ACTraversal: Ranking Crowdsourced Commonsense Assertions and Certifications

Tao-Hsuan Chang, Yen-Ling Kuo, and Jane Yung-jen Hsu

Department of Computer Science and Information Engineering
National Taiwan University
{doudi.tw,a33kuo}@gmail.com, yjhsu@csie.ntu.edu.tw

Abstract. Building commonsense knowledge bases is a challenging undertaking. While we have witnessed the successful collection of large amounts of commonsense knowledge by either automatic text mining or *games with a purpose* (GWAP), such data are of limited precision. Verifying data is typically done with repetition, which works better for very large data sets. Our research proposes a novel approach to data verification by coupling multiple data collection methods. This paper presents *ACTraversal*, a graph traversal algorithm for ranking data collected from GWAP and text mining. Experiments on aggregating data from two GWAPs, i.e. Virtual Pets and Top10, with two text mining tools, i.e. SEAL and Google Distance, showed significant improvements.

1 Introduction

Codifying several million pieces of commonsense knowledge into machine usable forms has proved to be time-consuming and expensive. In 1984, a team of knowledge engineers [6] started to craft the Cyc knowledge base using CycL, a logic-based language. Extreme care was taken to ensure its correctness. In contrast, Open Mind Common Sense (OMCS) [8] took the Web 2.0 approach by collecting voluntary contributions of commonsense sentences from online users. In recent years, with the advances in human computation, large amounts of data can be crowdsourced or mined from the web efficiently. Despite improvements in the efficiency of knowledge acquisition, those methods are limited in their precision.

Several methods have been developed to guarantee the precision of crowdsouced knowledge. Frequency is the most used approach to filtering out noisy data in crowdsourcing, which takes advantage of the huge amount of user input for data verification. Mechanism design is another approach to eliciting correct answers from players of GWAPs. For example, Verbosity [1] uses sequential verification to verify answers, and Virtual Pets [5] improves the quality of answers by punishinig malicious behaviours in community interaction.

Unfortunately, these mechanisms may not work in practice. The data collected are subject to pollution due to tricks by players. Speer et al. [9] found that half of the data collected by Verbosity should be rejected by OMCS since they are only sound-alike or look-alike clues used for guessing the answers. In their experiment, half of the data were filtered, yet the remaining data were still had lower quality than the assertions in OMCS. In a single GWAP, players may learn tricks from experiences, using the tricks to get game points but contribute little to the knowledge base. Even the mechanism is well designed, the quantity and quality may be limited due to those tricks.

In order to improve the precision without modifying the original GWAP, one can develop a separate game for data verification [3]. Meanwhile, text mining methods has been proved it is useful to learning sentences from the web[2]. By coupling GWAP or text mining methods, the precision may also be improved. Nevertheless, the existing text mining depends on properly defined seeds as training examples and templates, while GWAP tend to produce redundant data over time. As a result, aggregating both GWAP and text mining are better than coupling only one type of methods for building large commonsense knowledge bases.

This paper proposes an approach, *ACTraversal*, to improving the precision of commonsense knowledge collected by aggregating GWAP with text mining components. The proposed *ACTraversal* is a universal graph traversal aggregation for ranking common sense assertions and certifications.The assertions and certifications of commonsense knowledge are defined as below:

- Assertions: Sentences composited by subject-relation-object triples.
- Certifications: Evidences indicating partial-order of associated assertions confidence level. Assertion with higher confidence level is associated with higher order certification. Every assertion is associated with at least one certification.

In the following sections, we first compare the pros and cons of existing verification methods of GWAP and text mining techniques, and introduce the proposed ACTraversal. We then present our prototype implementation, followed by the experiments designed to show that

- ACTraversal can improve quality of the partial order ranking produced by a single component,
- ACTraversal can aggregate data from multiple components, and
- ACTraversal is efficient on large dataset.

2 Approaches to Knowledge Verification

This section presents an overview of approaches to knowledge verification and discusses their pros and cons to show the advantages in coupling the approaches.

Games with a Purpose. Games With A Purpose (GWAP) [10] utilizes computer games to gather players and guides them to perform tasks. The quality of collected sentences is guaranteed via the mechanism of the game. It has demonstrated its efficiency in commonsense knowledge collection. For example, Verbosity [1] and Virtual Pets [5] have collected over a million English and Chinese commonsense sentences respectively within a year with acceptable precision.

