow$ vertices in $L_{(|L|+1)/2}$;
(5) **else**
(6) $N \leftarrow$ select the optimal vertices according to G_c;
(7) **end if**
(8) Crowdsourcing N in parallel and color G;
(9) Remove the colored vertices;
(10) **end while**
(11) **return** colored V

Algorithm 5: Question selection: TopologicalSorting.

The serial path-based vertex-selection algorithm can only publish a single fact to a crowdsourcing platform at a time, which is unable to crowdsource candidate facts simultaneously and results in long latency. To overcome this drawback, we extend the path-based algorithm to support parallel settings, which select multiple vertices and publish the corresponding candidate facts simultaneously to the crowdsourcing platform in each iteration. The pseudo code is shown in Algorithm 4. We first identify the β disjoint paths and use the optimal vertex selection strategy discussed in Section 4.3.2 to select one vertex from each path to conduct crowdsourcing in parallel. When there is no path with length greater than 1, we select the optimal vertices according to G_c. Based on the answers of these vertices, we color the graph. Next we remove the colored vertices and repeat the above step until all the vertices are colored.

However, the parallel algorithm may generate conflicts. For example, if t_i is colored Green and t_j is colored Red, then there is a conflict on t where $t_i \Rightarrow t$ and $t \Rightarrow t_j$, because t is inferred as Green based on t_i but is inferred as Red based on t_j. To address this confliction, we can use majority voting to vote t's color and randomly choose one if a tie occurred.

4.3.4. Topological-Sorting-Based Algorithm. Note that the maximal matching can be computed in $O(\beta|V|^2)$ [30], which is too slow in practice when used for a large knowledge base. To address this issue, we perform a topological sorting on the vertices. We first identify the set of vertices with zero in-degree, denoted by L_1. Then we delete them from the graph and find another set of vertices whose in-degrees are zero, denoted by L_2. We repeat this step until all vertices are deleted. Suppose there are $|L|$ sets, $L_1, L_2, \ldots, L_{|L|}$. Obviously vertices in each L_i have no in-edges (as their in-degrees are 0). Therefore, each L_i can be considered as an independent set.

We design a topological-sorting-based algorithm to improve the time efficiency of the maximal matching. It first computes topological-sorted sets $L_1, L_2, \ldots, L_{|L|}$ in G_p. And then it crowdsources vertices in the middle set $L_{(|L|+1)/2}$ in parallel. When $|L| \leq 1$, it selects optimal vertices according to G_c to conduct crowdsourcing. Based on the results of these vertices, it colors the graph and removes the colored vertices and $L_{(|L|+1)/2}$ from the set L. It repeats the above step and iteratively colors all vertices. The pseudo code of the topological-sorting-based algorithm is shown in Algorithm 5.

4.4. Tolerating Errors. There are two types of possible errors in our graph-based framework. The first type is caused by workers' errors and the second type is propagated through

Input: $G = (V, \mathscr{E})$
Output: All vertices in G are colored as Green or Red
(1) **while** there exist uncolored vertices in V **do**
(2) Select a set of uncolored vertices to conduct crowdsourcing;
(3) **for** each crowdsourced t_i with an answer **do**
(4) **if** *crowd confidence* ≥ 0.8 **then**
(5) color t_i and related vertices using coloring strategy;
(6) **else**
(7) color t_i Blue;
(8) **end if**
(9) **end for**
(10) **end while**
(11) Learn a logistic regression model to the machine confidence scores from Green and Red vertices;
(12) Predict colors of facts in Blue vertices;
(13) **return** colored V

Algorithm 6: Error-tolerant graph coloring.

Table 1: Statistic characters of dataset.

Dataset	Category	Relation	Total
Candidate	836K	182K	1.02M
Promotion	354K	64K	418K
Ontological Relation	18K	52K	70K
Test	2002	2546	4546
Training	4777	5089	9866

inference rules. For example, suppose a candidate fact t_i is actually incorrect. However, the workers wrongly label it as correct. This error is caused by workers' errors. Consider a contradictive fact t_j of t_i, whose labels are correct. Our graph-based algorithms could wrongly label it as incorrect using inference rules. This error is propagated through inference rules. We will discuss how to address these errors in our framework as follows.

Confidence of Workers' Answer. To tolerate workers' errors, we assign each candidate fact to multiple workers and aggregate their answers. There are many methods to compute the confidence of workers' answers. We use majority voting as an example and any other techniques can be integrated into our framework. Suppose each candidate fact is assigned to z workers and $y > z/2$ workers vote a consensus answer (e.g., Yes) and $z - y$ workers vote the other answer (e.g., No). The confidence of the voted answer is $c = y/z$.

Error-Tolerant Coloring. For each crowdsourced fact, if the confidence of workers on this fact is high, for example, greater than 0.8, we use inference rules to label related candidate facts; otherwise, we label it uncertain (color the vertex in graph as Blue) and do not use it to infer the labels of other candidate facts. For the Green and the Red vertices, we take them as ground truths as their answers have large confidences. Then we utilize them to color Blue *(*uncertain*)* vertices. Specifically, we use facts in Green and Red vertices to learn a logistic regression model on the machine confidence scores (provided by the information extraction system) and predict the labels of facts in Blue vertices.

