Information Modelling and Knowledge Bases XXXIII
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA210480

On Semantic Spatiotemporal Space and Knowledge with the Concept of "Dark-Matter"

Xing Chen[1] and Kiyoki Yasushi[2]
[1] *Department of Information & Computer Sciences*
Kanagawa Institute of Technology, Japan
[2] *Graduate School of Media and Governance, Keio University, Japan*

Abstract. It is highlighted for machine learning models implementing functions based on data training without program coding. Artificial neural network is one of the efficient machine learning models. Different from the other machine learning models like artificial neural network, we have presented semantic computing models which represent "meaning" of machine learning results. In our model, semantic spaces are created based on training data sets. Data calculations are performed on the space. Data are mapped to semantic spaces and presented as points in semantic spaces. The mapped positions of data represent the "meaning" of data. In this paper, we first present our new discovery in the formation of semantic spaces. We use the word "matter" to represent features of semantic spaces which are related to the non-temporal data. As the same time, we use the word "dark-matter" to represent the features of semantic spaces which are temporally changed. We use the word "energy" to represent matrixes which are used in the semantic computations to generate output data. We reveal that the "dark-matter" is the spatiotemporal matrix and present a mechanism of "memory" for implementing the semantic computation. The most important contribution of this paper is that we developed a new mechanism for implementing machine learning with "knowledge" in the "memory." In the paper, we use case studies to illustrate the concepts and the mechanism. At the beginning, we present an example on creating a semantic space from a "chaotic state" to an "ordered state." After that, we use examples to illustrate the mechanism of the "memory" and the semantic computation. The space expansion and the space division are also illustrated by examples.

Keywords. Artificial intelligent, semantic space, spatiotemporal space, multiple semantic space transmitting, machine learning

1. Introduction

Program coding is generally required for implementing functions on computers. Although many program languages and developing tools are proposed, it is still a hard work for program coding. Moreover, it is not always possible to create models to implement required functions. Machine learning is one of the methods to implement functions and models without the requirement of program coding. The functions or models can be implemented or created through training process. During the training

[1] Xing Chen, 1030 Simo-Ogino, Atsugi-shi, Kanagawa 243-0292, Japan; chen@ic.kanagawa-it.ac.jp

process, training data are used, which are constructed by input data and expected output data. Based on the training data, todays machine learning technics are applied to wide application areas, such as image recognition, object classification, data retrieval, etc.

Different from the other machine learning models like artificial neural network, we have presented semantic computing models which can present "meaning" of learning results [1, 2, 3, 4]. In our model, input data are mapped through mapping matrixes into semantic spaces and presented as points in semantic spaces. Distances of the points present the "meaning". In our models, semantic calculation is transmitted to calculate Euclidean distances of those points. For example, in the case for implementing semantic query, a query data set is mapped into a semantic space and summarized as a point in the space. Retrieval candidate data are also mapped into the semantic space and summarized as other points. Euclidean distance is calculated between the query point and each retrieval candidate point. When the distance of a retrieval candidate is shorter than a given threshold, its relative retrieval candidate is extracted as the query output.

In our method, creating a semantic space is a basic operation. This space can be created by applying one of the following two methods, Mathematical Model of Meaning (MMM) [3, 4] and Semantic Feature Extracting Model (SFEM) [1, 2] . The difference of the two methods is that, in MMM, a common data set, for example an English-English diction is used to create the space. But in SFEM, a different data sets are used to create different spaces according to the requirement of applications.

After the semantic space is created, input data will be mapped to the space and the Euclidean distance calculation between the mapped data points in the space will be performed. Mapping matrixes are required to map input data into the semantic space. In SFEM, different mapping matrixes are required when the semantic space model is applied in different application areas [5- 13]. Therefore, we developed many methods to create mapping matrixes and apply the model in the areas of semantic information retrieving [8, 9, 13], semantic information classifying [10], semantic information extracting [11], and semantic information analyzing on reason and results [12], etc. We furtherly developed a method to create the mapping matrixes through deep-learning [14]. Same as the semantic computation model, the multiple matrix calculation is also the basic computation for implementing artificial neural networks and deep-learning computation [15-18].

As true and false judgement are the basic computation required by machine learning, we present a mechanism to implement the basic logic computation based on the semantic space model [19]. Furthermore, we designed and conducted experiments for simulating unmanned ground vehicle control based on the mechanism [20]. Based on our studies presented in [19] and [20], we noticed that time factors are important in the semantic computation model.

In this paper, we first present our new discovery in the formation of semantic spaces. We use the word "matter" to represent features of semantic spaces which are related to the non-temporal data. As the same time, we use the word "dark-matter" to represent the features of semantic spaces which are temporally changed. We use the word "energy" to represent matrixes which are used in the semantic computations to generate output data. We reveal that the "dark-matter" is the spatiotemporal matrix and present a mechanism of "memory" for implementing the semantic computation. The most important contribution of this paper is that we developed a new mechanism for implementing machine learning with "knowledge" in the "memory." In the paper, we use case studies to illustrate the concepts and the mechanism. At the beginning, we

present an example on creating a semantic space from a "chaotic state" to an "ordered state." After that, we use examples to illustrate the mechanism of the "memory" and the semantic computation. The space expansion and the space division are also illustrated by examples.

In the following, we first briefly review the semantic computation model and the mechanism to implement logic computations based on the semantic computation in section 2. After that, we illustrate the concept of the "matter," "dark-matter," "energy" and "knowledge" with examples in section 3. In this section, we also illustrate how to create the semantic space from the "chaotic state" to the "ordered state" with an example. In this example, the concepts of the "chaotic state" and the "ordered state" are illustrated. In the same time, the mechanism of "knowledge" for data retrieval on the semantic space and the relation between "knowledge" and "dark-matter" are illustrated. In section 4, we illustrate the concept of the space expansion from subspaces and dividing a space into subspaces. Finally, we will present our conclusions in section 5.

2. The semantic computation models and their applying to logic calculations

In this section, we first briefly review two semantic computation models, the Semantic Feature Extracting Model (SFEM) and the Mathematical Model of Meaning (MMM). After that, we review the mechanism to implement logic computations based on the semantic computation.

