A Novel Clustering-based Approach for SaaS Services Discovery in Cloud Environment

Kadda Beghdad Bey[1], Hassina Nacer[2], Mohamed El Yazid Boudaren[1] and Farid Benhammadi[1]
[1]*Informatics Systems Laboratory, Ecole Militaire Polytechnique, Algiers, 16111, Algeria*
[2]*MOVEP Laboratory, University of Science and Technology, USTHB, Algiers, Algeria*

Keywords: Cloud Computing, Resource Allocation, Software as a Service (SaaS), Services Discovery, Web Service, Multi-agents Systems, Clustering Methods, Matching.

Abstract: Cloud computing is an emerging new computing paradigm in which both software and hardware resources are provided through the internet as a service to users. Software as a Service (SaaS) is one among the important services offered through the cloud that receive substantial attention from both providers and users. Discovery of services is however, a difficult process given the sharp increase of services number offered by different providers. A Multi-agent system (MAS) is a distributed computing paradigm-based on multiple interacting agents- aiming to solve complex problems through a decentralized approach. In this paper, we present a novel approach for SaaS service discovery based on Multi-agents systems in cloud computing environments. More precisely, the purpose of our approach is to satisfy the user's needs in terms of both result accuracy rate and processing time of the request. To establish the interest of the proposed solution, experiments are conducted on a simulated dataset.

1 INTRODUCTION

Scientific applications are becoming increasingly complex and distributed. They are faced with the challenges of the information heterogeneity used and the decentralization of processing and controls. Thanks to the evolution of IT, especially the web and virtualization, a new way has been created to offer IT as a service in an economic context and, at the same time, solve all computational and storage problems. Cloud computing is proposing to be the most appropriate solution. It is an internet-based computing, where computing resources, softwares and information hosted in an ultra secure data center are provided to clients on demand.

In recent years, cloud computing has taken a considerable step as a new paradigm for distributed computing. It is becoming an ideal system for distributed computing communities, thanks to the advantages of the proposed services and the easy access to the information without limitation nor place or time. The deployment of cloud solution enables users to develop their different services in a distributed environment consisting of a large number of computing problems free resources with a very low operating cost. Services discovery remains however, one of the most difficult and important problems that Internet users meet, particularly due to the large number of services published by different providers. The diversity and evolution of services presented in different levels, as well as the non-standardization of the description languages of cloud services, search services in a cloud environment is a very difficult task in order to meet all client requirements (Parhi et al., 2015). Moreover, Multi-Agent Systems (MAS) propose a new metaphor based on social concept which is the "agent" notion. They are a promising paradigm for modeling, analysis and design applied to many development and research areas such as solving distributed problems. Multi-agent systems are often used to solve complex problems based on decentralized approach where a set of agents contribute collectively to find a solution.

With the exponential growth in the number and functionality of cloud services, as well as the diversity of technologies used for their presentation, the problem of determining a strategy for SaaS services discovery in cloud has become a challenging issue (Pirro et al., 2010). In this context, we present a new SMAs-based approach to SaaS service discovery in the cloud using two complementary modules. The first is responsible for SaaS services publication offered by providers that are organized in "clusters",

Bey, K., Nacer, H., Boudaren, M. and Benhammadi, F.
A Novel Clustering-based Approach for SaaS Services Discovery in Cloud Environment.
DOI: 10.5220/0006328205460553
In *Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS 2017) - Volume 1*, pages 546-553
ISBN: 978-989-758-247-9
Copyright © 2017 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

in order to reduce the search space. For this purpose, the new clustering algorithm is proposed based on some Data mining techniques in order to resolve the overlapping problem. The second one provides, on the other hand, the selection of the most relevant SaaS services based on a set of agent behavior, which ensure an efficient management of cloud resources and enable the matching algorithms execution.

The remainder of the paper is organized as follows. Section 2 summarizes some previous works related to SaaS services discovery in cloud system. The definition and modeling problem of SaaS service discovery are introduced in Section 3 where the proposed SaaS service discovery system-based on multi-agent are also presented. Simulation results and related discussion are presented in Section 4. Section 5 concludes the paper.

