Hindawi
Scientific Programming
Volume 2021, Article ID 6661035, 12 pages
https://doi.org/10.1155/2021/6661035

Research Article

A Web Service Clustering Method Based on Semantic Similarity and Multidimensional Scaling Analysis

Chuang Shan and Yugen Du

Shanghai Key Laboratory of Trustworthy Computing, School of Software Engineering, East China Normal University, Shanghai 200062, China

Correspondence should be addressed to Yugen Du; ygdu@sei.ecnu.edu.cn

Received 5 November 2020; Revised 7 February 2021; Accepted 22 April 2021; Published 5 May 2021

Academic Editor: David Ruano-Ordás

Clustering web services is an effective method to solving service computing problems. The key insight behind it is to extract the vectors based on the service description documents. However, the brevity of natural language service description documents typically complicates the vector construction process. To circumvent the difficulty, we propose a novel web service clustering method to vectorize documents based on the semantic similarity, which can be calculated via WordNet and multidimensional scaling (WMS) analysis. We utilize the dataset from the ProgrammableWeb to conduct extensive experiments and achieve prominent advances in precision, recall, and F-measure.

1. Introduction

Through the rapid development of Internet technology [1], clustering web services has become an effective method to solving service discovery [2–4], service composition [5, 6], and service recommendation [7]. Firstly, there are increasing enterprises and institutions that encapsulate software functions or data into web services and publish them to the network. For instance, the number of services in ProgrammableWeb has grown from fewer than 3,000 in 2011 to more than 20,000 by 2020, which remarkably increases the difficulty of managing web services. Secondly, when users query the web services they need, the service discovery system generally searches all related web services and returns ordered ones, where the ranking index is mainly based on the relevance to the query. However, it is intractable to search through the entire whole web services space and obtain accurate results. According to the work made by Zhang et al., clustering web services can improve the performance of service discovery by reducing the search space [2]. Thirdly, service composition is proposed to select appropriate web services from the repository to build functional web services. However, the scale of the repository will influence the efficiency of finding and sorting multiple web services with various functions. Clustering web services that matches the clusters to the requirements of developers can successfully alleviate this dilemma in the light of the research results of Xia et al. [5].

The premise of clustering web services is to extract vectors corresponding to service description documents, which are mainly constructed based on keyword or semantic features. Unfortunately, despite the effectiveness of clustering web services towards a variety of web service tasks, its applicability is hindered by extracting vectors from natural language service description documents. The service description document is an important basis for clustering web service, which is commonly implemented by Web Services Description Language (WSDL) document or natural language web service description document. Though WSDL document written in Extensible Markup Language (XML) can offer plentiful convenient functions such as describing the service in combination with Web Ontology Language (OWL), its construction procedure is quite complex. Therefore, some companies and institutions, such as ProgrammableWeb, leverage natural language to describe web services to generate succinct service description documents,

where each keyword appears almost once. However, the brevity of these documents leads to extra problems when clustering web services extracts' two types of target features. Specifically, extraction of keyword features highly depends on the frequency of keyword occurrence. Similarly, the corpus of service description documents can hardly establish so that the corresponding probabilistic topic model is difficult to construct to extract the semantic features that refers to the probability distribution of a document on different topics.

This paper proposes a novel approach that constructs vectors via differences between documents instead of document features. We mainly cluster the web services described in natural language. The operation object is the natural language service description document, and the dataset is from ProgrammableWeb. The main contributions of this paper are based on (1) designing of an algorithm to calculate the similarity between documents, (2) proposing a methodology to convert similarity data into distance data, which is an important prerequisite for multidimensional scaling analysis, and (3) the implementing principal component analysis (PCA) methods on the vectors corresponding to the service documents to determine the appropriate clustering algorithm.

The rest of our work is structured as follows. Section 2 compares existing work. Section 3 introduces the study materials. We explain the study methods in detail in Section 4. We explain the experimental process in detail in Section 5. Finally, Section 6 summarizes the main conclusions and highlights future work.

2. Related Work

In this section, we will separately introduce the related works on WSDL service description document clustering and natural language service description document clustering.

2.1. WSDL Service Description Document. In the early days, there were many web services described using WSDL documents, so many scholars paid attention to the clustering of such web services. Paik and Kumara et al. used the ontology model in service clustering [8, 9], which greatly improved the service clustering effect. Some scholars used WordNet to calculate the semantic similarity [8, 10], but the algorithm they proposed is not suitable for natural language documents. We proposed an algorithm for calculating the semantic similarity between natural language documents using WordNet. In addition, some scholars also used context-aware methods to improve service clustering [9, 11]. Liang et al. used tag information in WSDL document clustering to improve the clustering effect [12]. Considering the sparse semantics of WSDL documents, Gu et al. used open data to increase semantic information before clustering [13]. Because of the too much useless information of WSDL documents, Agarwal et al. used a probability model to filter useless information before clustering [14]. Sun et al. added neural networks to service clustering [4]. In general, these methods have a common limitation, and they are not suitable for processing service documents described in natural language. For example, in literature [8], separate ontology is constructed for different "element" data, and the "element" includes <definitions>, <types>, <messages>, and <portType>. However, in natural language documents, there is no "element," so it is very difficult to construct ontology.

2.2. Natural Language Service Description Document. Because WSDL documents are too complex to construct, some companies and organizations now use natural language to describe web services. Some scholars focus on the clustering of natural language service description documents. Muth and Inkpen used the term frequency-inverse document frequency (TF-IDF) to extract keywords and then clustered web services according to keywords [15]. However, the service description documents are too short to extract keywords. Some scholars used the latent Dirichlet allocation (LDA) to build a probabilistic topic model and calculated the probability distribution of each service description document on each topic so as to achieve document vectorization and then clustered the vectors [2, 16]. The premise of LDA is to construct the unigram model. However, the corpus of service description document is too few, and the constructed unigram model is weak. Some scholars used the Word2Vec training external corpus to expand service documents to improve the effect of LDA training [17, 18]. However, the size and type of the corpus seriously affect the degree of improvement. Lizarralde used deep variational autoencoders in this work to solve this problem [3]. Cao et al. used the Doc2Vec model to train the service document dataset, converted each document into a vector, and then clustered the vectors [19]. However, there is no reference basis for the selection of vector dimensions, which increases the uncertainty of the results. Zou et al. first trained the WE-LDA model to obtain the probability-topic distribution of each document, then trained the recurrent convolutional neural network (RCNN) to obtain a fitting model from each service document to the probability-topic distribution, and finally clustered the document-feature vectors [20, 21]. However, the structure of RCNN is very complicated, the training effect of RCNN depends on adjusting the parameters, and the training results of the LDA model greatly increase the uncertainty of RCNN. So, it is very difficult to get a suitable model by adjusting parameters. In short, the problem of these methods comes from the uncertainty caused by mining service document features. We noticed that it is difficult to extract features from short documents, but it is easier to compare the differences between short documents, so we use WordNet to quantify document differences and then use multidimensional scaling analysis to construct vectors corresponding to the documents and finally cluster the vectors. In our method, only very few parameters need to be adjusted and our work on adjusting parameters has a theoretical and experimental basis.

