CrowdTC: Crowdsourced Taxonomy Construction

Rui Meng*, Yongxin Tong†, Lei Chen*, Caleb Chen Cao*
*Department of Computer Science and Engineering, HKUST, Hong Kong SAR, China
†SKLSDE Lab and IRI, Beihang University, China
*{rmeng,leichen,caochen}@cse.ust.hk †yxtong@buaa.edu.cn

***Abstract*—Recently, taxonomy has attracted much attention. Both automatic construction solutions and human-based computation approaches have been proposed. The automatic methods suffer from the problem of either low precision or low recall and human computation, on the other hand, is not suitable for large scale tasks. Motivated by the shortcomings of both approaches, we present a hybrid framework, which combines the power of machine-based approaches and human computation (the crowd) to construct a more complete and accurate taxonomy. Specifically, our framework consists of two steps: we first construct a complete but noisy taxonomy automatically, then crowd is introduced to adjust the entity positions in the constructed taxonomy. However, the adjustment is challenging as the budget (money) for asking the crowd is often limited. In our work, we formulate the problem of finding the optimal adjustment as an *entity selection optimization* (ESO) problem, which is proved to be NP-hard. We then propose an exact algorithm and a more efficient approximation algorithm with an approximation ratio of $\frac{1}{2}(1-\frac{1}{e})$. We conduct extensive experiments on real datasets, the results show that our hybrid approach largely improves the recall of the taxonomy with little impairment for precision.**

I. Introduction

With the advanced semantic web and Web 2.0, significant interests have been growing in using taxonomies to ease information management. Many applications have been observed to benefit from using taxonomies, such as webpage classification [1] and short-term understanding [2]. In the past few years, many works have been conducted to construct taxonomies, i.e., Freebase [3], YAGO [4], Probase [5], etc.

Existing approaches for taxonomy construction can be categorized mainly into two mainstreams: machine-based automatic construction [5], [6], [4] and human-based manual construction [7], [3]. Each approach has its own advantages and disadvantages, in terms of accuracy, efficiency and cost, respectively. Furthermore, the machine-based automatic approaches have two categories: *pattern-based* and *clustering-based*. *Pattern-based* approaches adopt syntactic patterns to extract and discover relationships, which have high accuracy if the patterns are carefully chosen. However, these approaches suffer from the sparse coverage problem since high quality patterns are rare. A. Ritter [8] tries to improve the recall by exploring coordinate relations to learn more potential "isA" patterns. However, adopting more patterns would induce noises and impair the accuracy. Therefore, pattern-based approaches often suffer from the issue of either low recall or low precision. *Clustering-based* approaches cluster terms based on semantic similarities on some quantifiable features. These approaches can have a high coverage, however, they do not have high accuracy compared to *pattern-based* approaches.

Recently, crowdsourcing has gained its popularity in various domains to handle human intrinsic tasks, such as data cleaning [9], [10], topic discovery [11], etc. Some works have been conducted to explore the power of the crowd in knowledge extraction [12], [13]. However, they either let the crowd to carry the whole burden of the taxonomy construction or target at fact extraction which employs the crowd to fill in the relationships between given entity pairs.

Based on the above discussion, we find that neither automatic techniques nor the crowd-based approaches could derive a satisfactory taxonomy. To address the challenges, we propose a hybrid machine-crowdsourcing framework to construct a taxonomy with high accuracy and coverage. Note that the coverage of our work is defined as the number of entities the taxonomy covers [5]. In our framework, we divide the taxonomy construction into two steps. The first step is to build a taxonomy automatically which aims at a high coverage. We first construct a partial taxonomy based on syntactic patterns, then incrementally cluster all extracted entities based on the semantic features. Though have a higher coverage, the enriched taxonomy has much noise which impair its precision. Therefore, in the second step, we take advantage of the crowd power to adjust the entity positions in the "complete but noisy" taxonomy, constructed in the first step.

However, it is impossible to employ the crowd to adjust every entity in the taxonomy when processing large scale corpus with limited budget. To assist the crowd and reduce the burden of adjustment, we judiciously select entities to be adjusted and give candidate positions for each selected entity. Therefore, we need to make a decision and pick the most "beneficial" entities to ask and adjust. To evaluate the benefit, we model the utility function; also, each adjustment task is associated with a cost, which is proportional to the number of human intelligence tasks (HITs) needed for the adjustment operation. Finally, the adjustment problem is formulated as an *entity selection optimization* (ESO) problem. We prove the ESO problem is NP-hard and propose an exact algorithm and an approximation algorithm subsequently.