Verbosity is a two-player game in which the Narrator gives clues to help the Guesser figure out a secret word. The clues are collected as commonsense knowledge about the corresponding secret word. Virtual Pets utilizes the interactions in communities to collect knowledge via question-answering between online users. Players in Virtual Pets answer questions such as "Spoon is used for ___?" in exchange for food for their pets. Both games use frequency as filters to verify the assertions. This approach succeeded in building useful knowledge bases within a relatively short time and with extremely low costs.

However, previous research found that the precision of answer is not perfect. The precision can be further improved by coupling with a verification game [3]. For example, Top10, a FamilyFeud-like verification game, can be used for verifying the concepts of an assertion collected by Virtual Pets. The precision of assertions collected by Top10 and Virtual Pets is significantly higher than the assertions collected by only a single game. This approach shows the opportunity to aggregate GWAP to gain precision.

Text Mining. Some types of general knowledge, such as categorical relations, can be extracted automatically [7,4,2] from a corpus. The data quality is ensured via text mining or machine learning techniques. Previous work has demonstrated that pattern-based and list-based extraction methods can be combined to achieve significant improvements in recall [4]. Never-Ending Language Learner (NELL) [2] also showed that it is possible to collect sentences with high precision and improve the learning process itself by using coupled machine learning components. This approach shows the opportunity to couple text mining to gain precision.

However, these coupled methods are limited to their constraints. While text mining is restricted on certain knowledge domains, GWAP produces fewer assertions. By coupling GWAP and text mining methods, the data can be further verified. In addition, aggregation of the non-overlapping data can also improve the coverage of knowledge base. With larger coverage, the aggregation approach can verify more assertions.

3 ACTraversal

To get the best of both worlds, we propose ACTraversal (ACT); a universal ranking aggregation by graph traversal on assertions and certifications. By utilizing partial-order of certifications, ACT ranks assertions from multiple components, constructing a total-order of all assertions. This section introduces the data structure, graph, assumptions, and the algorithms used in ACTraversal.

3.1 Data Structure

The proposed ACT uses assertions and their certifications from multiple components.

Assertions. Commonsense knowledge is comprised of assertions, which are defined as "subject-relation-object" triples in this paper. For example, the corresponding assertion for "dog is an animal" is "dog-`IsA`-animal". Both subject and object of an assertion are concepts, which are represented in plain text. For the sake of simplicity, we assume that all the texts have only one sense since the concepts in the same text but different senses can be stored as different instance in this system. The relation of an assertion comes from a predefined relation set which can be incrementally enlarged. Also, the relations are directed edges connecting two concepts. For example, "Subject-`IsA`-Object" and "Object-`IsA`-Subject" are regard as different assertions. If every corresponding element in two triples are the same, the system treats them as the same assertions.

Certifications. Assertion with higher confidence level is associated with higher order certification. Every assertion is associated with at least one certification. Certifications are the evidences indicating partial-order of associated assertions confidence level. Each assertion may associate with multiple certifications from different sources (i.e. GWAP and text mining components.) For example, both "2 players answered dog-`IsA`-animal in Verbosity" and "NELL learned dog-`IsA`-animal with 100.0% confidence" are valid certifications for "dog-`IsA`-animal". Since each component returns assertions with their certifications, every assertion must associate with at least one certification. In addition, an assertion can also associate with multiple certifications from a single source. For example, "UserA vote for dog-`IsA`-animal on OMCS website" and "UserB vote for dog-`IsA`-animal on OMCS website" are all valid certifications. Multiple certifications in a single component can be aggregated into a single certification. For example, the above certification can be aggregated as "2 users vote for dog-`IsA`-animal on OMCS website'. By doing so, it reduces the computational complexity and hide personal information.

AC Graph. The graph of assertions and certifications (i.e. the AC graph) in ACT has two types of nodes and two types of edges. The nodes are *Assertion* and *Certification*. Each distinct assertion and certification corresponds to an *Assertion* node and *Certification* node, respectively. The edge type between *Assertion* and *Certification* is *Cross*. The edge type between *Certifications* is *Order*. The *Cross* edge is bidirectional, and the *Order* edge is unidirectional. There is no edge between *Assertion* nodes. Each node has a ranking score, and each edge has a weight. Weight of *Cross* edge is the confidence score of its source component. For component with higher confidence score, its associated certifications are more trustworthy. Weight of *Order* edge is set to a uniform score since it only indicates the direction of traversal. Figure 1 shows a sample graph containing 3 assertions and 3 certifications from 2 components.