3. Study Materials

The experimental data in this paper come from the ProgrammableWeb website. This article uses the WordNet database to calculate semantic similarity, and we will introduce them in detail below.

3.1. ProgrammableWeb. ProgrammableWeb is an information and news source about the Web as a programmable platform. It is a subsidiary of MuleSoft and has offices in San Francisco, CA. The website publishes a repository of web APIs, mashups, and applications and has documented over 22000 open web APIs and thousands of applications in October 2020. It has been called the "journal of the API economy" by TechCrunch [22]. The data in ProgrammableWeb mainly include category, description document, tag, and calling method (see website https://www.programmableweb.com/) (see Figure 1). This paper uses description documents as the main body for service clustering. "Tag" is the auxiliary information given by web service developers, which helps us to preprocess service documents. "Category" is the classification given by web service developers, and we use it as the evaluation index of clustering.

3.2. WordNet. WordNet is an English dictionary established and maintained by the Cognitive Science Laboratory of Princeton University [23]. Because it contains semantic information, it is different from a dictionary in the usual sense. WordNet groups the entries according to their meanings. Each group of entries with the same meaning is called a *Synset*. WordNet provides a short, summary definition for each *Synset* and records the semantic relationship between different *Synsets*. A word may have multiple meanings, which are in different *Synsets* (see Table 1).

Synset contains a variety of semantic relations, such as upper and lower relation, antisense relation, and whole and part relation (see Figure 2). Based on these relationships, the semantic similarity between *Synsets* can be calculated. A word may have multiple semantics and parts of speech corresponding to different *Synsets*. Therefore, the two words have different semantic similarities in different *Synsets* (see Table 2). We have to choose one of them as the semantic similarity between two words. Some of the existing methods choose the maximum value [24, 25]. This is the basis for calculating the semantic similarity between documents.

4. Study Methods

This section consists of three parts. Section 4.1 introduces the method of calculating the semantic similarity between two service documents. Section 4.2 introduces the method of using semantic similarity to calculate the vector corresponding to the service document. Section 4.3 introduces how to select the appropriate algorithm to cluster the vectors.

The main study methods of this paper are based on (1) obtaining preprocessed documents (PD) through tags and WordNet, (2) calculating the semantic similarity between PDs and then obtaining the semantic distance matrix, (3) using the multidimensional scaling to analyze the semantic distance matrix to obtain the semantic distance vector (SDV) corresponding to each web service, and (4) using the K-means algorithm to cluster the SDVs to achieve clustering of web services (see Figure 3). The multidimensional scaling is used to translate "information about the pairwise "distances" among a set of n objects or individuals" into a configuration of n points mapped into an abstract Cartesian space [26].

4.1. Calculate Semantic Similarity. The basis of calculating document semantic similarity is to calculate the semantic similarity between words. We enumerate all the semantic similarities of two words in different *Synsets* and select the largest as the semantic similarity of the words [24, 25].

Before calculating the semantic similarity of documents, preprocessing is required. General preprocessing methods include removing punctuation and stop words. This paper considers the particularity of Web service description documents. Except for stop words, there are many words that have nothing to do with document semantics. "Tag" is the auxiliary information given by web service developers according to the research results of Jingli et al. In [27], using tags can filter out the words that are not related to the topic; according to the research results of Shi et al. [28], the more tags two web services have duplicates, the more likely they are to belong to the same category. Therefore, in the process of document preprocessing, we keep words that are semantically similar to tags, thereby removing words that have nothing to do with the subject of the document. We use D to represent the service description document, T to represent the document tag collection, and PD to represent the preprocessed document (see Algorithm 1).

Regarding the threshold α, since the semantic similarity calculation result of WordNet is between 0 and 1, we adopt an intermediate value strategy and take α as 0.5.

The semantic similarity between the two PDs is determined by the words in the PD (see Figure 4). We can calculate the maximum semantic similarity between each word and all the words on the opposite side. The semantic similarity between two PDs is divided by the sum of length after the similarity is accumulated, which can ensure the symmetry (see Algorithm 2).

The semantic similarity calculated by WordNet is between 0 and 1.

$$
\begin{aligned}
&0 < \text{Wordnet.similary}\left(A_i, B_j\right) < 1,\\
&\text{SUM}\left(\text{PD1}\right) = \sum_{i=1}^{m} \text{Max}\left(\text{Wordnet.similary}\left(A_i, B_j\right)\right) \quad (1 < j < n),\\
&\text{SUM}\left(\text{PD2}\right) = \sum_{j=1}^{n} \text{Max}\left(\text{Wordnet.similary}\left(B_j, A_i\right)\right) \quad (1 < i < m).
\end{aligned}
\tag{1}
$$

So, we can get

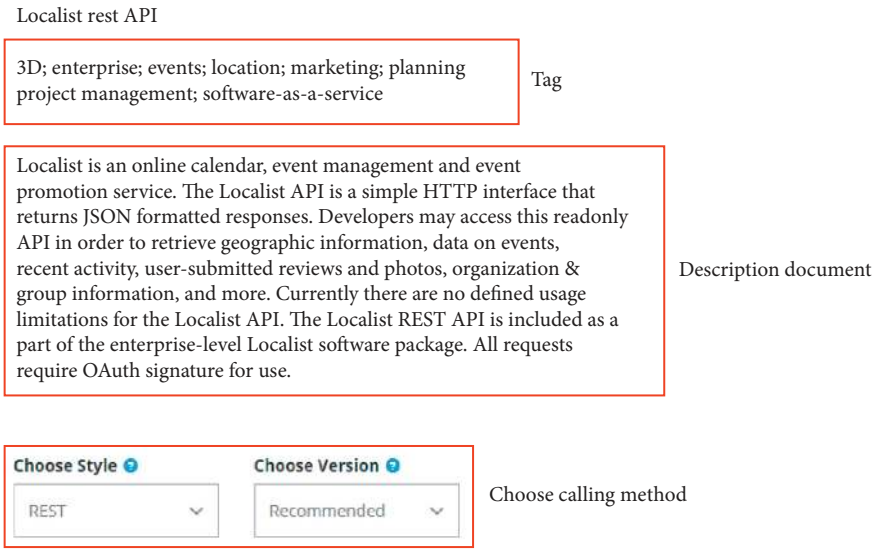

Figure 1: A web service with the category "Calendars" on ProgrammableWeb [40].

