International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Comparing K-means, K-medoids and ISODATA Clustering Algorithms for a Cloud Service Search Engine

Walaa Reda, Hanan Elazhary, Ehab Hassanein

Abstract*: The evolution of cloud computing over the past few years is potentially one of the major advances in the history of computing. Cloud computing theoretically provides all computing needs as services. Accordingly, a large number of cloud service providers exist and the number is constantly increasing. This presents a significant problem for a user to find a relevant service provider, and calls for developing a specialized search engine to help users select suitable services matching their needs. Towards this goal, we developed a search engine that crawls the web sites of various service providers, extracts service attributes from their JavaScript Object Notation (JSON) files and normalizes the attributes in a service table. Those attributes are clustered using one of three different algorithms (K-means, K-medoids, and ISODATA). The requirements of a given user are then matched against the centroids of the various clusters to help obtain the closest match. In this paper, we compared the three algorithms with respect to time and accuracy. The ISODATA algorithm exhibited the best performance.

***Keywords*: Cloud Computing, Cloud Services, ISODATA, K-means, K-medoids.**

I. INTRODUCTION

Cloud computing can be defined as a large-scale and distributed computing paradigm driven by economies of scale, in which a pool of dynamically-scalable, virtualized, managed computing power, platforms, storage, and applications are delivered on demand to the customers through the Internet [1]. In fact, various cloud services can be deployed and accessed by the users online [2]. This is usually referred to as Everything as a Service (XaaS) [3]. Recently, cloud computing has been redefined [4] as "*a computing paradigm for providing anything as a service such that the services are virtualized, pooled, shared, and can be provisioned and released rapidly with minimal management effort. For the users, the services can be accessed conveniently, ubiquitously, across the network, dynamically, and on demand; can be configured with minimal interaction with the service provider; and are elastic and metered on a pay-per-use basis*." Unfortunately, there is shortage in standards and mechanisms to help users discover different types of cloud services that suit their needs [5].

A web search engine is a software system that is designed to crawl and search for information on the World Wide Web. Search engines typically index tens to hundreds of millions of web pages involving a comparable set of distinct terms [6], [7]. Users can use general search engines (such as Google) to search for cloud services, but they will be provided merely with providers' links. To go deeply into the details of each service and to compare alternatives from different providers, a user needs to browse each provider's web page, analyze the provided cloud services and compare the prices one by one. In other words, the user needs to browse the services manually. This is impractical due to many reasons that include [8]:

1) A user can only search for a specific service, not for a specific service specification.
2) Some of the provided web sites may be irrelevant.
3) Cloud service providers use different formats to describe the provided services and the possible Service Level Agreements (SLAs) on their web pages.
4) The web page browsing process is time consuming
5) The comparison among available cloud services needs special capabilities.

Accordingly, another approach in the literature suggests developing a specialized search engine [3] for this task. In other words, a cloud-specific search engine is required for discovering different cloud services and clustering search results to help efficiently and effectively in finding an appropriate service that closely matches the user's needs. The goal is to provide users with a Quality of Service (QoS) [9]. This paper presents a search engine developed for this purpose. This search engine not only crawls and indexes web pages, but it also has the ability to extract relevant service information from web pages employing JavaScript Object Notation (JSON) files. It then exploits clustering algorithms to cluster the discovered services and simplify the task of finding the one that closely matches the user's requirements. The three explored algorithms are K-means [10], K-medoids [10], and ISODATA [11].

Manuscript published on 30 September 2019
* Correspondence Author
Walaa Reda*, Information Systems Department, Cairo University, Cairo, Egypt. Email: walaareda27@yahoo.com
Hanan Elazhary, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia; Computers and Systems Department, Electronics Research Institute, Cairo, Egypt. Email: helazhary@uj.edu.sa
Ehab Hassanein, Information Systems Department, Cairo University, Cairo, Egypt. Email: e.ezat@fci-cu.edu.eg

Though the three algorithms are available to the user, the paper provides a comparative study among the three to highlight their pros and cons.