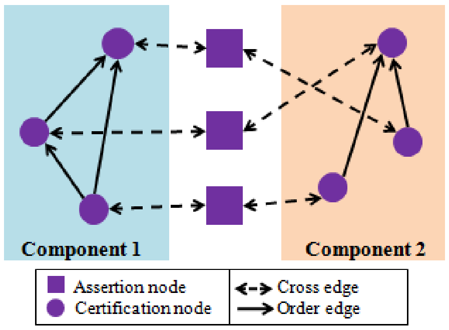


Fig. 1. A sample AC graph

Assumptions. Two assumptions are made in using the assertions and certifications:

- The ranking scores of assertions are positively correlated to ranking score of associated certifications.

– The certifications of a single component are in a partial-order indicating the confidence level of associated assertions.

The first assumption means a good assertion has a strong certification, and the strength of certification depends on the assertion it attests to. The second assumption implies that the confidence scores of assertions are comparable through certifications. For example, “2 users vote for dog-`IsA`-animal on OMCS website’ has higher confidence score than “1 users vote for dog-`IsA`-pet on OMCS website’. However, “2 users vote for and 1 user votes against bird-`CapableOf`-fly on OMCS website’ is not comparable with “1 users vote for bird-`IsA`-creature on OMCS website.’ These assumptions hold for components that use frequency as filter to verify data. Therefore, GWAP that use frequency of answers/votes and text mining components that output assertions with probabilities/rankings can be formalized as components of ACTraversal.

3.2 ACTraversal Algorithm

ACTraversal (ACT) is a graph traversal algorithm to rank assertions and certifications on a weighted directed graph. It takes the output assertions and certification of GWAP or text mining components as input. The confidence score of each component is predefined according to the result of previous study. The output of ACT is a ranking list of assertions and certifications with the converged scores.

AC Graph Construction. ACT builds the graph according to the input pairs of assertions and certifications. For each pair, the assertion and certification are connected by *Cross* edges. After reading all the pairs from a component, ACT compares the certifications and adds a *Order* edge from lower order (weak certification) to higher order (strong certification). For example, if every element of a 2-ary certification is greater than another 2-ary certification, its order must be higher. So, ACT adds an edge from (2,1) to (2,3), but it does not build edge between (2,1) and (1,2). The time complexity of building AC graph for a single component is $O(|Cross| + |Certification|^2)$.

Algorithm 1. $ACGraph(Components)$

```
Require: $Components$ with output pairs of assertion and certification
    {$ACG$ is a weighted directed graph}
 1: for all $C \in Components$ do
 2:     for all $assert, cert \in C.ACpairs()$ do
 3:         $ACG.addNode(assert, cert)$
 4:         $ACG.addBidirectEdge(assert, cert, C.confScore)$
 5:     end for
 6:     for all $cert1, cert2 \in ACG.Certs$,
        s.t. $cert1 < cert2$ do
 7:         $ACG.addUnidirectEdge(cert1, cert2)$
 8:     end for
 9: end for
10: return $ACG$
```

Once constructing the AC graph, ACT assigns the weights to nodes and edges. All nodes are assigned weight evenly; with sum of weight of all nodes is equal to 1. The weight of edges are assigned as defined in AC graph. Algorithm 1 describes the details of building AC graph.

AC Graph Traversal. The traversal algorithm proceed iteratively until the ranking score of node converges. In each iteration, scores of nodes are updated while the weight of edges remains the same. For each *Assertion* node, ACT assigns the weighted average of the linked *Certification* nodes to it as the new score. The weight used here is weight of *Cross* edge. For each *Certification* node, ACT collects two sources of scores, and then assign the node by averaging scores from the two sources. One source is the score of its associated *Assertion* nodes. ACT computes the weighted average scores of *Assertion* nodes as the first source. The other source is the score of lower order *Certification* nodes. ACT passes score of lower order *Certification* evenly to its linked higher order neighbors through the *Order* edges. After updates of the scores, ACT redistributes a small portion of scores to every node as random restart. In the end of each iteration, ACT normalizes the scores to make their sum equal to 1. Algorithm 2 demonstrates the details of ACTraveral. The time complexity of this algorithm is $O(iteration \times (N+E))$.