2.1. The semantic computation models

In the semantic space model, we create a semantic space by a pre-selected training data set into several clusters. In SFEM model [1, 2], a data set is used where each of the clusters has a feature that some common features of data frequently appear among the data sets in the same cluster but rarely appear in the data sets of the other clusters. The common features which frequently appear in a cluster C_i are referred to as C_i's key features. By using the training data clusters, we construct a matrix, which is referred to as *K-C* matrix. In the *K-C* matrix, each of the rows corresponds to a key feature set K_i, and each of the columns corresponds to a cluster. The ij^{th} entry of the matrix is the number of the key features in the set K_i appearing in the cluster C_j. Because key features in the set K_i only appear in the cluster C_i, therefore, if i is not equal to j, $i \neq j$, the value of the ij^{th} entry is 0. Therefore, the *K-C* matrix is a diagonal matrix. That is to say that an orthogonal space is created. The value of the ii^{th} entry of the matrix is the number of the elements of the set K_i, $|K_i|$.

Next step in the semantic space model is to map data into the semantic space. By using the *K-C* matrix, each cluster is represented as a q dimensional vector. We use $\mathbf{C}_i$ to represent the vector of the cluster C_i. We use a unit vector $\mathbf{c}_i$, where its norm is 1 ($|\mathbf{c}_i|=1$), to represent the cluster vector $\mathbf{C}_i$ as $\mathbf{C}_i=|K_i|\mathbf{c}_i$. When the data are classified into q clusters, we obtain q cluster vectors. Therefore, we define q unit vectors $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_q$, to represent the q cluster vectors. We refer the vector space constructed by the q unit vectors to as the "*space*". Because the inner product of two different unit vector is 0, $(\mathbf{c}_i \cdot \mathbf{c}_j)=0$, $i \neq j$, and there are q unit vectors, the space is a q dimensional orthogonal space.

When data d_j, is vectorized according to the key features, the count of the occurrences of the C_i's key features in the data d_j is defined to $e_{i,j}$. We set the value of $e_{i,j}$ based on the following rule: *when a key feature in the key feature set K_i appearing in the data d_j, it is counted only once*. If the number of the key feature sets is q, the data d_j is vectorized to a q dimensional vector. We represent the vector of the data d_j as $\mathbf{d}_j$:

$$\mathbf{d}_j = \begin{bmatrix} e_{1,j} \\ e_{2,j} \\ \vdots \\ e_{q,j} \end{bmatrix}$$

We use v_t to represent the counted value of a key feature t. In the following, we use the expression $\sum_{t \in K_i} \{v_t\}$ to represent the sum of $v_{t_1} + v_{t_2} + \cdots + v_{t_a}$, where, t_1, t_2 ... t_a are the elements of the key feature set K_i:

$$\sum_{t \in K_i} \{v_t\} = \left\{ v_{t_1} + v_{t_2} + \cdots + v_{t_a} \middle| t_1 \in K_i, t_2 \in K_i, \cdots, t_a \in K_i \right\}.$$

In this way, the calculation for $e_{i,j}$ is represented by the following formula:

$$e_{i,j} = \sum_{t \in K_i} \{ v_t \mid if\ t \in d_j\ v_t = 1\ else\ v_t = 0 \},$$

where, K_i is the set of the C_i's key features and v_t is the counted value of one of the C_i's key feature t. If the data d_j contains the key feature t, v_t is set to 1. If the data d_j does not contain the key feature t, v_t is set to 0.

With the definition of the retrieval space, we express the data vector $\mathbf{d}_j$ as

$$\mathbf{d}_j = \sum_{i=1}^{q} e_{i,j} \mathbf{c}_i .$$

In this way, data are mapped onto the q dimensional space.

The third step is to calculate Euclidean distances for data retrieval, classification or recognition. Take the data query as an example. In the processing of data query, the retrieval candidates are mapped in the semantic space and summarized as retrieval candidate points. A query is also mapped in the semantic space and summarized as a query point. We use two methods to implement the Euclidean calculation. The first method is to calculate the distances between the retrieval candidate points and query point. The second method is to is to select a subspace by a given query and mapped the query into the original point of the subspace and calculate the length of each retrieval candidate points from the original point. That is to calculate the norms of the retrieval candidate points. By ranking the retrieval candidates based on the norms of their relative points, the query result is obtained.

When a subspace is selected from the semantic space based on queries, the subspace is a v-dimensional space which is a part of the q-dimensional space, where v is smaller than q. The v-dimensional subspace correlates to v clusters. The subspace is selected by the following steps.

(1) When a query Q is given, the data which contain the same features in the query is searched. Data that have the same feature as those in the query are

extracted.

(2) From all the component items of the selected data vector, the cluster vector $\mathbf{c}_i$ is extracted where the related component item $|e_{i,j}\mathbf{c}_i|$ has the maximum value.

$$\left|e_{i,j}\mathbf{c}_i\right| = MAX\left(e_{1,j}, e_{2,j}, \cdots, e_{q,j}\right)$$

We use a vector **q** referred to as the query vector to represent the extracted clusters. We add the extracted cluster vector $\mathbf{c}_i$ to the vector **q**,

$\mathbf{q}=\mathbf{q}+\mathbf{c}_i$; where the initial value of **q** is **0**.

(3) A retrieval subspace *S* corresponding to the query is selected from the entire retrieval space by calculating the inner product of $\mathbf{c}_i$ and **q**. If the value of the inner product $\mathbf{c}_i \cdot \mathbf{q}$ is greater than or equal to a threshold ε, which is referred to as the subspace selection threshold, $\mathbf{c}_i$ is added to *S*.

These steps (2) and (3) are repeated for each extracted data vector.

When the subspace *S* is obtained, the data vector on the subspace is projected and represented as

$$\mathbf{d}_j = \sum_{i=1}^{q} \left\{ e_{i,j}\mathbf{c}_i \middle| \mathbf{c}_i \in S \right\}.$$

In the second step, the data are ranked by calculating the norms of the data vectors on the selected subspace:

$$\left|\mathbf{d}_j\right| = \left| \sum_{i=1}^{q} \left\{ e_{i,j}\mathbf{c}_i \middle| \mathbf{c}_i \in S \right\} \right|.$$

In MMM, the semantic interpretation is performed as projections of the semantic space dynamically, according to contexts, as shown in Figure 1.