Table 1: Different meanings in different *Synsets* of the word "people."

Synset	Meaning
people.n.01	(Plural) any group of human beings (men or women or children) collectively
people.n.02	The body of citizens of a state or country facilities for research and teaching
people.n.03	Members of a family line
People.n.04	The common people generally

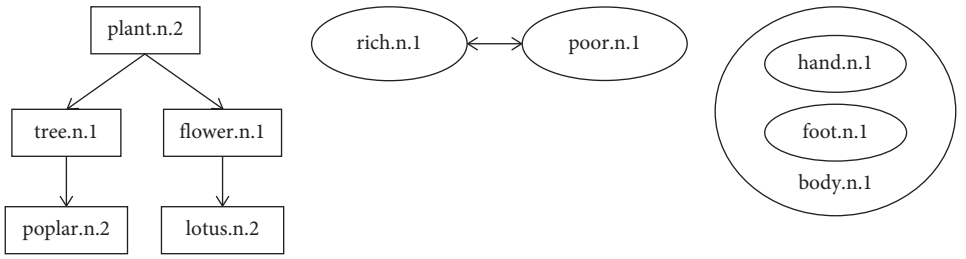

Figure 2: Three relationships between *Synset*: (a) upper and lower relation; (b) antisense relation; (c) whole and part relation.

Table 2: Semantic similarity between "people" and "citizenry" in different *Synsets*.

	citizenry.n.01
people.n.01	0.33333
people.n.02	1
people.n.03	0.1250
People.n.04	0.33333

$$0 < \mathrm{SUM}(\mathrm{PD1}) < m,$$
$$0 < \mathrm{SUM}(\mathrm{PD2}) < n, \tag{2}$$
$$0 < \mathrm{sim}(\mathrm{PD1}, \mathrm{PD2}) = \frac{\mathrm{SUM}(\mathrm{PD1}) + \mathrm{SUM}(\mathrm{PD2})}{m+n} < 1.$$

We assume that the number of web service description documents is n. Through this algorithm, we can get a

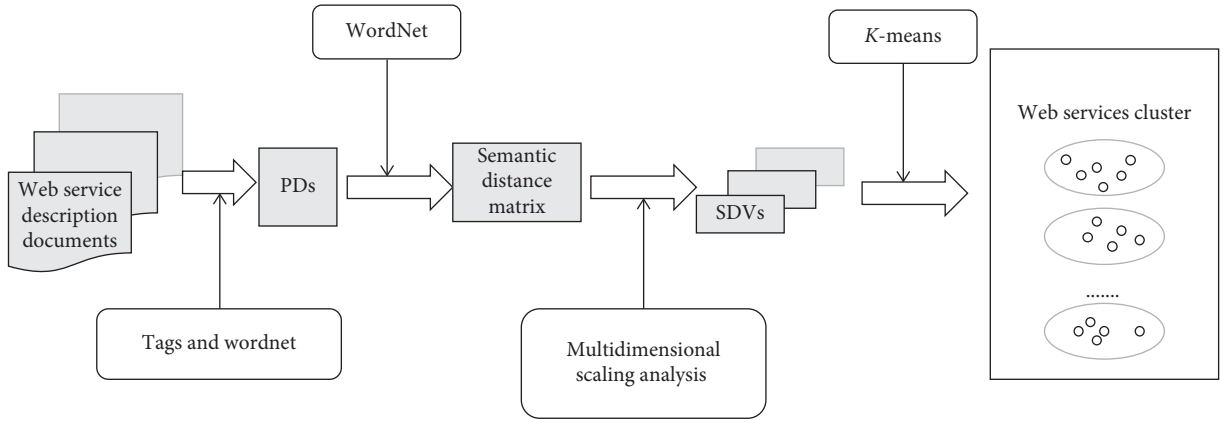

Figure 3: The framework of our web services clustering method.

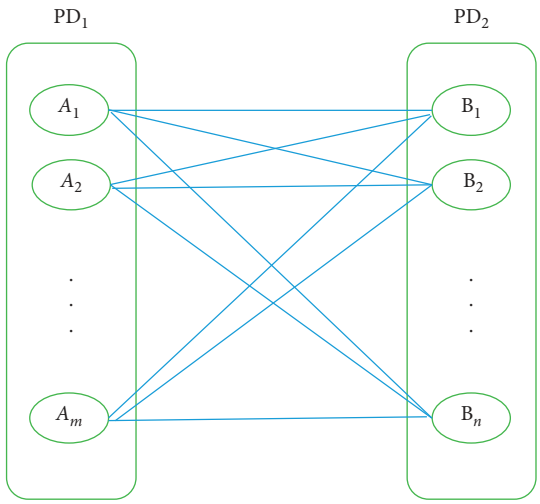

Figure 4: The semantic similarity between two PDs. The length is m and n, respectively.

semantic similarity matrix called $\mathrm{SIM} = (\mathrm{sim}_{ij})_{n\times n}$. The matrix elements are between 0 and 1, the larger the element value, the higher the semantic similarity. The sim_{ij} represents the semantic similarity between the i-th and j-th documents. Obviously, all diagonal elements are 1.

4.2. Multidimensional Scaling Analysis. The problem solved by the multidimensional scaling method is as follows. When the similarity (or distance) of each pair of n objects is given, the representation of these objects in multidimensional space is determined, and the original similarity (or distance) is expressed as much as possible. In other words, two semantic similar web services are represented by two points close to each other in multidimensional space, which creates conditions for clustering [29]. We first introduce data concepts related to multidimensional scaling.

4.2.1. Similar Data and Distance Data

Similar Data. This is the data representing the similarity of two objects. The larger the value is, the higher the similarity is. "Semantic similarity" in the previous article is the *similar data*.