To summarize, we have made the following contributions:

- To the best of our knowledge, we are the first to explore the hybrid framework to combine machine-crowd intelligence towards completeness in taxonomy construction
- We formulate the taxonomy adjustment problem as an *entity selection optimization* (ESO) problem and prove it

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.77

is NP-hard. Then, we design an exact solution and an approximation algorithm.
- We conduct extensive experiments to demonstrate that our approach outperforms existing automatic approach and is scalable enough for large corpus.

II. Problem Definition

Definition 1 (Hypernym/Hyponym Relation): A word or a noun phrase X is a hyponym of a word or noun phrase of Y if X is a subtype or instance of Y. It's also called an "isA" relation.

Definition 2 (Entity, Concept and Instance): Each entity corresponds to a noun or noun phrase extracted from the given corpus. These entities have hypernym/hyponym relations. An instance is such an entity that has no hyponym and a concept (also a category) is an entity that has instances or concepts as its hyponyms.

Definition 3 (Taxonomy): Given the corpus of a certain domain, denoted as DC, a taxonomy is a tree structure, denoted as $T = (N, R)$. Each node $e_i \in N$ represents an entity (a concept or an instance) extracted from DC; each directed edge $r_{ij} \in R$ represents a hypernym/hyponym relation, i.e., $r_{ij} = (e_i, e_j)$ means that e_i is the hypernym of e_j.

Definitions 1, 2 and 3 give the concepts of hypernym, entity and taxonomy. We regard the taxonomy containing all entities in N as the ***full taxonomy*** and a ***partial taxonomy*** is a tree containing only a subset of entities in N. The workflow of the hybrid human-machine framework is shown in Figure 1. The main focus of our work is on the *crowd-asssited entity adjustment*, which improves the quality of the taxonomy constructed automatically. Now we formally define the problem of *crowd-assisted taxonomy adjustment*.

Problem Statement. Given a taxonomy T_n constructed automatically from a domain-specific corpus and a crowd-sourced budget B_0, the problem of crowd-assisted taxonomy adjustment aims at improving the quality of T_n as much as possible under the given budget constraint B_0.

III. A Human-Machine Framework

A. Machine-based Taxonomy Construction

In this subsection, we focus on the machine-based taxonomy construction. The syntactic patterns are first adopted to construct an initial, partial taxonomy and the entities extracted from the corpus are added to enrich the taxonomy based on the taxonomy metric scores.

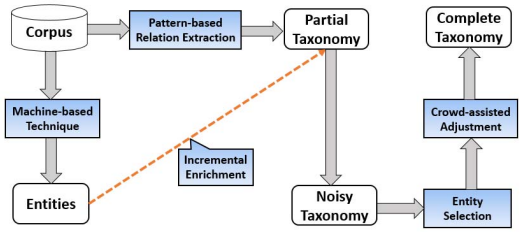


Fig. 1. Hybrid Machine-Human Workflow

Partial Taxonomy Construction. We construct the partial taxonomy based on syntactic patterns. In our work, we adopt "*Hearst Patterns*", which have been widely used and shown to have a high of precision [14].

Taxonomy Enrichment. Due to the limited coverage of syntactic patterns, we further improve the coverage of the taxonomy through enrichment procedure. We extract all entities in the corpus through NLP techniques, then use *contextual* and *co-occurrence* features to derive the taxonomy metric scores. We then incrementally add the entities into the taxonomy. A full taxonomy is constructed by adding entities one by one, which yields a series of partial taxonomies. For each entity, we need to find its optimal position in the partial taxonomy. In our work, we adopt the technique of [15], which followed the *minimum evolution tree* selection criteria to guide the position selection during enrichment process.

