

An innovative and efficient approach for searching and selecting web services operations

Sara Rekkal¹, Kahina Rekkal²

¹Laboratory of LRI, Computer Science Department, Badji Mokhtar–Annaba University, Annaba, Algeria

²Electrical Engineering Department, Salhi Ahmed–Naama University Center, Naama, Algeria

Article Info

Article history:

Received Feb 15, 2024

Revised Sep 12, 2024

Accepted Oct 1, 2024

Keywords:

Clusters

Quality of services

Response time

Searching for web services

Selection of web services

Web services operations

ABSTRACT

The marketing of web services on the internet continues to increase, resulting in an increasing number of web services and, therefore, operations offering equivalent functionalities. As a consequence, finding an appropriate web service (operation) for a particular task has become a difficult challenge, taking a lot of time and leading to an insufficient selection of relevant services. This work aims to propose a new approach facilitating the search and localization of relevant web services (operations) in an acceptable time while ensuring the totality of the response. This approach is divided into three crucial phases. The first step involves collecting web services from various universal description, discovery, and integration (UDDI) registries and different domains and forming specialized sub-registries. The second phase involves the extraction of operations from various services, followed by a similarity study whose goal is the formation of clusters of similar operations. The third phase processes user requests by identifying the desired features. A list of operations is then provided to the client, including the non-functional properties, from which they select the one that best meets their needs and begin to invoke it.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sara Rekkal

Laboratory of LRI, Computer Science Department, Badji Mokhtar–Annaba University

23000 Annaba, Algeria

Email: sara.rekkal@univ-annaba.dz

1. INTRODUCTION

The computer science field has evolved a lot in recent years. The size and complexity of software are constantly growing. This is due to the new and numerous needs that are more and more complex. With this evolution, several approaches have emerged to improve the productivity and efficiency of software, including service-oriented architecture (SOA), which has been accompanied by an exponential increase in the number of services and service providers.

In this movement, the search for the best services represents a real challenge. Finding the right service to invoke implies having researched and discovered a set of potential services. It is then a question of searching and selecting the most relevant to invoke. Searching for and selecting relevant web services presents several challenges related to:

- Heterogeneity of technologies: Various technologies and protocols (*e.g.*, simple object access protocol (SOAP), representational state transfer (REST), remote procedure call (gRPC)) can be used to construct web services. Because of these technologies' diversity, the search and selection of services can be tailored to meet the unique requirements of an application that is more complex.

- b. Service quality: It might be challenging to assess the level of services that are offered. Cost, effectiveness, availability, dependability, security, and other factors can all affect how good a service is. It is critical to establish selection processes that consider these factors.
- c. Dynamic environments: These settings allow for frequent additions, deletions, and updates of services. Effective management and adaptation to changes in these dynamics is crucial for service research and selection.
- d. Security: To prevent assaults and protect data privacy, it is necessary to ensure the security of specific services.
- e. Complexity management: Searching and choosing among the many options offered might get complicated. It is critical to have efficient systems in place for filtering, categorizing, and presenting services so that users can choose more easily.
- f. Interoperability: There may be problems with interoperability when web services are established by various entities. To guarantee successful integration across chosen services, these distinctions must be considered during the service search and selection process.
- g. Absence of representation of non-functional properties: Choosing services that best satisfy user needs is penalized in the absence of quality of service (QoS) representations [1]–[3].

Effective solutions must be developed to address these challenges and enable efficient and secure use of web services in distributed applications. This paper first addresses the “complexity management” issue by proposing an innovative method that allows a good representation of web service operations classified by their functionalities to facilitate the search and selection process. It is important to clarify at this point that a client seeks a service for its functionalities, so there is an interest in orienting the service search towards the operation search. Once the operation is found, the service that offers it is discovered. Second, this approach solves the challenge of a “dynamic environment.” Similar operations are grouped into clusters in order to replace a failing operation with another similar one. Moreover, this method ensures the updating of the database in case of deletion or modification of an operation by the service provider. Furthermore, the quality of services is treated differently compared to existing work. The system offers customers the corresponding operations as well as their qualities, and it is up to them to select those that suit them best. Finally, this work ensures security by encapsulating both the client and the provider. A mediator system ensures the invocation of the desired operations and returns the responses to the client. The remainder of this document is structured as follows: section 2 discusses related work, section 3 introduces the proposed approach, section 4 outlines the experimental results, and section 5 concludes the paper.