Distance Data. This is contrary to similar data. The larger the value is, the lower the similarity is.

Only the distance data can be directly used for multidimensional scaling analysis [29].

4.2.2. Distance Matrix. A matrix $\mathrm{DIS} = (\mathrm{dis}_{ij})_{n\times n}$ of order $n \times n$, dis_{ij} is the distance between the i-th object and the j-th object if the following condition is met:

$$\begin{aligned} &\mathrm{DIS} = \mathrm{DIS}^T, \\ &\mathrm{dis}_{ij} \geq 0,\ \mathrm{dis}_{ii} = 0, \quad i, j = 1, 2, 3, \ldots, n. \end{aligned} \tag{3}$$

Then, the matrix DIS is a distance matrix.

If there is a positive integer r and there are n points in R^r, $X_1, X_2, \ldots, X_n$, such that

$$\mathrm{dis}_{ij}^2 = \left(X_i - X_j\right)^T \left(X_i - X_j\right), \tag{4}$$

then DIS is called the Euclidean distance matrix [30]. In fact, there is a simpler way to determine whether the distance matrix is a Euclidean distance matrix, which we will introduce in later chapters.

4.2.3. Similarity Coefficient Matrix. A matrix $C = (c_{ij})_{n\times n}$ of order $n \times n$, c_{ij} is the similarity coefficient between the i-th object and the j-th object if the following condition is met:

$$\begin{aligned} &C = C^T, \\ &c_{ij} \leq c_{ii}, \quad i, j = 1, 2, 3, \ldots, n. \end{aligned} \tag{5}$$

Then, matrix C is a similarity coefficient matrix.

If the data are not a distance matrix, it must be transformed into a distance matrix by a certain method in order to carry out multidimensional scaling analysis.

Therefore, the semantic similarity matrix SIM is not suitable for multidimensional scaling analysis. We need to translate semantic similarity into "semantic distance" through inversion. We define "semantic distance" as a value from two service description documents, between 0 and 1.

The smaller the semantic distance value, the higher the semantic similarity. The semantic distance matrix is $\mathrm{DIS} = (\mathrm{dis}_{ij})_{n\times n}$. We need to use appropriate functions to reverse the semantic similarity. There is a classic transformation function (see equation (6)) [31–33]. However, from the experimental point of view, the effect of this function is not satisfactory.

$$\mathrm{dis}_{ij} = \left(c_{ii} + c_{jj} - 2c_{ij}\right)^{1/2}. \tag{6}$$

The sigmoid function is a commonly used activation function in neural networks [34]. This function expression is shown in the following equation:

$$S(x) = \frac{1}{1 + e^{-x}}. \tag{7}$$

The sigmoid function is an increasing function, and it cannot activate the data between 0 and 1, so we need to deform it. We use this function (see equation (8)) to reverse the data.

$$\mathrm{NS}(x) = \frac{1}{1 + e^{\sigma(x-\mu)}}. \tag{8}$$

μ and σ are adjustment coefficients. To ensure that NS is a minus function, μ and σ should be positive real numbers. We adjust μ and σ for many times through experiments and determine that when $\sigma = 20$ and $\mu = 0.3$, we can get better results.

So, we can calculate DIS by the following equation:

$$\mathrm{dis}_{ij} = \begin{cases} \mathrm{NS}\left(\mathrm{sim}_{ij}\right), & i \neq j \\ 0, & i = j \end{cases} \tag{9}$$

Let n points in r-dimensional space be expressed as $X_1, X_2, \ldots, X_n$ and expressed by matrix as $X = (X_1, X_2, \ldots, X_n)^T$. If the corresponding point of the i-th web service description document is X_i, then the coordinate of X_i is marked as follows:

$$X_i = \left(X_{i1}, X_{i2}, \ldots, X_{ir}\right) = \mathrm{SDV}_i. \tag{10}$$

The purpose of multidimensional scaling analysis is to calculate X. We call X a fitting composition of the semantic distance matrix DIS.

Let $B = (b_{ij})_{n\times n}$, where B is called the central inner product matrix of X, and the construction of matrix B is the premise of multidimensional scaling analysis [29]. Let us first construct matrix $A = (a_{ij})_{n\times n}$ according to the following equation:

$$a_{ij} = -\frac{1}{2}\mathrm{dis}_{ij}^2. \tag{11}$$

Next, the matrix $H = (h_{ij})_{n\times n}$ is constructed according to equation (12). In equation (12), I_n is an identity matrix of order n, E_n is a square matrix of order n, and any element of matrix E_n is 1.

$$H = I_n - \frac{1}{n}E_n. \tag{12}$$

Finally, matrix B is constructed as follows:

$$B = HAH. \tag{13}$$

We calculate the n eigenvalues of B and arrange them to obtain

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n. \tag{14}$$

The eigenvectors corresponding to the n eigenvalues are

$$e_1, e_2, e_3, \ldots, e_n. \tag{15}$$

The sufficient and necessary condition for the semantic distance matrix DIS to be Euclidean distance matrix is $|B| \geq 0$ [29]. We will discuss two cases of $|B|$.

(1) When $|B| \geq 0$, DIS is a Euclidean distance matrix, and all eigenvalues are nonnegative:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_n \geq 0. \tag{16}$$

The dimension of coordinate Xi is r. We need to construct X by using the eigenvector corresponding to r maximum eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_r$. r can be determined by accumulating the eigenvalues and calculating the proportion of the accumulated sum to the sum of all eigenvalues.

$$\frac{\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_r}{\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_n} \geq \alpha. \tag{17}$$

α is the threshold given in advance, generally 80% [29]. Then, $e_1, e_2, e_3, \ldots, e_r$ are selected to construct X.

(2) When $|B| < 0$, DIS is a non-Euclidean distance matrix. And there are negative eigenvalues.

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_s \geq 0 > \lambda_{s+1} \geq \ldots \geq \lambda_n. \tag{18}$$

r can be determined by accumulating the eigenvalues and calculating the proportion of the accumulated sum to the sum of absolute values of all eigenvalues.

$$\frac{\lambda_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_r}{\left|\lambda_{1|}\right| + \left|\lambda_2\right| + \left|\lambda_3\right| + \ldots + \left|\lambda_n\right|} \geq \alpha. \tag{19}$$

α is the threshold given in advance, generally 80%. Then, $e_1, e_2, e_3, \ldots, e_r$ are selected to construct X.