The pseudo code of our error-tolerant coloring algorithm in shown in Algorithm 6. It uses the coloring strategy only for the vertices with high-confidence answers (line 5) and utilizes the logistic regression model to color the vertices with low-confidence answers (lines 11-12).

5. Experiments

In this section, we evaluate our methods and report experimental results.

5.1. Experimental Setup

Datasets. NELL [4] generates a knowledge base iteratively. In each iteration, NELL uses facts learned from the previous iteration and a corpus of web pages to generate a new set of candidate facts. NELL selectively promotes those candidates that have a high confidence from the extractors and obey ontological constraints with the existing knowledge base to build a high-precision knowledge base. We use extractions of the $165th$ iteration of NELL released by [7] to evaluate our method, containing over 1 M extractions, with a manual labelled test set consisting of 4546 instances and a training set consisting of 9866 instances. There are 70 K ontological relations in total. Table 1 shows the statistics of the data set. The training set can be used to calibrate the confidence scores from the original system. When training data is not available, we can adopt the confidence provided by the information extraction system as the probability.

We calculate contradictoriness scores among all candidate facts and select candidate facts from the test set for

crowdsourcing. We use a threshold 0.5 for the confidence score. For crowdsourced data, a fact is treated as correct only when more than half of crowd answers are "Yes." We compare our methods with other popular methods in terms of the quality, the number of questions, and the number of iterations. To evaluate the quality, we use three metrics, that is, precision, recall, and *F1*. Suppose the set of correct facts is S_T, and the set of facts that an algorithm reports as correct is S_P. Then the precision is $p = |S_T \cap S_P|/|S_P|$, the recall is $r = |S_T \cap S_P|/|S_T|$, and the F-measure is $F1 = 2pr/(p + r)$.

Crowdsourcing on MTurk. We use MTurk for crowdsourcing. We post all candidate facts in the test set to MTurk and record the crowd's answers in a local file F. During our experiments, when a method requests to crowdsource candidate facts, we retrieve answers from F instead of posting facts to MTurk. This ensures that all methods utilize the same set of crowdsourced results, for the fairness of comparisons. We take four microtasks as one HIT, where each microtask contains a candidate fact. To assist workers to understand the fact, we provide a description of each category or relation and use a human format for each entity (see Box 1 as an example). We pay \$0.02 each time a worker completes an HIT and \$0.01 to MTurk for publishing each HIT. We assign each HIT to five workers. We require that each worker has an approval rate greater than 95%. This setting intends to ensure that all workers provide reasonably accurate answers to the HITs.

5.2. Experimental Results. We first compare our method with state-of-the-art methods for knowledge refining. Then we evaluate our rank function, question selection strategies, and error-tolerant techniques, respectively.

5.2.1. Evaluation of Our Methods. In order to evaluate the effectiveness of our proposed techniques, we compare our methods Rank, Graph, and Graph+ (graph-based method with error-tolerant techniques) with two recent methods for cleaning automatically extracted knowledge bases, that is, MLN [8] and PSL [7], using previously reported results on the same evaluation set. We also compare with the default strategy used by the NELL [4] project to choose candidate facts to include in the knowledge base.

MLN [8]. This method defines a Markov logic network (MLN) to perform jointly probabilistic inference over candidate facts. We compare our method against the best-performing MLN model from [8], which expresses ontological constraints, and candidate and promoted facts trough logical rules. The MLN method reports an output with a 0.5 marginal probability cutoff, which maximizes the *F1* score.

PSL [7]. This method uses probabilistic soft logic (PSL) to jointly reason candidate facts and identify coreferent entities, which can perform inference more efficiently. The PSL method reports results using a soft-truth threshold 0.55 to maximize *F1*.