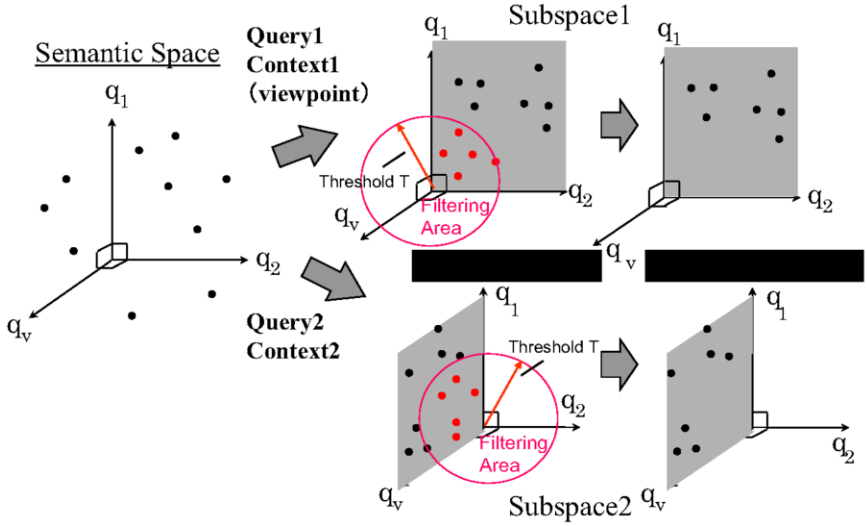

Figure 1. Semantic interpretation according to contexts in MMM

In the Mathematical Model of Meaning (MMM) [4, 7], an orthogonal semantic space is created for semantic associative search. Retrieval candidates and queries are

mapped onto the semantic space. The semantic associative search is performed by calculating the correlation of the retrieval semantic space.

In MMM, the acquisition of information or knowledge is performed by semantic computations. Context-dependent interpretation means that information is dynamically extracted by a semantic computation with context-recognition. The method realizes the computational machinery for recognizing the meaning of contexts and obtaining the semantically related information to the given context. MMM is essentially different from those methods. The essential difference is that this method provides dynamic recognition of the context. That is, the "context-dependent interpretation" is realized by dynamically selecting a certain subspace from the entire semantic space. The other methods do not provide the context dependent interpretation, that is, their space is fixed and static. The outline of MMM [4, 7] is summarized as the following:

The semantic associative computing algorithm is extended to include a deep-learning process in the MMM semantic space in the following steps:

(1) A set of *m* words is given, and each word is characterized by *n* features. That is, an *m* by *n* matrix ***M*** is given as the data matrix.

(2) **"Context words" and "image"** are characterized as **"context"** by using the *n* features and representing them as *n*-dimensional vectors.

(3) The context words and **"image"** are mapped into the orthogonal semantic space by computing the Fourier expansion for the *n*-dimensional vectors.

(4) A set of all the projections from the orthogonal semantic space to the invariant subspaces (eigen spaces) is defined. Each subspace represents a phase of meaning, and it corresponds to "**context."**

(5) A subspace of the orthogonal semantic space is selected according to the given **"context"** expressed in *n*-dimensional vectors, which are given as **"context"** represented by **"a sequence of words"** and **"image."**

(6) The most correlated information resources to the given **"context"** are extracted as the selected subspace by applying the metric defined in the semantic space.

2.2. Implement logic computations based on the semantic space model

Logical design is performed based on truth tables. In the truth table, all output values are given to all possible input values. The input data values and output data vaudes are Boolean values. That is, the value can only be '1' or '0'. For example, if there are two logical input data, x_1 and x_2, all the possible input data values of the input data x_1 and x_2 are (0, 0), (0, 1), (1, 0) and (1, 1), in which (a, b) means the value of x_1 is 'a' and the value of x_2 is 'b'. As shown in Table 1, each input data pair is given a corresponding output value in the truth table. The output values '0', '0', '0' and '1' are given to the input values (0, 0), (0, 1), (1, 0) and (1, 1) in the "and" logic truth table. Based on the truth table, logical formulas are derived. For example, for the "and" logic, an equation

$$y=x_1*x_2,$$

is derived, where '*' presents logic "and". In the same way, an equation

$$y= \tilde{}x_1*x_2+x_1*\tilde{}x_2$$

is also derived for the "xor" logic, where '*', '+' and '~' present logic "and", "or" and "not", respectively.

Table 1. Truth Table of "and," "or" and "xor" logic

"and" logic

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

"or" logic

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

"xor" logic

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

A logical system is constructed by the derived formulas from the truth table. Boolean algebra is applied to simplify the derived formulas in order to reduce the complexity of the designed system. For example, the formula

$$a*b+a*c+\tilde{}a*\tilde{}c+b*c$$

can be simplyfied as

$$b+a*c+\tilde{}a*\tilde{}c.$$

In the following, we use an example to illustrate how to implement "and," "or" and "xor" combination logical calculation based on the semantic calculation model. First, we present a data set with two inputs x1 and x2 and three outputs corresponding to "and," "or" and "xor," as shown in Table 2.

Table 2. State transition table

x_1	x_2	Y_{and}	Y_{or}	Y_{xor}
0.2	0.1	0.1	0.2	0.1
0.1	0.9	0.2	0.8	0.9
0.9	0.1	0.1	0.9	0.8
0.9	0.8	0.9	0.8	0.2

The values in data set are set as: when a value is close to 0, it might be logic '0;' when a value is close to 1, it might be logic '1.' If a data value is 0.5, it might be logic '0' or logic '1.' For the given data set, $x_1 = 0.9$ and $x_2 = 0.1$, it means that the input might be $x_1 = 1$ and $x_2 = 0$. The "and" output of it might be '0.' The "and" output of it might be '0.' The "or" output of it might be '1,' and the "xor" output might be '1.'

Set a data set as a matrix M. A well-known method of the principal component analysis is the Singular Value Decomposition (SVD), which is a matrix computation widely used in spectral analysis, eigenvector decomposition and factor analysis. The computation is performed on a matrix with different entities on the rows and the columns. When SVD is performed on the matrix M, this matrix is decomposed into three other matrixes that contain "singular vectors" and "singular values". We call these three matrixes as U, S and V:

$$M = U*S*V'$$

where, S is a diagonal matrix that contains singular values, matrixes U and V are left and right matrix of S, respectively. V' is the transposed matrix of V. The matrix V has orthonormal columns, that is

$$V'*V=I$$

where I is the identity matrix.

We call the space U*S is the semantic space created by the matrix M. As

$$M = U*S*V'$$
$$M*V = U*S*V'*V$$
$$M*V = U*S*I$$
$$M*V = U*S,$$

we call the matrix V as the *space mapping matrix*. That is, any matrixes of data sets which have the same number of columns of the matrix M can be mapped to the semantic space through the mapping matrix V.

