



Numerical Similarity Algorithms for Cloud Service Discovery and Selection System

Samer Hasan^{1*} Vatsavayi Valli Kumari¹

¹*Department of Computer Science & System Engineering,
College of Engineering (A), Andhra University, India*

* Corresponding author's Email: samer.hasan@yahoo.com

Abstract: Cloud computing delivers computing services over the Internet based on pay as you use financial model. Cloud service providers publish service advertisements in different formats on the Internet. Thus, cloud consumers should explore all provider websites using available search engines to find the appropriate cloud service. Unfortunately, existing search engines give a huge list of unrelated results that makes consumers spend a lot of time and effort to find the best matched cloud service. In this paper, we present a layered architecture for cloud service discovery and selection system to automate cloud service discovery and selection process, and remove the barriers between cloud service providers and consumers. Additionally, we present novel numerical algorithm for cloud services matching and compare it with existing algorithms. Proposed algorithm (XNSim) is independent of any external attribute value, while existing algorithm (SNSim) depends on the max and min values of the service attribute and (MNSim) algorithm depends on the max value of the service attribute. Comparison is done based on four parameters (number of matched services, execution time, average score and recall) to find the advantages and disadvantages of each one. XNSim algorithm showed better performance and more effectiveness over MNSim and SNSim.

Keywords: Cloud service, Service Discovery, numerical similarity, cloud computing.

1. Introduction

Cloud computing [1] enables ubiquitous and on-demand network access to a shared pool of configurable computing resources. There are three basic abstract delivery models for cloud services (SaaS, PaaS and IaaS). In Software as a Service (SaaS), consumers use applications running on providers' infrastructure. In Platform as a Service (PaaS), consumers deploy applications onto providers' infrastructure. Finally, in Infrastructure as a Service (IaaS), consumers deploy and arbitrary software and have a full access to the operating system. Nowadays, finding the appropriate cloud service is a time-consuming and tedious task. Firstly, Consumer should use the available search engines like (Google, Bing and Yahoo) with the appropriate keywords to find all cloud provider websites. Secondly, consumer should make a list of all

available services with their features. Finally, consumer selects the best appropriate cloud service and uses it. Unfortunately, available search engines are not designed to give a small set of exact matched cloud services. On the contrary, existing search engines show all the websites that have the search keywords without any semantic like (ParkCloud, CurrencyCloud). In addition to this, Cloud computing have some special characteristics [2] that makes the discovery process a hard and time consuming process. Some of these characteristics are: lack of standards, varied types and architectures of cloud services, dynamic behaviour of the cloud services and interoperable between cloud providers, tremendous growth in the number of cloud service providers and the geographical distribution of the cloud infrastructures that cross the country borders. Buyya et al. [3] wrote in 2013 "The discovery of cloud services is mostly done by human

intervention: a person (or a team of people) looks over the Internet to identify offerings that meet his or her needs. We imagine that in the near future it will be possible to find the solution that matches our needs by simply entering our request in a global digital market that trades cloud computing services.” They added “In this cloud marketplace, cloud service providers and consumers, trading cloud services as utilities”. In this paper, we present a layered architecture for cloud service discovery and selection systems to i) automate cloud service discovery and selection process, ii) reduce the time and effort of finding cloud services, iii) make service providers more visible to all consumers, iv) create a shared understanding of cloud service domain and v) improve the overall user experience. Additionally, we present novel numerical algorithm (XNSim) for cloud services matching that is independent of any external attribute values. XNSim algorithm reduces cloud services matching time and increases system efficiency. Furthermore, we compare between proposed algorithm and two existing algorithms. First one SNSim is presented in [4] depends on the max and min values of the service attribute and the second one is MNSim algorithm that depends on the max value of the service attribute only. Comparison is done based on four parameters (number of matched services, execution time, average Score and recall) to find the advantages and disadvantages of each one. The rest of this paper is organized as follows. Section 2 demonstrates cloud services discovery and selection challenges. Section 3 presents a layered architecture for cloud service discovery. Section 4 presents related works. Section 5 presents problem definition and matching algorithms. Section 6 presents the experiment results. Section 7 presents a conclusion and future works.