The rest of the paper is organized as follows: Section II provides background about the three clustering algorithms considered in the paper. Section III discusses related work in the literature. Section IV describes the developed search engine. Section V describes the experiments and results, in addition to a discussion. Finally, Section VI provides the conclusion and possible directions for future work.

II. BACKGROUND

In this section we discuss the three clustering algorithms considered in this paper.

A. K-Means Clustering Algorithm

The objective of the unsupervised K-means clustering algorithm [10] is to group data into K clusters such that similar data points or samples are grouped together. K is a predefined number based on experience or experimentation. The algorithm starts by selecting K random data points that represent the potential centers or centroids of the target clusters. It then iteratively assigns each data point to the closest centroid typically using Euclidean distance, and recalculates the centroids. The new centroid, which is the mean of the objects in the cluster, can be a logical one, not a real data object. The Euclidean distance d_{ij} between two points $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ described by p numeric attributes is computed as shown in equation (1), where *sqrt* stands for the square root.

$$d_{ij} = \text{sqrt}\left[(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+\dots+(x_{ip}-x_{jp})^2\right] \quad (1)$$

A stopping criterion is used such as when no change is encountered in one of the iterations (convergence) or a maximum number of iterations is reached. Fig. 1 shows a flow diagram of the algorithm.

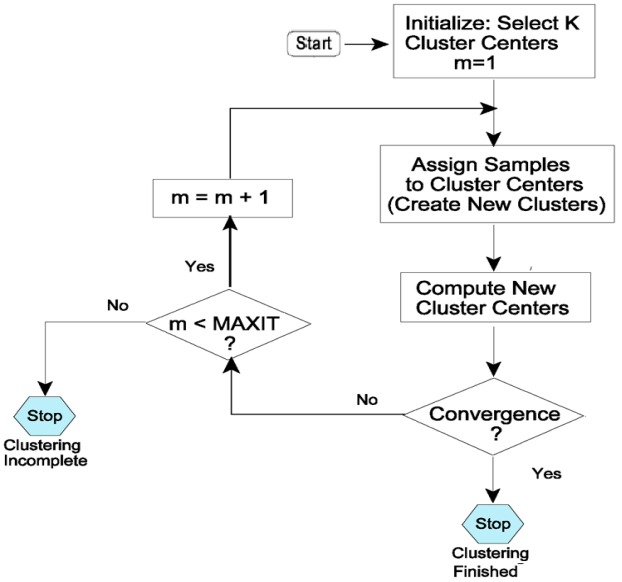

Fig. 1. Flow diagram of the K-means algorithm.

B. K-Medoids Clustering Algorithm

The K-means clustering algorithm is affected by outliers. Hence, the K-medoids algorithm [10] has been introduced to address this problem. It differs from the K-means algorithm in two aspects as follows:

1) Actual objects are used to represent the clusters (instead of the mean)
2) Partitioning of clusters is done based on minimizing the absolute error for all data points, The absolute error is the sum of the distances between each data point and the representative object of its cluster, as shown in equation (2), where E is the absolute error to be minimized, o_i is the representative object of cluster C_i, and p represents data points in the cluster.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} dist(\boldsymbol{p}, \boldsymbol{o}_i) \quad (2)$$

It is clear that the K-medoids problem is NP-hard, and so many heuristic and approximate algorithms have been proposed to speed up the computation [10], [12], [13].