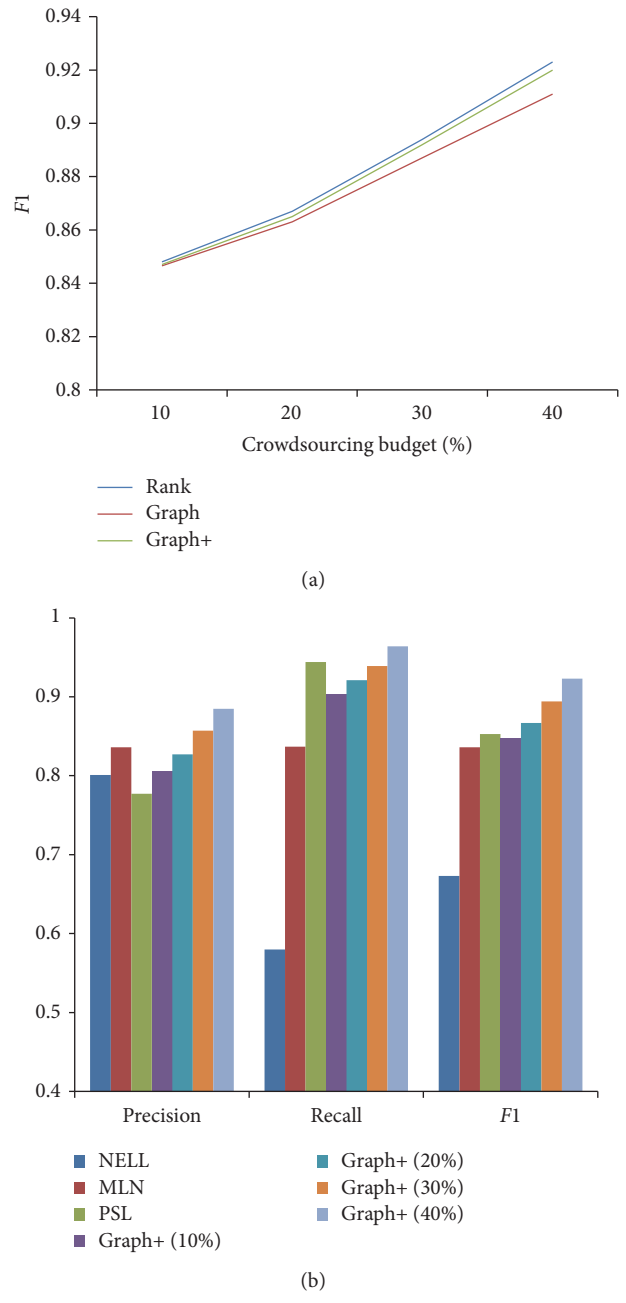

Figure 3: Evaluation of proposed methods. (a) Quality comparison of proposed methods for different crowdsourcing budgets. (b) Quality comparison with the state-of-the-art methods.

NELL [4]. We also compare the default strategy used by the NELL project to choose candidate facts to include in the knowledge base. We take the promoted facts as its result.

Given a budget B (e.g., 40% of candidate facts), our Rank method selects top B candidate facts to conduct crowdsourcing at a time. For the Graph and Graph+ methods, we construct a graph with the top B candidate facts and use the topological-sorting algorithm to select questions for crowdsourcing. Figure 3(a) shows a comparison of the overall performance of our Rank, Graph, and Graph+ methods. We