When adding an entity, the partial taxonomy T^{l+1} is the one that induces the least change against the previous one T^l:

$$T^{l+1} = \arg\min_{T'} \left| \mathit{Info}(T^l) - \mathit{Info}(T') \right| \tag{1}$$

$\mathit{Info}(T)$ is the information score of T, which is defined as the sum of the taxonomy metrics among all entity pairs, $\mathit{Info}(T) = \sum_{i \le j, e_i, e_j \in E} d(e_i, e_j)$. Where, $d(e_i, e_j)$ is the semantic distance between entity e_i and e_j. We use *cosine similarity* for contextual feature and *normalized Pointwise Mutual Information (nPMI)* for co-occurrence feature, the similarity scores is a weighted combination of different feature functions. Note that we convert the similarity score to distance metric through Inverse Min-Max-Normalization.

Based on the selection strategy in Equation 1, each entity can find its optimal position. By incrementally insert all entities into the initial, partial taxonomy, we construct a "complete but noisy" taxonomy.

B. Crowd-assisted Taxonomy Adjustment

1) Entity Adjustment Utility: We measure the utility of adjusting an entity as the linear combination of information gain for entity position decision and the expected amendment entity numbers.

Entity Position Uncertainty. In the enrichment procedure, each entity can have a set of candidate positions, the normalized similarity score can be considered as a distribution among all the candidates. The more skewed distribution, the lower uncertainty of a distribution. We adopt *Shannon Entropy* as the evaluation metric for the uncertainty, denoted as:

$$U(e_i) = -\sum_{e_j \in CL(e_i)} \frac{sim(e_i, e_j)}{Z} \log \frac{sim(e_i, e_j)}{Z} \tag{2}$$

where $sim(e_i, s_j)$ is the similarity score of e_i and e_j and Z is the normalization factor computed by $Z = \sum_{e_j \in CL(e_i)} sim(e_i, e_j)$. After adjustment, the correct position is fixed and therefore the entity position uncertainty is reduced to 0, i.e. $U'(e) = 0$. The *information gain* of the candidates equals to the delta of uncertainty which is exactly the *entity position uncertainty*, i.e., $\triangle U(e_i) = U(e_i) - U'(e_i) = U(e_i)$.

Position Amendment Benefit. For adjustment operation, we adopt the *subtree-based* adjustment, which means that if e_i is moved from p_a to p_b, then all nodes in the subtree rooted at e_i will be moved simultaneously. That is, adjusting an entity is equivalent to adjusting the subtree rooted at that entity, denoted as ST_{R_i}. As the automatic taxonomy induction techniques incrementally add new entities into the partial taxonomy, an error position would affect the decision of subsequent entities. Furthermore, due to the hierarchical structure of taxonomy, if an entity e_i is in the wrong position (e.g., the "isA" relationship between e_i and its parent $Pa(e_i)$ is wrong), then all the descendants of e_i are misled and therefore should also be adjusted. As the taxonomies usually have shallow depth, e.g. the maximum and average level of Probase are 7 and 1.086 and the maximum and average level of Freebase are both 1. Therefore, in our work, we only consider the effect on the selected entity and its direct descendants.

Consider adjusting an entity e_i. If the position of e_i is moved from a wrong position to an appropriate position, then we regard the position of this entity as being improved. There are two cases for the adjustment: 1) if the position is unchanged according to the crowd answer, no improvement of the position; 2) if the position of e_i is changed, position of e_i is improved, and the position improvement of its children, e.g, e_j, is the probability that "isA" relationship between e_i and e_j is correct. We use expected number of improved entities to define the *position amendment benefit*. Given an entity set E, the adjustment of each entity will benefit itself and its children, which are not in the selection set, in probability, denoted as:

$$B_E(e_i) = P_{change}(e_i)[1 + \sum_{e_j \in Child(e_i) \setminus E} Pr(e_i, e_j)] \quad (3)$$

where, $Pr(e_i, e_j)$ is the probability that the "isA" relation between e_i and e_j is correct, $Pr(e_i, e_j) = \frac{sim(e_i, e_j)}{Z}$, where $sim(e_i, e_j)$ and Z are similar to those in Equation 2 and $P_{change}(e_i) = 1 - P_r(e_i, Pa(e_i))$. Based on the Equation 3, we have the aggregated benefit of the selected entity set E:

$$B(E) = \sum_{e_i \in E} B_E(e_i) \quad (4)$$

The utility of adjusting an entity should take both the *entity position uncertainty* and *position amendment benefit* into consideration. We model the utility of each entity as follows:

$$Uti_E(e_i) = \lambda \cdot U(e_i) + (1 - \lambda) \cdot B_E(e_i) \quad (5)$$

where $\lambda \in (0, 1)$ is a parameter used to balance of $U(e_i)$ and $B_E(e_i)$. Similar with the aggregated position amendment benefit, the aggregated utility of a given set is:

$$Uti(E) = \sum_{e_i \in E} Uti_E(e_i) \quad (6)$$

2) Cost Modelling: For each adjustment task, the crowd needs to pick the "best" hypernym from the candidate list, known as the *Max Discovery Problem* [16].

