Crowdsourcing for Web Service Discovery

Fatma Slaimi[1,2(✉)], Sana Sellami[2], Omar Boucelma[2], and Ahlem Ben Hassine[1]

[1] National School of Computer Science (ENSI), University of Manouba, Manouba, Tunisia
[2] Aix-Marseille Université, CNRS, LSIS UMR 7296, 13397 Marseille, France
{fatma.slaimi,ahlembh}@gmail.com,
{fatma.slaimi,sana.sellami,omar.boucelma}@univ-amu.fr

Abstract. Over last decade, research in Web service discovery has brought a variety of techniques to find out responses for a Web service request. While the accuracy of matchmaking approaches has continuously improved, human contributions remain a key ingredient of the process. In this paper, we propose an approach called Crowd4WS (Crowdsourcing for Web service discovery) to complement and refine matchmaking approaches by using crowdsourcing techniques. We describe our approach and present the results of experiments on a known collection of RESTful services described with hRESTS.

Keywords: Web services discovery · Matchmaking · Crowdsourcing

1 Introduction

Web services discovery has been considered as one of the key challenges for achieving efficient service oriented computing. During the last decade, the advent of the semantic Web has led to the development of several semantic matchmaking techniques and systems [1–5]. Matchmaking process aims to discover the most relevant services for a user request. Services are ranked according the similarity scores (a service versus a query). Previous works [6] have shown that matchmaking remains a semi-automatic process, that leading to some false positive results and still may require some form of human assistance.

More recently, with the rapidly growing of Social Web, Crowdsourcing has gained momentum and proved useful in many practical applications. In a nutshell, crowdsourcing [7] [8] can be considered as a model in which a problem is divided into sub-problems and distributed among a group of people (called the crowd or workers). Each sub-problem may result in one or several microtask(s), performed by several workers on behalf of some organizations. Tasks are performed individually or in a collaborative way. Crowdsourcing domains applications examples are: content classification, objects ranking, image annotation, etc.

In this paper we describe Crowd4WS (Crowdsourcing for Web service discovery), a crowdsourcing approach and implementation. Given a subset of relevant precomputed Web services (candidate returned by a matchmaker), and a Web

C. Debruyne et al. (Eds.): OTM 2015 Conferences, LNCS 9415, pp. 451–464, 2015.
DOI: 10.1007/978-3-319-26148-5_30

service request, Crowd4WS asks the crowd to assess the relevance of the services candidates. The outcome of the system is a set of Web service(s) that are the most relevant to a service request. For the experimentation described in this paper, Crowd4WS takes as an input a subset of relevant Web services generated by SR-REST, an hybrid matchmaker described in a previous work [9].

Despite a reasonable number of crowdsourcing applications and platforms, to the best of our knowledge, the crowdsourcing computational paradigm has not been used in the context of the Web services discovery domain: One of the main reasons being the lack of expertise (experts) in such domain. Given this context, we believe the contributions of the work described in this paper are as follows:

(1) we proposed to leverage existing matchmaking techniques/systems with the involvement of the crowd;

(2) we came up with an approach that can use different models such as, for example, a probabilistic model to detect faulty workers;

(3) we used real datasets (Web services descriptions) with a real (although small) set of workers, while many crowdsourcing related work use synthetic data.

The remainder of the paper is organized as follows. Section 2 surveys some related works on matchmaking approaches and crowdsourcing. In section 3, we describe a motivating example. Section 4 presents our approach based on crowdsourcing for Web service discovery (Crowd4WS). In section 5, we describe the system architecture and detail the experimental settings and results in section 6. Section 7 concludes the paper.

2 Related Work

Matchmaking. To leverage semantic descriptions of Web services, several matchmaking approaches have been proposed, most of them exploiting SAWSDL [1] [10] or OWL-S annotated services [3]. These matchmakers can be logic-based, non-logic-based or hybrid, i.e., combining of logic and non-logic similarity functions. Recently, RESTful matchmaking approaches and systems have also been proposed in the literature [9][4]. Such systems, like XAM4SWS [2] for instance, compare service operations, inputs, outputs and similarities. In [4] a graph-theoretic approach called semantic flow matching is proposed: the approach matches REST Web services, specified in WADL (Web Application Description Language) and uses linguistic knowledge and domain-specific heuristics. In [9], we developed a matchmaking approach based on several similarity measure functions that exploits different elements of RESTful descriptions.