2. RELATED WORK

Web services remain extremely complex technologies, generating various challenges. In this context, numerous initiatives have been undertaken to guarantee both their availability and customer satisfaction. Huang and Zhao [4] devise a service discovery approach using multidimensional service representations, employing methods like Word2Vec, embeddings from language models (ELMo), and term frequency-inverse document frequency. They construct a similarity matrix based on word frequency, static context, and dynamic context features, enhancing accuracy through convolution, pooling, and optimization in a neural matching network. Candidate services are rated using predicted scores from the matching network, yielding target services for specific queries. Chen and Kuang [5] introduce a novel method for web service retrieval, focusing on comparing the similarity of service interfaces. Their algorithm automatically adjusts keyword weights in requests to improve result accuracy, assigning lower weights to keywords shared across the service set.

Achir *et al.* [6] present a taxonomy that they use to categorize service discovery methods in an internet of things (IoT) setting. They assess these techniques based on a range of factors, going into how well they work in various situations and circumstances as well as their benefits and drawbacks. They also point out difficulties and offer ideas for future avenues for this field of study. Zarei and Gaedke [7] present distributed service discovery (DISCO) “Web service discovery chatbot,” a chatbot-driven approach that lets consumers choose web services with ease. DISCO makes the interaction simpler and removes the requirement for users to be technically knowledgeable by using conversational interfaces.

In order to find online services in social repositories, Lizarralde *et al.* [8] suggest extracting characteristics from service descriptions using variational autoencoders. To train the autoencoder, they use a dataset of 17,113 services from the “*ProgrammableWeb.com*” application programming interface (API). Nguyen *et al.* [9] present a sophisticated clustering-based technique for web service suggestions with the goal of achieving a wider range of results. They provide a distinct set of recommendations by taking into account functional interest, diversity traits, and quality of service (QoS) preferences. By use of web service graph

construction and varied-width clustering, they improve the quality of online service suggestions by improving recommendation accuracy.

Abramowicz *et al.* [10] present an architecture for filtering and clustering web services that makes use of user and application profiles that are specified using web ontology language for services (OWL-S). In order to improve data refinement and save execution time, filters compare service-related clusters using clustering analysis. A clustering technique is used in [11] to group heterogeneous services and concentrates on web service discovery using OWL-S in conjunction with web services description language (WSDL) to define service semantics.

A mediator built on a global ontology drawn from local ontologies obtained by hospital information systems (HIS) is used in [12] to provide a semantic interoperability architecture. To handle possible semantic conflicts, web services are tagged with local ontologies from applications. The mediator helps compose web services for complicated queries, but it does not go into great depth about how to implement it.

Using a machine learning-based methodology to predict QoS features from source code measurements and assess service reputation through credibility and usage history, Rangarajan [13] present a novel architecture for web service discovery and selection. Dynamic Hilbert clustering, a technique for classifying online services according to convex set similarity, is introduced in [14]. By calculating the mathematical similarity of SOAP messages, the method groups them into very similar clusters.

Mohammed *et al.* [15] propose a deep learning framework to enhance service composition quality for cloud users, with a focus on location awareness. Their structure initially reduces data dimensions and integrates particle swarm optimization with a deep learning long short-term memory network to accurately estimate QoS values, considering location. The framework aims to optimize service selection to reduce consumer costs and demonstrates superior prediction and composition accuracy compared to existing models on real datasets.

Song and Co [16] presents a technique for developing unified modeling language (UML)-based cloud services, integrating cloud and UML modeling elements into a hierarchical metamodel. This facilitates the creation of service-oriented cloud modeling methodologies based on model-driven architecture (MDA) and model-view-controller (MVC), simplifying the design of cloud applications and enabling hierarchical reuse models.

3. PROPOSED APPROACH

As previously stated, this research aims to introduce a methodology for discovering and selecting web services operations. It is important to note that clients typically seek services based on their functionalities. Therefore, it is advantageous to focus the search on finding specific operations. Once the desired operation is identified, the corresponding service is identified too. To achieve this objective, a reorganization of the web services space is mandatory. Since it is difficult, even impossible, to modify the universal description, discovery, and integration (UDDI) registries, a mediator was proposed. The latter performs the following tasks (which will be discussed in detail):

- Organizing web services according to their different fields.
- Forming communities of similar operations.
- Handling user requests.