When SVD is performed to the matrix of the given data set,

$$M = \begin{bmatrix} 0.2 & 0.1 & 0.1 & 0.2 & 0.1 \\ 0.1 & 0.9 & 0.2 & 0.8 & 0.9 \\ 0.9 & 0.1 & 0.1 & 0.9 & 0.8 \\ 0.9 & 0.8 & 0.9 & 0.8 & 0.2 \end{bmatrix},$$

we get three matrixes:

$$U = \begin{bmatrix} -0.1 & -0.1 & 0.1 & -1.0 \\ -0.5 & 0.6 & -0.6 & 0.0 \\ -0.5 & 0.3 & 0.8 & 0.1 \\ -0.6 & -0.7 & -0.2 & 0.1 \end{bmatrix}$$

$$S = \begin{bmatrix} 2.5 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.9 & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.8 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.5 & -0.4 & 0.6 & 0.1 & -0.5 \\ -0.4 & 0.0 & -0.7 & -0.2 & -0.5 \\ -0.3 & -0.6 & -0.3 & 0.5 & 0.5 \\ -0.6 & 0.1 & 0.1 & -0.6 & 0.5 \\ -0.4 & 0.7 & 0.1 & 0.6 & 0.0 \end{bmatrix}.$$

The semantic space U*S is

$$\begin{bmatrix} -0.3 & 0.0 & 0.1 & 0.0 & 0.0 \\ -1.3 & 0.6 & -0.5 & 0.0 & 0.0 \\ -1.3 & 0.2 & 0.6 & 0.0 & 0.0 \\ -1.6 & -0.7 & -0.2 & 0.0 & 0.0 \end{bmatrix}.$$

This is a five-dimensional space. Each row of the matrix represents a mapping point of the data set. As it is a four rows matrix, that means four points are mapped to the semantic space. It is worth to notice that the values of the last two columns of the matrix is zero, therefore, we remove the last two rows and get a new matrix P,

$$\begin{bmatrix} -0.3 & 0.0 & 0.1 \\ -1.3 & 0.6 & -0.5 \\ -1.3 & 0.2 & 0.6 \\ -1.6 & -0.7 & -0.2 \end{bmatrix}.$$

The meaning of removing the last two rows of the matrix is that the semantic space is compressed from a five-dimensional space to a three-dimensional space.

Strictly speaking, that is each of the value of the last two rows is smaller than a threshold. In this example, the absolute value of the threshold is set to 0.003.

As U'*U=I, that is

$$U'*U=\begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{bmatrix},$$

the data set is mapped to an orthogonal space. This orthogonal characteristic is not changed to the compressed space:

$$P'*P = \begin{bmatrix} -0.3 & -1.3 & -1.3 & -1.6 \\ 0.0 & 0.6 & 0.2 & -0.7 \\ 0.1 & -0.5 & 0.6 & -0.2 \end{bmatrix} * \begin{bmatrix} -0.3 & 0.0 & 0.1 \\ -1.3 & 0.6 & -0.5 \\ -1.3 & 0.2 & 0.6 \\ -1.6 & -0.7 & -0.2 \end{bmatrix}$$

$$= \begin{bmatrix} 6.2 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.0 \\ 0.0 & 0.0 & 0.6 \end{bmatrix}.$$

As the data set is mapped to an orthogonal space, Euclidean distance calculation can be applied to calculate the distance of a new mapped point to the points already in the space.

For example, give a new input set $x_1 = 0.7$, $x_2 = 0.3$. As is different from the given data set, we should calculate four outputs Y_{and}, Y_{or} and Y_{xor} for the given input data set. As the output value is not known, we set the values of the three outputs as 0.5, that means it might be logic '0' or logic '1.' Thus we get a vector

[0.7 0.3 0.5 0.5 0.5].

Mapping the vector to the semantic space,

[0.7 0.3 0.5 0.5 0.5]* V,

we get a mapping point p_x represented as a three-dimensional vector,

p_x - [-1.1 -0.2 0.2].

Dividing each rows of matrix P, we get four mapping points of the given data set in the semantic space, p_1, p_2, p_3 and p_4, represented as four vectors:

p_1 = [-0.3 0.0 0.1],
p_2 = [-1.3 0.6 -0.5],
p_3 = [-1.3 0.2 0.6],
p_4 = [-1.6 -0.7 -0.2].

Calculating the Euclidean distances of p_x to p_1, p_2, p_3 and p_4, we get four values: 0.64, 1.01, 0.44 and 0.58. Among them, the smallest value is the third one, 0.44. That is, point p_x is most close to the point p_3. The value of p3 in the data set is

[0.9 0.1 0.1 0.9 0.8],

thus, we set the output value as $Y_{and} = 0.1$, $Y_{or} = 0.9$ and $Y_{xor} = 0.8$ for the input value $x_1 = 0.7$ and $x_2 = 0.3$.

3. The concept of the "matter," "dark-matter," "antimatter," "energy" and "knowledge"

In this section, we use a case study to illustrate the concept of the "matter," "dark-matter," "antimatter" and "energy." For the logical computation as reviewed in Section 2, we use two matrixes M and E to implement it,

$$Y = M*E,$$

where M represents an input space, E is a mapping matrix to map input data to the output space, "*" represents matrix multiply and Y is the matrix represents output data.

As shown in Figure 2, if input data is a two-dimensional logical value matrix, all the possible input data are (0, 0), (0, 1), (1, 0) and (1, 1), which is a four-rows and two-columns matrix. Take it as an example, we use four data 0.0, 0.1, 0.2 and 0.3 as an index of the input matrix. As the input data are that we use to obtained output results, that is, the input data are "visible" to us, we refer to the data in the index as "***matter***." For example, the data "0.2" in first column of the space matrix X is a kind of "matter" which indicates the input data "10." In order to calculate the invers matrix of X, we use random data to fill the other three columns of X. We refer to the data in these columns as "***dark-matter***." As these columns are filled with random data, we refer to the space X as "chaotic space."