2. Cloud service discovery and selection challenges

Cloud services have some special characteristics that make their discovery process different from normal web service discovery process. Firstly, lack of standards for cloud service advertisement and discovery where each cloud service provider advertises cloud services without any standards. On the other hand, Web services adopt standard languages (Unified Service Description Language (USDL) and Web Services Description Language (WSDL)) to expose service interface and adopt the Universal, Description, Discovery and Integration (UDDI) to publish service advertisement to

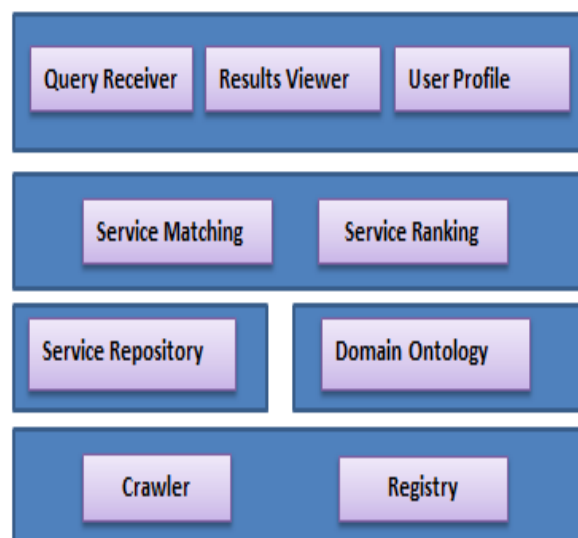


Figure.1 Architecture for cloud service discovery and selection system

registries. Secondly, cloud providers supply cloud services at different levels (data logic, business logic and infrastructure) with varied features that make Cloud service identification and categorization a complicated problem. Thirdly, finding the appropriate cloud service using general search engine is a time-consuming process with tremendous growth of the cloud services. Fourthly, legal challenge due to geographical distribution of cloud provider data centres that cross the country borders. Fifthly, dynamic behaviour of cloud services where new services appear around the clock while the old ones disappear. Sixthly, interoperable issue between cloud service providers. Seventhly, Cloud service is a membership service [3] where each cloud consumer should have an account with cloud provider for authentication and authorization.

3. Layered architecture for cloud service discovery and selection system

As shown in Fig. 1, the proposed architecture is divided into four layers.

- First layer uses crawler search engine or registry (or both) to collect cloud service advertisements from cloud providers.
- Second layer consists of domain ontology that creates a shared understating of cloud service domain and service repository that stores the collected cloud service advertisements.
- Third layer contains two components service matching component and Service ranking component. Service matching component matches between consumer

request and available cloud services. Service ranking component ranks the matched results.

- Forth layer contains three components: query receiver component, results viewer component and user profile component. Query receiver component receives user queries and translate them into appropriate format. Results viewer component displays the ordered matched list of cloud services. User profile component stores the user behaviour and feedback to improve the search results.

3.1 System operations

Fig. 2 shows in details cloud service discovery and selection operations as follows:

- Cloud service providers advertise cloud service descriptions in two approaches: central registry and /or websites.
- Service Identifier classifies and categorizes the detected cloud services into predefined categories.
- Service Identifier stores the identified cloud services into services repository or database.
- Cloud service consumer enters query (text or predefined parameters) through the user Interface.