3.1. Organizing web services according to their different fields

This organization collects WSDLs from UDDIs across various fields and categorizes them into specialized sub-registers. Each WSDL is assigned to a specific domain, such as weather, math operations, or finance, to streamline the research process. This structured organization simplifies access and enhances the efficiency of service discovery, as illustrated in Figure 1.

3.2. Forming communities of similar operations

This step is of utmost importance for the following:

- a. Research: If an operation is discovered, all similar ones are discovered too. This reduces search time.
- b. Substitution: One operation is replaced by another if the latter fails.

3.3. Communities' formation steps

A community is defined as a set of similar operations. Two operations are considered similar if they share identical inputs and outputs. Let O and O' be two operations extracted from different web services $S1$ and $S2$. The similarity between O and O' is calculated according to the Algorithm 1:

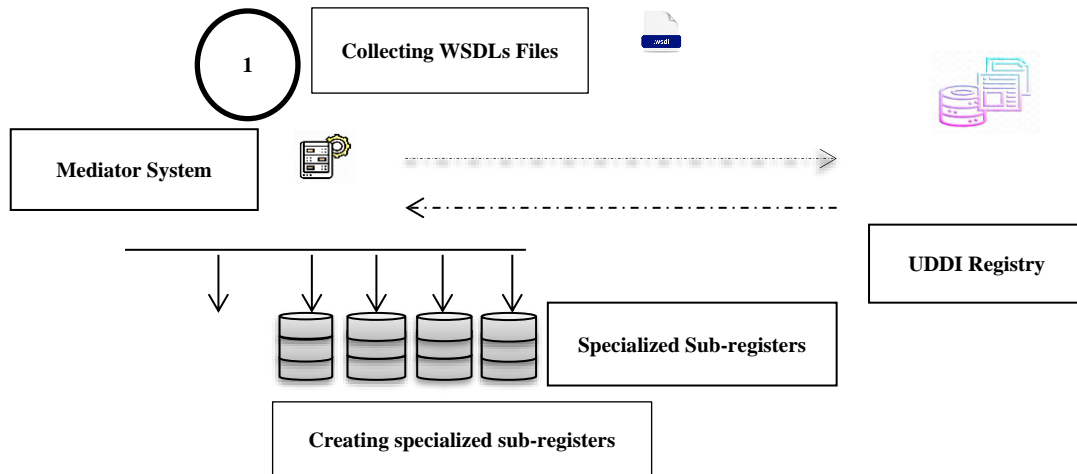


Figure 1. Organizing web services by field

Algorithm 1. Proposed algorithm

double SimWord (word1, word2)

// SimWord(): Measures the degree of similarity between two words.

If word1 or word2 is not in WordNet then

Score ← Jaro-Winkler(word1, word2)

else

Score ← Wu-Palmer(word1, word2)

return Score

double SimIdent (Ident1, Ident2)

// SimIdent(): Measures the degree of similarity between two identifiers. An identifier can be made up of several words. Hungarian-Maximum-Matching finds a matching of maximum scores and calculates their average.

for (int i = 0; i < Ident1.length; i++)

for (int j = 0; j < Ident2.length; j++)

Mat[i][j] = SimWord(Ident1[i], Ident2[j])

FinalScore = Hungarian-Maximum-Matching (Mat[i][j])

return (FinalScore)

double SimType (Type1, Type2)

// SimType(): Measures the degree of similarity between two types.

Score ← Min [Sim (T, T'), Sim (T', T)]/2

return Score.

double SimParts(partList 1, partList 2)

// SimParts(): Measures the degree of similarity between two parameters. Each parameter has an identifier and a type.

ScoreSimIdent ← SimIdent (partList 1.ident, partList 2.ident)

ScoreSimType ← SimTypes (partList 1.type, partList 2.type)

FinalSimScore ← (ScoreSimIdent + ScoreSimType)/2

return FinalSimScore.

double SimMessages(msg1, msg2)

// SimMessage(): Measures the degree of similarity between two messages (input/output). Each message has a list of parameters.

for (int i = 0; i < msg1.partList.length; i++)

for (int j = 0; j < msg2.partList.length; j++)

Mat[i][j] = SimParts (msg1.partList[i], msg2.partList[j])

FinalScore = Hungarian-Maximum-Matching (Mat[i][j])

return FinalScore

double SimOps(O, O')

// SimOps(): Measures the degree of similarity between two operations.