It is not practical to show the crowd worker a large number of entities, we need to decompose the task into a number of sub-tasks, each of which has the cardinality less or equal to HIT size limit s, and then aggregate the results. Therefore, different entity adjustment tasks could be decomposed into various numbers of sub-tasks which leads to different costs.

As the taxonomy is a tree structure, each entity (node) in the taxonomy has one parent, therefore, we need a "single-winner" among the candidate entities. We adopt the *plurality rule* as our voting strategy. For each sub-task t, we ask k workers to vote for the best one, the entity with maximum votes is selected. For decomposing and aggregating algorithm, we adopt *Tournament Max Algorithm* [16] to get the optimal single winner.

Cost Function. For the entity e, with the hypernym candidate list CL of size N, the cost of conducting *entity adjustment task* is proportional to the total number of HITs. For obtaining the single winner, the number of HITs needed is:

$$N_h(e) = k \cdot \sum_{i=1}^{m} \lceil \frac{N}{s^i} \rceil \quad (7)$$

where, m satisfies $s^{(m-1)} \leq N \wedge s^m \geq N$, s is the micro-task size threshold and k is the replication factor. Based on the HIT numbers, given each HIT price is p, then we can have the cost of each entity-adjustment:

$$cost(e) = p \cdot N_h(e) \quad (8)$$

3) Entity Selection Optimization: According to utility and cost models, which entities should be chosen to ask the crowd can be considered as the following *Entity Selection Optimization* problem.

Definition 4 (Entity Selection Optimization (ESO)): Given a complete but noisy taxonomy $T_n = (N, R)$, where N denotes the entities, R denotes the hypernym/hypomyn relations and a specific budget B_0, this problem is to select a set of entities E to adjust under the given budget B_0 so that the utility function is maximized.

$$\begin{aligned} max \quad & \textit{Uti(E)} \\ s.t. \quad & \sum_{i=1}^{m} cost(e_i) \leq B_0 \end{aligned} \quad (9)$$

where, *Uti(E)* is defined in Equation 6 and E is the subset of N that we picked for conducting entity adjustment tasks.

The ESO problem is a *NP-hard* problem, which can be proved by a reduction from *Knapsack problem*.

Theorem 1: The *Entity Selection Optimization* (ESO) problem is NP-hard.

The proof could be found in the full version of this paper [17].

IV. Solutions of Entity Selection Optimization

Since the ESO problem is NP-hard, we propose an exact algorithm as well as a more efficient approximation algorithm.

A. Exact Algorithm

Brute Force Algorithm. The straightforward approach for the problem is to enumerate all the subsets of N in brute force. There are totally 2^N possible combinations for entity selection. For each combination, we need to compute the cost and utility, the complexity is $O(N \cdot p)$, where p is the maximum number of children among all entities in N, therefore the time complexity of this algorithm is $O(p \cdot N \cdot 2^N)$.

Dynamic Programming (DP) Algorithm. To solve the problem of selecting optimal subset from $N = \{e_1, e_2, \cdots, e_n\}$, we decompose the given instance into smaller sub-problems and obtain the solution recursively. Our problem differs from the traditional knapsack problems in that the utility of each entity changes with each choice decision. Here, we use $Uti_{D_i}(i, b)$ to denote the optimal utility value of first i entities, with the budget less than or equal to b and set D_i, where the set $D_i = Child(e_i) \cap E_i$. Firstly, we number the entities by post-order traversal which ensures if one entity's number is i then all its children's number is less than i. We use E_i to denote the selection result of first i entities. When considering entity e_{i+1}, its children entities have already been processed, which means D_i is fixed. For the optimal solution of first $(i+1)$ entities, we consider all subsets of first i entities, which is equivalent to all subsets of entity i's children set. For each situation, we can get the current selection result with first i entities, i.e., E_i, and D_i, then the utility of $(i+1)^{th}$ entity can be computed. The recursive step is defined as follows:

$$Uti_{D_{i+1}}(i+1, b) = \max_{\forall D_i : D_i \cap Child(e_{i+1}) \subseteq D_{i+1}} \begin{cases} Uti_{D_i}(i, b) & \text{if } e_{i+1} \notin D_{i+1} \\ Uti_{D_i}(i, b - cost(i)) + Uti_{D_i}(e_i) & \text{if } e_{i+1} \in D_{i+1} \end{cases} \quad (10)$$