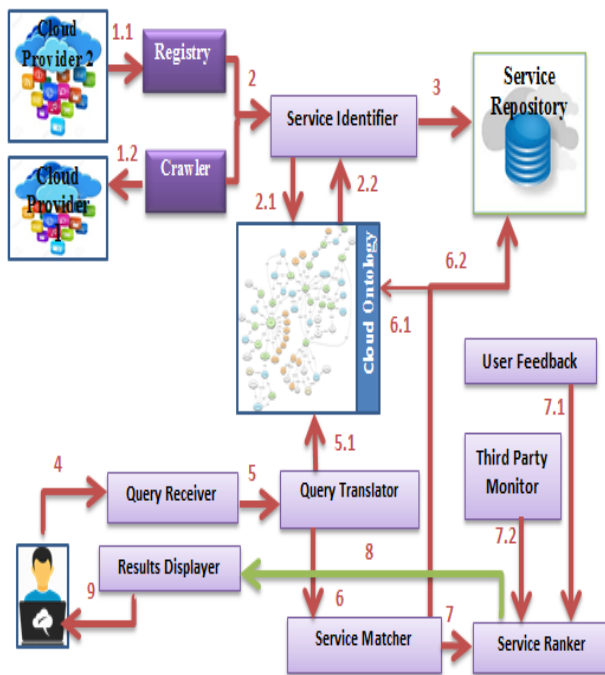


Figure.2 System operations

- Service Matching component receives consumer translated queries.
- Service Ranking component receives the matched services list.
- Results Viewer displays the ranked list to the cloud consumer.
- Cloud consumer selects the best matched service based on the task requirements.

4. Related works

Researchers proposed different architectures and frameworks for cloud services discovery and selection systems as following: [5, 6] presented an XML framework to model the existing cloud services in XML documents and query these documents using XQuery. [7] proposed a framework for comparing and ranking cloud services based on Service Measurement Index (SMI) identified by Cloud Service Measurement Index Consortium (CSMIC). In [8] Semantic search engine matches services based-on SPARQL language. [9] presented a Cloud Recommender System based on Owl ontology and SQL matching approach where consumers' requests are expressed as SQL queries. [10] built an Ontology based on Web Ontology Language (OWL) and uses SPARQL as query language with Protégé built-in semantic reasoner, unfortunately SPARQL language need experienced users. [11, 12, 13] treated cloud service discovery as Multi-Criteria Decision Making (MCDM) and apply different methods of MCDM like Analytic Hierarchy Process (AHP) to select the matched services. [14] matched between cloud service and user request based on SQL semantics, procedures and views. To overcome the limitation of SPARQL language [15] used WordNet ontology to expand the service description and user request semantically. In [16] user Interface receives user requests then generates SPARQL queries based-on natural language processing approach. [17] presented cloud service search and selection system based-on Skyline algorithm and SQL to meet user requirements. In [18] system used regular expressions, algebraic operations and SQL for matching user requests to service descriptions. [19] Presented four matching methods based-on ontology and QoS attributes: same comparison, equivalence matching, containing reasoning and similarity matching. [20] used cosine similarity for semantic matching. In [21] providers advertise their services based-on common metadata description and system use Description Logics (DLs) for service matching. [22] Calculated the cosine similarity between demand vector and each cluster centre. To overcome the limitations of proposed architectures and

automate the cloud service discovery and selection process, we presented a distributed layered architecture where user can enter a request for cloud service and get a ranked list of best matched cloud services. Although the large number of researches was done in the field of cloud service discovery and selection in the past few years, numerical similarity gained very low interest from researchers. [4, 19, 23, 24, 25] developed a cloud service discovery and selection systems and proposed three types of cloud services matching: similarity reasoning, compatibility reasoning, and numerical reasoning. [26] implemented three methods of service matching: Similarity, Equivalent and Numerical reasoning. [27] proposed three types of reasoning: numerical, similarity and compatibility. Numerical matching algorithms in all above proposed systems are dependent on external values of cloud service attribute, which increase search time and reduce system efficiency. To overcome those limitations, we present a novel algorithm for cloud services matching that is independent of any external attribute value.

5. Cloud services numerical similarity

5.1 Problem definition

The problem is finding the best matched cloud services (MCS) advertised by the cloud providers to satisfy the cloud user request (UR) in distributed and heterogeneous environment with the lowest effort and time. We represent the cloud services matching problem in mathematical model as follows:

1- We define each cloud service advertised by cloud provider as a set of attributes as follows:

$$CS_i = \{sa_1, sa_2, sa_3 \dots sa_n\} \quad (1)$$

$|CS_i| = n$ represents the total number of cloud service attributes. i represents cloud service unique ID. $sa_1, sa_2 \dots sa_n$ represent cloud service attributes like: RAM, Storage, CPU, Price, etc.

2- We define user request as a set of attributes as follows:

$$UR = \{ua_1, ua_2, ua_3 \dots ua_m\} \quad (2)$$

$|UR|=m$ represent the total number of cloud service attributes in user request. $ua_1, ua_2 \dots ua_m$ represent

cloud service attributes like: RAM, Storage, CPU, Price, etc.