Next, X is calculated as follows:

$$X = \left(\sqrt{\lambda_1}\, e_1, \sqrt{\lambda_2}\, e_2, \ldots, \sqrt{\lambda_r}\, e_r\right) = \left(x_{ij}\right)_{n\times r}. \tag{20}$$

Each line in X corresponds to a web service description document, and the i-th line is Xi. Next, we need to cluster $X_i\,(1 \leq i \leq n)$.

4.3. Clustering Algorithm. The existing clustering methods are mainly divided into: layering, partitioning, density-based, model-based, grid-based, and soft computing methods [35]. We project the SDV into a two-dimensional space through PCA [36]. We analyzed the distribution of SDV projections and believed that the partitioning

clustering method [37] is suitable for processing our data. We compared each partitioning clustering method through experiments and chose the *K*-means algorithm. The *K*-means algorithm is a classic unsupervised learning clustering method, which is used in this paper for service clustering. [38, 39].

At this point, our research methods are all introduced.

5. Experimental Results and Analyses

5.1. Experimental Data. Our experimental data are real data crawled from the ProgrammableWeb. As of October 3, 2020, there were 21956 web services on ProgrammableWeb, totaling 425 categories [40]. The number of web services covered by different topics varies greatly. For example, there are 1020 web services in the category *Financial* and only one web service in the category IDE. The number is too unbalanced, which seriously affects the clustering effect. For this experiment, we select the categories that contain more than 400 web services. There are 11 categories ($CG = \{CG_1, CG_2, \ldots, CG_{11}\}$), including 6533 web services (see Table 3). These classifications are completed by the developers who publish these web services and are generally considered to be accurate.

5.2. Evaluating Indicator. We use three indexes to evaluate the clustering effect, which are precision, recall, and F-measure. We cluster 6533 web services into 11 clusters, which are expressed as $NG = \{NG_1, NG_2, \ldots, NG_{11}\}$. The three indexes are defined as follows:

$$\text{precision}(NG_i) = \frac{|CG_i \cap NG_i|}{|NG_i|},$$

$$\text{recall}(NG_i) = \frac{|CG_i \cap NG_i|}{|CG_i|},$$

$$F - \text{measure}(NG_i) = \frac{2 * \text{Precision}(NG_i) * \text{Recall}(NG_i)}{\text{Precision}(NG_i) + \text{Recall}(NG_i)}. \tag{21}$$

5.3. Comparison Method. Our method is compared with these five methods. The introduction is as follows:

(1) *TF-IDF + K* [15]. Keywords are extracted by word frequency and inverse document word frequency, and document-keyword vectors are constructed with keywords. *K*-means is used to cluster the document-keyword vectors.

(2) *LDA + K* [16]. We use latent Dirichlet allocation to model the documents and then get the topic-word matrix and document-topic vectors. *K*-means is used to cluster document-topic vectors.

(3) *Doc2Vec + K* [19]. We use the Doc2Vec model to train the documents and convert the documents into vectors. *K*-means is used to cluster document vectors.

(4) *RCNN + LDA + K* [20]. First train the LDA model to obtain the probability-topic distribution of each document and then train the RCNN network to obtain a fitting model from each service document to the probability-topic distribution. In this process, the feature vector of each service document can be obtained. Finally, cluster the document-feature vectors by *K*-means.

(5) *CMD + CT + K*. The classical transformation function is used to process similar data, and then multidimensional scaling analysis is carried out (see equation (6)). Finally, *K*-means clustering is used. We want to demonstrate the effectiveness of our new transformation method through this experiment.

(6) *WMS*. The new method proposed in this paper.

In order to compare the performance of the methods more objectively, we use Algorithm 1 to preprocess documents for all six methods, .

5.4. Comparison of Experimental Results.

(1) Algorithm implementation.

The distance matrix DIS with a dimension of 6533 can be obtained by processing the experimental data using the method designed above. We need to determine if the DIS is a Euclidean distance matrix. We calculated the eigenvalues and eigenvectors of the DIS and got 6533 eigenvalues (see Figure 5):

A total of 2032 eigenvalues are negative, so DIS is a non-Euclidean distance matrix. Let us take $\alpha = 80\%$, and when $r = 50$, equation (19) is satisfied, so we take the vector dimension as 50.

(2) Precision comparison of 6 methods on 5 categories (see Figure 6).

(3) Recall comparison of 6 methods on 5 categories (see Figure 7).

(4) F-measure comparison of 6 methods on 5 categories (see Figure 8).

(5) The average precision, recall, and F-measure of the 6 methods on 11 categories (see Table 4).

5.5. Result Analysis. From the experimental results, the TF-IDF + *K* method is the worst because the service description document is too short to extract keywords although the clustering effect is improved by adding context information. It should be noted that the LDA + *K* method, the Doc2Vec + *K* method, and the RCNN + LDA + *K* method contain a large number of random processes, resulting in different operation results in each operation. In contrast, the WMS method proposed in this paper not only has stable results but also has the best clustering effect. From the results, the clustering effect of the Doc2Vec + *K* method and the RCNN + LDA + *K* method is poor. We believe that these two methods rely on the continuity of the document, but our preprocessing (see Algorithm 1) destroys the continuity of the document. In contrast, the LDA + *K* method and the

```
Input: D, T, α
Output: PD (PD is initialized to empty)
Begin:
FOR each tag in T do:
  flag ⟵ 0;
  FOR each word in D do:
    IF WordNet.similarity (tag, word) >α do:
      PD.add (word);
      flag ⟵ 1;
    ENDIF
  ENDFOR
  IF flag = 0 do:
    PD.add (tag);
  ENDIF
ENDFOR
RETURN PD;
END
```

Algorithm 1: Preprocessing of a service description document.

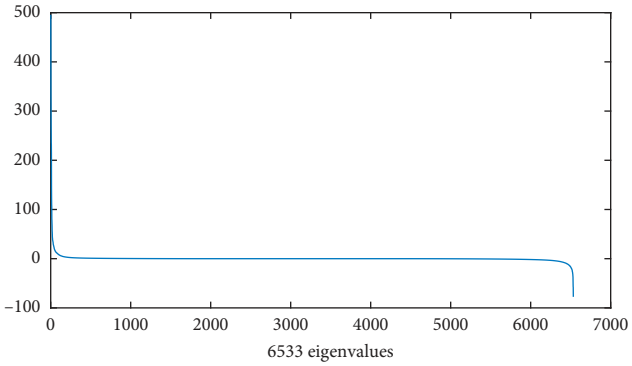

Figure 5: 6533 eigenvalues of distance matrix DIS.