2 RELATED WORKS

Developing efficient systems for services discovery and composition has become an essential task. This is mainly due to the evolution in number and functionality of services and technologies of development tools in distributed systems in general and cloud computing in particular, in order to determine and select the best services desired by customers. Indeed, service discovery is becoming increasingly critical in areas where the names or the service description based on inputs/outputs are not sufficient for service publication through different providers or for services identification by users. The success of SaaS services involved the adoption of this technology by different service providers through the cloud, which induces the increasing of SaaS number and makes the discovery problem a tedious task. SaaS services discovery is an emerging area of cloud computing research which aims to automatically detect the services consisting of a set of applications and data components in a cloud.

In literature, plethora of solutions for SaaS services discovery and research has been proposed (Li and Chen, 2014) (Guerfel et al., 2015) (Fan et al., 2015b). The aim of these discovery approaches in cloud environment is to determine an optimal resources allocation and to satisfy the user's needs in terms of results accuracy rate and the processing time of the customers' requests. (Fan et al., 2015b) proposed an integrated SaaS-based personalization framework to facilitate the preferences collection and the corresponding SaaS services delivery. The adapted approach by authors for the design and development of this framework is based on the synthesis principle of various models and techniques in a novel way. This proposal is an extension of a semantic client-side personalization approach presented in (Fan et al., 2015a) by integrating Rich Client and Semantic Web for SaaS-based personalization in order to improve the efficiency in gathering user profile data and reducing the overhead of server-side computing.

The work by (Wu et al., 2011) presents a resource allocation algorithm for SaaS providers in order to minimize infrastructure cost and SLA violations. The aim of this approach is to ensure that SaaS providers are able to manage the dynamic change of customers, mapping customer requests to infrastructure level parameters and handling heterogeneity of Virtual Machines. (Alfazi et al., 2014) proposed a novel ontology-based approach for cloud service discovery. The proposed approach has the capability to generate the ontology semi-automatically using new concepts from documents related to cloud services. (Chen and Li, 2011) proposed a Cloud solution called SRC (Service Registry on the Cloud), which represents an extension of keywords based model but deployed as an application on the cloud. The model SRC stores semantic descriptions of Web services as well as the evaluation of the state dynamic of QoS under the GFS (Google File System) file in the Cloud, and use the Map Reduce mechanism for dealing with these files. GoDiscovery (Elshater et al., 2015) is a discovery system that aims to present effective discovery approach using statistical modeling and indexing techniques. It generates Term Frequency-Inverse Document Frequency (TF-IDF) model for corpus Service, then it builds a tree index K-Dimensional for the model search. On the other hand, a new approach based on multi-agents system for ser- vices discovery in cloud environment has been introduced by (Han and Sim, 2010). The proposed prototype of Cloud Services Discovery System (CSDS) consists of a search engine and three agents: Query Processing Agent that locates information resources by running regular research engines, Filtering Agent which relieves users of long and painful tasks, and Cloud Service Reasoning Agent which consults the cloud ontology for reasoning about the relationships between cloud services. (Parhi et al., 2006) proposed a framework based on multi-agent system to support the description and discovery of cloud services. The authors involve an artificial intelligence approach to efficiently interpret the user requirements based on both functional and nonfunctional demands and fuzzy constraints. The main objective of this approach is to reduce the search space due to implementation of distributed service oriented architecture.

3 SaaS DISCOVERY SYSTEM

As described in previous sections, SaaS services discovery is an emerging area of research which allows to automatically detecting services in a cloud environment. This section describes the proposed agent-based system for SaaS service description and discovery in cloud which is a hybridation of two approaches in order to meet the limits encountered by the developers.