The initial settings are:

$$Uti_{D_1}(1, b) = \begin{cases} Uti_{\emptyset}(e_1) & \text{if } D_1 = \{e_1\} \wedge b \geq cost(e_1) \\ 0 & \text{if } D_1 \wedge 0 \leq b \leq B_0 \\ \infty & \text{others(illegal)} \end{cases} \quad (11)$$

Note that to get the selection result, we use $Selection_{D_i}(i+1, b)$ to record the selection strategy of $(i+1)^{th}$ entity under the budget of b and set D_i. And the selection result of different situations can be obtained recursively, the algorithm for computing the selection result is shown in Algorithm 2 and the pseudocode of the DP algorithm is shown in Algorithm 1 .

Complexity. As shown in Algorithm 1, line 2 has N loops, line 3 has B_0 loops and line 5 has 2^p loops for each subset of children set. The complexity in line 5 and lines 6∼ 11 are $O(N)$, respectively. Therefore the total time complexity is $O(N^2 \cdot B_0 \cdot 2^p)$. In real cases, $p \ll N$, therefore, the DP algorithm is more efficient than the brute force algorithm. The space complexity of algorithm 1 is $O(N \cdot B_0 \cdot 2^p)$.

B. Approximation Algorithm

Although, the DP gives optimal solution to the *ESO* problem, the space complexity is $O(N \cdot B_0 \cdot 2^p)$, which is intractable, especially when the n is large in practice. Therefore we explore an approximation algorithm which significantly enhances the efficiency and has the approximation guarantee.

Algorithm 1: Dynamic Programming Algorithm (DP)

Input: A set of entities $\{e_1, e_2, \cdots, e_n\}$, a given budget B_0
Output: Maximum utility value and corresponding selection set E

```
1  Initialize Uti_{D_1}(1, b) according to Equation 11;
2  for i=0 to N − 1 do
3      for b=0 to B_0 do
4          S_i ← FindSelection(i, b, Uti) ;
5          for each all D_i ⊂ Child(e_i) ∩ S_i do
6              if cost(i + 1) ≤ b and
               Uti_{D_i}(e_{i+1})+Uti_{D_i}(i, b − cost(i + 1)) ≥ Uti_{D_i}(i, b) then
7                  Uti_D(i, b) = Uti_D(e_i)+Uti_D(i, b − cost(i))
                   D_{i+1} ← D_i ∩ Child(e_{i+1}) ∪ e_{i+1};
8                  Selection_{D_{i+1}}(i + 1, b) = true;
9              else
10                 D_{i+1} ← D_i ∩ Child(e_{i+1});
11                 Uti_{D_{i+1}}(i + 1, b) = Uti_{D_i}(i, b);
12                 Selection_{D_{i+1}}(i + 1, b) = false;
13 E = FindSelection(N − 1, B_0, Uti);
14 return E
```

Algorithm 2: FindSelection

Input: Entity index i,Budget b, Utility result $Utility$
Output: Selection set E

```
1  Initialize: E ← ∅;
2  while i ≥ 1 ∧ b ≥ 0 do
3      for all D_i do
4          if Uti_{D_i}(i, b) = max Uti(i, b) ∧ Selection_{D_i}(i, b) == true
           then
5              E ← E ∪ e_i;
6              b ← b − cost(e_i);
7      i ← i − 1;
8  return E;
```

Theorem 2: The utility function defined is monotone and submodular.

The proof could be found in the full version of this paper [17].

Based on the Theorem 2, the greedy algorithm is shown in Algorithm 3. Starting from an empty set, each time we select the entity which gives best utility increase for each unit.

Lemma 1: The greedy algorithm achieves an approximation factor of $\frac{1}{2}(1 - \frac{1}{e})$.