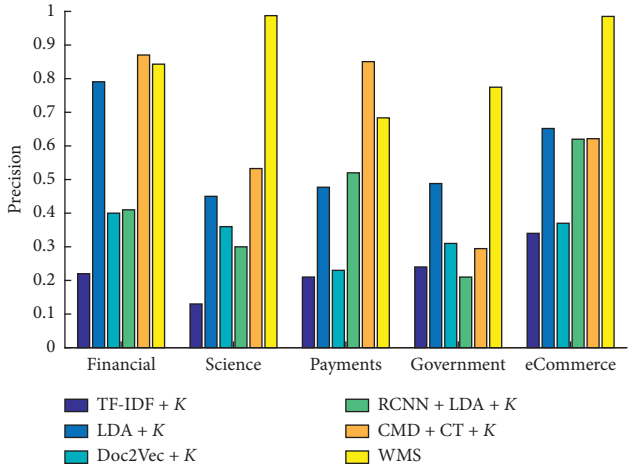

Figure 6: Precision of 6 methods on top 5 categories.

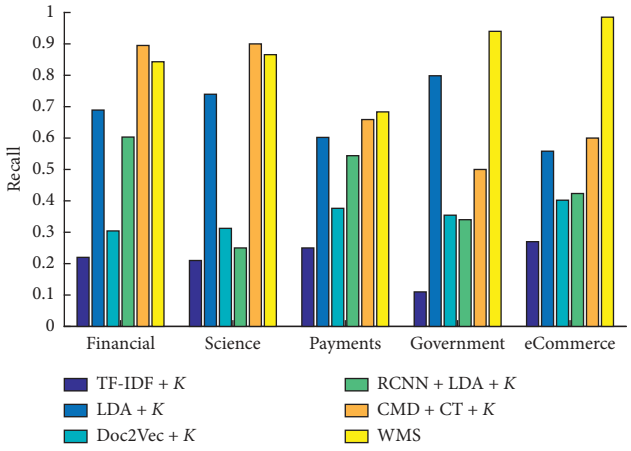

Figure 7: Recall of 6 methods on 5 categories.

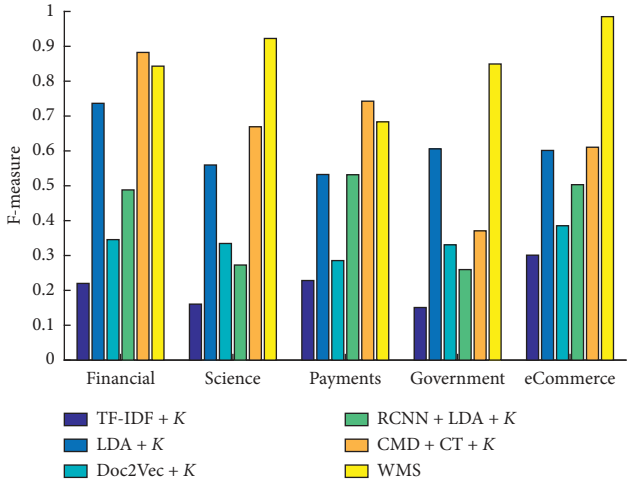

Figure 8: F-measure of 6 methods on 5 categories.

```
Input: PD1 (the length is m), PD2 (the length is n)
Output: sim (the semantic similarity between two PDs)
Begin:
SUM ⟵ 0;
FOR i ⟵ 1 to m do:
   MAX ⟵ 0;
   FOR j ⟵ 1 to n do:
      IF WordNet.similarity (A_i, B_j) >MAX do:
         MAX ⟵ WordNet.similarity (A_i, B_j);
      ENDIF
   ENDFOR
   SUM ⟵ SUM + MAX;
ENDFOR
FOR j ⟵ 1 to n do:
   MAX ⟵ 0;
   FOR i ⟵ 1 to m do:
      IF WordNet.similarity (A_i, B_j) >MAX do:
         MAX ⟵ WordNet.similarity (A_i, B_j);
      ENDIF
   ENDFOR
   SUM ⟵ SUM + MAX;
ENDFOR
sim ⟵ SUM/(m + n);
RETURN sim;
END
```

Algorithm 2: The process of calculating the semantic similarity between two PDs.

WMS method do not have any requirements for document continuity, so the clustering effect is better. And we improved the method of transposing data in multidimensional scaling analysis. Experiments prove that our improvement is effective.

5.6. Selection of Clustering Algorithm. We project the SDV into a two-dimensional space through PCA (see Figure 9).

We can see from Figure 9 that the clusters of SDV data are roughly distributed around a certain center in an elliptical shape, and some clusters are more fused. The partitioning clustering method is suitable for processing such data [39]. And our data belong to numerical type data. There are three typical partitioning clustering methods suitable for processing numerical type data: *K*-means, *K*-medoids [41], and Clustering for Large Application (CLARA) [42]. We compared the average precision, average recall, and average F-measure of the three methods (see Table 5). It is finally determined that *K*-means has the best clustering effect.

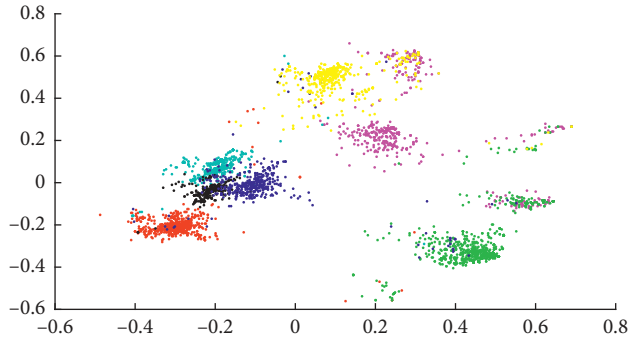

Figure 9: SDVs projection in two-dimensional space.

5.7. Supplementary Notes. Here, we show how to determine $\mu = 0.3$ and $\sigma = 20$ in equation (8). The symmetry center of sigmoid function is $(\mu, 0)$, so μ plays the role of data segmentation. We count the distribution of elements in the matrix SIM (see Figure 10).

Table 3: The distribution of web services in top 18 categories.