3.1 Definition and Modeling Problem

The Cloud represents today a new platform of distributed computing where users search, discover and share information. In this context, the cloud service discovery process, in particular SaaS services are fundamental and allow one to make the link between information published by SaaS providers and users queries. In general, such a process is based on a "textual" or "keywords" research, but this type of research cannot always identify the most relevant services. So the challenge is to satisfy the user by responding to his request with the suitable services automatically. Figure 1 represents conceptual diagram of the proposed SaaS service discovery system in cloud environment. It comprises different phases, including: Services collection, Service description and clustering, Requests generation, Matching process, Service Selection and Personalization.

Despite the considerable number of proposed SaaS services discovery solutions in the literature, the discovery problem remains a very active area of research. Several reasons have motivated the scientists to propose other solutions to improve the existing performances discovery systems. Thus, we have explored this track to propose a new discovery mechanism consisting of two main layers:

- *Publication Layer*: is responsible for the description and organization of the proposed services by suppliers on different clusters in an appropriate manner in order to stock them in the services database after validation;

- *Discovery layer:* is the process that allows one to find all the relevant services based on their characteristics, in order to facilitate the services search. The discovery process is responsible for the generation of the user's query and matching process that compares the services description saved in a services database with those of the user query to identify services that are relevant.

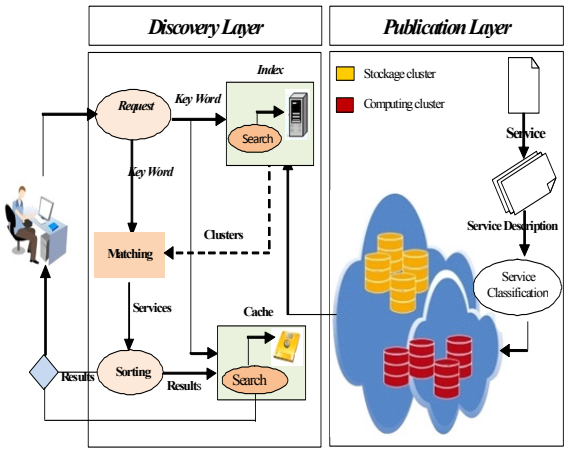

Figure 1: Architecture of the proposed discovery system.

The system modeling was made by the multi-agent system to have an intelligent and effective management of cloud resources and each principal task must be supervised and executed by a specific agent. Furthermore, agents must cooperate and collaborate, and their behaviors should be harmonious to avoid possible conflicts in the dynamic resource management. The last point is important for the combination of two future technologies namely, Agents and Cloud technology.

This combination allows the creation of a new generation of service-oriented agents, which designed to push limitations and capabilities of the resource management, towards management systems more intelligent and more appropriate to customer needs. Figure 2 shows an overall view of the proposed SaaS services discovery system modeled by the SMAs. The five main SaaS service discovery agents are: Agent administrator, agent interface, agent index, agent matching and agent resource.

Interface Agent: is responsible for transmitting client requests to regeneration module of query for translating them to SQL requests before sending them to the Manager Agent. Moreover, it displays the results of SaaS service discovery found by the Multi-agents system to clients.

Manager Agent: provides the following functionalities: (i) the reception of the request after regeneration and the creation of a customer file to store the functional and non-functional requirements of each request; (ii) sends the request to search agents namely cache agent, index agent and matching agent; (iii) the reception of different services that are found by the cache agent as well by the matching agent, and sends the results to the interface agent.

Index Agent: aims to find the appropriate cluster index. It hosts the search mechanism after that receives the manager request to find the appropriate cluster. Then, it communicates with the agent manager to confirm that finished the search in order to send the identified cluster to the matching agent.

Matching Agent: shuts off after identification of the cluster found by the index agent and executes the matching algorithm to sort the SaaS of the selected cluster. Finally, it sends the final results concerning the SaaS service found to the manager agent.

Resource Agent: periodically finds the available resources in the Cloud. When an agent wants to run those appropriate algorithms, resource agent consults the state of available processing resources for running and distributing the different tasks in order to improve the performance of SaaS service discovery systems.