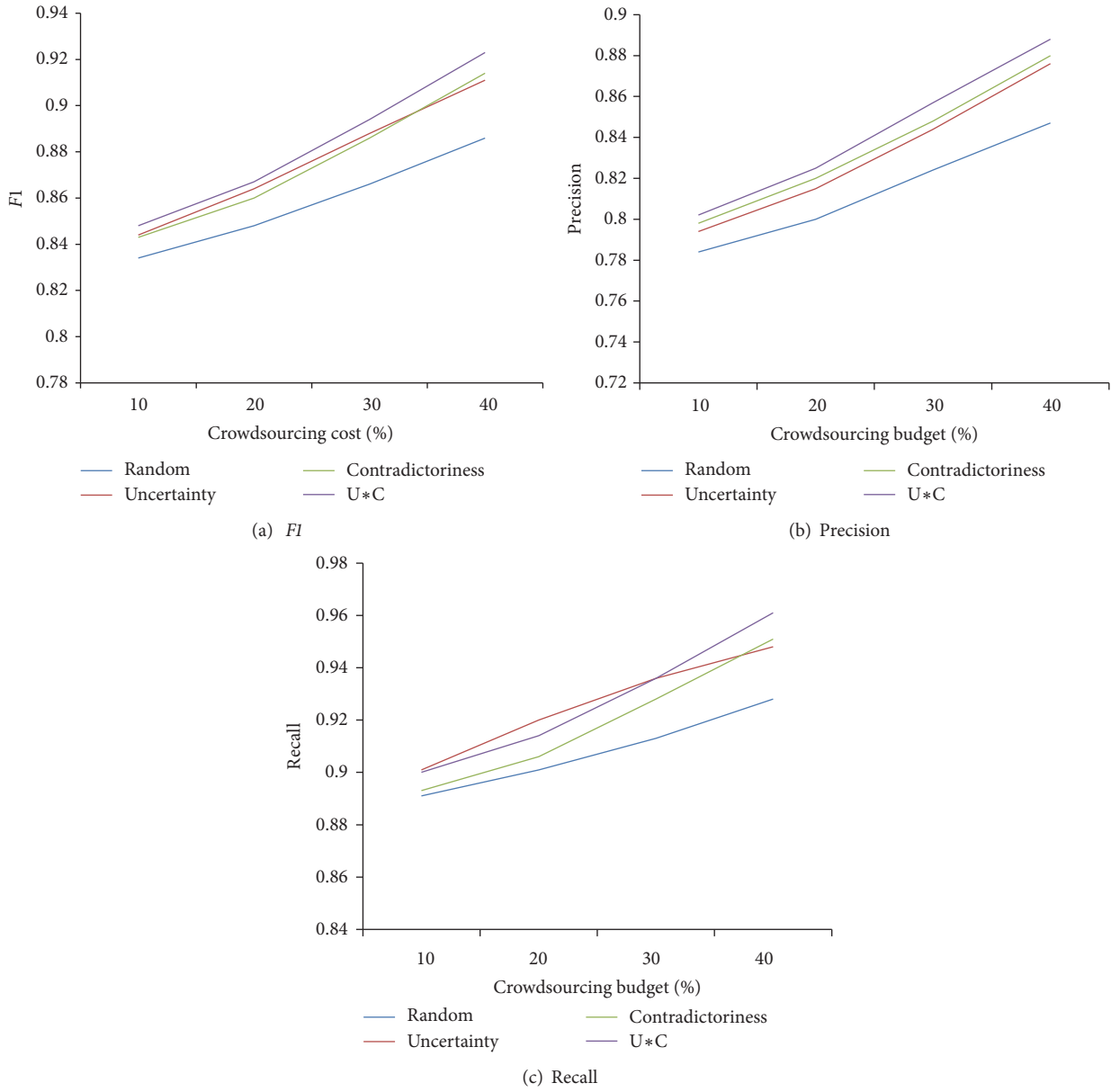

Figure 4: Evaluation of different ranking functions.

report the results of our methods under different budgets. From Figure 3(a), we can see that, given a larger crowdsourcing budget, our method can obtain a higher performance. Graph+ and Rank achieve a similar quality. However, Graph+ asks fewer questions than Rank, as shown in Section 5.2.3. Graph+ outperforms Graph, because Graph+ can tolerate workers' errors by not coloring unconfident vertices and thus avoids enlarging the errors by a wrong colored vertex. Figure 3(b) shows a comparison of the overall performance with the state-of-the-art methods. From Figure 3(b), we can see that MLN and PSL perform well in precision or recall, respectively. Our method improves both precision and recall. Overall, our method improves significantly on *F1*. With a reasonable budget (above 20% test instances), our method outperforms both MLN and PSL methods in terms of *F1*.

5.2.2. Evaluation on Rank Function. In this experiment, we evaluate our ranking function, which is a key for selecting crowdsourcing candidate facts in the rank-based method. This function, denoted as U∗C, quantifies the usefulness of a candidate fact by considering both its uncertainty and its contradictoriness with other facts. We compare U∗C against following baselines. (1) Method *Uncertainty* considers only uncertainty scores. (2) Method *Contradictoriness* considers only contradictoriness scores. (3) Method *Random* selects candidate facts for crowdsourcing randomly.

Figure 4 shows the results of *F1* using different ranking functions. The U∗C method achieves the highest *F1*. The Uncertainty method achieves the highest recall at the beginning but the speed of improvement slowing down with the increment of the budget. This is because there are