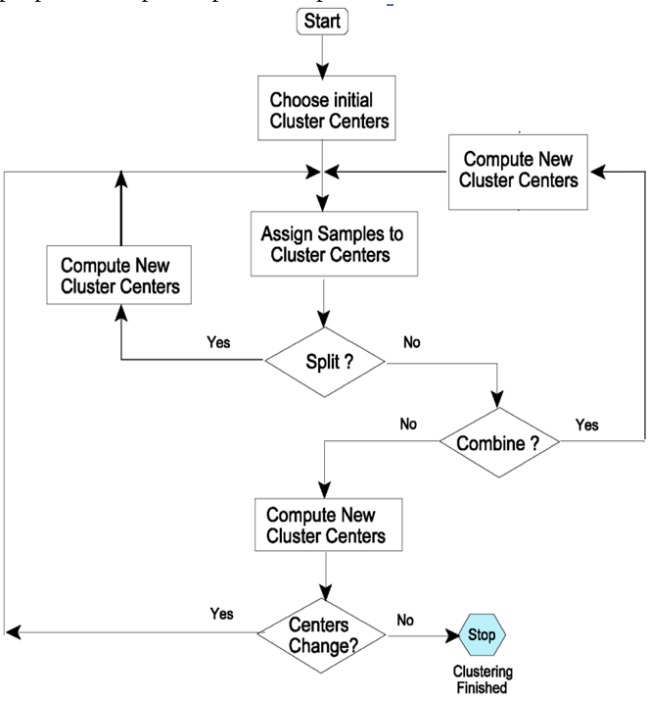

Fig. 2.Flow diagram of the ISODATA algorithm.

C. ISODATA Clustering Algorithm

ISODATA [11] stands for iterative self-organizing data analysis technique which allows the number of clusters to be adjusted automatically by iteratively merging close clusters and splitting clusters with large standard deviation(s). The ISODATA algorithm is more flexible than its K-means counterpart at the expense of having to experiment with more parameters. The algorithm starts with random centroids or cluster centers to which data points or samples are assigned based on minimum distance as in the K-means algorithm.

The algorithm, however, proceeds in a different manner as shown in the flowchart in Fig. 2. Iteratively, the algorithm attempts to split or merge clusters and reassign samples to the new clusters by following the following steps:

1) Clusters are split in case the standard deviation within any cluster is greater than a user-defined threshold
2) Clusters are merged in case the distance between them is less than another user-defined threshold

This continues until a predefined stopping criterion is satisfied. Possible stopping criteria are as follow:

1) the average distances between the centroids are smaller than a user-defined threshold
2) the average change in the distances between the centroids is smaller than a user-defined threshold
3) the maximum number of iterations is reached

The algorithm can be also controlled using other parameters such as:

1) the minimum number of samples per class
2) the maximum number of classes desired
3) the maximum number of combined classes each iteration

Similar to the K-medoids algorithm, many heuristic and approximate algorithms have been proposed to speed up the computation [14, 15].

III. RELATED WORK

Several research studies in the literature have been concerned with using search engines for automated discovery of suitable cloud services matching the user QoS requirements. One approach proposed the used of ontologies to match the user QoS requirements against features of the candidate cloud services. Nevertheless, they differed in how cloud service information is extracted before the matching process starts. For example, Han and Sim [16] used general search engines such as Google for searching web sites and then used an agent to filter them before the matching process starts. But, the filtering details were not provided. They also used agents to realize their proposed search engine. Nithya et al. [6] assumed that cloud service providers register their services into the database of a specialized search engine. Kang and Sim [8] made a similar assumption, but also provided the details of a web portal for communicating with the users. Both of them also proposed the use of agents to realize their systems. Liu et al. [17] proposed using agents to extract information of candidate cloud services by periodically accessing the web services of the corresponding cloud service providers. This is in addition to using agents to realize their search engine as in the above research studies. Moreover, they exploited an ontology to translate the user QoS requirements to facilitate matching them against the retrieved cloud service information.

Other researchers approached the problem in a different manner. For example, Elazhary [3] proposed a framework for a more comprehensive cloud service broker that addresses all aspects of the process including discovery, normalization, and matching and ranking of cloud services against the users' QoS requirements. They also proposed a decision support module to negotiate the available services with the users in case an exact match is absent. This is in addition to service monitoring and evaluation techniques to ensure that the QoS is maintained by cloud service providers and to provide users with reliable consultation. Gong and Sim [18] developed a specialized search engine that utilizes the K-means clustering algorithm for finding the best matching cloud services against the user QoS requirements. To deal with non-numeric data, they proposed using simple set intersection to reflect the similarity of two entries that represent the user's requirements and a candidate cloud service. In this paper, we propose to develop a search engine that covers a considerable portion of the proposed framework above [3] and to consider other possible clustering algorithms.