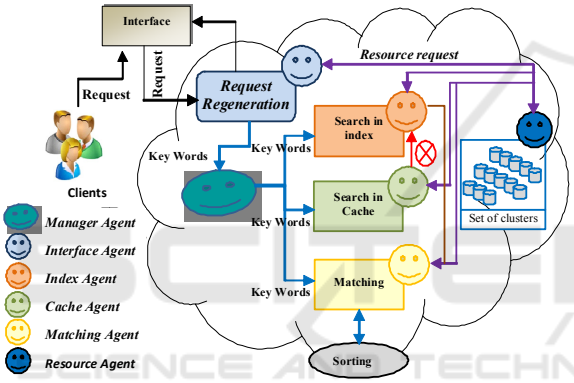

Figure 2: System architecture modeled by SMAs.

3.2 Proposed Clustering Algorithm for SaaS Services Publication

The work by (Alfazi et al., 2015) proposes a clustering algorithm for SaaS service based on grouping by functionality. The principle is to choose randomly from the SaaS services database one service as a "reference service" for the first cluster. Then, it calculates the similarity between this reference service and other services of cloud sources. All services with a higher similarity than a predefined threshold "T" will be removed from the set of cloud services source and added to the first cluster. This procedure is repeated until all services are affected to clusters (Alfazi et al., 2015). Although presented algorithm achieves a good classification of SaaS services, there are gaps in the choice of references for each cluster services. Also, there may be an overlap problem between clusters due to misallocation of services.

Figure 3 illustrates this problem using this clustering approach. To overcome this drawback, we propose an improved algorithm by adding another treatment between the ranked service as reference and the already classified services in previous clusters. This calculation is used to ensure a good affectation of appropriate service to each cluster and each similarity computation will be saved for comparison with respect to the new reference service the maximum will be chosen. Algorithm 1 describes in detail the steps of services affectation to different clusters according to our approach, which solves the overlap problem.

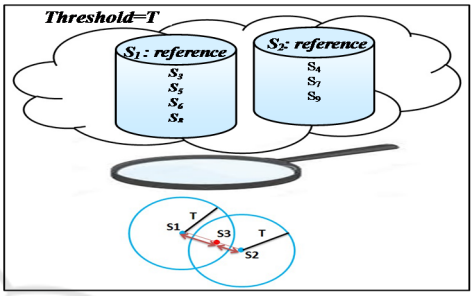

Figure 3: Presentation of overlap problem.

Similarity Computing: The "similarity" parameter used in the previous algorithm, is calculated by following the steps below:

1. Calculation of the number of occurrences of each concept in each service, as shows in table 1.
2. Calculation of the frequency of each concept in each service, using the following equation:

$$TF(S_i, C_j) = \frac{N_{ij}}{\sum_{k=1}^{m}(C_m)} \quad (1)$$

3. The similarity between two services S_i and S_j is given by the following formula:

$$Similarity = \frac{\vec{S_i} \times \vec{S_j}}{S_i \times S_j} \quad (2)$$

Such as:

$$S_i = < TF(S_i, C_1), TF(S_i, C_2), TF(S_i, C_3), \ldots TF(S_i, C_n) >$$

Table 1: Calculation of the occurrences number for each concept in each service.

	Concept 1 (C_1)	*Concept 2 (C_2)*	.....	*Concept N (C_N)*
Service 1 (S_1)	N_{11}	N_{12}	.....	N_{1n}
Service 2 (S_2)	N_{21}	N_{22}	.....	N_{2n}
⋮				
Service m (S_m)	N_{m1}	N_{m2}	.....	N_{mn}