Since the *Assertion* and *Certification* are positively correlated, they can be viewed as the attributes of each other. Also, the partial order of *Certification* nodes are used for estimating authority. The ACTraversal can be considered as a process to 1) average the attributes and 2) compute the authority. With the two steps, we can estimate the rank of *Assertion* and *Certification* at the same time. Figure 2 shows the interaction result of *Certification* and *Assertion* in ACT.

Algorithm 2. $ACTraversal(Components, \lambda)$

```
Require: $Components$ with output pairs of assertion and certification
    {$ACG$ is a weighted directed graph}
 1: $ACG \leftarrow ACGraph(Components)$
 2: repeat
 3:   for all $Node \in ACG$ do
 4:     if $Node$ is $Assertion$ then
 5:       $score \leftarrow weightedAvg(neighborCert(Node))$
 6:     else if $Node$ is $Certification$ then
 7:       $score \leftarrow weightedAvg(neighborAssert(Node))$
 8:       for all $cert \in predecessorCert(Node)$ do
 9:         $score \leftarrow score + (\frac{cert.score}{cert.outDegree})$
10:       end for
11:       $score \leftarrow score/2$
12:     end if
13:     $Node.score \leftarrow score$
14:   end for
15:   $ACG.RandomRestart(\lambda)$
16:   $ACG.NormalizeScores()$
17: until score is converged
18: return $ACG.rankingScore$
```

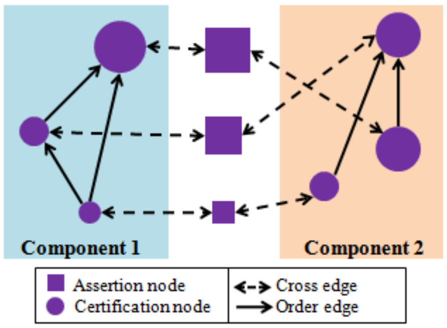


Fig. 2. ACT on sample AC graph. The size is the result weight.

4 Implementation

We evaluated ACT in terms of its quality of aggregation results and efficiency. In the following section, we will first describe the components in our implementation. Second, we will show that ACT can rank assertions in components with partial order certifications to achieve higher precision. Then we will demonstrate the quality of ranked assertions after aggregating the data from multiple components. Finally, we will compare the running time of ACT in dataset of different sizes.

4.1 Components

In this experiment, we chose 2 GWAP and 2 AutoExtraction components that can output pairs of assertion and certification. The components are VirtualPets (VP) [5], Top10 [3], SEAL [11], GoogleCount (GC)[1]. The components are chosen based on previous studies[5,3,11]. The input, output, and the formalization of certification of each component are described in the following paragraphs.

Virtual Pets. Virtual Pets (VP) is a game in which players teach common sense to their own virtual pets. It outputs assertions with play logs, which are the frequency, good rank, and bad rank. We formalized these logs as certifications in 3-ary tuple, (+frequency, +good rank, -bad rank). The partial order of certifications is set if every element in one certification is greater than or equal to another.

Top10. Top10 is a family-feud-like game in which players guess the top 10 concepts of a given question to get high scores. The input of game is a list of assertions ranked by their frequency. It outputs assertions with play logs, which are the frequency, good votes, and bad votes. We formalized these logs as certification in 3-ary tuple, (+frequency, +good votes, -bad votes). The partial order of certifications is set if every element in one certification is greater than or equal to another.

[1] Google search engine, http://www.google.com

SEAL. Set Expander for Any Language(SEAL) is an algorithm that uses common wrappers of a given set of seeds to auto extract instances from the web. For example, given "tea" and "milk", both are liquid, as input seeds of SEAL, it can extract "water". It extracts the wrapper around seeds from web pages, and then use wrapper the extract new instances. A possible extracted wrapper of the example above is "drink X.</li>", where X is the extraction. It outputs the top 100 instances retrieved using a set of seeds belonging to a given category. We formalized the assertions as "instance-IsA-category" triples. Also, we defined the mutually exclusive relations between categories. If an instance belongs to two mutually exclusive categiries, we add a flag on the assertion. The certifications can then be formalized in 2-ary tuples, (-rank, -flag), which is in partial order relationship.