Category	Number
Financial	1020
Tools	856
Payments	657
Messaging	650
E-commerce	627
Enterprise	508
Social	497
Mapping	470
Science	434
Government	412
Security	402

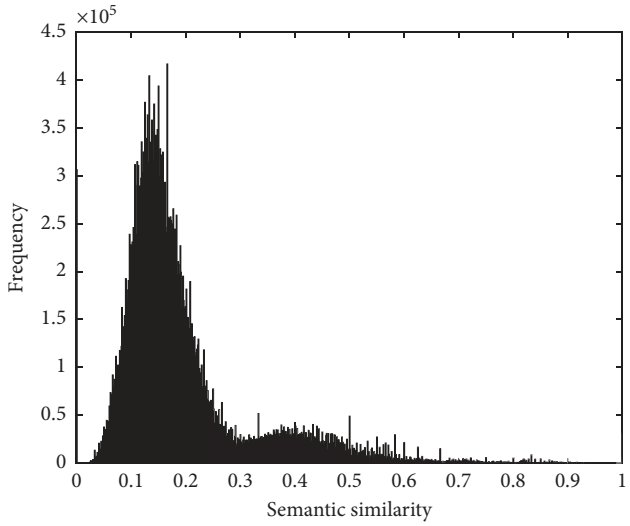

Figure 10: The distribution of elements in the matrix SIM.

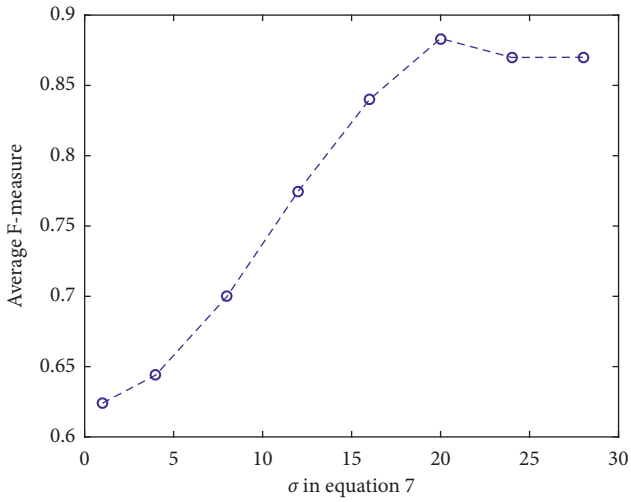

Figure 11: The changes of average F-measure by adjusting σ

It can be found that 0.3 is the segmentation point of frequency, so $\mu = 0.3$. After that, determine $\mu = 0.3$ and record the changes of average F-measure by adjusting σ in equation (8) (see Figure 11).

Table 4: Comparison of average precision, recall, and F-measure of six methods.

Method	Precision	Recall	F-measure
TF-IDF + K	0.1936	0.2010	0.1986
LDA + K	0.5827	0.6461	0.5994
Doc2Vec + K	0.3502	0.3096	0.3203
RCNN + LDA + K	0.5231	0.5763	0.5496
CMD + CT + K	0.7816	0.8311	0.7967
WMS	0.8935	0.8734	0.8795

Table 5: Comparison of average precision, recall, and F-measure of three methods.

Method	Precision	Recall	F-measure
K-medoids	0.8087	0.8650	0.8206
CLARA	0.8138	0.8705	0.8269
K-means	0.8935	0.8734	0.8795

As can be seen from Figure 11, when $\sigma = 20$, the average F-measure has good result.

6. Conclusions

In this paper, we propose a web service clustering method based on semantic similarity and multidimensional scaling analysis. We first used WordNet to calculate the semantic similarity between documents and then obtained the semantic distance matrix. Then, we used multidimensional scaling analysis to get the SDVs. Finally, we used the K-means algorithm to cluster the SDVs. Most of the existing methods vectorize documents by extracting document features. We have proposed a new idea to vectorize documents by comparing the differences between documents. The improvement of the vectorization method leads to the improvement of the clustering effect. Multidimensional scaling analysis is the core of our method. The experimental results show that our method is better than existing methods in precision, recall, and F-measure. And our method is more deterministic than the method based on deep neural network and LDA. And we improved the method of transposing data in multidimensional scaling analysis. Experiments prove that our improvement is effective.

We believe that our method has a major flaw; our algorithm relies on tags and is less robust. For future work, we will improve Algorithm 2 to get rid of the dependence on tags. In addition, service clustering cannot be directly useful to users. For future work, we will use the service clustering method in this article as a basis to improve service composition, service discovery, and other web service tasks.

Data Availability

The data used to support the results of this study are obtained from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (61572195) and the special fund of Shanghai Economic and Information Commission (sheitc160306).

References

[1] J. Pasley, "How BPEL and SOA are changing web services development," *IEEE Internet Computing*, vol. 9, no. 3, pp. 60–67, 2005.

[2] N. Zhang, J. Wang, K. He, Z. Li, and Y. Huang, "Mining and clustering service goals for RESTful service discovery," *Knowledge and Information Systems*, vol. 58, no. 3, pp. 669–700, 2019.

[3] 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, Article ID 102231, 2020.

[4] C. Sun, L. Lv, G. Tian, Q. Wang, X. Zhang, and L. Guo, "Leverage label and word embedding for semantic sparse web service discovery," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–8, Article ID 5670215, 2020.

[5] B. Xia, Y. Fan, W. Tan et al., "Categoryaware API clustering and distributed recommendation for automatic mashup creation," *IEEE Transactions on Services Computing*, vol. 8, no. 5, pp. 674–687, 2014.

[6] B. Cao, X. Liu, M. M. Rahman et al., "Integrated content and network-based service clustering and web apis recommendation for mashup development," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 99–113, 2017.

[7] M. Shi, J. Liu, D. Zhou et al., "A probabilistic topic model for mashup tag recommendation," in *Proceedings of IEEE International Conference on Web Services (ICWS)*, pp. 444–451, San Francisco, CA, USA, June 2016.

[8] B. T. G. S. Kumara, I. Paik, W. Chen, and K. H. Ryu, "Web service clustering using a hybrid term-similarity measure with ontology learning," *International Journal of Web Services Research*, vol. 11, no. 2, pp. 24–45, 2014.

[9] R. A. H. M. Rupasingha, I. Paik, and B. T. G. S. Kumara, "Specificity-aware ontology generation for improving web service clustering," *IEICE Transactions on Information and Systems*, vol. E101-D, no. 8, pp. 2035–2043, Aug. 2018.