IV. THE PROPOSED SEARCH ENGINE

In this section, we discuss the details of the implemented cloud service search engine.

Id	Provider	OS_Type	OS_Name	OS_Version	SQLSupport	VCPU	RAM	InstanceStorage	SSDStorage	PriceHourly
1	amazon	windows	server	2008	TRUE	16	1024	1200	80	50
2	amazon	linux	centos	5.7	TRUE	2	10	100	0	0.2
3	rackspace	linux	rhel	5.7	FALSE	8	4	800	0	1.4
4	gogrid	windows	server	2008	FALSE	32	64	1000	80	7.2
5	amazon	windows	server	2008	TRUE	16	1024	1300	80	5
6	gogrid	linux	ubuntu	12.04	FALSE	64	256	600	40	0.1
7	appharbor	linux	ubuntu	12.04	FALSE	1	1024	10	20	0
8	appharbor	linux	ubuntu	12.04	FALSE	2	1024	15	80	0.06
9	appharbor	linux	ubuntu	12.04	FALSE	4	1024	20	80	0.34

Fig. 3. A service table.

A. Cloud Service Discovery and Normalization

There are several ways to collect data regarding candidate cloud services. One possibility is to collect data manually, but this is impractical because the cloud services may change and because of the increasing number of service providers. Hence, it is better to crawl for each cloud service provider [18].

In the proposed search engine, crawlers extract required service information from the JSON files of the service providers' sites. JSON files are alternatives to XML files, which are increasingly used on the web since they are more readable and writable by humans in comparison to XML. The extracted information (service attributes) is then normalized into a cloud service table as shown in Figure 3.

B. Clustering Algorithms

After normalizing the collected data, we use one of three possible clustering algorithms to cluster the data before classifying the user's requirements into the best matching cluster.

The three clustering algorithms that we employ in our search engine are as follows:

1) K-means algorithm [10]
2) An efficient version of the K-medoids algorithm, called Partitioning around Medoids (PAM) [12].
3) An efficient version of the ISODATA algorithm [15].

As shown in Figure 3, some service attributes are numeric such as the number of virtual CPU cores, the memory size (RAM), the instance storage, the SSD storage, and the hourly price; while others are non-numeric such as the provider name, the operating system type, and SQL support. Since Euclidean distance is used with numeric data only, in the above clustering algorithms, we replaced it with a dissimilarity measure for attributes of mixed types [10].

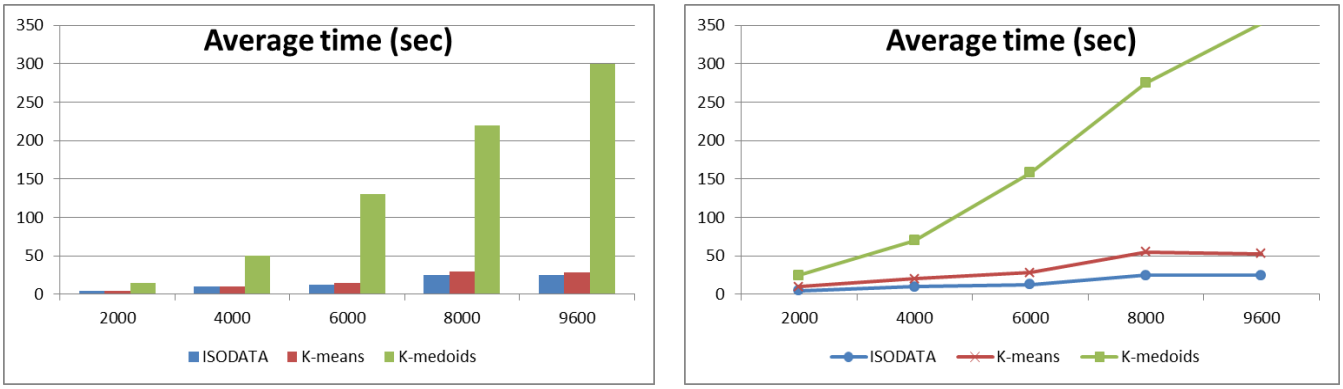

Fig. 4. Time performance of the clustering algorithms.