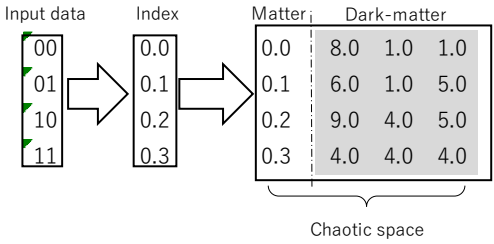

Figure 2. Creating a "chaotic space"

Before change the space from "chaotic space" to ordered state, let's introduce a concept of "time" used in this paper. We refer to the "time" as state transition. Considering our system as a state machine, the state of the system will be changed as the time lasted. If the states of a system are not changed, the time of the system is stopped. That is, as the time lasted, states transition is happened continuously. Figure 3 is an example of state transition diagram. In the figure, a circle and the number in the circle represents a state, an arrow represents a state transiting from one to another one. The numbers by the arrows represent the condition for the state transition. For example, if the number is "0.0," it means if the index number if "0.0," the state will be transited from state "0.0" to the state "0.1." In the example, we use the numbers of the index to indicate states.

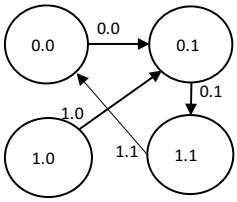

Figure 3. An example of state transition

Next, we introduce how to create the "ordered space" from the "chaotic space" as the time lasted. The example of the state transition diagram shown in Figure 4 is used to illustrate it. In the example, state "0.0" transits to the state "0.1" at the first step. Figure 4 (b) shows this state transition. In the next step, the state "0.1" transits to the state "0.3." Figure 4 (c) shows this state transition. At last, when the state transition diagram shown is represented as Figure 4 (d), we say that we created an "ordered state" from the "chaotic state" shown in Figure 4 (a). In the figure, the "matter" is represented in the first row of the matrix, and the "dark-matter" is represented in the second, third and fourth row of the matrix, painted gray.

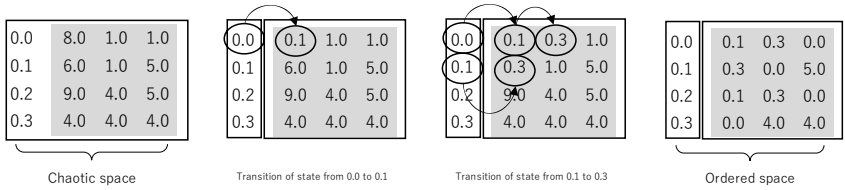

Figure 4. Changing the "chaotic space" to "ordered space"

From Figure 4, we can find that the "dark-matter" is a space represents state transition. It determines the state transition rules of the "matter" in space which we created. Based on the "ordered space," we refer to the inverse matrix of the "ordered space" as the "anti-matter space." If X^{-1} is an inverse matrix of an "ordered space" X, data in the matrix X^{-1} are referred to as "***anti-matter***." An example of the "anti-matter space" is shown in Figure 5 (b), which is the inverse matrix of the space represented as a matrix X shown in Figure 5 (a).

By applying antimatter to the desired result, we create a matrix. We refer to this matrix as "***energy***." Taking the "and," "or" and "xor" logical calculation results as shown in Table 1 as an example, for all possible input data values, the related output result values are shown in Figure 5 (c), represented as Y_{and}, Y_{or} and Y_{xor}. Three different kinds of "energy", E_{and}, E_{or} and E_{xor} are represented as three vectors in Figure 5 (d), where "energy" is created by the matrix multiplying result of "antimatter" and calculating results:

$$E = X^{-1} * Y .$$

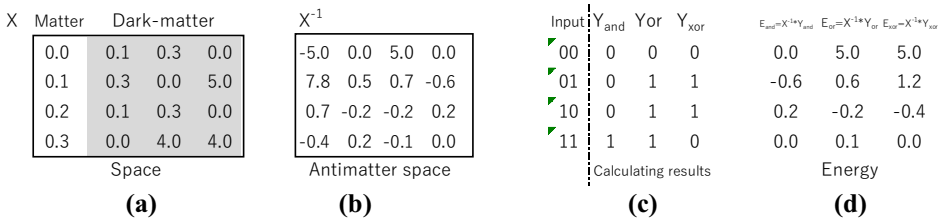

Figure 5. The concept of the "matter," "dark-matter," "anti-matter" and "energy."

For a given input, we use a function *mem* which we refer to as "memory" to extract its relative row vector from the space,

$$M = mem(\text{Input}).$$

The row vector M is composed by "matter" and "dark-matter," we simply call it "matter." For example, if the input value is a vector [0, 1], its relative row vector [0.2, 0.1, 0.3, 0.0] is extracted from the space matrix X,

$$M = mem([0,1]),$$

$$M = [0.2\ 0.1\ 0.3\ 0.0] \begin{bmatrix} 0.0 \\ -0.6 \\ 0.2 \\ 0.0 \end{bmatrix}$$

$$M = 0 .$$

We implement the semantic computation by apply energy to the matter,

$$Y = M*E.$$

As an example, when energy E_{and} is applied to the matter [0.2, 0.1, 0.3, 0.0], we get a value "0" which is the logic "and" calculation, "0 and 1." In Figure 6, we summarize the two bits logical "and," "or" and "xor" calculations. When the matter shown in Figure 6 (a) is multiplied by the energy shown in Figure 6 (b), the logical calculations "and," "or" and "xor" are performed. The calculation results are shown in Figure 6 (c).

M			
0.0	0.1	0.3	0.0
0.1	0.3	0.0	5.0
0.2	0.1	0.3	0.0
0.3	0.0	4.0	4.0

(a)

E_{and}	E_{or}	E_{xor}
0.0	5.0	5.0
-0.6	0.6	1.2
0.2	-0.2	-0.4
0.0	0.1	0.0

(b)

$M*E_{and}$	$M*E_{or}$	$M*E_{xor}$
0	0	0
0	1	1
0	1	1
1	1	0

(c)

Figure 6. Implementing calculation by applying "energy" to "matter"

As "energy" E is calculated with invers of matrix X,

$$E = X^{-1}*Y ,$$

the calculation of M*E can be written as

$$M*E = M*(X^{-1}*Y) .$$

In the case that the matrix M calculated through the memory function mem is the same as the matrix X, we obtain the equation,

$$\begin{aligned} M*E &= M*(X^{-1}*Y) \\ &= X*(X^{-1}*Y) \\ &= X*X^{-1}*Y \\ &= I*Y \\ &= Y, \end{aligned}$$

where, I is a unit matrix. It is required that the memory function should recall the original matrix X based on the input.

A simple way to implement the memory function is to use a list with the columns of input data and the matrix M as shown in Figure 7 (a). In the list, each input data

corresponds a row of matrix M. As shown in the figure, if the input data is [1, 0], the output of the memory function *mem*([1, 0]) is a row vector [0.2, 0.1, 0.3, 0.0].