3- We define the matching score (ms) between user request UR and cloud service CS_i as follows:

$$ms(UR, CS_i) = \frac{\sum_{j=1}^k Sim(ua_j, sa_j)}{k} \quad (3)$$

ua_j and sa_j represent the attribute value in user request and cloud service respectively. $K = \text{Max}(n, m)$ the max number between number of attributes in user request and available cloud service. $Sim(ua_j, sa_j)$ represents the similarity algorithm adopted by the system.

4- We define the following:

- Th is matching threshold for cloud services discovery and selection system.
- Ideal matching $ms = 1$ then Cloud service is exactly matching the user request.
- Anti-Ideal matching $ms = 0$ then Cloud service is not related to the user request.
- Matched service: $ms \geq th$.
- Unmatched service: $ms \leq th$

5- We define matched cloud services list MCS as a list of all available services that have a matching scores bigger than threshold. $ms > th$.

5.2 Numerical similarity algorithms

Numerical similarity is responsible for similarity between the attribute values of cloud services. It determines how much a value of attribute A is similar to another value of the same attribute A?. As an example, assume that cloud consumer is looking for a solution with 10 GB RAM and EC2 offers a solution with 12GB RAM while GoGrid offers a solution with 1GB RAM. It's very clear that EC2 solution is more similar to consumer request than GoGrid solution. Numerical similarity algorithm for cloud service discovery and selection system should be a symmetrical function with respect to user requested value. As a result to symmetrical function, values that have the same distance from user requested value should have the same similarity score. As an example, if user request is 10 GB RAM, GoGrid offer is 15 GB RAM and Amazon offer is 5 GB RAM then GoGrid distance is 5 and amazon distance is 5, so the two offers should have the same similarity score: $Sim(10, 15, RAM) = Sim(10, 5, RAM)$. Cloud service discovery and

selection system need to retrieve all alternatives with matching score (ms) bigger than threshold. Numerical similarity gained less interest from researchers than Semantic similarity. One of the early works on the numerical matching problem for cloud services is presented in [4]. Proposed algorithm increase the search time by considering min and max value of cloud attribute when calculating matching score ms between user request and cloud service. To overcome these limitations, we present two algorithms for numerical matching in cloud services. First one is MNSim which depend only on max value of cloud service attribute and the second one is XNSim which is independent of any external value. The following paragraphs explain each algorithm in details and paragraph 6 will show the experimental results for each algorithm.

5.2.1 Sim numerical similarity (SNSim)

In [4] Kang and Sim presented an algorithm (We refer it as SNSim in this paper) to calculate the Numerical Similarity between two attribute values as follows:

$$SNSim(x, y, a) = 1 - \frac{|x-y|}{\max((\max a - x), (x - \min a))} \quad (4)$$

As shown in fig. 3 $|x - y|$ represents the distance between user requested value x and alternative service value y . $\max a$ and $\min a$ represent the max and min value in service attribute a for all available cloud services respectively. $(\max a - x)$ represents the distance between user requested value and max value of attribute a in all alternative services. $(x - \min a)$ represents the distance between consumer requested value and min value of attribute a in all alternative services. SNSim algorithm need to consider the max and min values of cloud service attribute in all available cloud services to calculate the similarity with user request. SNSim is not the best algorithm for dynamic domain like cloud services.

5.2.2 Max numerical similarity (MNSim)

As a first improvement of SNSim presented in [4] We define MNSim algorithm between two attribute values based on the max value of this attribute only as follows:

$$MNSim(x, y, a) = 1 - \frac{|x-y|}{\max a} \quad (5)$$

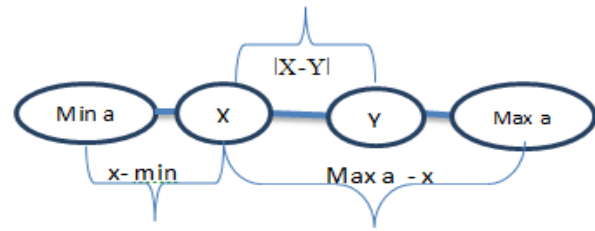


Figure.3 Sim numerical similarity

$\max a$ is the max value for attribute (a) in all available cloud services. Max numerical similarity (MNSim) algorithm need only to find the max value of cloud service attribute in all available cloud services to calculate the similarity with user request.