GoogleCount. GoogleCount is an algorithm that computes the number of web pages Google search engine returned for a given sentence. The input assertions are represented as natural language sentences. The output of GoogleCount is the assertions with page counts. We formalized the counts as certifications. One certification is in higher order only if its associated count is strictly greater than others.

In our experiment, we evaluated the precision of top 800 instances ranked by 1) each sigle component, 2) ACT on each single component, or 3) ACT on the four components, with λ equals to 0.01. We recruited 46 graduate students to label the top 800 assertions returned by each method as ground truth. Each label is contributed by at least 3 students. A label is true only if over half of annotators vote true.

4.2 Single Component Ranking

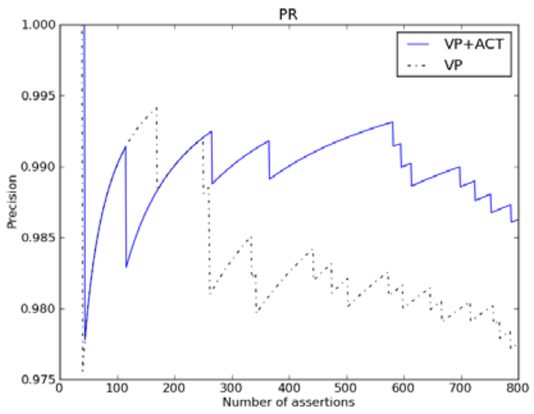
In the single component ranking, we compare ACTraversal with the serializing heuristic which sort results by frequency. This heuristic is previously used by the existing component, so we apply ACTraversal to examine the improvement. Table 1 lists the number of distinct assertions each component outputs. Figure 3 shows the precision of top 800 instances ranked by ACT on single component with single component precision as baseline. The improvement of precision in VP, Top10, and SEAL demonstrates the re-ranked lists produced by ACT are better than their original ranking lists. Therefore, ACT can be viewed as a ranking algorithm that turns partial order certifications into a better total order ranking of assertions. The result of GC is omitted in figure 3 because ACT does not produce new ranking list for a total order ranking.

Table 1. Number of assertions output by components

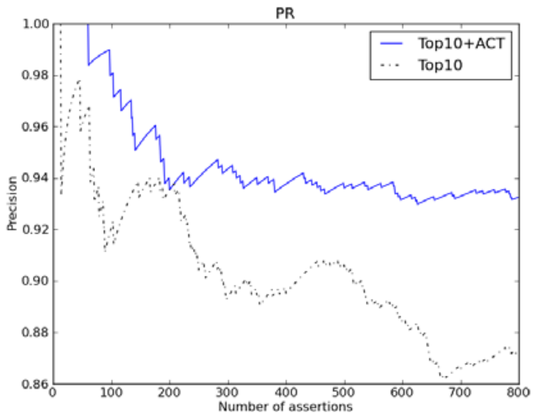
Component	VP	Top10	SEAL	GC
Number	378k	378k	2011	49989

4.3 Multiple Components Aggregation

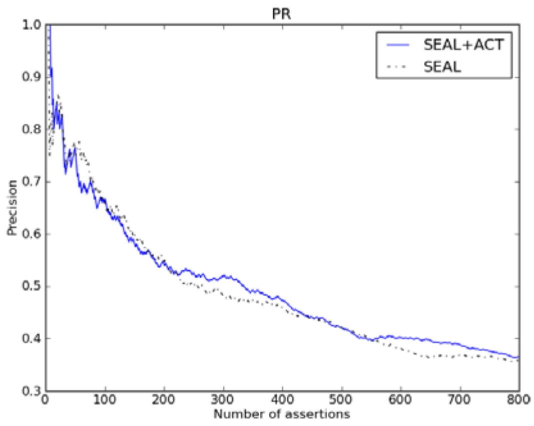
We first experimented on different confidence scores used by ACT to verify the impact of parameters. Then, we tested on dataset of different sizes to evaluate the efficiency of ACT.