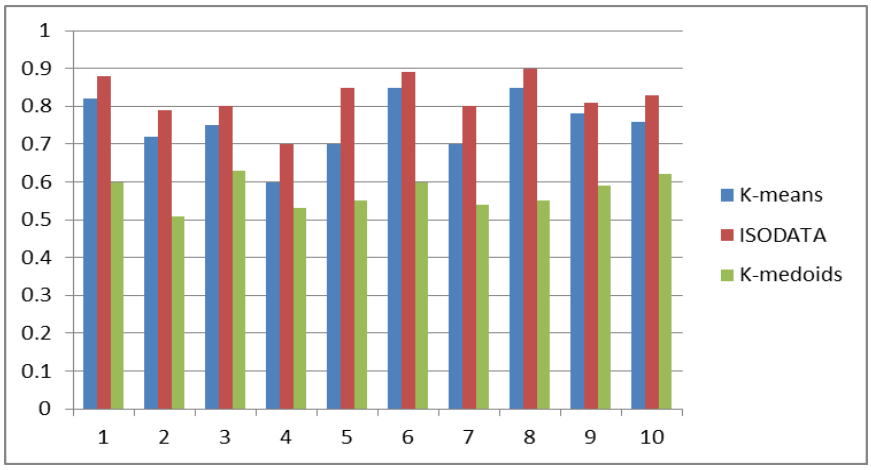

Fig. 5. Accuracy of the clustering algorithms.

C. The Matching Process

Given some user's requirements, the search engine matches them against the centroids of the various clusters and returns the closest matching cluster. Fine-grained matching then proceeds to select the closest service within the cluster. It is worth emphasizing that we use the dissimilarity measure for attributes of mixed types mentioned above [10].

D. The User Interfaces

The search engine provides three web-based interfaces for the admin, the provider, and the user respectively.

1) The admin can:
   - start crawling to a specific URL to obtain the JSON file and extract attributes of cloud services
   - run clustering offline
   - create/edit/delete a given provider profile
   - search for a suitable service matching her requirements like a normal user
2) The provider can:
   - send a request to the admin to create her profile
   - log into the system
   - edit her profile
   - edit her account
   - search for a suitable service matching her requirements like a normal user
3) The user can search for a suitable service matching her requirements.

V. EXPERIMENTAL RESULTS AND DISCUSSION

We gathered real data from service providers such as Amazon EC2, Rackspace, GoGrid and AppHarbor as shown in Figure 3. The normalized service attributes included Service Provider, OS Type, OS Name, OS Version, SQL Support, Virtual CPU, RAM, Instance Storage, SSD Storage, and Hourly Price. Then we experimented with the three clustering algorithms.

A. Time

To compare the three clustering algorithms with

respect to the average clustering time, for each algorithm, we experimented with data of various sizes; 2000, 4000, 6000, 8000 and 9600. For each case, the experiment was repeated ten times and the average was computed. As shown in Figure 4, the ISODATA achieved significantly better performance than K-means and K-medoids.

B. Accuracy

We experimented with ten different user queries with increasing number of requirements, and measured the accuracy in terms of the similarity between the matched cluster centroid and the user requirements. As shown in Figure 5, the ISODATA algorithm also showed the best performance in comparison to K-means and K-medoids

VI. CONCLUSION

This paper presented a specialized cloud service search engine that can crawl the web sites of the service providers to automatically extract the service attributes from their JSON files and then normalize them in a service table. The attributes are then clustered for subsequent matching with a given user's requirements set to select the closest match. Towards this goal we compared the K-means, K-medoids, and ISODATA clustering algorithms; and the ISODATA showed the best performance in terms of time and accuracy.