5.2.3 X numerical similarity (XNSim)

Cloud services are dynamic domain where new cloud services appear around the clock, while others disappear. Using dependent algorithms like SNSim [4] and MNSim for numerical matching in cloud services will increase the search time and reduces the system efficiency. Dependent algorithms need to consider external values to calculate the numerical similarity between user request and available cloud service. To overcome the limitations and disadvantages of dependent algorithms we present X numerical similarity algorithm (XNSim). XNSim algorithm is an independent algorithm that calculates the numerical similarity between user requested value and available cloud service value without considering any external values like (min and max). Proposed algorithm is based only on the requested attribute value and cloud service attribute value. It's independent of any external values as follows:

$$XNSim(x, y) = \begin{cases} 1 - \frac{|x-y|}{x}, & y < 2x \\ 0, & y \geq 2x \end{cases} \quad (6)$$

X represent the attribute value of user request and y represent the attribute value of available cloud service. If $y < 2x$ then y is similar to x and the similarity value is $XNSim(x, y)$. If $y \geq 2x$ then the distance between attributes is big and similarity is zero, thus cloud consumer need to change the query to get different results. XNSim calculates the numerical similarity with user request directly without any additional values like (min and max). XNSim is independent of any external attribute values.

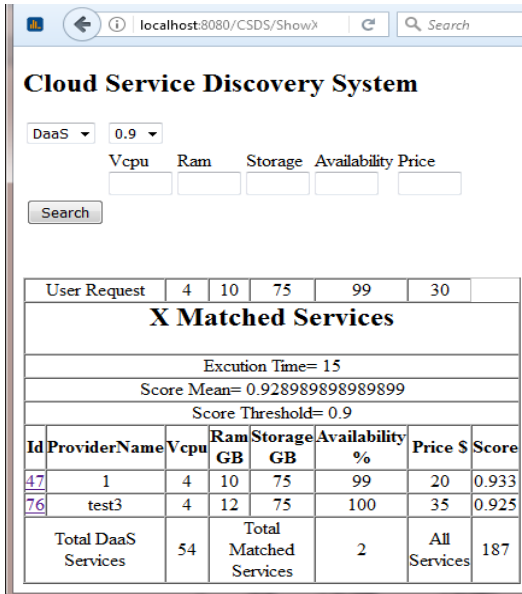


Figure.4 System output for XNSim algorithm

6. Results and discussion

We collected 187 cloud service advertisements from cloud provider websites using Google search engine and we used a core I5 laptop with 4 GB Ram for Experiments. We determined four parameters to compare proposed algorithms (MNSim and XNSim) with SNSim numerical matching algorithm presented in [4]. Comparison parameters are: number of matched services, execution time, average score and recall. Comparison is done based on one user request ($VCPU=4$, $Ram=10GB$, $Storage=75GB$, $Availability=99%$, $Price=30 USD/month$) and threshold values $\{0.01, 0.2, 0.4, 0.6, 0.8, 0.9\}$. Fig. 4 shows matched cloud services (MCS) list for X numerical similarity algorithm for $threshold=0.9$.

6.1 Number of matched services

The total number of matched services represents the number of services that will be displayed as a response for cloud service request. The bigger number of matched services needs more effort and time from consumers to find the best one. On the other hand, the smaller number of matched services means less chance for user to find the appropriate service. Fig. 5 shows the total number of matched services for each algorithm based on different values of threshold. MNSim algorithm shows the biggest number of matched services for all thresholds, on the other hand, SNSim algorithm shows the lowest