As studied in [6] matchmaking solutions came back with false positive and negative results and most of them fail to discover all the relevant services according to the user request. Existing matchmakers have to use and combine several similarity measures and deal with aggregating methods of matching scores to resolve discovery issues. For example, in [6][11] authors show that many matchmakers do not consider semantic similarity between services (e.g., synonym relations) and an improvement of semantic similarity is still needed.

Crowdsourcing. Crowdsourcing has gained momentum to overcome computational tasks that require human assistance. Crowdsourcing has been recently proposed as a solution to a variety of research problems such as ontology alignment, schemas matching or images annotation to cite a few. For example, in [12] authors used crowdsourcing techniques to validate correspondences between schemas: they designed questions with contextual information that may help workers to answer. Crowdmap [8] is a system that collects human contributions via crowdsourced microtasks. For a pair of ontologies, Crowdmap splits the alignment problem into individual microtasks, publishes them on labor market online, collects results and aggregates them. This process has improved the precision of results returned by an automatic ontology alignment process. In [13], crowdsourcing has been used to validate the results of automated images search on mobile devices. Crowdsourcing techniques have also been successfully applied for several data management problems leading to systems such as CrowdSearch [14] or CrowdScreen [13]. In [10] Shen et al. deployed crowdsourcing methods for schema matching. More recently, approaches that use crowdsourcing to assess entities extracted form Web pages (text) to URIs was proposed in the literature: ZenCrowd [15] combines probabilistic reasoning with crowdsourcing to exhibit correspondences from text entities to linked object data.

To summarize, Matchmaking works have shown that there is no ideal matching and matchmaking may still require some form of human assistance. Crowdsourcing has gained momentum and proved useful in many practical applications e.g objects ranking, semantic Web, Linked data, ontology engineering etc. Systems such as ZenCrowd demonstrate how crowdsourcing can contribute to the Semantic Web. All these studies have shown the crowdsourcing benefits and underline the need of human intelligence to effectively handle difficult tasks.

3 Motivating Example

To motivate our approach, let's consider a simple example, described in [9], where a user is looking for a Web service that provides information about car prices. The car price request is described as follow: User request (*name:* car_price, *operation:* get_price, *input:* car, *output:* price).

The process can be summarized as follows:

1. First, we apply SR-REST, an hybrid matchmaker on the car prices example. This yields a set $S = \{$*Toyota car price, auto price color, bicycle price, car year, car report and car recommended price*$\}$, S being considered as the most relevant services;
2. Looking carefully at S, it is obvious that *bicycle price* service (*name:* bicycle_price, *operation:* get_price, *input:* bicycle, *output:* price) is a false positive (a bicycle is not a car), although this service proves highly similar to a car price service, mainly because the matchmaking algorithm does not require a total mapping between the inputs/inputs of user's request and a service [6];
3. When submitting S to the crowd, services that obviously do not match the user request, are simply removed, although returned by the matchmaker.

Fig. 1 below illustrates our crowdsourcing process. People, called *workers* are asked to perform a specific work, usually decomposed into a set of elementary tasks called *microtasks*. A microtask represents a task that could be handled by a worker in a reasonable amount of time. A single microtask may be performed by many workers in order to limit the bias of an individual work. In adopting a crowdsourcing approach, we aim (1) at assessing the results returned by the matchmaking, and (2) returning a set of ranked Web services.