As future work, we will extend the search engine to deal with data formats other than JSON to increase the number of handled service providers. We also intend to deal with more attributes of various data types.

REFERENCES

1. I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," in *Proc. IEEE Grid Computing Environments*, Texas, 2008, pp. 1-10.
2. L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *Proc. Grid Computing Environments Workshop*, Austin, TX, USA, 2008, pp. 42-51.
3. H. Elazhary, "User-centric cloud service broker," *International Journal of Computer Applications*, 154(7), 2016.
4. H. Elazhary, "Internet of Things (IoT), mobile cloud, cloudlet, mobile IoT, IoT cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *Journal of Network and Computer Applications*, 128, pp. 105-140, 2019.
5. P. Sheu, S. Wang, Q. Wang, K. Hao, and R. Paul, "Semantic computing, cloud computing, and semantic search engine," in *Proc. IEEE International Conference on Semantic Computing*, 2009.
6. G. Nithya, M. Engels, S. Gayathri, and D. Ganesh, "Cloud based search engine," *Journal of Engineering and Applied Sciences*, 10(5), 2015.
7. L. Sun, H. Dong, F. Hussain, O. Hussain and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications*, 45, 2014.
8. J. Kang and K. Sim, "A cloud portal with a cloud service search engine," in *Proc. International Conference on Information and Intelligent Computing*, 2011.
9. H. Elazhary and S. Gokhale, "Integrating path computation and precomputation for quality-of-service provisioning," in *Proc. ISCC'04*, Alexandria, Egypt, 2004.
10. J. Han, M. Kamber, and J. Pei, *Data Mining Concepts and Techniques*, 3rd ed., Morgan Kaufmann, 2012.
11. A. Vassilaros, *ISODATA*. Available: http://web.pdx.edu/~jduh/courses/Archive/geog481w07/Students/Vassilaros_ISODATA.pdf
12. *K-Medoids clustering with example*. Available: https://www.geeksforgeeks.org/ml-k-medoids-clustering-with-example/
13. H. Park and C. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications*, 36(2), pp. 3336-3341, 2009.
14. N. Memarsadeghi, N. Netanyahu, and J. LeMoigne, "A fast implementation of the ISODATA clustering algorithm," *International Journal of Computational Geometry and Applications*, 2006.
15. N. Memarsadeghi, D. Mount, N. Netanyahu, and J. LeMoigne, *A fast implementation of the ISODATA clustering algorithm*. Available: https://www.cs.umd.edu/~mount/Projects/ISODATA/
16. T. Han and K. Sim, "An ontology-enhanced cloud service discovery system," *in Proc. the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2010.
17. D. Liu, W. Xing, X. Che, and P. Bao, "A centralized service discovery approach for agent-based cloud computing system," *The Open Cybernetics & Systemics Journal*, 9, pp. 526-535, 2015.
18. S. Gong and K. Sim, "CB-cloudle: A centroid-based cloud service," in *Proc. the International MultiConference of Engineers and Computer Scientists*, Hong Kong, 2014.

AUTHORS PROFILE

Walaa Reda earned her B.Sc. from the Department of Computer Science and Information Systems, MUST University, 6th of October, Egypt. Currently she is a teaching assistant in the same department and is pursuing her M.Sc. degree in computers and information systems at Cairo University.

Hanan Elazhary earned her B.Sc. and M.Sc. degrees from the Department of Electronics and Communications Engineering, Cairo University. She earned her Ph.D. degree in Computer Science and Engineering from the University of Connecticut, USA. Currently, she is a professor in the College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia, and the Computers and Systems Department, Electronics Research Institute, Cairo, Egypt. She has numerous publications on cloud computing research.

Ehab Hassanein earned a Master degree in Computer Science from Northwestern University in 1989 and a Ph.D. degree from the same university in December 1992. He is the former Chairman of the Information Systems Department, Faculty of Computers and Information, Cairo University, Egypt. He is currently a professor in the same department.