Algorithm 1: *Clustering algorithm for Cloud SaaS services.*
Input: - A set of SaaS services $S= \{S_1, S_2, \ldots\ldots S_N\}$;
$T=$ *Threshold* ;
// T is a fixed value of similarity threshold
Output: A set of Clusters $C= \{C_1, C_2, \ldots\ldots C_k\}$;
$K=1$;
Affect S_1 to C_k and remove it from S ;
while $S \neq \phi$ **do**
  $Simil_{max} = -1$;
  for $m = 1 \rightarrow k$ **do**
    Compute Simil (S_i, S_{rk}) ;
      // S_{rk} *is the reference service of cluster k*
    if ($Simil > Simil_{max}$) **then**
      $Simil_{max} \leftarrow Simil$
      $K' \leftarrow K$
    end if
  end for
  if ($Simil_{max} > T$) **then**
    Affect S_i to $C_{k'}$;
    Remove S_i from S ;
    Save Simil (S_i, $S_{rk'}$) ;
  else
    Create new C_{k+1} ;
    Affect S_i to C_{k+1} ;
    S_i is S_{rk+1} ;
    // reassignment of exist services in C_k
    for ($m=1$, $m<k$, $m++$) **do** *// number of cluster*
      $j=2$;
      while (C_m *not end*) **do**
        Compute Simil (S_i, S_{jm}) ;
        if $Simil > S_{jm}$ *.Simil* **then**
          // S_{jm} *.Simil ; similarity between* S_{jm} *et* S_{rm}
          Affect S_{jm} to C_{k+1} ;
          Remove S_j from C_m ;
          Save the new similarity ;
        end if
        $j=j+1$;
      end while
    end for
    $K=k+1$;
  end if
end while

3.2 SaaS Services Discovery Approach

The proposed approach is composed of two parts: the cluster identification component and the services matching component. The first is responsible of identifying the cluster functionally responding to the user query. The second is based on the use of the "cluster index" that simplifies and speeds up the search operation. The index contains the functional description of each cluster which is expressed by its reference service. The search algorithm computes the similarity between the reference services of each cluster and the keyword vector of the user query and chooses the maximum similarity in order to ensure the desired functionality. Once the cluster is selected, the role of the services matching component is to select the services with high similarity to the user preference using a matching algorithm based on similarity between query concepts and all functional description parameters of services expressed in the WSDL document. This algorithm is applied to find, among the cluster services, the most similar SaaS services having greater probability to user's preference.

The primary problem of discovery approaches lies in the choice of concepts matching techniques (i.e., similarity measures) and / or the matching algorithms optimization. In the proposed discovery approach, we have chosen to compute similarity with SimWP (Wu and Palmer, 1994) that finds the most similar concepts between two ontologies. It calculates the semantic proximity between two concepts through ontology arcs. The similarity measure depends only on the concepts depths, and since most ontologies have limited depths compared to the concepts number, then the execution time of the measurement is always acceptable. It was decided for the similarity computing to use the semantic lexical database WordNet. First, we calculate the similarity between two concepts. The calculation principle is given by formula (3):

$$\boldsymbol{SimComp(c_1, c_2)} = \frac{2 * depth\big(SS(c_1, c_2)\big)}{depth(c_1) + depth(c_2)} \quad (3)$$

with: - *Depth*(C_1) *and*(C_2)*: is the number of arcs that separate* C_1 *and* (C_2) *to the ontology root O.*
- *SS*(C_1,C_2)*: is the Smallest Subsumption of* C_1 *and* C_2 *(i.e., the common ascendant between* C_1*and* C_2 *farthest away from the root).*

Having computed the similarity between two concepts, we now calculate the SimWP similarity between the user query and services, which are both expressed by a set of concepts. A similarity computing example using ontology is shown in Figure 4. Let $T_1 = (C_1, C_2, ..., C_n)$ denote the request concepts and $T_2 = (C'_1, C'_2, ..., C'_n)$ the service concepts, based on the above computing, we determinate a similarity matrix between T_1 and T_2, as follows:

$$M = T_1 * T_2 = \begin{bmatrix} SimComp(C_1, C'_1) & \ldots & SimComp(C_1, C'_n) \\ \vdots & \ddots & \vdots \\ SimComp(C_n, C'_1) & \ldots & SimComp(C_n, C'_n) \end{bmatrix} \quad (4)$$

Then, we construct the MaxL sequence by searching the existing maximum in the matrix and removing the row and column to which this maximum element belongs. The previous step is

repeated until the number of the matrix elements is equal to zero:

$$\boldsymbol{MaxL} = (SimCom_{max1}, SimCom_{max2}, \cdots, SimCom_{maxk}) \quad (5)$$