However, in some cases, one input value may correspond two or more rows of matrix M. It will happen if the input data is obtained from a senor, but the resolution of the sensor is not high enough to provide the required input data value. As shown in Figure 7 (b), although two-bit input data values are required, but the efficient input values are only one-bit values. As we can not tell the difference for the first, second and the third, fourth rows, we use the medium value between 0.0 to 0.1, and that between 0.2 to 0.3, that is, "0.05" and "0.25," respectively as the index of "00" and "10," as shown in Figure 7 (b).

Input	M			
00	0.0	0.1	0.3	0.0
01	0.1	0.3	0.0	5.0
10	0.2	0.1	0.3	0.0
11	0.3	0.0	4.0	4.0

(a)

Input	M			
00	0.05	0.1	0.3	0.0
00	0.05	0.3	0.0	5.0
10	0.25	0.1	0.3	0.0
10	0.25	0.0	4.0	4.0

(b)

Figure 7. Implementing calculation by applying "energy" to "matter"

We add the "dark-matter" in the memory function as "***knowledge***" to implement the recall function. As shown in Figure 8 (a), at the beginning of the system, we know that the next state is "0.1." If the input is "00," the "dark matter" vector [0.1, 0.3, 0.0] is added to the "matter" [0.05]. Thus, the output the memory function is a row vector, [0.05, 0.1, 0.3, 0.0]. If the input is "10," the "dark matter" vector [0.1, 0.3, 0.0] is added to the "matter" [0.25]. Thus, the output the memory function is a row vector, [0.25, 0.1, 0.3, 0.0]. Applying "energy" to these output vector, as shown in Figure 8 (b), the logical calculations are correctly implemented.

Input	M			
00	0.05	0.1	0.3	0.0
00	0.05	0.3	0.0	5.0
10	0.25	0.1	0.3	0.0
10	0.25	0.0	4.0	4.0

(a)

$M*E_{and}$	$M*E_{or}$	$M*E_{xor}$
0	0	0
0	1	1
0	1	1
1	1	0

(b)

Figure 8. Implementing calculation by applying "energy" to "matter"

As described above, "dark-matter" represents the rule of the space that we created. Based on the rule, we know that when the state transition happens, which row of the matrix M should be extracted. As the example of Figure 8 (a), at the beginning, the state of the system is "0.0." If the input data is "10", the third and fourth rows relate to the input data. As the next state of the state "0.0" is "0.1," the third row of the matrix M is extracted. That is, "knowledge" is the mechanism of the memory function which extract rows as the output from the space M based on the rule of the space. We refer to the space M as the "semantic space." We refer to the process of extracting rows from the matrix M related to the input data and current state as the "semantic retrieval." We refer to the processing of applying the matrix E to the matrix M as the "semantic computation."

4. The concept of "space expansion" and "space division"

In this section, we use a case study to illustrate the concept of "space expansion" and "space division." We use a virtual agent in this case study. The agent moves on a plane surface with obstacles. As shown in Figure 9 (a), the start point of the agent is marked as "S" and the goal is "G." The row number is indicated by two-bit numbers same as the column. The agent has five actions, "Non," "Left," "Right," "Up" and "Down," which are assigned with the number "0.0," "0.1," "0.2," "0.3" and "0.4," as shown in Figure 9 (b). The agent moves from the position "01" row and "00" column, as shown in Figure 9 (a). We use four bits number to indicated the position of the agent. Thus, the start position is "0100" and the goal position is "1111." When the agent starts to move, its next position is "0101." We use arrows to show the movement and the actions of the agent. Taking the positions as the states, we get a state transition diagram through the movement of the agent. That is, the "rule" of the semantic space is defined. Based on the "rule," we create the semantic space.

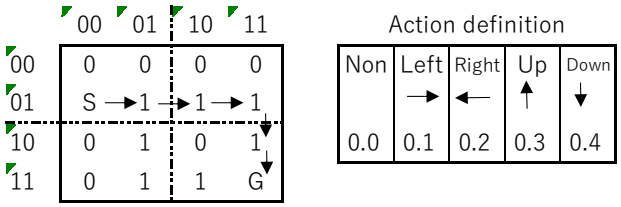

Figure 9. The plane surface for an agent moving and the definition of the agent action

In this example, there are 16 positions. Although we can create a 16 states space, but we prefer to use the 4 states space in order to introduce the concept "sub-space." As shown in Figure 9 (a), we divide the plain surface into four blocks, each part has 4 positions. As mentioned above, we use row-column number (abbr. RC) to indicate the positions. In the first block, the four positions are "0000," "0001," "0100" and "0101." As shown in Figure 10 (a), we set an index to the four positions as "0.0," "0.1," "0.2" and "0.3," respectively. The agent can move from the position "0100" to the position "0101." Taking the index number to represent the four possible states, we can draw a state transmission diagram, which has four states, as shown in Figure 10 (b). As the start position of the agent is at position "0100," the initial state is "0.2." As the agent is required to move left, the calculation result is set to "0.1" as defined in the Figure 9 (b). When the agent is moved left, the state transmission is happened from the state "0.2" to the state "0.3," as shown in Figure 10 (b). Based on the state transmission diagram, we create a space for the first block as shown in Figure 11 (a).

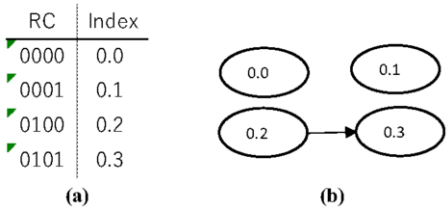

Figure 10. Implementing calculation by applying "energy" to "matter"

As shown in Figure 11, four spaces X_1, X_2, X_3 and X_4 are created for the plain surface. Each space corresponds to a block of the surface. We refer to these spaces as sub-spaces. A sub-space X_i is created based on the states and following the "rule" defined on the space. "Energy" E_i of the sub-space is created by applying the action matrix defined based on the "rule" of the space to the inverse matrix of the sub-space, X_i^{-1} . In Figure 11, four sub-spaces are created. As the shown in Figure 11 (d), no "rule" is defined on the fourth block, the created sub-space X_4 is a chaotic space. As no action is required on this block, "energy" is a zero vector.