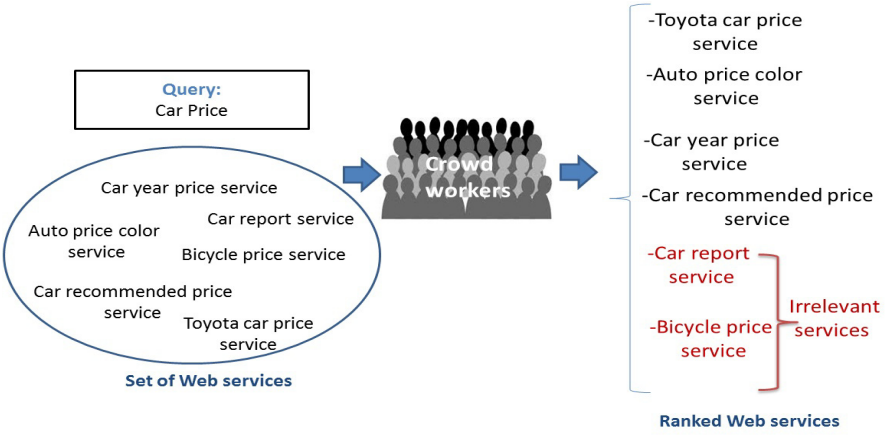

Fig. 1. Illustration of the Crowdsourcing Process

4 Crowd4WS: Crowdsourcing for Web Service Discovery

Crowd4WS, consists of two main steps (microtasks): (1) Validation step, where the crowd checks whether a service matches a request, and hence discard irrelevant services and (2) Ranking step , where the services are ranked by the crowd accordingly with an order of preference (relevance).

Validation. Given a user's query R, a set S of n Web services, $S = \{s_1, s_2, ..., s_n\}$, and a set W of k workers ($\{w_1, ..., w_k\}$), the validation step returns a set RS of relevant services accordingly to three relevance levels RL = {relevant, possibly relevant, irrelevant}. Each worker assigns a relevance score to a s_i expressing how relevant is the returned service s_i to the query R.

Ranking. The crowd will perform a microtask pairwise comparison between Web services. Each worker has to order the set of services. The symbol $>$ is used to define the order; for example a $>$ b means that the worker prefers the object a to the object b (if we consider a and b as two services, a $>$ b means

that a is more relevant then b). This is a preliminary step for inferring a ranking (total ordering) among the results (Web services). We use also an aggregation mechanism to aggregate answers collected from crowd. For each microtask we will apply a set of aggregation functions as described below.

Definition 1 (Results Aggregation): The aggregation process takes as input the set of all answers collected among the workers. More formally, this is represented by a matrix A where a_{ij} represents the answer of a worker w_i for a service s_j.

$$\mathrm{A}= \begin{pmatrix} a_{11} \cdots a_{1n} \\ \vdots \quad \vdots \quad \vdots \\ a_{k1} \cdots a_{kn} \end{pmatrix}$$

The result of aggregation process is a set of aggregated values representing the relevance level assigned for the service s_i.

Several aggregation techniques are proposed in the literature to compute the aggregated value such as, majority voting [16] and expectation maximization (EM) [17]. While majority voting aggregates responses for each service independently, the EM method aggregates simultaneously all responses. The majority voting collects and aggregates worker answers for each service.

In our work, a ranked list of services would be generated; our aim is to compute a relevance score for each service. For this reason we will apply an aggregation method based on the majority voting approach. We define a numerical equivalent for each alternative in order to compute a global score per service, then services will be ordered according to these scores.

Definition 2 (Aggregation Score): We denote Sc(S) as the aggregation function applied to compute the score of each candidate Web services from the worker's answers. This function is defined as follow:

$$Sc(s_i) = \frac{\sum_{i=1}^{k}(r_i)}{k}. \tag{1}$$

Where k is the number of workers who have validated service s_i ; and r_i is a relevance number accordingly to the level chosen by the worker such that:

$$\begin{cases} r_i = 1, & \text{if answer=relevant} \\ r_i = 0.5, & \text{if answer=possibly relevant} \\ r_i = 0, & \text{if answer=irrelevant.} \end{cases}$$

After aggregating all answers, an ordered list of Web services will be generated. This list will be submitted to the crowd to rank the services in this list from the best to the worst one. This ranking microtask implies both pairwise

comparison between the services and an aggregation of the different results (see Definition 3).

Definition 3 (Ranking Aggregation): The "total" ordering of relevant Web services is given by the aggregation of orders supplied by the workers. The ordering of a worker w_k is denoted by:ϕ_l=$\{(i,j), s_i \succcurlyeq s_j\}$ l=$\{1,..,k\}$.