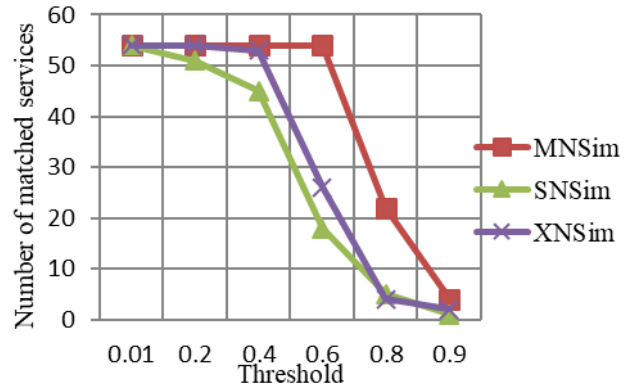


Figure.5 Total matched services per threshold

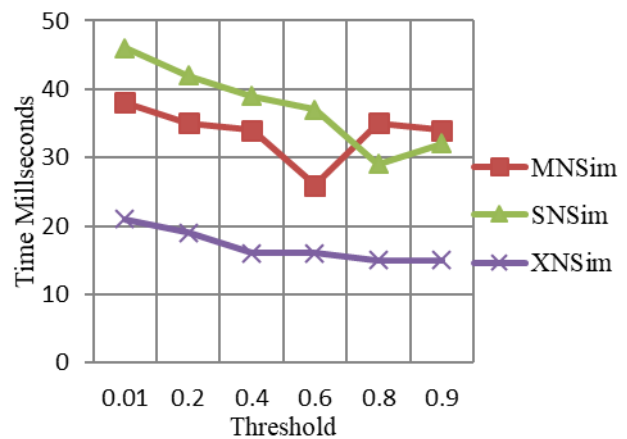


Figure.6 Execution time per threshold

number of matched services. XNSim algorithm shows an average number for matched services for all thresholds.

6.2 Execution time

Execution time is an important factor for any information retrieval system. The success of cloud service discovery and selection system is depending on the time that it takes to retrieve the matched cloud services. Execution time is the period between submitting the query and displaying the results on the screen for cloud consumer. It's not include the time needed to collect cloud service advertisements, identify and classify those cloud service. Fig. 6 shows the execution time for each algorithm based on different values of threshold. SNSim algorithm shows high execution time for all the thresholds because it calculates the distance between x value and the max and min values of cloud service attribute. MNSim shows high execution time at threshold = 0.8 and threshold = 0.9 because of the large number of matched services. Finally, XNSim sustains the lowest values of execution time for all thresholds because it is independent of min and max value of the cloud service attribute.

6.3 Average Score

Average matching score is an important parameter for cloud service discovery and selection algorithms. It's playing the main role in determining the best matching threshold (th) for cloud service discovery and selection system to satisfy cloud consumer requirements with lowest time and effort. Fig. 7 shows the average score for each algorithm based on the different values of threshold. Matching score affects the total number of matched services for each threshold. MNSim shows the highest average score for all the thresholds. On the other hand, SNSim and XNSim show almost the same values of average score for all thresholds.

6.4 Recall

Number of matched service doesn't show the accuracy and efficiency of information retrieval system. Recall and precision evaluate the completeness and effectiveness of information retrieval system [28]. Recall represents the number of matched services out of the total number of

relevant services. As shown in fig. 8 MNSim algorithm shows the highest recall percent for all thresholds while SNSim shows the lowest percent. XNSim shows an average percent between MNSim and SNSim for all thresholds.

7. Conclusion

Cloud computing becomes an important part of human life as an individual or organizations. The current process of finding the appropriate cloud service is tedious and time consuming for the end user. In this work we presented a layered architecture for cloud service discovery and selection system to automate cloud service discovery and selection process and remove the barriers between cloud providers and consumers. In proposed architecture cloud service provider and consumer can trade cloud service as a utility where cloud consumer can enter a request and get an ordered list of matched services. Proposed architecture automates cloud service discovery and selection process and makes service providers more visible to all consumers. Additionally, we presented a novel algorithm (XNSim) for numerical matching in cloud services which is independent of any external value. XNSim showed better results for cloud service discovery and selection system. XNSim algorithm reduced the search time and improved system efficiency in comparison with SNSim algorithm presented in [4] that depends on the max and min values of the service attribute, and MNSim that depends on the max value of the cloud service attribute. For the future work, we would like to collect more cloud service advertisements to expand our dataset and we would like to examine more cloud service matching algorithms to enhance the performance and the results.