ScoreIdent ← SimIdent (O.Ident, O'.Ident')

ScoreSimInpMsg ← SimMessages (O.inputMessage, O'.inputMessage)

ScoreSimOutMsg ← SimMessages (O.OutputMessage, O'.OutputMessage)

FinalScore ← (ScoreIdent + (ScoreSimOutMsg + ScoreSimInpMsg)/2)/2

return FinalScore.

Key Points:

- a. *SimWord()* employs Wu-Palmer similarity [17] when both words exist in WordNet; otherwise, it utilizes Jaro-Winkler similarity [18].
- b. The Hungarian maximum matching algorithm [19] determines the maximum scores and computes their average similarity score.
- c. *SimType()* relies on a similarity table for types, as detailed in sources [20], [21].

3.4. Assigning a function to the communities

The function is the task performed by the operation. The name of the operation is usually meaningful and represents its task. Therefore, it is used to describe the function of each community. The function may have several keywords (taken from different operations; repeated words are removed) referring to it, as shown in Figure 2.

3.5. Handling user requests

3.5.1. Search steps

As known, user queries can be simple (seeking one feature) or complex (trying to locate multiple features in one query). Simple queries pose no problem, unlike complex ones. To remedy this and facilitate the search see Figure 2, it is recommended to:

- Simplify the complex request into sub-requests, allowing the client to determine the number of features they wish to search for at the beginning;
- Provide a specific number of keywords referring to each desired function;
- Retrieve those keywords and match them with those listed in the table, resulting in a list of operations from which the client can select the one that best suits their needs.

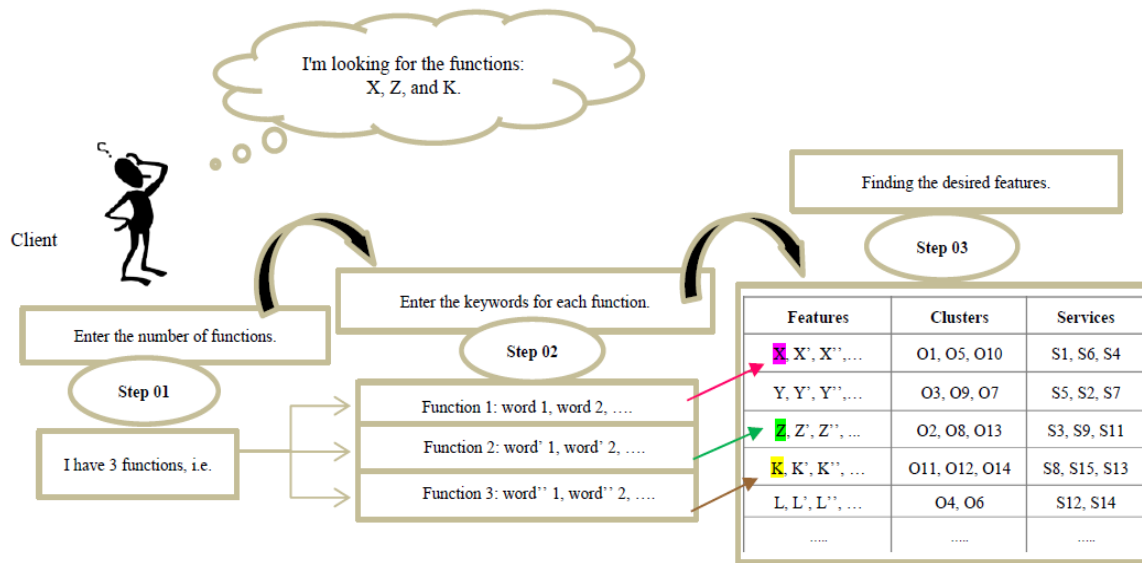


Figure 2. Search steps

3.5.2. Text similarities

In this work, it is necessary to assess the similarity between two bags of words: the first represents the keywords entered by the user, and the second represents the keywords describing the function provided by each community. There are several methods for calculating this measure, including Cosine, Jaccard, Dice, Pearson, and Euclidean, among others. According to [22], [23], the performances of Cosine similarity, the Jaccard coefficient, and the Pearson coefficient are very close and significantly better than those of the Euclidean distance. According to [24], the Dice index and the Jaccard index have similar performances. In this work, Jaccard similarity [25] was chosen; it is simple and effective, and the results are satisfying.