After collecting the answers of all k workers that performed pairwise comparisons, we obtain a set ϕ of ranked Web services, where ϕ represents the set of orders given by the k workers and ϕ_l represents the order given by a worker l.

$$\phi = \cup \phi_l. \tag{2}$$

Results collected from the crowd must be aggregated in order to find a total order. We are using the the Bradley-Terry model [18] [5] that, given two objects x and y, evaluates the probability that x is preferred to y. More precisely, if $\alpha, \beta \geqslant 0$ respectively denote the relevance scores of x and y, this probability is computed as:

$$Pr(x \succcurlyeq y) = \frac{\alpha}{\alpha + \beta} \tag{3}$$

We assume that there are n candidate services $\{s_1, .., s_n\}$ and a pool of k workers $\{w_1, .., w_k\}$. The set of (pairs of) services evaluated by a worker w_k is denoted by $S_k = \{(i,j) : s_i \succcurlyeq s_j\}$, where $s_i \succcurlyeq s_j$ represents that the worker prefers s_i to s_j , we apply the Bradley-Terry, we have:

$$Pr(s_i \succcurlyeq s_j) = \frac{\alpha_i}{(\alpha_i + \alpha_j)} \tag{4}$$

Where α_i is the relevance score of the service s_i. The score of each service can be estimated in using maximum likelihood method.

A global ranking over n candidates services can be obtained by sorting the vector of scores $= (\alpha_1, \cdots, \alpha_n)$ where α_i is the score of the service s_i.

Example. We consider the previous example of car price service (see section 3). Table 1 represents the pairwise comparison list (see definition 3) submitted for ranking to 5 workers. We computed the number of times a service s_i was preferred to s_j. Then we applied the Bradley and Terry model in using XLSTAT[1], to obtain a final ordering.

A matrix M can be obtained where M_{ij} represents the number of time a service i is considered as better than a service j.

$$M = \begin{pmatrix} - & 3 & 4 & 3 \\ 2 & - & 4 & 1 \\ 1 & 1 & - & 2 \\ 2 & 4 & 3 & - \end{pmatrix} \tag{5}$$

[1] www.xlstat.com

Table 1. Pairwise Comparison Results

service 1	service 2	service 1 wins	service 2 wins
Auto color price car (s_1)	Car year price (s_2)	3	2
Auto color price car (s_1)	Toyota car price (s_3)	4	1
Auto color price car (s_1)	Car recommended price (s_4)	3	2
Car year price (s_2)	Auto color price car (s_1)	2	3
Car year price (s_2)	Toyota car price (s_3)	4	1
Car year price (s_2)	Car recommended price (s_4)	1	4
Toyota car price (s_3)	Auto color price car (s_1)	1	4
Toyota car price (s_3)	Car year price (s_2)	1	4
Toyota car price (s_3)	Car recommended price (s_4)	2	3
Car recommended price (s_4)	Auto color price car (s_1)	2	3
Car recommended price (s_4)	Car year price (s_2)	4	1
Car recommended price (s_4)	Toyota car price (s_3)	3	2

The choice of the symbol - for the diagonal entries is arbitrary. This matrix will be used as input to the Bradley and terry model (Definition 3). Results are presented by table 2 below.

Table 2. Bradely and Terry Aggregation Results

test	number of wins	number of losses	percentage of wins	percentage of losses
s_1	20	10	66.67	33.33
s_2	15	15	50.00	50.00
s_3	7	23	23.33	76.67
s_4	18	12	60.00	40.00

Based on the percentage of wins, a simple ordering may be obtained: the "best service" is the one with the best percentage.

5 Crowd4WS Architecture

As depicted in Fig. 2, Crowd4WS takes as inputs a set of Web services generated by a matchmaker and a user query and submit them to the crowd for validation and ranking. The output of this crowdsourcing process is a collection of ranked services corresponding to the user query.

The crowdsourcing workflow consists mainly on: generating microtasks, publishing microtasks, collecting and aggregating workers' answers.

Microtask Generator: generates two kinds of microtask: 1) a validation microtask and 2) a ranking microtask. The validation microtask is used to verify the similarities between the Web services and the user query. While the second microtask performs a ranking as described above.

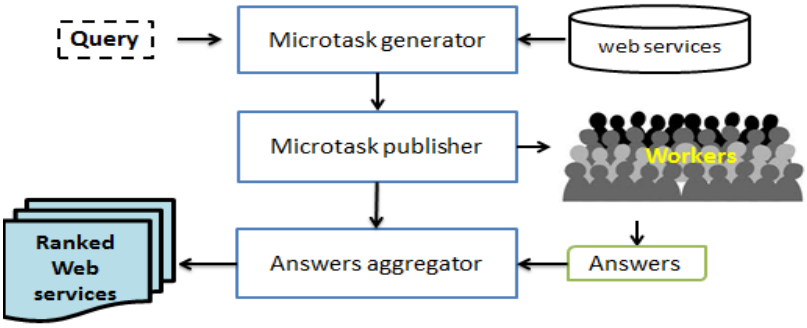

Fig. 2. System Architecture

Microtask Publisher: posts microtasks to the crowd.