(a) Precision of ACT on VP



(b) Precision of ACT on Top10



(c) Precision of ACT on SEAL

Fig. 3. Precision improvement by ranking single component with ACT

Confidence Score. We examined three possible settings of aggregation weighting parameters. The first setting uses uniform weight for all components. The other two settings are based on the precision of each component. The precision used for calculating the confidence score is the precision of top 200 assertions returned by ACT on each single component. One parameter setting uses the precision as the weight. The other parameter setting comes from the following formula, which is inverted logistic function:

$$confScore = -\ln(\frac{1}{precision} - 1) \tag{1}$$

The weights are then scaled to [0,1] by dividing the maximum weight. Table 2 lists the precision and weights used in this experiment. Figure 4 shows the precision of the three parameter settings for multiple components aggregation and ACT on each component. Compared to the uniform and precision weighted setting, inverted logistic weight has comparable high precision in aggregating both noisy and high quality components.

Table 2. Confidence Score in experiment

Component	VP	Top10	SEAL	GC
Uniform	1	1	1	1
Precision	0.99	0.935	0.525	0.745
Inverted Logistic	1.000	0.580	0.022	0.233

Table 3. Experiment on each size setting

Questions	Assertions	Certifications	Precision	Building Time	Traverse Time	Iterations
1250	138398	2007	0.94	34s	2m26s	56
2500	183628	2103	0.9525	1m2s	4m6s	58
5000	258997	2204	0.96375	1m54s	8m54s	62
10000	395612	2294	0.9725	3m10s	11m18s	64
20000	626077	2328	0.98125	6m39s	23m43s	64

Size of Dataset. We used different number of questions (i.e. the concept-relation pair used in VP and Top10) to generate dataset of different sizes. Table 3 lists the number of questions, number of the corresponding assertions returned by the four components, and the precision of top 800 assertions returned by ACT on dataset of different sizes. We observed that the precision was increased as the size of total assertions. Table 3 also gives the time in building graph and time in traversing graph of different sizes. Using linear regression to predict the execution time, the traversal time is $y = 4E - 05x - 2.8577, R^2 = 0.9739$ each iteration, and the graph construction time is $y = 0.0007x - 78.865, R^2 = 0.9894$. Both of them are nearly linear time.