Therefore, the SimWP similarity computing is given by the following formula:

$$\boldsymbol{SimWP} = (1/k) \sum_{i=1}^{k} SimCom_{max\mathrm{i}} \quad (6)$$

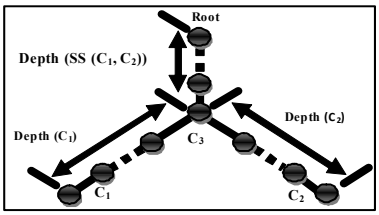

Figure 4: Example of SimComp measure computing.

4 SIMULATION RESULTS

In this section, we present an evaluation of the proposed SaaS services Discovery system for cloud environment. The proposed system is composed of two complementary modules: Publishing and Discovery SaaS service. Let us first present the test environment in which our system will be developed. We describe the used platform, the services data base that will constitute the simulation data, and the WordNet database as a calculation tool. WordNet is a lexical database for the English language developed by the cognitive science laboratory of Princeton University. Its purpose is to list, classify and relate in various ways the semantics and lexical content of the English language. For the SaaS services database, we used an open source corpus OWLS-TC4 version. This database describes a set of services web through OWLS documents, it contain 1083 web services distributed in 7 classes according to the conceived ontology. To evaluate the performance of the proposed system, we conducted two measurement experiments: the first part is devoted to evaluating to the Clustering algorithm while the second one aims at approving the discovery process using several types of simulation. We evaluate the different algorithms through a set of simulation tests.

4.1 Parameters Adjustment

To study and compare discovery systems, a measure must be defined to evaluate their performance. Different measures are used in the literature, which makes comparisons often difficult. In this paper, we choose two measures which are the precision rate and the recall rate. Then, Accuracy is used to evaluate the proposed Clustering algorithm. To evaluate the discovery mechanism, we choose the Recall and Accuracy rates, and the Execution Time. The formulas of all such metrics are defined as following:

Accuracy (A): is the ratio between the correct answers provided by the system and the total number of responses:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{T_p}}{\boldsymbol{T_p} + \boldsymbol{F_p}}$$

with ***Tp*** (True Positive) being a correct result, and considered as valid by the system; whereas ***Fp*** (False Positive) is an erroneous result, but considered valid by the system.

Recall (R): is the ratio between the correct answers provided by the system and the actual number of correct answers belonging to the data base:

$$\boldsymbol{Recall} = \frac{\boldsymbol{T_p}}{\boldsymbol{T_p} + \boldsymbol{F_n}}$$

with:

- ***Fn***: False Negatives (a correct result, and considered false by the system);

F-measure: is the harmonic means that takes into consideration both Accuracy *(A)* and Recall *(R)*:

$$\boldsymbol{F_{Measure}} = \frac{\boldsymbol{2 * A * R}}{\boldsymbol{R + A}}$$

4.2 Clustering Parameters

The proposed clustering algorithm is based on two parameters, which are the similarity threshold between services and service description types. For that, a study was carried out on the number of clusters formed with different service samples (917, 647, 307,170 services), using three different types of description: ***input***, ***output*** and ***input + output***. This clustering computing is implemented with different threshold values (0.4, 0.5, 0.6, 0.7). Table 2 illustrates the obtained results for determining the similarity threshold value and the service description type used in our Clustering approach.

According to the results obtained, the number of clusters described by the ***input*** and ***input + output*** descriptions is proportional to the number of services for the different thresholds values. Also, for the services using the output description, the cluster number is almost constant. This shows that there is services diversity describing by input or i***nput + output***, because with few services we have a limited

Clusters number of services. On the other hand, increasing the number of services lead to two cases: (i) with the ***input*** description, the clusters number remains almost the same because the new services will be classified in existing clusters; and, (ii) with the input and ***input*** + ***output*** description, the clusters number increases because there is no similarity between the services and the existing clusters. Hence, the output type is the best-suited description whereas the best threshold value is 0.5 because the clusters number is small and the services have been grouped according to their objectives.