3.5.3. Threshold determination

The degrees of similarity returned by the Jaccard measure range between [0, 1]. According to our observations, there are five subintervals:

- [0, 0.2]: No similarity.
- [0.2, 0.5]: Little similarity.
- [0.5, 0.7]: Average similarity.
- [0.7, 0.9]: Very similar.
- [0.9, 1.0]: Similar.

Thus, the similarity threshold starts from 0.7, meaning that two bags are considered similar if and only if they have a score greater than or equal to 0.7. Consequently, the concerned function or community (and all its operations) will be returned.

3.5.4. The selection process

The last step returns a list of operations from which the client selects one operation over another based only on its non-functional properties. To illustrate this idea, consider the following example: a client searches for function 1. The entire cluster (O1, O2, O3) is returned, and their non-functional criteria are also displayed as shown in Figure 3. The client then selects the operation that best suits their needs.

Functions	Id-Ops	Names-Ops	Services	properties
Function 1	1	O1	S1	P 1
	2	O2	S9	
	3	O3	S3	
Function 2	4	O'1	S6	P 2
	5	O'2	S5	
.....

Costs	Response times	Availability	To choose
C1	T1	Yes	
Free	T2	Yes	✓
Free	T3'	No	

Figure 3. Selection results

4. RESULTS AND DISCUSSION

4.1. Experimental environment

The experimentation was carried out using a set of real web services sourced from different domains, including messaging, weather, and reservation systems. These services were carefully selected to ensure a diverse representation of use cases, covering a wide spectrum of practical applications. This diversity was crucial for thoroughly evaluating the robustness of our approach across various scenarios. Consequently, it allowed us to assess how efficiently our method could handle real-world complexities.

4.2. Application development

The application used for this research was developed using NetBeans, which is a robust integrated development environment (IDE) based on Java. The primary objective of this application is to streamline and optimize the process of searching for relevant operations. To achieve this, the application groups similar operations into clusters, organizing them based on common features. This clustering step simplifies the subsequent search by narrowing down the options and making it easier to locate the desired functionalities efficiently.

4.3. Methodology for reorganizing web services space

4.3.1. Organizing web services by domain

The first step in our approach is to organize web services WSDL according to their specific application domains, such as weather, messaging, and reservations. This initial categorization helps structure the available services and facilitates their subsequent management. By grouping web services into distinct domains, we lay the foundation for more targeted and efficient searches.

4.3.2. Formation of communities of similar operations

Once the web services are organized by domain, we proceed with a more detailed analysis to identify and group similar operations within each domain. These “communities” of operations are formed based on functional similarities, meaning operations that perform comparable tasks. This grouping has a dual purpose: it simplifies the search by allowing similar operations to be quickly found when a relevant operation is identified, and it facilitates the substitution of operations when an alternative is needed.

4.4. Search process

The application manages user queries based on the desired functionalities. When a user specifies the operations they need, the search is launched across the previously formed communities of operations. Thanks to this clustering organization, not only is the exact operation quickly found, but all similar operations are also proposed, offering the user a comprehensive set of relevant options as shown in Figure 4.