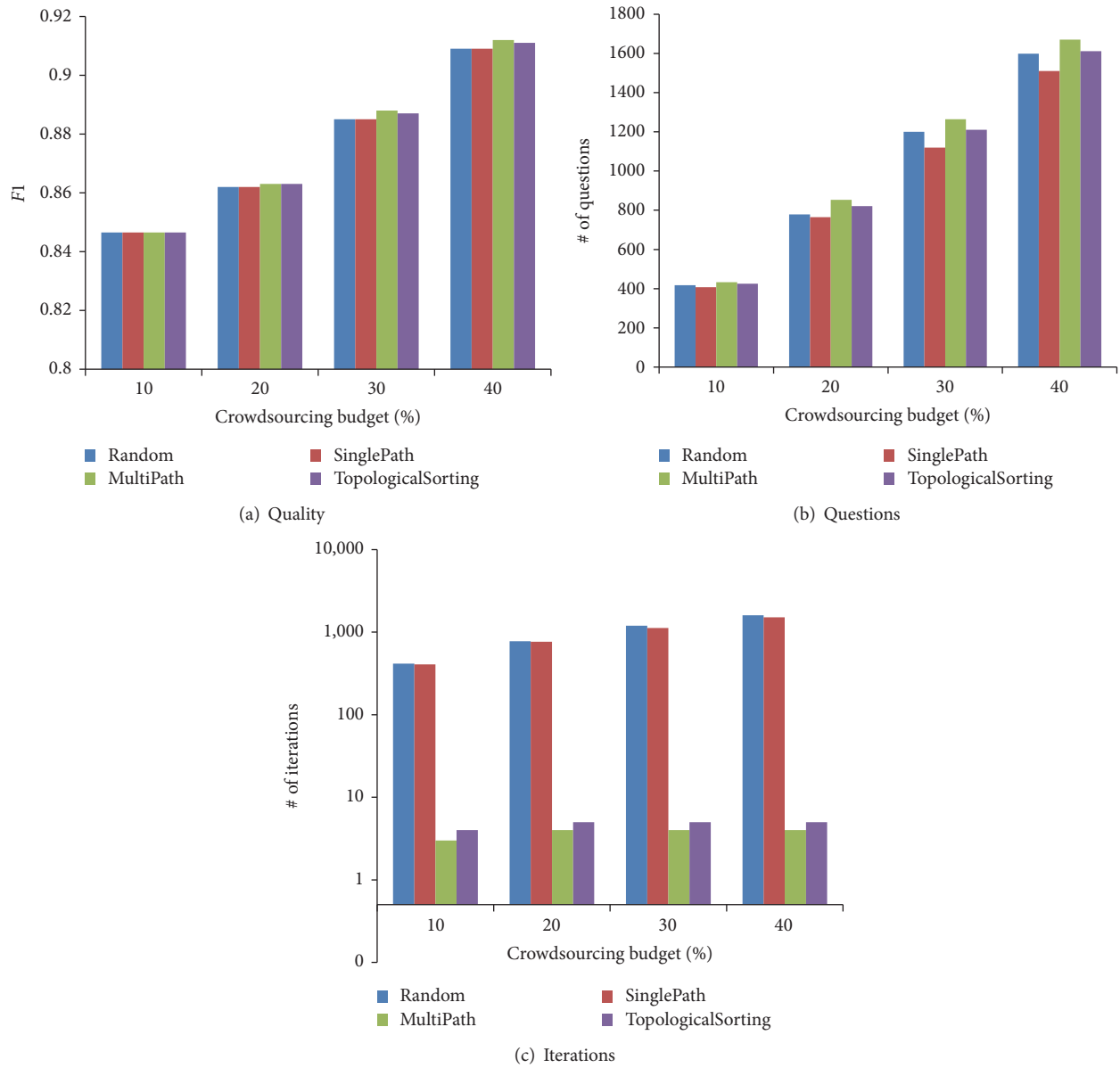

Figure 5: Evaluation of question selection strategies on the test dataset.

many false positives and negatives among candidate facts with confidences around the threshold, which have higher uncertainty scores. The error rate drops quickly when the difference between confidence and threshold increases, while considering contradictoriness can still help detect potential erroneous facts effectively. The *Contradictoriness* method achieves better precision than *Uncertainty*. *Random* consistently performs the worst.

5.2.3. Evaluation on Question Selection. From Section 5.2.1, we can see that the Graph+ method has a similar quality with the Rank method. In this section, we focus on the efficiency of question selection algorithms in terms of the number of questions and the number of iterations. We evaluate the path-based and topological-sorting-based question selection algorithms proposed in Sections 4.3.3 and 4.3.4. We compare four algorithms: (1) Random: which randomly selects a vertex in each iteration. (2) SinglePath: which selects a vertex from the longest path in each iteration. (3) Multipath: which selects multiple vertices from multiple disjoint paths in each iteration. (4) TopologicalSorting: which selects multiple independent vertices based on topological sorting in each iteration. We compare them in terms of the quality, the number of questions, and the number of iterations, shown in Figure 5.

From Figure 5(a), we can see that the four methods achieve the similar quality, because different question orders do not affect the quality based on inference rules. From Figure 5(b), we can see that the two parallel algorithms Multipath and TopologicalSorting crowdsource a few more