Table 2: Definition of the Clustering parameters.

Threshold	*Description*	*Services number*			
		170	307	647	917
		Number of clusters			
0.4	Input	28	51	80	117
	Output	20	25	56	86
	In + Out	30	45	75	95
0.5	Input	35	73	112	198
	Output	32	35	36	55
	In + Out	40	56	80	135
0.6	Input	36	67	118	169
	Output	35	39	41	61
	In + Out	30	62	100	151
0.7	Input	19	38	56	100
	Output	31	42	50	101
	In + Out	58	99	151	24

4.3 Performance Measurement

In order to evaluate the proposed clustering algorithm, we conducted various experiments and comparisons. For this, we have used a set of 548 services distributed in 8 classes according to the conceived ontology (Klusch and Kapahnke, 2010). However, it is necessary to randomly select the services of each cluster and verify if the chosen cluster by the indexing approach is the appropriate one. Table 3 shows a sample of test results and performance measure in terms of Precision and Recall provided by our Clustering algorithm. As shown in Table 3, the proposed clustering algorithm has higher precision and recall for all identified cluster categories. For instance, the maximum error rate of our algorithm is 0.104 for the address information category. This error is due to the informal query processing which is a set of concepts.

Table 3: Performance measures of the proposed approach.

Cluster Categories	*Number of services*	*Precision %*	*Recall %*
Multimedia video	75	94.9	97.4
Services price	101	93.5	97.1
Job	81	95.2	97.5
Authors information	47	94	100
Address information	69	89.6	93.2
Hotel	93	93.9	97.8
Travel distance	54	93.1	98.1
Government	28	90.3	100

The rest of this section is devoted to present and validate the obtained results through the proposed multi-agent system for SaaS services discovery considering different types of simulation. Performance assessment will be achieved based on three metrics: Precision, Recall and F-measure. For this purpose, we use a set of requests selected randomly. We also use the similarity measure defined in section 3.3, and the value of threshold T is 0.5. Table 4 shows an example of processing the request *R < hotel info by country and town >* delivered by our discovery system. The maximum value of similarity is 0.75 obtained by the SaaS services discovery system. This is due to the choice of description method of the request.

The performance measurements of the proposed discovery approach are shown in figure 5. We present a study about the threshold influence on the Recall, Precision and F-measure rates using the previous request R. We note that when the extraction threshold of the results increases, the Recall rate decreases whereas the precision rate increases. An ideal discovery system provides answers whose Precision and Recall are equal to 1. When the extraction thresholds are less than 0.4, the Recall rate is high. Consequently, the system meets the needs of clients, but they are not accurate.

Table 4: Discovery result of the request R.

Name of service	**Similarity**	**Proposed approach**	**Human expert**
Country hotel service	0.75	Accept	Accept
Capital city country hotel service	0.75	Accept	Accept
City hôtel info	0.65	Accept	Accept
Inside geopolitical entity info	0.6	Accept	Accept
Person city country info	0.59	Accept	Refuse
Europe city hotel info	0.41	Refuse	Accept

To evaluate discovery systems, one must choose a good tradeoff between Recall and Precision. To this end, we use the comprehensive F-Measure. Figure 4 illustrates comparison results according to different thresholds values. We note that the maximum value of this harmonic mean is 0.84, whose Recall and Accuracy rates are at the maximum, which shows that our system is efficient. Another important remark is that our clustering-based discovery system has a significantly better response times than discovery systems without Clustering.

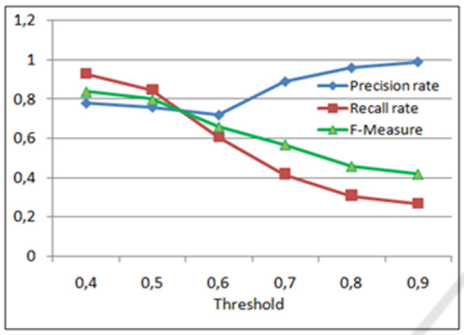

Figure 5: Comparison of performance measurement.