Answers Aggregator: collects and aggregates the answers from different workers/microtasks. Answers returned by the crowd might be incorrect for several reasons, such as tasks misunderstanding, leading to errors or even not to finding the right answer. To solve this problem, we need aggregation mechanisms, applied in the Answer Aggregation component.

5.1 Microtask User Interface

Workers are provided with two micro task user interfaces.

Validation Microtask. Recall that during the validation phase, workers have to decide how relevant is a service to a query. As illustrated in Fig. 3, a microtask consists mainly of a question (How relevant is a service?), relevance being expressed as relevant, possibly relevant or irrelevant. The information about the query, the candidate Web service and the alternatives are also displayed to assist the workers (i.e with a simple click on the More Info field).

Ranking Microtask. The goal of the ranking microtask (Fig. 4) is to collect workers preferences on a pairwise comparison basis. The relevant services resulted from the Validation microtask are ranked by the crowd accordingly with an order of preference (relevance). For each pair of services and a request, workers assign a preference order 1 (being the best) or 2 to indicate that a service is more relevant than another.

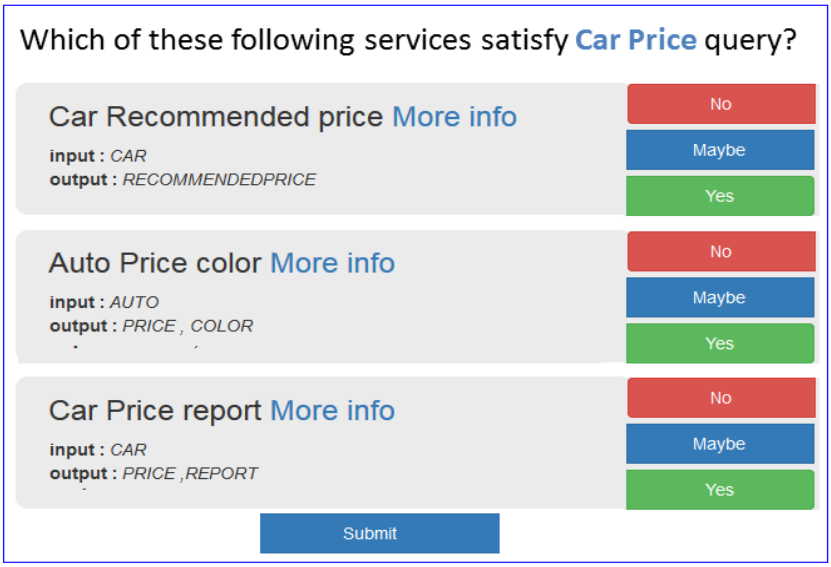

Fig. 3. Validation Microtask UI

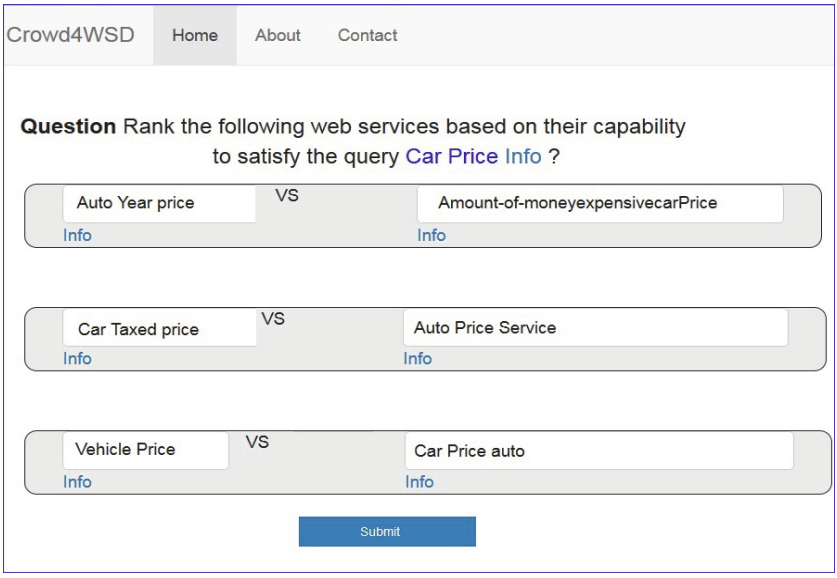

Fig. 4. Ranking Microtask UI

6 Experimentation

Many crowdsourcing research prototypes use commercial platforms [19], such as Amazon Mechanical Turk. We find it easier to implement our own system: a video illustrating the system is accessible[2].