Proof: According to Theorem 2, the utility function of *ESO* problem is monotone and submodular, the greedy algorithm in Algorithm 3 has the approximation ratio of $\frac{1}{2} \cdot (1 - \frac{1}{e})$ as shown in Theorem 3 of [18]. ■

Complexity. The greedy algorithm in Algorithm 3 takes $O(N^2)$, the while loop has N iterations and in each iteration (lines 4-9) takes $O(N)$ time to select the s^*. (Note that the computation of $\triangle_U(S_1, e)$ is $O(p)$, where $p \ll N$).

V. EXPERIMENTS

A. Experimental Setup

Dataset. We conduct experiments on a real dataset of a corpus on Computer domain, which is a collection of academic paper abstracts, crawled from Springer [1]. The dataset is pre-

[1] http://www.cse.ust.hk/~rmeng/CrowdTC/Ex.zip

Algorithm 3: Approximation Algorithm by Greedy Selection

Input: A set of entities $E = \{e_1, e_2, \cdots, e_n\}$, a given budget B_0
Output: Maximum utility value and corresponding selection set S

$S_1 \leftarrow \emptyset$; $L \leftarrow E$;
$S_2 \leftarrow arg \max_{\forall e_i \in E} Uti_{\emptyset}(e_i)$;
while $L \neq \emptyset$ **do**
  for *each entity* $e \in L$ **do**
    $\triangle_U(S_1, e) \leftarrow U(S_1 \cup e) - U(S_1)$;
    $g(e) \leftarrow \frac{\triangle_U(S_1,e)}{cost(e)}$;
  $s^* \leftarrow arg \max_{e \in L} g(e)$;
  if $cost(S_1) + cost(s^*) \leq B_0$ **then**
    $S_1 \leftarrow S_1 \cup s^*$;
  $L \leftarrow L \backslash s^*$;
return $arg \max_{S \in \{S_1, S_2\}} Uti(S)$

TABLE I
STATISTICS OF TAXONOMY

Taxonomy	# entities	Avg # of child	Max # of child
Partial Taxonomy	10788	1.29	677
Enriched Taxonomy	34420	1.09	689
Sampled Taxonomy	27635	0.85	5

processed in several steps: first, we adopt syntactic based technique on the corpus and construct a partial taxonomy, among which only 3.38% sentences are observed to have *"Hearst Patterns"*; second, we extract all entities (noun phrases) from corpus as the candidates for subsequent enrichment. For entity extraction, we use NLTK Chunker.

The statistics of the partial taxonomy show that the maximum number of children is 677. The DP algorithm is intractable due to the limited memory. Therefore, to illustrate the efficiency and effectiveness of DP algorithm, we sample a smaller taxonomy ($P = 5$) from the enriched taxonomy. The detail information of the partial taxonomy and the sampled taxonomy is shown in Table I.

Evaluation Metric. For the taxonomy evaluation, we adopt *precision* and *coverage* as the evaluation metric:

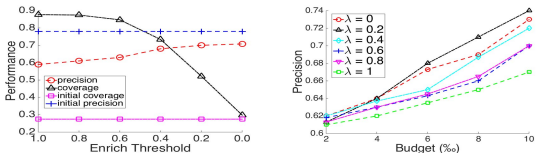
$$Precision = \frac{\#of\ correct\ pairs}{\#of\ totally\ extracted\ pairs} \quad (12)$$

$$Coverage = \frac{\#of\ entitiesinthetaxonomy}{\#of\ entities\ extracted\ from\ corpus} \quad (13)$$

As we do not have the ground truth for the taxonomy, we use coverage metric as the indicator of recall. For precision, we randomly sample 100 "ancestor-descendant" pairs and check the correctness manually.

Parameters. There are several parameters needs evaluation: weight parameter λ, budget B_0 and distance filter parameter θ for candidate filtering. For λ and θ, we vary it from 0$\sim$1; for B_0, we compute the total cost for adjusting every entity in the taxonomy, $B_{total} = \$69,011$ and vary the cost budget from 2‰$\times B_{total}$ to 10‰$\times B_{total}$.