5 CONCLUSIONS

In this paper, we have introduced a new prototype for SaaS services discovery in cloud environment based on multi-agent system. The modeling scheme of this agent-based discovery system is divided in several main elements including services description and publication, clustering approach, request generation, matching process and discovery method. To improve the quality of SaaS service discovery, we have proposed a new algorithm for clustering SaaS services in cloud based on their semantic description. Experimental results show that our approach yields a significant performance improvement in terms of accuracy and requests processing time. An interesting future direction would be: (i) to define and use a SaaS service description model based on non-functional categorization attributes and other features of SaaS services; and, (ii) to utilize for each domain services other ontologies in order to improve search accuracy.

REFERENCES

Alfazi, A., Noor, T. H., Sheng, Q. Z., and Xu, Y. (2014). *Towards ontology-enhanced cloud services discovery.* In International Conference on Advanced Data Mining and Applications, pages 616–629. Springer.

Alfazi, A., Sheng, Q. Z., Qin, Y., and Noor, T. H. (2015). *Ontology-based automatic cloud service categorization for enhancing cloud service discovery.* In Enterprise Distributed Object Computing Conference (EDOC), 2015 IEEE 19th International, pages 151–158. IEEE.

Chen, H.-p. and Li, S.-c. (2011). *SRC: a service registry on cloud providing behavior-aware and qos-aware service discovery*. In Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference, pages 1–4. IEEE.

Elshater, Y., Elgazzar, K., and Martin, P. (2015). *Godiscovery: Web service discovery made efficient.* In Web Services (ICWS), 2015 IEEE International Conference, pages 711–716. IEEE.

Fan, H., Hussain, F. K., and Hussain, O. K. (2015a). *Semantic client-side approach for web personalization of SaaS-based cloud services*. Concurrency and Computation: Practice and Experience, 27:2144–2169.

Fan, H., Hussain, F. K., Younas, M., and Hussain, O. K. (2015b). *An integrated personalization framework for SaaS-based cloud services*. Future Generation Computer Systems, 53:157–173.

Guerfel, R., Sba¨ı, Z., and Ayed, R. B. (2015). *Towards a system for cloud service discovery and composition based on ontology*. Computational Collective Intelligence, pages 34–43. Springer.

Han, T. and Sim, K. M. (2010). *An ontology-enhanced cloud service discovery system*. In Proceedings of the International MultiConference of Engineers and Computer Scientists, volume 1, pages 17–19.

Klusch, M. and Kapahnke, P. (2010). *OWLS-TC*, version 4.0, http://projects.semwebcentral.org/projects/owlstc.

Li, S. and Chen, H.-p. (2014). *A context-aware framework for SaaS service dynamic discovery in clouds*. International Conference on Algorithms and Architectures for Parallel Processing, pages 671–684. Springer.

Parhi, M., Pattanayak, B. K., and Patra, M. R. (2014). *A multi-agent-based QoS-driven web service discovery and composition framework*. ARPN Journal of Engineering and Applied Sciences, VOL. 9, NO. 4, APRIL 2014.

Parhi, M., Pattanayak, B. K., and Patra, M. R. (2015): A multi-agent-based framework for cloud service description and discovery using ontology. Intelligent Computing, Communication and Devices, pages 337–348. Springer.

Pirro, G., Trunfio, P., Talia, D., Missier, P., and Goble, C. (2010). *Ergot: A semantic-based system for service discovery in distributed infrastructures*. In Cluster, Cloud and Grid Computing (CCGrid), 10th IEEE/ACM International Conference, pages 263–272. IEEE.

Wu, L., Garg, S. K., and Buyya, R. (2011). *SLA-based resource allocation for software as a service provider (saas) in cloud computing environments*. Cluster, Cloud and Grid Computing (CCGrid), 11th IEEE/ACM International Symposium, pages 195–204.

Wu, Z. and Palmer, M. (1994). *Verbs semantics and lexical selection*. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pages 133–138. Association for Computational Linguistics.