Regarding the workers, we relied on our academic network which is composed mainly staff (lab members) and students at our university. We implemented a Web application that generates microtasks questions from the test collection hREST-TC1[3]. This collection is derived from SAWSDL-TC1 collection and is composed of 25 queries (from different domains: communication, food, economy, medical, travel and education), 895 services and 24 ontologies used to semantically annotate the inputs and outputs parameters of Web services. For each query a relevant set is provided.

The experimental results are detailed below.

6.1 Experimentation Setup

Configuration Parameters for Crowdsourcing. In the crowdsourcing environment, each microtask should be executed by a number of workers. Based on common practices in the crowdsourcing context for similar tasks[20] [8], the number of workers assigned to each task is limited to 5. As microtasks are simple to accomplish and require only few minutes to be performed, each worker is asked to fulfill a set of validation (or ranking) tasks.

Experimental results show that it takes between few minutes to receive answers from each worker (depending on the number of microtasks that should be attributed). In order to evaluate the number of tasks to be assigned to each worker, we measured the time required to perform a set of microtasks by these workers. We will pay attention to the time consumed by worker's to fulfil a set of grouped microtasks. We varied the number of micro tasks per groups between 1 and 30 microtasks. Fig. 5 illustrates the obtained results. In this figure, we show that the time consumed increased when we have more than 10 microtasks.

Then the user interface design and the time required to execute a microtask influenced the crowdsourcing results. Our concern is how to improve the user interface to make more easily the user interaction. Based on these results, we limited the number of microtasks per group to 6 in order to get the maximum number of answers.

6.2 Evaluation Results

Worker's Expertise. To evaluate our approach, first we addressed the problem of the lack of expertise of workers. We have implemented a simple model based on a quiz as illustrated in Fig. 6 to check the expertise of the worker.

[2] http://www.lsis.org/sellamis/Projects.html
[3] http://semwebcentral.org/projects/hRESTS-tc/

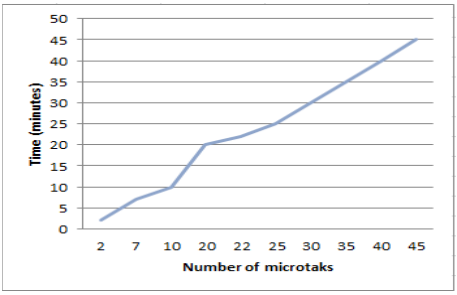

Fig. 5. Time per Set of microtasks

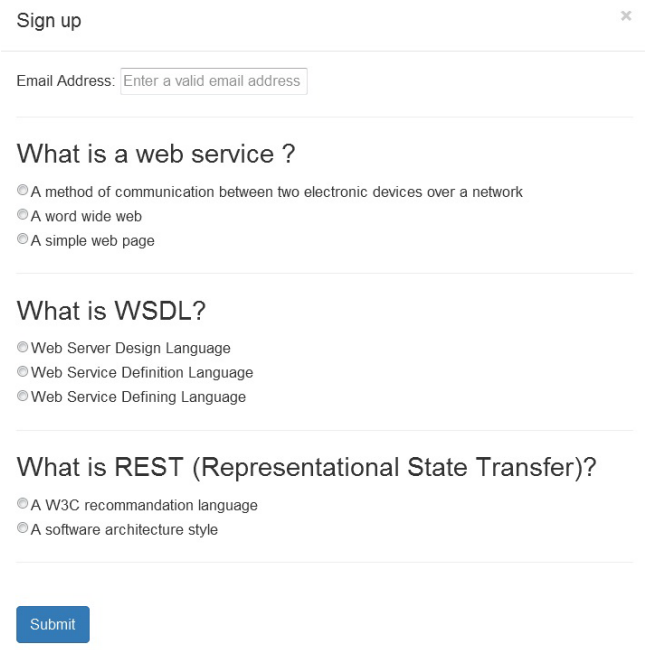

Fig. 6. Worker's Quiz

Workers are asked to answer a set of questions about Web services before executing validation and ranking microtasks. The results of theirs answers will be used to measure a degree of expertise. If the worker makes more than one mistake, he is considered as non expert and the system does not consider her answers.