Crowdsourcing on AMT. We use Amazon Mechanical Turk (AMT) to conduct crowdsourcing tasks. For each HIT, we design it as a single choice question and ask the worker to select the most appropriate hypernym of given word from several choices. The maximum number of choices of each HIT is set to 10 (the maximum number of choices on AMT for single-choice questions). For those entities which have a candidate list size larger than 10, we separate the candidates into several groups, size of each does not exceeds 10. For these entities, the process of collecting answers has several rounds. In each round, we first separate the candidates into groups and generate a HIT for each group, then we collect the answers and aggregate the answers to produce a new candidate list, which is used to generate HITs for the next round. The collecting process will finish until we get the final answer. We pay \$0.01 to each worker for a HIT. To make sure the quality of answers, we assign each HIT to three workers and aggregate their answers by majority voting strategy.



(a) Evaluation of θ (b) Evaluation of λ

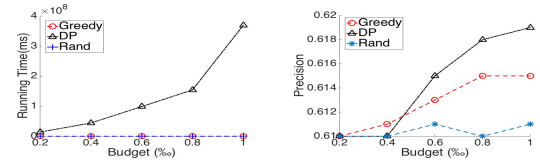
Fig. 2. Parameter Evaluation

B. Evaluation of Techniques

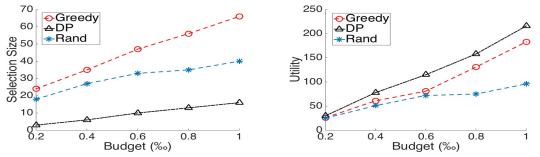
Parameter Setting. By decreasing the θ value, the filtering power is increasing (more entities have no candidate). We compare the coverage and accuracy of the enriched taxonomy with various θ values, as shown in Figure 2(a). For the weight parameter of utility function in the Equation 6, we empirically learn the value of λ by varying it between 0$\sim$ 1. For each value of λ, we examine the accuracy under different budget settings. The results are shown in Figure 2(b). According to the results, we set $\theta = 0.8$ to have a high coverage and $\lambda = 0.2$.

Comparison Adjustment Algorithms. We compare the DP algorithm and greedy algorithm with the naive random selection strategy. In the naive algorithm, we select the entity that does not exceed the remaining budget randomly. As the DP algorithm has a space complexity of $O(N \cdot B_0 \cdot 2^p)$, we only test it on the sampled taxonomy, due to the memory constraint. We vary the budget and examine the running time of entity selection, selection utility value, selection entity number and the precision of adjusted taxonomy based on the answers from crowd, the results are shown in Figures 3, 4. Due to the memory limit, for the DP part, we use a small budget range to conduct the comparison, 0.2‰$\sim$ 1.0‰.

In Figure 3, we can see that the DP algorithm achieves better accuracy compared with the greedy selection algorithm. However, the DP solution is much more cost than the greedy one. Also, the memory cost is not scalability for large datasets. From Figure 3 and Figure 4, we can see that the greedy algorithm outperforms the naive random selection algorithm in both accuracy, adjusted entity number and utility value. From the comparisons, we conclude that the taxonomy constructed by our method which combines machine techniques with crowdsourcing techniques achieves the precision nearly 75% and coverage 87%. Taxonomy constructed by our approach improves the accuracy of the noisy enriched taxonomy (61%)

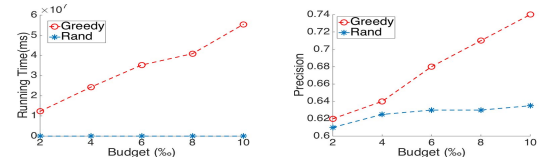


(a) Running Time (b) Precision

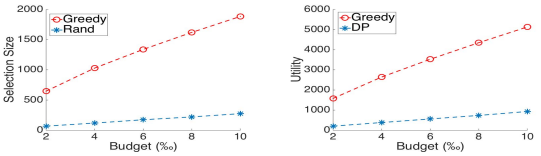


(c) Selection Size (d) Utility

Fig. 3. Comparison of Algorithms on Sampled Taxonomy



(a) Running Time (b) Precision



(c) Selection Size (d) Utility

Fig. 4. Comparison of Algorithms on Real Taxonomy

and have a much larger coverage compared with the initial partial taxonomy (27%).

VI. Related Work

Many study have been conducted on taxonomy construction, either manually or automatically. The manual approaches, which construct taxonomies by domain experts or collaboratively by community members, e.g. Freebase [3], has the limitation of scalability. The automatic approaches [5], [4], [19] construct taxonomy based on syntactic patterns; the works [20], [6] build taxonomy via hierarchical classification or incremental clustering approaches.