[10] A. Konduri, "Clustering of web services based on semantic similarity," MS thesis, University of Akron, Akron, OH, USA, 2008.

[11] B. T. G. S. Kumara, I. Paik, and Y. Yaguchi, "Context-aware web service clustering and visualization," *International Journal of Web Services Research*, vol. 17, no. 4, pp. 32–54, 2020.

[12] T. Liang, Y. Chen, W. Gao, M. Chen, M. Zheng, and J. Wu, "Exploiting user tagging for web service Co-clustering," *IEEE Access*, vol. 7, pp. 168981–168993, 2019.

[13] Y. Gu, H. Cai, C. Xie, L. Jiang, Y. Gu, and A. Liu, "Utilizing semantic information from linked open data in web service clustering," in *Proceedings of 2016 International Conference on Progress in Informatics and Computing (PIC)*, pp. 654–658, Shanghai, China, December 2016.

[14] N. Agarwal, G. Sikka, and L. K. Awasthi, "Enhancing web service clustering using Length Feature Weight Method for service description document vector space representation," *Expert Systems with Applications*, vol. 161, Article ID 113682, 2020.

[15] A. Muath and D. Inkpen, "Clustering the topics using TF-IDF for model fusion," in *Proceedings of Phd Workshop on Information & Knowledge Management ACM*, Napa Valley, CF, USA, October 2008.

[16] C. Lin, Y. He, C. Pedrinaci, and J. Domingue, "Feature LDA: a supervised topic model for automatic detection of web API documentations from the web," in *Proceedings of International Semantic Web Conference Springer*, Berlin, Heidelberg, November 2012.

[17] M. Shi, J. Liu, D. Zhou, M. Tang, and B. Cao, "WE-LDA: a word embeddings augmented LDA model for web services clustering," in *Proceedings of 2017 IEEE International Conference on Web Services (ICWS)*, pp. 9–16, Honolulu, HI, USA, June 2017.

[18] Y. Zhao, K. He, and Y. Qiao, "ST-LDA: high quality similar words augmented LDA for service clustering," in *Proceedings of International Conference on Algorithms and Architectures for Parallel Processing*, pp. 46–59, Springer, Guangzhou, China, November 2018.

[19] X. Zhang, J. Liu, B. Cao, Q. Xiao, and Y. Wen, "Web service recommendation via combining Doc2Vec-based functionality clustering and DeepFM-based score prediction," in *Proceedings of 2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pp. 509–516, Melbourne, Australia, December 2018.

[20] G. Zou, Z. Qin, Q. He, P. Wang, B. Zhang, and Y. Gan, "DeepWSC: a novel framework with deep neural network for web service clustering," in *Proceedings of 2019 IEEE International Conference on Web Services (ICWS)*, pp. 434–436, Milan, Italy, July 2019.

[21] G. Zou, Z. Qin, Q. He, P. Wang, B. Zhang, and Y. Gan, "DeepWSC: clustering web services via integrating service composability into deep semantic features," *IEEE Transactions on Services Computing*, vol. 9, 2020.

[22] "ProgrammableWeb" from Wikipedia, 2020, https://en.wikipedia.org/wiki/ProgrammableWeb.

[23] "WordNet" from Wikipedia, 2020, https://en.wikipedia.org/wiki/WordNet.

[24] J. Gonzalo, M. F. Verdejo, I. Chugur, and J. Cigarran, "Indexing with WordNet synsets can improve text retrieval," arXiv preprint cmp-lg/9808002, 1998.

[25] C. Strapparava and A. Valitutti, "Wordnet affect: an affective extension of wordnet," *Lrec*, vol. 4, pp. 1083–1086, 2004.

[26] A. Mead, "Review of the development of multidimensional scaling methods," *The Statistician*, vol. 41, no. 1, pp. 27–39, 1992.

[27] Z. Jingli, N. Xuejun, Q. Leihua et al., "Web clustering based on tag set similarity," *Journal of Computers*, vol. 6, no. 1, pp. 59–66, 2011.

[28] M. Shi, J. Liu, B.-Q. Cao, Y. Wen, and X. Zhang, "A prior knowledge based approach to improving accuracy of web services clustering," in *Proceedings of 2018 IEEE International Conference on Services Computing (SCC)*, IEEE, San Francisco, CA, USA, July 2018.

[29] M. A. A. Cox and T. F. Cox, "Multidimensional scaling," *Handbook of Data Visualization*, Springer, Berlin, Heidelberg, pp. 315–347, 2008.

[30] J. C. Gower, "Properties of Euclidean and non-Euclidean distance matrices," *Linear Algebra and Its Applications*, vol. 67, pp. 81–97, 1985.
[31] W. Härdle and L. Simar, "Multidimensional scaling," in *Applied Multivariate Statistical Analysis*Springer, Berlin, Heidelberg, 2003.
[32] C. K. I. Williams, "On a connection between kernel PCA and metric multidimensional scaling," *Machine Learning*, vol. 46, no. 1/3, pp. 11–19, 2002.
[33] W. K. Härdle and L. Simar, "Multidimensional scaling," in *Applied Multivariate Statistical Analysis*Springer, Berlin, Heidelberg, 2019.
[34] X. Yin, J. A. N. Goudriaan, E. A. Lantinga et al., "A flexible sigmoid function of determinate growth," *Annals of Botany*, vol. 91, no. 3, pp. 361–371, 2003.
[35] L. Rokach, "A survey of clustering algorithms," *Data Mining and Knowledge Discovery Handbook*, Springer, Boston, MA, USA, pp. 269–298, 2009.
[36] S. Wold, E. Kim, and G. Paul, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1-3, pp. 37–52, 1987.
[37] J. Swarndeep Saket and P. Sharnil, "An overview of partitioning algorithms in clustering techniques," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 5, no. 6, pp. 1943–1946, 2016.
[38] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.
[39] M. Agarwal, R. Jaiswal, and A. Pal, "k-means++ under approximation stability," *Theoretical Computer Science*, vol. 588, pp. 37–51, 2015.
[40] ProgrammableWeb, 2020, https://www.programmableweb.com/.
[41] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for K-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.
[42] L. Kaufman and P. J. Rousseeuw, —*Finding Groups in Data*‖, John Wiley, New York, NY, USA, 1990.