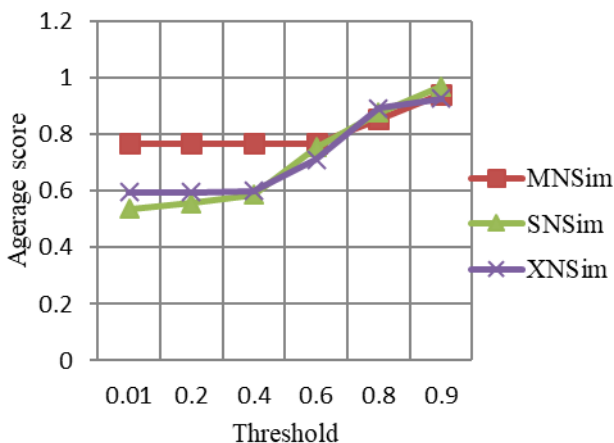


Figure.7 Average score per threshold

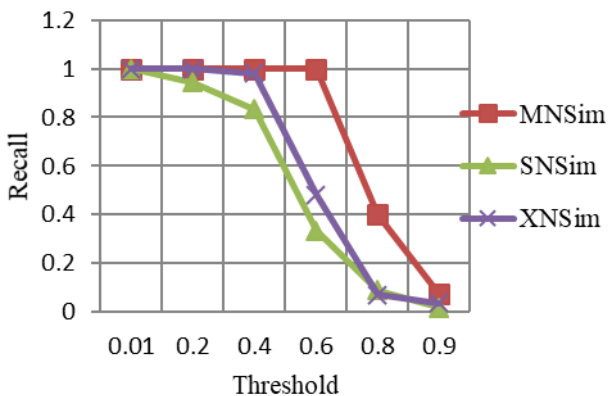


Figure.8 Recall per threshold

References

- [1] The NIST Definition of Cloud Computing (SP 800-145), 2011, is available at: <http://dx.doi.org/10.6028/NIST.SP.800-145>.
- [2] J. Gracia and E. Mena, "Semantic heterogeneity issues on the web", *IEEE Internet Computing*, Vol.16, No.5, pp. 60-67, 2012.
- [3] R. Buyya, C. Vecchiola, and S. Thamarai Selvi, "Mastering cloud computing: Foundations and applications programming", Morgan Kaufmann Publishers Inc., San Francisco, CA, 2013.
- [4] J. Kang, and K. M. Sim, "Towards agents and ontology for cloud service discovery", In: *Proc. of 2011 International Conference on Cyber-Enabled Distributed Computing and*