Recently, the increasing popularity of crowdsourcing brings new trends to leverage the power of crowd in taxonomy construction. In [12], [21], the crowd is used for categorizing items for taxonomy construction. In [7], each "isA" relationship is voted by the crowd and take it as the input for taxonomy induction. S. K. Kondreddi [13] proposes a hybrid approach that combines information extraction technique with human computation for knowledge acquisition, in which the crowd are asked to compile relationships between entities.

VII. Conclusion

In this paper, we propose a hybrid framework which combines the power of automatic machine-based approaches and human computation (the crowd) to construct a complete and accurate taxonomy. In this work, the core problem is formulated as the *Entity Selection Optimization (ESO)* problem, which is proven to be NP-hard. To solve this optimization problem, we propose an exact algorithm and a more efficient approximation algorithm with a $\frac{1}{2}(1-\frac{1}{e})$ approximation factor. Finally, we have verified our proposed algorithms through extensive experimental studies.

VIII. Acknowledgement

This work is supported in part by the Hong Kong RGC Project N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2014CB340303 , NSFC Grant No. 61328202/61502021, NSFC Guang Dong Grant No. U1301253, Microsoft Research Asia Gift Grant, Google Faculty Award 2013 and Microsoft Research Asia Fellowship 2012.

References

[1] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma, "Support vector machines classification with a very large-scale taxonomy," *SIGKDD*, vol. 7, 2005.
[2] Y. Song, H. Wang, Z. Wang, H. Li, and W. Chen, "Short text conceptualization using a probabilistic knowledgebase," in *IJCAI*, 2011.
[3] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD*, 2008.
[4] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW*, 2007.
[5] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," in *SIGMOD*, 2012.
[6] X. Liu, Y. Song, S. Liu, and H. Wang, "Automatic taxonomy construction from keywords," in *SIGKDD*, 2012.
[7] D. Karampinas and P. Triantafillou, "Crowdsourcing taxonomies," in *ESWC*, 2012.
[8] A. Ritter, S. Soderland, and O. Etzioni, "What is this, anyway: Automatic hypernym discovery." in *AAAI*, 2009.
[9] Y. Tong, C. C. Cao, C. J. Zhang, Y. Li, and L. Chen, "Crowdcleaner: Data cleaning for multi-version data on the web via crowdsourcing," in *ICDE*, 2014.
[10] C. J. Zhang, L. Chen, Y. Tong, and Z. Liu, "Cleaning uncertain data with a noisy crowd," in *ICDE*, 2015.
[11] Y. Tong, C. C. Cao, and L. Chen, "Tcs: efficient topic discovery over crowd-oriented service data," in *SIGKDD*, 2014.
[12] L. B. Chilton, G. Little, D. Edge, D. S. Weld, and J. A. Landay, "Cascade: crowdsourcing taxonomy creation," in *CHI*, 2013.
[13] S. K. Kondreddi, P. Triantafillou, and G. Weikum, "Combining information extraction and human computing for crowdsourced knowledge acquisition," in *ICDE*, 2014.
[14] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *COLING*, 1992.
[15] H. Yang and J. Callan, "A metric-based framework for automatic taxonomy induction," in *ACL*, 2009.
[16] P. Venetis, H. Garcia-Molina, K. Huang, and N. Polyzotis, "Max algorithms in crowdsourcing environments," in *WWW*, 2012.
[17] R. Meng, Y. Tong, L. Chen, and C. C. Cao. (2015) Crowdtc:crowdsourced taxonomy construction. [Online]. Available: http://www.cse.ust.hk/~rmeng/CrowdTC/CrowdTC-TechReport.pdf
[18] S. Khuller, A. Moss, and J. Naor, "The budgeted maximum coverage problem," *Inf. Process. Lett.*, vol. 70, 1999.
[19] N. Nakashole, G. Weikum, and F. Suchanek, "Patty: a taxonomy of relational patterns with semantic types," in *EMNLP*, 2012.
[20] K. Punera, S. Rajan, and J. Ghosh, "Automatically learning document taxonomies for hierarchical classification," in *WWW*, 2005.
[21] J. Bragg, D. S. Weld *et al.*, "Crowdsourcing multi-label classification for taxonomy creation," in *HCOMP*, 2013.