Crowdsourcing Evaluation. We conducted a series of tests to evaluate the impact of both validation and ranking processes. We measured the average precision of the validation process and then the average precision of both validation and ranking process. For our evaluation, we use precision and recall measures

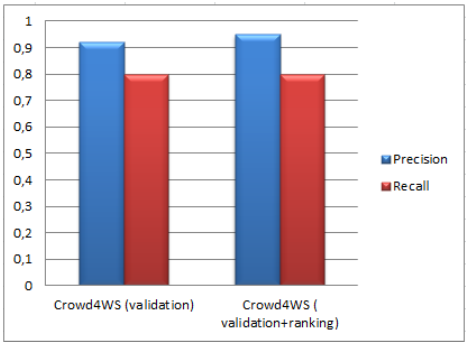

Fig. 7. Crowd4WS Precision and Recall (validation and ranking)

to evaluate the effectiveness of our approach. These measures are described as follows: Precision= $\frac{|A \cap B|}{B}$, and Recall= $\frac{|A \cap B|}{A}$ Where A is the set of all relevant services for a request and B the set of all retrieved services for a request. We then compute the average precision and recall.

As shown in Fig. 7, Ranking microtask has positive impact on the returned list of services. In fact we note that irrelevant services are at the end of the list while the most relevant services are at the top of the list. Ranking microtask improves the precision but gives the same recall as does the validation step. Based on the definition of recall (i.e the number of relevant services that are retrieved by the workers) we conclude that crowdsourced ranking discards false positives but has no impact on the number of relevant retrieved Web services (the same set approved by the validation microtasks).

Crowdsourcing vs Matchmaking. Finally, we compared Crowd4WS with our previously proposed matchamker SR-REST by measuring their average precisions and recalls. Fig. 8 shows the obtained results. We noticed that the use of the crowd to validate services obtained through the matchmaking process leads to a noteworthy precision improvement. This is due to the fact that irrelevant Web services will be discarded as they are judged as impertinent to the user's research. The pruning of these services from the set of returned pertinent services vindicates the recall improvement.

To summarize, our experiments involved 25 different workers. Overall, they confirmed our first hypothesis: human intelligence could be used to enhance precision of automatic Web services discovery approaches. The crowd is able to detect false positive results given by the matchmaker and discards them from the set of returned results.

7 Conclusion

The work described in this paper demonstrates that crowdsourcing may complement and refine matchmaking approaches. The experimental numbers, expressed

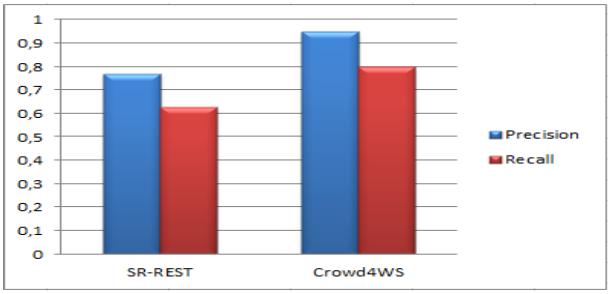

Fig. 8. Crowd4WS vs SR-REST precision and recall

in terms of precision, show a worthy improvement with respect to the number of candidates services provided by the matchmaking process: hence this leaves hope for further investigation in order to come up with an efficient service discovery platform that combines both matchmaking and crowdsourcing. This work can be considered as a first step towards hybrid human-automatic Web service discovery.

As mentioned earlier, we did not use a commercial platform such as AMT, because we need to rely on a "service-aware" crowd. Indeed, tagging a Flickr picture, or recognizing a landmark in a city, are examples of easy tasks; while deciding whether a Web service matches a user request is a harder task because one needs to have some expertise in the service oriented computing area. This remark raises several challenges that need to be tackled such as user's credibility (level of expertise), quality of results, etc.