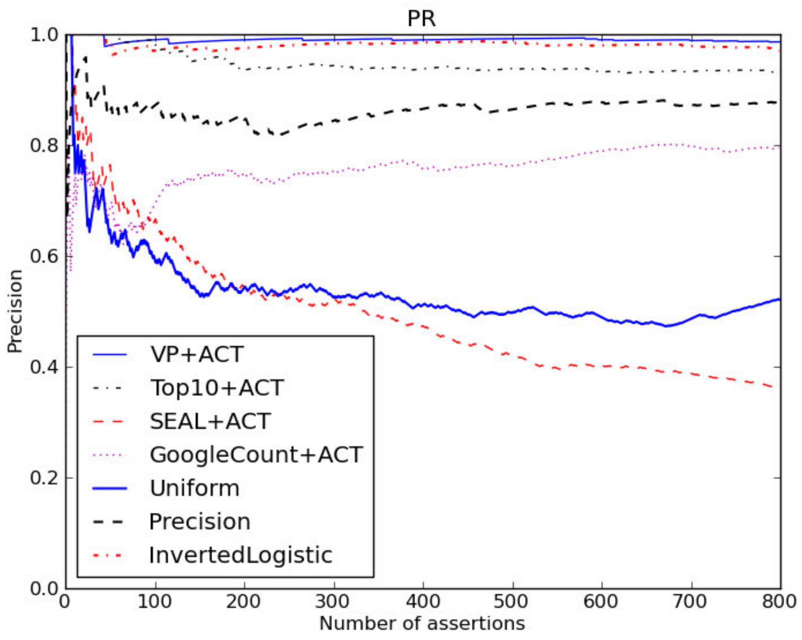


Fig. 4. Precision of ACT on each component and aggregation

5 Discussion

– *GWAP v.s. Text Mining* In previous study, text mining components are more productive in quantity[3,11]. While GWAP like Virtual Pets produces one thousand assertion-certification pairs a day, text mining components can produce the same quantity in few minutes. On the other hand, GWAP have higher precision than text mining components in our experiments. Meanwhile, text mining component is limited to certain domains, such as IsA relation, but GWAP can be used to collect variant assertions. From these observations, we know that the two types of components are good complement to each other. By aggregating the data from both components, we can take advantage of the two approaches.
– *Non-overlapping Data* The ACT can rank data from multiple sources without requirement of overlaps. ACTraversal ranks non-overlapping data by utilizing partial order from overlapping data. This property implies that we can infer a total ranking of confidence level of assertions. We believe the ranking result can be a guidance to bootstrap text mining component to collect more data. We can feed the assertions with high ranking score as the input of text mining components to collect more data. For the assertions with middle ranking scores, we can feed them as input of GWAP for verification. The assertions with low ranking score can then be filtered out. With this kind of coupling after aggregation of ACT, it is possible to get the data with higher quality and quantity.

– *Quantity of Assertions* In the experiment, we observed the quantity of assertions is positively correlated with the quality of top assertions. It implies that our two assumptions hold in our crowdsourced data. If good assertions are collected, they would be ranked in higher order. However, we also noticed the aggregation result is slightly less precise than the component with the best precision, i.e. ACT on VP. This slightly lower precision shows that ACT is not noise-proof. If stronger certification is not associated with better assertion, it would promote the bad assertion to the top. In our experiment, some noisy assertions were reported with high rank by GC component. Due to Google is a keyword based search engine, it cannot detect incorrect template filling. In this case, the count of Google retrieved data was high, but many of the results were not in the same meaning of our assertion. This errors can disturb the ranking order and bring up noises into results. Therefore, we suggest that components put in ACT should be verified to ensure that it 1) holds the assumptions and 2) has acceptable precision.

6 Conclusion

This paper presents the ACTraversal (ACT); a graph traversal algorithm to rank crowdsourced assertions and certifications. The partial order certifications are used for reconstructing the total order ranking of assertions. Our experiments of ACT on Virtual Pets, Top10, SEAL and GoogleCount shows that ACT successfully improve the precision of single component and construct total ranking of assertions from multiple components in nearly linear time.

References

1. von Ahn, L., Kedia, M., Blum, M.: Verbosity: A game for collecting common-sense knowledge. In: ACM Conference on Human Factors in Computing Systems (CHI Notes), pp. 75–78 (2006)
2. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka Jr., E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Twenty-Fourth Conference on Artificial Intelligence, AAAI 2010 (2010)
3. Chang, T.h., Chan, C.w., Hsu, J.Y.j.: Human computation game for commonsense data verification. In: Proceedings of the 2010 AAAI Fall Symposium Series on Commonsense Knowledge (2010)
4. Etzioni, O., Cafarella, M., Downey, D., Popescu, A.M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Methods for domain-independent information extraction from the web: an experimental comparison. In: Proceedings of the 19th National Conference on Artifical Intelligence, AAAI 2004, pp. 391–398 (2004)
5. Kuo, Y.L., Lee, J.C., Chiang, K.Y., Wang, R., Shen, E., Chan, C.W., Hsu, J.Y.j.: Community-based game design: experiments on social games for commonsense data collection. In: Proceedings of the ACM SIGKDD Workshop on Human Computation (2009)
6. Lenat, D.B.: Cyc: A large-scale investment in knowledge infrastructure. Communications of the ACM 38(11), 33–38 (1995)
7. Schubert, L., Tong, M.: Extracting and evaluating general world knowledge from the brown corpus. In: Proceedings of the HLT-NAACL Workshop on Text Meaning (2003)

8. Singh, P., Lin, T., Mueller, E.T., Lim, G., Perkins, T., Zhu, W.L.: Open mind common sense: Knowledge acquisition from the general public. In: On the Move to Meaningful Internet Systems, 2002 - DOA/Coop IS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002 (2002)
9. Speer, R., Havasi, C., Surana, H.: Using verbosity: Common sense data from games with a purpose. In: The Florida AI Research Society Conference (2010)
10. Von Ahn, L.: Games with a purpose. Computer 39(6), 92–94 (2006)
11. Wang, R., Cohen, W.: Language-independent set expansion of named entities using the web. In: Seventh IEEE International Conference on Data Mining, ICDM 2007, pp. 342–350. IEEE (2008)