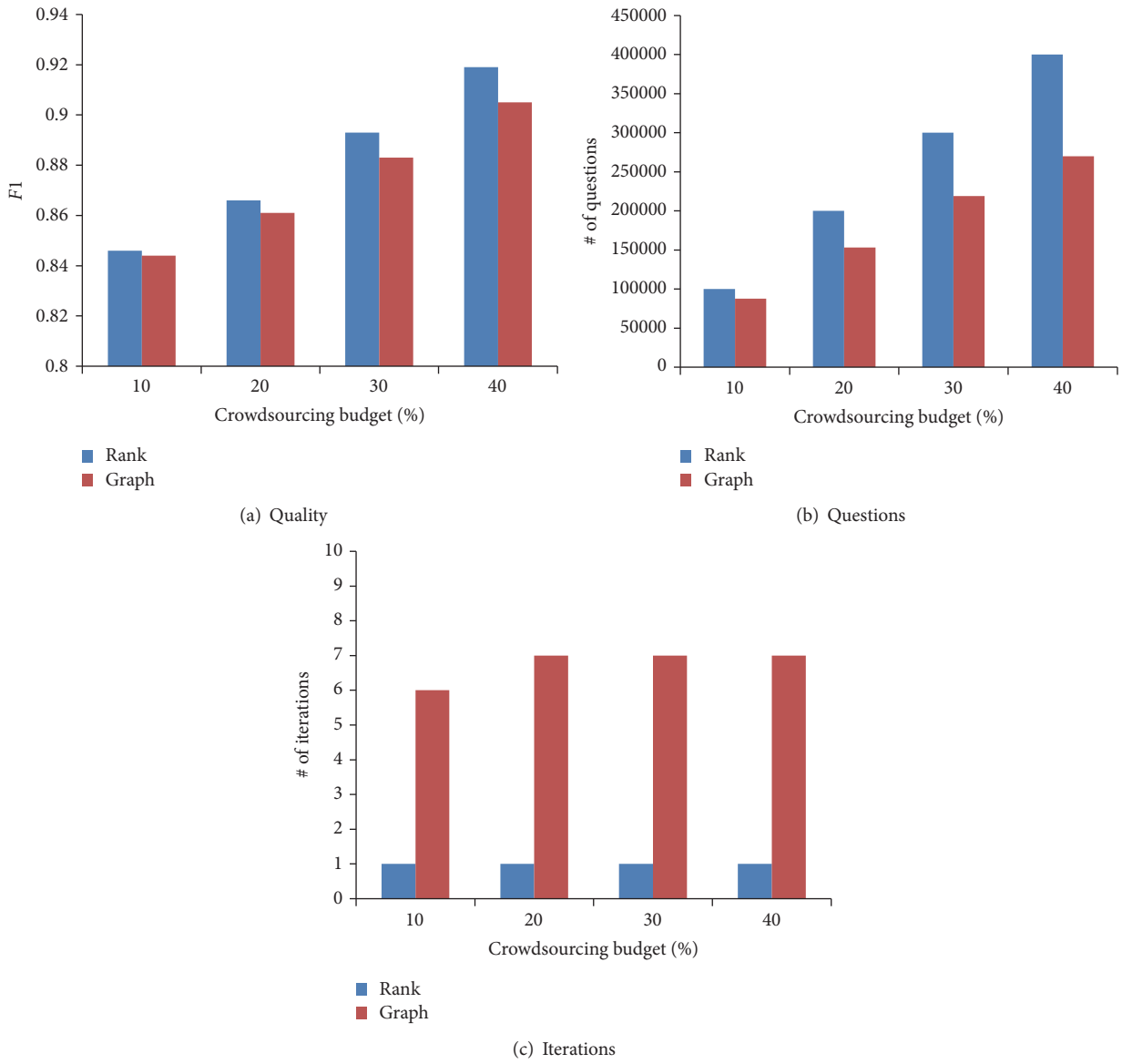

Figure 6: Evaluation of question selection strategies on the complete dataset.

questions than SinglePath. This is because Multipath may crowdsource vertices with ancestor-descendant relationships and TopologicalSorting may crowdsource vertices with the same descendants which can be avoided by our serial algorithm SinglePath based on the inference rules. TopologicalSorting outperforms Multipath because TopologicalSorting crowdsources independent questions in each iteration while Multipath may crowdsource dependent questions. SinglePath outperforms Random and reduces the number of questions. This is because SinglePath can effectively identify the boundary vertex using the optimal vertex search strategy. From Figure 5(c), the two parallel algorithms Multipath and TopologicalSorting significantly outperform SinglePath and Random as they crowdsource questions in parallel.

To evaluate our graph-based method (Graph) on reducing the number of questions, we conduct additional simulation experiments on the complete dataset, using NELL beliefs as ground truths and simulating workers with accuracy of 90%. Our experimental results are shown in Figure 6. Figure 6 shows that our graph-based method crowdsources fewer questions than our rank-based method (Rank). It saves even more than 30%, comparing with the rank-based method. This is because we can utilize the inference rules to prune many candidate facts that do not need to be crowdsourced. The rank-based method achieves a higher quality at the expense of crowdsourcing many more questions. Besides, the graph-based method only involves a few iterations, because it can crowdsource many questions in parallel.

5.2.4. Evaluation on the Error-Tolerant Solution. In this section, we evaluate the effectiveness of our error-tolerant solution (proposed in Section 4.4) by comparing two algorithms: (1) Graph: which does not consider errors; (2) Graph+: which