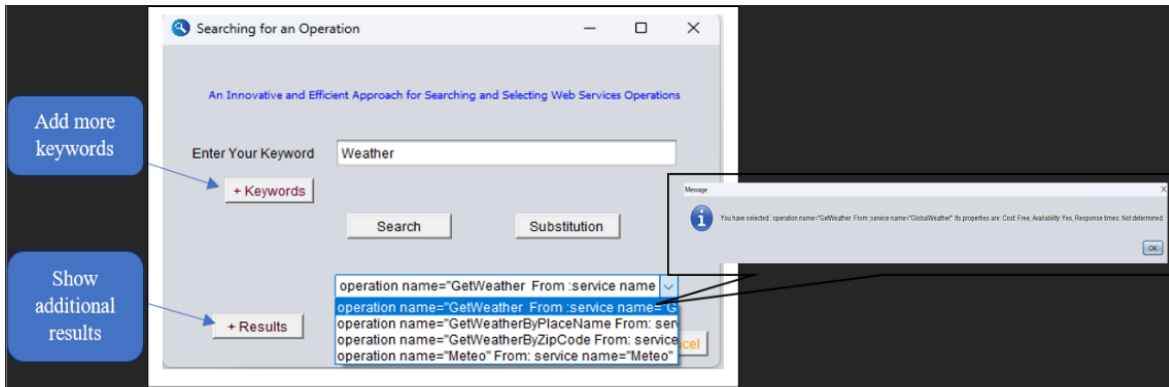


Figure 4. Search tool for web services operations

4.5. Validation and analysis

Since it is difficult, if not impossible, to find similar tools for comparison, we have used two measures, precision and recall, as shown in Figure 5, to validate the results, particularly the relevance of the selection before and after the reorganization of the web services space. These measures allow us to quantify the accuracy of our results, providing a clearer picture of how well our method identifies relevant services. Additionally, we conducted further analysis to evaluate the improvement in search time, as shown in Figure 6, which is crucial for understanding the overall efficiency of our approach.

We note a discernible improvement in the pertinent selection of web service operations after the rearrangement of the web services space. This improvement is represented in a higher accuracy measure, which indicates the relevance of the operations returned, and a higher recall measure, which indicates that all relevant operations were returned because of the clusters that were established. Additionally, this rearrangement has led to a significant decrease in search time. It is much more efficient to search in a well-structured environment, where actions are grouped into clusters and classified by domain, than in an unorganized one. The acquired findings demonstrate the indisputable efficacy of our methodology, emphasizing its capacity to precisely pinpoint pertinent actions. Our method's efficacy has been validated by meticulous.