- Knowledge Discovery*, China, pp. 483–490, 2011.
- [5] H. Ma, K. Schewe and H. Xie, “Using XML for cloud specification and XQuery for service discovery”, In: *Proc. of the 12th International Conference on Information Integration and Web-based Applications & Services*, Paris, France, pp. 603-611, 2010.
- [6] A. Ruiz-Alvarez and M. Humphrey, “An automated approach to cloud storage service selection”, In: *Proc. of the 2nd International Workshop on Scientific Cloud Computing*, San Jose, CA, USA, pp. 39-48, 2011.
- [7] S. Garg, S. Versteeg, and R. Buyya, “SMICloud a framework for comparing and ranking cloud services”, In: *Proc. of Fourth IEEE International Conference on Utility and Cloud Computing*, Washington, DC, USA, pp. 210-218, 2011.
- [8] M. Rodriguez-Garca, R. Valencia-Garca, F. Garca-Snchez, J. Samper-Zapater, and I. Gil-Leiva, “Semantic annotation and retrieval of Services in the cloud”, *Distributed Computing and Artificial Intelligence, Advances in Intelligent Systems and Computing*, Vol. 217, pp. 69-77, 2013.
- [9] M. Zhang, R. Ranjan, A. Haller, D. Eorgakopoulos, M. Menzel, and S. Nepal, “An ontology-based system for cloud infrastructure services discovery”, In: *Proc. of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Pittsburgh, Pennsylvania, United States, pp. 524- 530, 2012.
- [10] A. Tahamtan, S. A. Beheshti, A. Anjomshoaa, and A. M. Tjoa, “A cloud repository and discovery framework based on a unified business and cloud service ontology”, *2012 IEEE Eighth World Congress on Services*, pp. 203-210, 2012.
- [11] Z. U. Rehman, O. K. Hussain, and F. K. Hussain, “IaaS cloud Selection using MCDM methods”, In: *Proc. of IEEE Ninth International Conference on e-Business Engineering*, China, pp. 246-251, 2012.
- [12] M. Godse and S. Mulik, “An approach for selecting software-as-a-service (SaaS) product”, In: *Proc. of IEEE International Conference on Cloud Computing*, Bangalore, India, pp. 155-158, 2009.
- [13] J. Park, and H. Jeong, “The QoS-based MCDM system for SaaS ERP applications with social network”, *Journal of Supercomputing*, Vol. 66, No. 2, pp. 614-632. 2012.
- [14] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, “Declarative recommender system for cloud infrastructure services selection”, In: *Proc. of the 9th International Conference on Economics of Grids, Clouds, Systems, and Services*, pp. 102–113, 2012.
- [15] Y. Afify, I. Moawad, N. Badr, and M. Tolba, “A semantic-based software-as-a-service (saas) discovery and selection system”, In: *Proc. of the 8th International Conference on Computer Engineering & Systems*, Cairo, Egypt, pp.57-63, 2013.
- [16] B. Saravana Balaji, N. Kb. Karthikeyan, R. S. RajKumar, “Fuzzy service conceptual ontology system for cloud service recommendation”, *Computers and Electrical Engineering*, pp. 1–12, 2016.
- [17] M. Abourezq, and A. Idrissi, “A cloud services research and selection system”, In: *Proc. of International Conference on Multimedia Computing and Systems (ICMCS)*, Morocco, pp. 1195 – 1199, 2014.
- [18] G. Khan, S. Sengupta, and A. Sarkar, “WSRM: a relational model for Web service discovery in enterprise cloud bus (ECB)”, In: *Proc. of 3rd International Conference on Eco-friendly Computing and Communication Systems (ICECCS)*, Mangalore, India, pp. 217-222, 2014.
- [19] L. Liu, X. Yao, L. Qin, and M. Zhang, “Ontology based service matching in cloud computing”, In: *Proc. of IEEE International Conference on Fuzzy Systems*, Beijing, China, pp. 2544-2550, 2014.
- [20] M. A. Rodríguez-García, R. Valencia-García, F. G. Sánchez, and J. J. S. Zapater, “Ontology-based annotation and retrieval of services in the cloud”, *Knowledge Based Systems*, Vol. 56, pp. 15-25, 2014.
- [21] A. V. Dastjerdi, S. G. H. Tabatabaei, and R. Buyya, “An effective architecture for automated appliance management system applying ontology-based cloud discovery”, In: *Proc. of 2010 10th IEEEACM International Conference on Cluster Cloud and Grid Computing*, Melbourne, Australia, pp. 104-112, 2010.
- [22] H. Jia-jing, W. Jin-dong, W. Na, and N. Kan, “Service discovery method based on two-step clustering”, In: *Proc. of the 4th International Conference on Computer Science and Network Technology (ICCSNT)*, Harbin, China, pp. 220 – 224, 2015.

- [23] K. M. Sim, "Agent-based cloud computing", *IEEE Transactions on Services Computing*, Vol. 5, No. 4, pp. 564-577, 2012.
- [24] J. Kang and K. M. Sim, "Ontology and search engine for cloud computing system", In: *Proc. of International Conference on System Science and Engineering*, China, pp. 276- 281, 2011.
- [25] J. Kang and K. M. Sim, "Cloudle a multi-criteria cloud service search engine", In: *Proc. of the 2010 IEEE Asia-Pacific Services Computing Conference*, China, pp. 339-346, 2010.
- [26] T. Han and K.M. Sim, "An ontology-enhanced cloud service discovery system", In: *Proc. of the International Multi-Conference of Engineers and Computer Scientists Conference*, Hong Kong, pp. 17-19, 2010.
- [27] M. Barati and R. St-Denis, "An architecture for semantic service discovery and realizability in cloud computing", In: *Proc. of the 6th International Conference on the Network of the Future (NOF)*, Montreal, Canada, pp.1-6, 2015.
- [28] V. Raghavan, P. Bollmann, and G. Jung, "A critical investigation of recall and precision as measures of retrieval system performance", *ACM Transactions on Information Systems*, Vol.7, No.3, p.205-229, 1989.