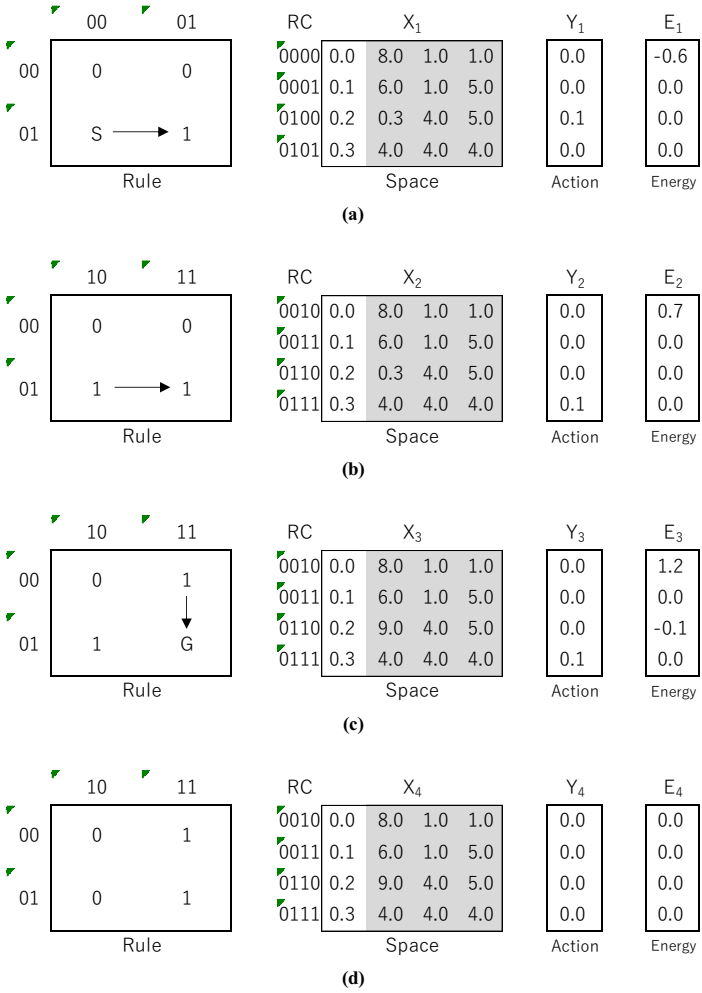

Figure 11. Creating four sub-spaces based on the movement of the agent

The semantic space of the plain surface is created through the creation of the sub-spaces. As shown in Figure 11, the semantic space is the same as the the sub-space X_1. When the sub-space X_2 is created, the semantic space is created with X_1 and X_2, and the number of the state of the semantic space are increased from 4 to 8. After the sub-space X_3 is created, the number of the state of the semantic space are increased to 16. That is, a sub-space must be created when a new state is added to the semantic space

and the number of the state of the semantic space are increased exponentially by 2. We refer to this process as the "***space expansion***."

Based on the sub-spaces, we can calculate actions of the agent on each block. However, we can not calculate the actions of the agent from one block to the others. For the four divided blocks, we define four positions, "0i0j," "0i1j," "1i0j" and "1i1j," where i and j are one-bit values "0" or "1." Using the four positions, we define relative index and states as shown in Figure 12. The agent starts at the position "0i0j," and moves to the position "0i1j." The goal of the agent is at the position "1i1j." Based on the "rule" of the movement of the agent, a space X with four states are created and "energy" E is also created by applying the agent action definition matrix Y to the inverse matrix of X, as shown in Figure 12.

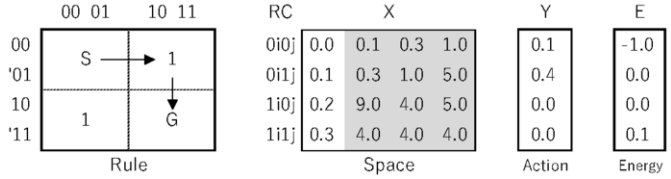

Figure 12. Creating the semantic space with four states

For the action of the agent in each block, we can create relative sub-space as shown in Figure 11. In summary, if the number of states for creating the semantic space are more than four, we divide them into two parts. In each part, if the number of states is still more than four, we divide them again until the number of the states in each part is smaller or equal to four. In the example shown in Figure 9 (a), the surface is divided twice into four blocks as shown in Figure 12. We refer to this process as "***space division***."

"Space division" is a process from global to local as shown in Figure 13. The results of the semantic computation on the global space shows the movement of the agent between blocks. Detail movements of the agent in each block are calculated on the local space.

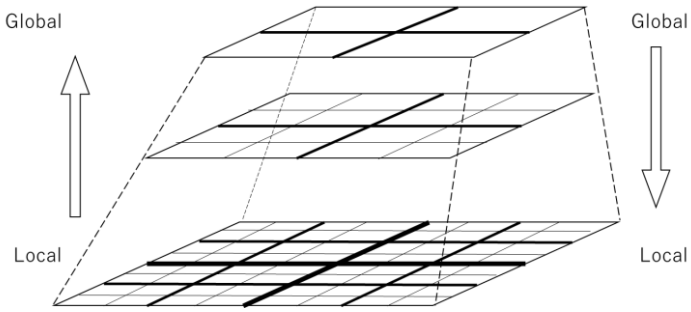

Figure 13. "Space expansion" and "Space division" among global and local

We use "space division" in the case that we have "knowledge", or in other words, we have "rules" from global to local. In the example shown in Figure 9 (a), the actions of the agent are shown from the start point to the goal point. That is, we have all

required "rules" for the agent moving on the surface. Therefore, we use "space division" to create sub-spaces for the agent movement between blocks and those in each block.

If we do not have all the required "rules," the semantic space creation is performed from local to global as shown in Figure 13. For example, if reinforcement learning is performed to the agent, "rule" and states are generated during the learning process. During the "rules" and states are generated, sub-spaces are created. Using reinforcement learning in a period of time, we can only get part of "rules" and states. All the other "rules" are remained to be found. Therefore, for the semantic space with all the possible states, only small parts of the space are in ordered state, the remained parts are in chaotic state as show in Figure 14.