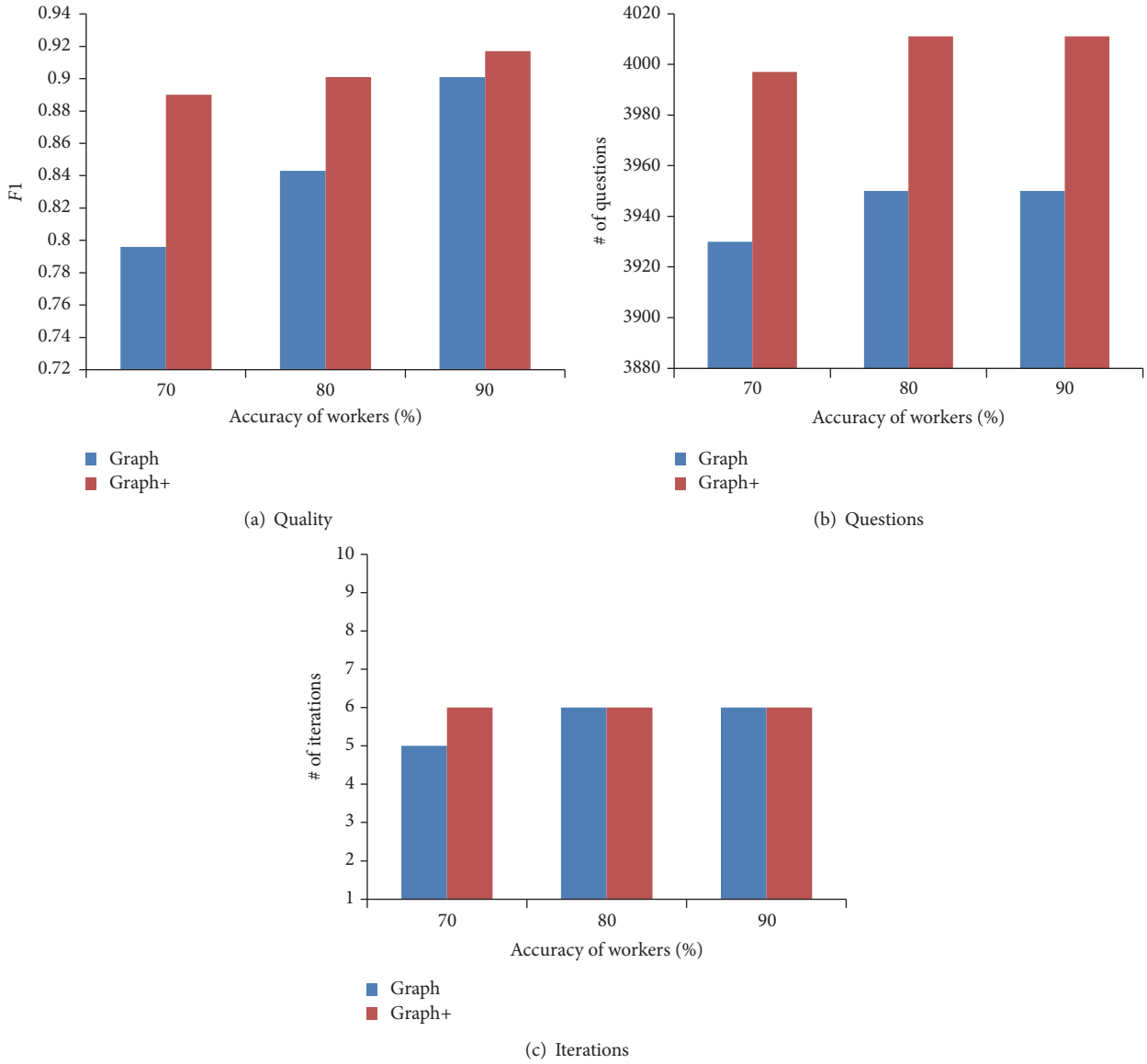

Figure 7: Evaluation of our error-tolerant technique.

extends Graph to tolerate errors. We use simulated workers and conduct evaluation under different accuracy levels of crowdsourcing workers (i.e., 70%, 80%, and 90%) on test dataset. We compare Graph+ with Graph in terms of quality, the number of questions, and the number of iterations. Our experimental results are shown in Figure 7.

From Figure 7, we can see that Graph+ achieves a higher quality than Graph, because it can tolerate the errors introduced by crowdsourcing workers and avoid error propagation along the inference rules. Graph+ significantly outperforms Graph for low-quality workers. With the increment of the accuracy level of workers, the improvement decreases. On the other hand, Graph+ crowdsources a little more questions than Graph. This is because Graph+ does not utilize the inference rules for some facts, so that it reduces the number of inferred facts. From Figure 7(c), we can see that the two methods have the same number of iterations. This is expected, since the only difference between Graph+ and Graph is that Graph+ does not infer the answers for some unconfident facts. The accuracy level of crowdsourcing workers has little impact on the number of questions and the number of iterations for both methods, because the number of questions and the number of iterations are determined by the graph structure. Therefore, we can use the error-tolerant technique to improve the quality of the knowledge base.

6. Conclusions

We proposed a cost-effective method for cleaning automatically extracted knowledge bases using crowdsourcing. Our method uses a ranking score to select the most beneficial candidate facts for crowdsourcing in terms of improving the quality of knowledge bases. We constructed a graph based on the semantic constraints and utilized the graph

to crowdsource questions and infer answers. We evaluated the effectiveness of our methods on real-world web extractions from NELL. Our experimental results showed that our method outperforms both MLN-based and PSL-based methods in terms of *F1* under a reasonable crowdsourcing cost.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was partially supported by Chinese NSFC (61170020, 61402311, and 61440053), Jiangsu Province Colleges and Universities Natural Science Research project (13KJB520021), Jiangsu Province Postgraduate Cultivation and Innovation project (CXZZ13_0813), and the US National Science Foundation (IIS-1115417).

References

[1] F. M. Suchanek and G. Weikum, "Knowledge bases in the age of big data analytics," *Proceedings of the VLDB Endowment*, vol. 7, no. 13, pp. 1713–1714, 2014.