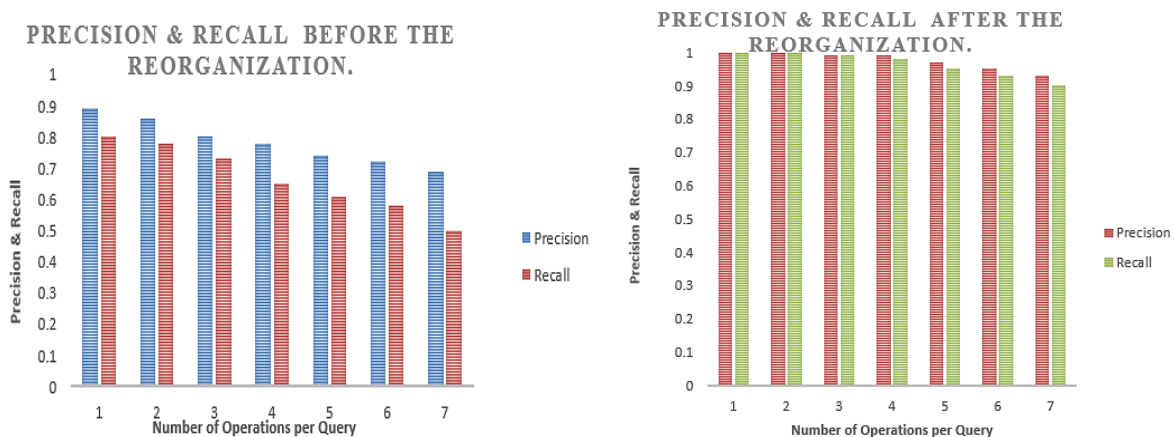


Figure 5. Precision and recall before and after the reorganization

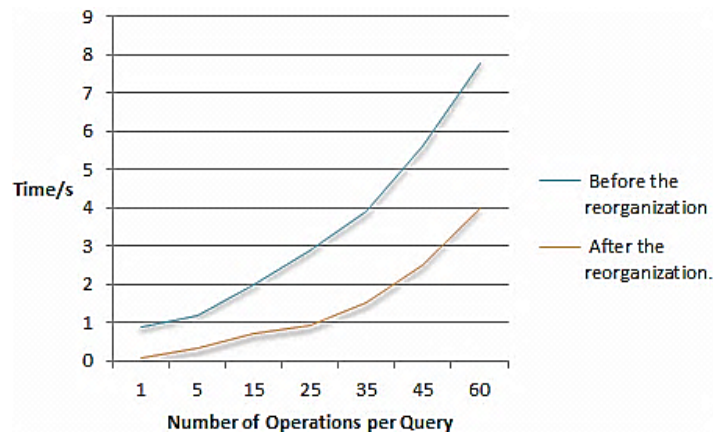


Figure 6. Response time

5. CONCLUSION

Over the last decade, an exponential increase in the number of services available on the web has been observed. This situation necessitates the creation of an efficient solution for the search and selection of web services. In this work, a simple yet highly effective approach was proposed, ensuring the completeness of the response as well as the relevance of the selection within an acceptable time frame. It involves dividing the client's request into several sub-functions and searching for the operations that perform these functions. The client then selects the one(s) that best meet their needs based on non-functional properties and begins to use it.




REFERENCES

- [1] Z. Gui, J. Cao, X. Liu, X. Cheng, and H. Wu, "Global-scale resource survey and performance monitoring of public OGC web map services," *ISPRS International Journal of Geo-Information*, vol. 5, no. 6, pp. 1–24, Jun. 2016, doi: 10.3390/ijgi5060088.
- [2] K. Hu, Z. Gui, X. Cheng, H. Wu, and S. C. McClure, "The concept and technologies of quality of geographic information service: improving user experience of GIServices in a distributed computing environment," *ISPRS International Journal of Geo-Information*, vol. 8, no. 3, pp. 1–26, Mar. 2019, doi: 10.3390/ijgi8030118.
- [3] Y. Cheng, W. Ge, and L. Xu, "Quality of geographical information services evaluation based on order-relation," in *Communications in Computer and Information Science*, 2018, pp. 679–688.
- [4] Z. Huang and W. Zhao, "A semantic matching approach addressing multidimensional representations for web service discovery," *SSRN Electronic Journal*, vol. 210, 2022, doi: 10.2139/ssrn.4076709.
- [5] K. Chen and C. Kuang, "Web service discovery based on maximum weighted bipartite graphs," *Computer Communications*, vol. 171, pp. 54–60, Apr. 2021, doi: 10.1016/j.comcom.2021.01.031.
- [6] M. Achir, A. Abdelli, L. Mokdad, and J. Benothman, "Service discovery and selection in IoT: a survey and a taxonomy," *Journal of Network and Computer Applications*, vol. 200, pp. 1–44, Apr. 2022, doi: 10.1016/j.jnca.2021.103331.
- [7] B. Zarei and M. Gaedke, "Disco: web service discovery chatbot," *IADIS International Journal on WWW/Internet*, vol. 18, no. 2, pp. 16–28, Dec. 2020, doi: 10.33965/ijwi_202018202.
- [8] I. Lizarralde, C. Mateos, A. Zunino, T. A. Majchrzak, and T.-M. Grønli, "Discovering web services in social web service repositories using deep variational autoencoders," *Information Processing & Management*, vol. 57, no. 4, pp. 1–19, Jul. 2020, doi: 10.1016/j.ipm.2020.102231.
- [9] H. H. Cuong Nguyen, B. T. Khiết, V. L. Nguyen, and T. T. Nguyen, "An effective method for clustering-based web service recommendation," *International Journal of Electrical and Computer Engineering*, vol. 12, no. 2, pp. 1571–1578, Apr. 2022, doi: 10.11591/ijece.v12i2.pp1571-1578.
- [10] W. Abramowicz, K. Haniewicz, M. Kaczmarek, and D. Zyskowski, "Architecture for web services filtering and clustering," in *Second International Conference on Internet and Web Applications and Services (ICIW'07)*, May 2007, doi: 10.1109/ICIW.2007.19.
- [11] R. Nayak and B. Lee, "Web service discovery with additional semantics and clustering," in *IEEE/WIC/ACM International Conference on Web Intelligence (WI'07)*, Nov. 2007, pp. 555–558, doi: 10.1109/WI.2007.82.
- [12] L. S. Tapsoba, Y. Traore, and S. Malo, "Towards an architecture for the interoperability of hospital information systems in Burkina Faso," *Studies in Health Technology and Informatics*, vol. 272, pp. 159–162, Jun. 2020, doi: 10.3233/SHTI200518.
- [13] S. Rangarajan, "QoS-based web service discovery and selection using machine learning," *ICST Transactions on Scalable Information Systems*, vol. 5, no. 17, pp. 1–8, May 2018, doi: 10.4108/eai.29-5-2018.154809.
- [14] N. A. Al-Musawi and D. Al-Shammary, "Dynamic Hilbert clustering based on convex set for web services aggregation," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 6, pp. 6654–6662, Dec. 2023, doi: 10.11591/ijece.v13i6.pp6654-6662.
- [15] A. M. Mohammed, S. S. A. Haytamy, and F. A. Omara, "Location-aware deep learning-based framework for optimizing cloud consumer quality of service-based service composition," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 1, pp. 638–650, Feb. 2023, doi: 10.11591/ijece.v13i1.pp638-650.
- [16] C.-Y. Song and E.-S. Cho, "A service-oriented cloud modeling method and process," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, pp. 962–977, Feb. 2020, doi: 10.11591/ijece.v10i1.pp962-977.