References

1. Klusch, M., Kapahnke, P., Zinnikus, I.: Adaptive Hybrid Semantic Selection of SAWSDL Services with SAWSDL-MX2. Int. J. Semantic Web Inf. Syst. **6**(4), 1–26 (2010)
2. Lampe, U., Schulte, S., Siebenhaar, M., Schuller, D., Steinmetz, R.: Adaptive matchmaking for RESTful services based on hrests and microwsmo. In: Binder, W., Schuldt, H., (eds.) ACM International Conference Proceeding Series, WEWST, pp. 10–17. ACM (2010)
3. Klusch, M., Kapahnke, P.: The iSeM matchmaker: A flexible approach for adaptive hybrid semantic service selection. J. Web Sem. **15**, 1–14 (2012)
4. Khorasgani, R.R., Stroulia, E., Zaane, O.R.: Web service matching for restful web services. In: Kienle, H.M., Bolchini, D., Tramontana, P., (eds.) WSE, pp. 115–124. IEEE (2011)
5. Chen, X., Bennett, P.N., Collins-Thompson, K., Horvitz, E.: Pairwise ranking aggregation in a crowdsourced setting. In: Leonardi, S., Panconesi, A., Ferragina, P., Gionis, A., (eds.) WSDM, pp. 193–202. ACM (2013)
6. Klusch, M., Fries, B.: Hybrid owl-s service retrieval with owls-mx: benefits and pitfalls. In: SMRR (2007)

7. Brabham, D.C.: Crowdsourcing. The MIT Press essential knowledge series. MIT Press (2013)
8. Sarasua, C., Simperl, E., Noy, N.F.: Crowdmap: crowdsourcing ontology alignment with microtasks. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 525–541. Springer, Heidelberg (2012)
9. Slaimi, F., Sellami, S., Boucelma, O., Ben Hassine, A.: Flexible matchmaking for RESTful web services. In: Meersman, R., Panetto, H., Dillon, T., Eder, J., Bellahsene, Z., Ritter, N., De Leenheer, P., Dou, D. (eds.) ODBASE 2013. LNCS, vol. 8185, pp. 542–554. Springer, Heidelberg (2013)
10. McCann, R., Shen, W., Doan, A.: Matching schemas in online communities: a web 2.0 approach. In: Alonso, G., Blakeley, J.A., Chen, A.L.P., (eds.) ICDE, pp. 110–119. IEEE (2008)
11. Klusch, M.: Service discovery. In: Alhajj, R., Rokne, J. (eds.) Encyclopedia of Social Networks and Mining (ESNAM). Springer (2014)
12. Hung, N.Q.V., Tam, N.T., Miklós, Z., Aberer, K.: On leveraging crowdsourcing techniques for schema matching networks. In: Meng, W., Feng, L., Bressan, S., Winiwarter, W., Song, W. (eds.) DASFAA 2013, Part II. LNCS, vol. 7826, pp. 139–154. Springer, Heidelberg (2013)
13. Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: algorithms for filtering data with humans. In: Candan, K.S., Chen, Y., Snodgrass, R.T., Gravano, L., Fuxman, A., (eds.) SIGMOD Conference, pp. 361–372. ACM (2012)
14. Yan, T., Kumar, V., Ganesan, D.: Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In: Banerjee, S., Keshav, S., Wolman, A., (eds.) MobiSys, pp. 77–90. ACM (2010)
15. Demartini, G., Difallah, D.E., Cudr-Mauroux, P.: Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab, S., (eds.) WWW, pp. 469–478. ACM (2012)
16. von Ahn, L.: Human Computation. Ph.D. thesis, School of Computer Science, Carnegie Mellon University (2005)
17. Dawid, A.P., Skene, A.M.: Maximum likelihood estimation of observer error-rates using the em algorithm. Applied Statistics **28**(1), 20–28 (1979)
18. Bradley, R.A., Terry, M.E.: The rank analysis of incomplete block designs – I. The method of paired comparisons. Biometrika **39**, 324–345 (1952)
19. Yuen, M.C., King, I., Leung, K.S.: A survey of crowdsourcing systems. In: 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (socialcom) Privacy, Security, Risk and Trust (passat), pp. 766–773. IEEE (2011)
20. Zhao, Y., Zhu, Q.: Evaluation on crowdsourcing research: Current status and future direction. Information Systems Frontiers, 1–18 (2012)