[2] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *Proceedings of the 16th international conference on World Wide Web*, pp. 697–706, 2007.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "DBpedia: a nucleus for a Web of open data," in *The Semantic Web*, pp. 722–735, Springer, 2007.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr., and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *AAAI*, vol. 5, p. 3, 2010.

[5] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam, "Open information extraction: The second generation," in *Proceedings of 22nd International Joint Conference on Artificial Intelligence, IJCAI 2011*, pp. 3–10, esp, July 2011.

[6] X. Dong, E. Gabrilovich, G. Heitz et al., "Knowledge vault: a web-scale approach to probabilistic knowledge fusion," in *20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2014*, pp. 601–610, usa, August 2014.

[7] J. Pujara, H. Miao, L. Getoor, and W. Cohen, "Knowledge graph identification," in *The Semantic Web-ISWC 2013*, pp. 542–557, Springer, 2013.

[8] S. Jiang, D. Lowd, and D. Dou, "Learning to refine an automatically extracted knowledge base using markov logic," in *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pp. 912–917, 2012.

[9] Z. Xia, X. Wang, X. Sun, Q. Liu, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, 2015.

[10] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 9, pp. 2546–2559, 2015.

[11] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[12] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast-but is it good?: evaluating non-expert annotations for natural language tasks," in *Proceedings of the conference on empirical methods in natural language processing*, pp. 254–263, Association for Computational Linguistics, 2008.

[13] L. Von Ahn and L. Dabbish, "Designing games with a purpose," *Communications of the ACM*, vol. 51, no. 8, pp. 58–67, 2008.

[14] J. Wang, T. Kraska, M. J. Franklin, and J. Feng, "CrowdER: Crowdsourcing entity resolution," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.

[15] W. W. Cohen, H. Kautz, and D. McAllester, "Hardening soft information sources," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pp. 255–259, USA, August 2000.

[16] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu, "Cleaning uncertain data with a noisy crowd," in *2015 IEEE 31st International Conference on Data Engineering*, pp. 6–17, 2015.

[17] Y. Zheng, B. Jeon, D. Xu, Q. M. J. Wu, and H. Zhang, "Image segmentation by generalized hierarchical fuzzy C-means algorithm," *Journal of Intelligent and Fuzzy Systems*, vol. 28, no. 2, pp. 961–973, 2015.

[18] X. Chu, J. Morcos, I. F. Ilyas et al., "Katara: a data cleaning system powered by knowledge bases and crowdsourcing," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1247–1261, 2015.

[19] B. Gu, X. Sun, and V. S. Sheng, "Structural minimax probability machine," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[20] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux, "ZenCrowd: Leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking," in *21st Annual Conference on World Wide Web, WWW'12*, pp. 469–478, France, April 2012.

[21] C. Gokhale, S. Das, A. Doan et al., "Corleone: hands-off crowdsourcing for entity matching," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 601–612, 2014.

[22] C. Sarasua, E. Simperl, and N. F. Noy, "Crowdmap: Crowdsourcing ontology alignment with microtasks," in *Proceedings of International Semantic Web Conference*, pp. 525–541, Springer, 2012.

[23] J. Fan, M. Lu, B. C. Ooi, W.-C. Tan, and M. Zhang, "A hybrid machine-crowdsourcing system for matching web tables," in *Proceedings of 30th IEEE International Conference on Data Engineering, ICDE 2014*, pp. 976–987, USA, April 2014.

[24] Z. Xia, X. Wang, X. Sun, Q. Liu, and N. Xiong, "Steganalysis of LSB matching using differences between nonadjacent pixels," *Multimedia Tools and Applications*, vol. 75, no. 4, pp. 1947–1962, 2014.

[25] Z. Xia, X. Wang, X. Sun, and B. Wang, "Steganalysis of least significant bit matching using multi-order differences," *Security and Communication Networks*, vol. 7, no. 8, pp. 1283–1291, 2014.

[26] S. K. Kondreddi, G. Weikum, and P. Triantafillou, "Human computing games for knowledge acquisition," in *Proceedings of the 22nd ACM international conference on Conference on information and knowledge management*, pp. 2513–2516, 2013.

[27] S. K. Kondreddi, P. Triantafillou, and G. Weikum, "Combining information extraction and human computing for crowdsourced knowledge acquisition," in *2014 IEEE 30th International Conference on Data Engineering*, pp. 988–999, 2014.

[28] B. Gu, V. S. Sheng, Z. Wang, D. Ho, S. Osman, and S. Li, "Incremental learning for ν-support vector regression," *Neural Networks*, vol. 67, pp. 140–150, 2015.

[29] B. Gu, V. S. Sheng, K. Y. Tay, W. Romano, and S. Li, "Incremental support vector learning for ordinal regression," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 7, pp. 1403–1416, 2015.

[30] S. Felsner, V. Raghavan, and J. Spinrad, "Recognition algorithms for orders of small width and graphs of small Dilworth number," *Order*, vol. 20, no. 4, pp. 351–364 (2004), 2003.