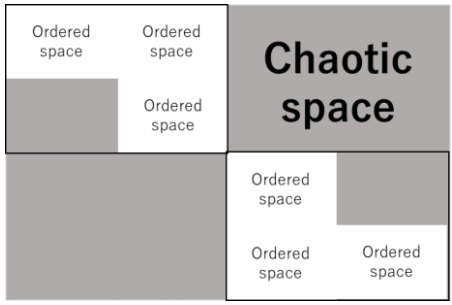

Figure 14. A semantic space creating through reinforcement learning

5. Conclusion and future work

In this paper, we presented that the semantic space is a spatiotemporal space. We introduced the concept "dark-matter" to reveal the temporal characteristic of the space. We defined state transmission as the "time" in a system performing semantic computation. In the paper, we illustrated following concepts and computations: A matrix X created with the states and state transmission diagram is referred to as the semantic space. The elements of the matrix representing states are referred to as "matter" and the elements representing the state transmission diagram are referred to as the "dark-matter." Its inverse matrix X^{-1} is referred to as the "anti-matter." The result of multiplying the "anti-matter" by the defined output is referred to as "energy." Memory function are defined to exact vectors representing "matter" and "dark-matter." "Knowledge" is required to be added to the memory function in the case that the input data cannot definitely indicate relative states. By applying "energy" to the output of the memory function, we get the semantic computation result related to the giving input. The "space expansion" is happened as the increasing of the "matters." When a state transmission diagram is given, we divide the diagram into sub-diagram and create relative sub-spaces. We refer to the process as the "space division." The divided semantic space is a pyramid shaped structure. Each layer of the pyramid is composed by sub-spaces. The top layer represents global state transmission diagram. The bottom layer represents details of the divided sub-diagram. Based on these concepts and mechanism, we developed a new mechanism for implementing machine learning. In this mechanism, machine learning is to calculate "energy" matrixes and "knowledge" is used to extract vectors from the semantic space based on the input data during the

semantic computation. Furthermore, we revealed that the semantic computation must be performed dynamically with the state transmission. We also revealed that the semantic computation must be performed from global to local to give rough and detailed calculation results. As our future work, we will use this new machine learning mechanism to implement artificial intelligence systems and furtherly confirm the effectiveness of the mechanism in practice.

References

[1] Chen, X. and Kiyoki, Y., "*A query-meaning recognition method with a learning mechanism for document information retrieval*," Information Modelling and Knowledge Bases XV, IOS Press, Vol. 105, pp.37-54, 2004.

[2] Chen, X. and Kiyoki, Y., "*A dynamic retrieval space creation method for semantic information retrieval*," Information Modelling and Knowledge Bases XVI, IOS Press, Vol. 121, pp.46-63, 2005.

[3] Kiyoki, Y. and Kitagawa, T., "*A semantic associative search method for knowledge acquisition*," Information Modelling and Knowledge Bases, IOS Press, Vol. VI, pp.121-130, 1995.

[4] Kitagawa, T. and Kiyoki, Y., "*A mathematical model of meaning and its application to multidatabase systems*," Proc. 3rd IEEE International Workshop on Research Issues on Data Engineering: Interoperability in Multidatabase Systems, pp.130-135, April 1993.

[5] Chen, X., Kiyoki, Y. and Kitagawa, T., "*A multi-language oriented intelligent information retrieval system utilizing a semantic associative search method,*" Proceedings of the 17th IASTED International Conference on Applied Informatics, pp.135-140, 1999.

[6] Chen, X., Kiyoki, Y. and Kitagawa, T., "*A semantic metadata-translation method for multilingual cross-language information retrieval,*" Information Modelling and Knowledge Bases XII, IOS Press, Vol. 67, pp.299-315, 2001.

[7] Kiyoki, Y., Kitagawa, T. and Hitomi, Y., "*A fundamental framework for realizing semantic interoperability in a multidatabase environment*, " International Journal of Integrated Computer-Aided Engineering, Vol.2, No.1(Special Issue on Multidatabase and Interoperable Systems), pp.3-20, John Wiley & Sons, Jan. 1995.

[8] Kiyoki, Y., Kitagawa, T. and Hayama, T., "*A metadatabase system for semantic image search by a mathematical model of meaning*, " ACM SIGMOD Record, Vol.23, No. 4, pp.34-41, Dec. 1994.

[9] Kiyoki, Y, Chen, X. and Kitagawa, T., "*A WWW Intelligent Information Retrieval System Utilizing a Semantic Associative Search Method*," APWeb'98, 1st Asia Pacific Web Conference on Web Technologies and Applications, pp. 93-102, 1998.

[10] Ijichi, A. and Kiyoki, Y.: "*A Kansei metadata generation method for music data dealing with dramatic interpretation*," Information Modelling and Knowledge Bases, Vol.XVI, IOS Press, pp. 170-182, May, 2005.

[11] Kiyoki, Y., Chen, X. and Ohashi, H.: "*A semantic spectrum analyzer for realizing semantic learning in a semantic associative search space*," Information Modelling and Knowledge Bases, Vol.XVII, IOS Press, pp.50-67, May 2006.

[12] Takano, K. and Kiyoki, Y.: "*A causality computation retrieval method with context dependent dynamics and causal-route search functions*," Information Modelling and Knowledge Bases, ISO Press, Vol.XVIII, pp.186-205, May 2007.

[13] Chen, X. and Kiyoki, Y.: "*A visual and semantic image retrieval method based on similarity computing with query-context recognition*," Information Modelling and Knowledge Bases, IOS Press, Vol.XVIII, pp.245-252, May 2007.

[14] Nitta T, "*Resolution of singularities introduced by hierarchical structure in deep neural networks*," IEEE Trans Neural Netw Learn Syst., Vol.28, No.10, pp.2282-2293Oct. 2017.

[15] Wiatowski, T. and Bölcskei, H., "*A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction*," IEEE Transactions on Information Theory, PP(99) · Dec. 2015.

[16] Hochreiter, S., Bengio, Y., Frasconi, P. and Schmidhuber, J. "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F. (eds.), *A Field Guide to Dynamical Recurrent Neural Networks*," IEEE Press, 2001.

[17] Hochreiter, S. and Schmidhuber, J., "*Long short-term memory*,". Neural computation, Vol.9, No.8, pp.1735-1780, 1997.

[18] Kalchbrenner, N., Danihelka, I. and Graves, "*A. Grid long short-term memory*," CoRR, abs/1507.01526, 2015.

[19] Chen, X. and Kiyoki, Y., "*On Logic Calculation with Semantic Space and Machine Learning*," Information Modelling and Knowledge Bases XXXI, IOS Press, Vol. 321, pp.324-343, 2019.
[20] Chen, X. Prayongrat, M. and Kiyoki, Y., "*A Concept for Control and Program Based on the Semantic Space Model*," Information Modelling and Knowledge Bases XXXII, IOS Press, Vol. 333, pp. 26-44, 2020.