- [17] M. Shenoy, "A new similarity measure for taxonomy based on edge counting," *International journal of Web & Semantic Technology*, vol. 3, no. 4, pp. 23–30, Oct. 2012, doi: 10.5121/ijwest.2012.3403.
- [18] O. Rozinek and J. Mareš, "Fast and precise convolutional Jaro and Jaro-Winkler similarity," in *2024 35th Conference of Open Innovations Association (FRUCT)*, Apr. 2024, pp. 604–613, doi: 10.23919/FRUCT61870.2024.10516360.
- [19] Y. Zeng, X. Wu, and J. Cao, "Research and implementation of Hungarian method based on the structure index reduction for DAE systems," *Journal of Algorithms & Computational Technology*, vol. 8, no. 2, pp. 219–231, Jun. 2014, doi: 10.1260/1748-3018.8.2.219.
- [20] E. Stroulia and Y. Wang, "Structural and semantic matching for assessing web-service similarity," *International Journal of Cooperative Information Systems*, vol. 14, no. 4, pp. 407–437, Dec. 2005, doi: 10.1142/S0218843005001213.
- [21] P. Plebani and B. Pernici, "URBE: web service retrieval based on similarity evaluation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 11, pp. 1629–1642, Nov. 2009, doi: 10.1109/TKDE.2009.35.
- [22] A. Huang, "Similarity measures for text document clustering," in *6th New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, Apr. 2008, pp. 49–56.
- [23] T. Z. Baharav, G. M. Kamath, D. N. Tse, and I. Shomorony, "Spectral Jaccard similarity: a new approach to estimating pairwise sequence alignments," *Patterns*, vol. 1, no. 6, pp. 1–10, Sep. 2020, doi: 10.1016/j.patter.2020.100081.
- [24] E. Negre, "Comparaison de textes: quelques approches...", hal-00874280, 2013. [Online]. Available: <https://hal.science/hal-00874280v1>
- [25] Ī. Kabasakal and H. Soyuer, "A Jaccard similarity-based model to match stakeholders for collaboration in an industry-driven portal," *The 7th International Management Information Systems Conference*, vol. 74, no. 1, pp. 1–9, Mar. 2021, doi: 10.3390/proceedings2021074015.

BIOGRAPHIES OF AUTHORS



Sara Rekkal    born in Bechar, Algeria, the author obtained her master's degree in computer science from the University of Bechar in 2015 and completed her doctorate in 2019 at the University of Ahmed Ben Bella Oran I. She is an assistant professor at Badji Mokhtar Annaba University in Algeria. Her primary areas of expertise include web services, multi-criteria decision support, embedded systems engineering and parallelism, metaheuristic algorithms, and natural language processing. For inquiries, she can be contacted at email: sara.rekkal@univ-annaba.dz.



Kahina Rekkal    born in Bechar, Algeria, the author earned her Dipl. Inf.-Ing. degree from the University of Bechar in 2009, a master's degree in 2012, and a doctorate in 2018, all from the same university. Currently serving as an assistant professor at Salhi Ahmed Naama University Center in Algeria, her professional focus encompasses channel coding, digital signal processing, optimization of trellis-coded modulation schemes, genetic algorithms, communication over multipath and fading channels, orthogonal frequency-division multiplexing (OFDM), channel equalizers, renewable energies, and Arabic natural language processing. For further communication, she can be contacted at email: rekkal@cuniv-naama.dz.