CNN Based Ontology Learning Algorithm and Applied in PE Data

Guan Li

Abstract—**Convolutional neural networks have received extensive attention and been applied to regular data like image processing recently. This paper applied the convolutional neural network to calculate the ontology similarity, and proposed an ontology learning algorithm based on CNN. Meanwhile, a sports ontology covering most sports concepts was constructed, and the detection of the new ontology learning algorithm is proved to be effective for P.E. ontology.**

Index Terms—**ontology, PE, sport data, similarity measuring.**

I. Introduction

ONTOLOGY is a structured data model, and its conceptual structure can be represented by graphs. It has powerful conceptual semantic expression function and get theoretical support from the mathematical logic, so it is widely used in common application fields in artificial intelligence and pattern recognition such as graphics, semantic networks, information retrieval, and extended query. Moreover, ontology is widely used in interdisciplinary research, such as medicine, pharmacy, genetics, botany, chemistry, GIS, etc. For the application of ontology in interdisciplinary subjects, one can refer to Huitzil et al. [1] and [2], Benitez-Andrades et al. [3], Garcia-Diaz et al. [4], Nembaware et al. [5], Trappey et al. [6], Ong et al. [7], Liu et al. [8], Hafeez et al. [9], and Azevedo et al. [10].

In recent years, as the amount of data processed by ontology increases, more and more machine learning methods have been applied to ontology similarity calculation and ontology mapping. Qiu [11] proposed a new ontology mapping constructing by using low rank distance matrix optimization trick. Gao and Chen [12] gave an approximation analysis of ontology learning algorithm in linear combination setting. Gao et al. [13] presented a discrete dynamics approach for sparse calculation and applied it in ontology similarity measuring. Wu et al. [14] determined a disequilibrium multi dividing ontology learning algorithm which considers different statuses of ontology data. Margin based ontology sparse vector learning algorithm was considered in Gao et al. [15] and was applied in "GO" ontology. Chen et al. [16] raised an ontology-driven framework for similarity measuring in terms of vector learning techniques. Lan et al. [17] suggested a new ontology similarity computation and ontology mapping in light of distance matrix learning trick. He et al. [18] determined an ontology similarity measurement algorithm using kernel principal component analysis and spectral cut-off regression, and applied it in "GO" ontology. Wu et al. [19] presented new algorithms for ontology similarity measure and ontology mapping by determining the similarity matrix of ontology, and the optimisation strategy and iterative procedures are designed in terms of metric distance learning tricks. Zhu et al. [20] proposed a boosting based ontology sparse vector computation algorithm. More researchers on ontology can be referred to [21], [22] and [23].

This work aims to introduce the CNN based ontology learning algorithm and construct a PE ontology. In the following parts, we first describe the convolutional neural network and new ontology learning algorithm. Then, we state the PE ontology and the corresponding experiment data, finally some future ongoing works are raised.

II. CNN based ontology learning algorithm

A. Setting of convolution neural network

Early neural networks can't be used to perform nonlinear calculations. However, the BP neural networks enabled neural networks to perform them. Before the BP neural network, the network was not too deep, generally 3 or 4 layers, because the gradient calculation of the underlying neurons would be very complicated, and the training of the network would be more difficult. The BP neural network shows that the errors of the network can be passed back layer by layer, and the errors collected by the neurons of the previous layer can be used by the next layer. However, there are some disadvantages of BP algorithm which are (1) Network over-fitting problem. After the network has gone through many layers, its parameters will become more, but the amount of data will become smaller. (2) The problem of gradient disappearance or gradient explosion. The so-called gradient disappearance results from the existence of nonlinear fitting function in the network. If its derivative is less than 1, the error will be continuously multiplied by a number less than 1 after being passed layer by layer. Hence, the error of the underlying neuron will be particularly small. Its parameters cannot be updated. On the contrary, if the derivative of the nonlinear fitting function is greater than 1, the error is multiplied by a number greater than 1. It will form a diffusion, which is the gradient explosion problem. (3) The BP algorithm still has theoretical problems: the solution obtained through error back propagation will fall into the local optimal solution, and the global optimal solution cannot be obtained. (4) The interpretability is relatively poor. We don't know why some neurons have very large weights and some others have very small weights. After changing these weights, what impact will the network output have? In this context, convolutional neural networks are slowly introduced and quickly promoted.

Manuscript received December 14, 2020; revised May 19, 2021. This work was supported in part by the Guangdong Higher Education Teaching Reform Project: Research on the Management System of College Student Health Promotion Based on Sports Data, and Yuejiao Gaohan [2018] No. 180 Research on the Management System of College Students' Health Promotion Based on Exercise Data.

G. Li is a Lecturer of Department of Computer Science, Guangdong University Science and Technology, Dongguan 523083, China (phone: 15622522344; fax: 0769-86211865; e-mail: for_liguan@163.com).

Let $f(x)$ and $g(x)$ be two integrable functions on $\mathbb{R}$, and the continuous form convolution is defined as follows:

$$\int_{-\infty}^{\infty} f(\tau)g(x-\tau)d\tau.$$

Discrete spatial convolution is expressed as

$$y_n = \mathbf{x} * \mathbf{w} = \sum_{k=1}^{K} w_k x_{n-k},$$

where $\mathbf{x}$ is the signal and $\mathbf{w}$ is the convolution kernel. For example, the convolution kernel on a two-dimensional plane image is generally a 3×3 sliding frame.

Some basic concepts about convolutional neural networks are listed below:

Kernel size: also known as the convolution operation receptive field. In two-dimensional convolution, it is usually set to 3, that is, the size of the convolution kernel is 3×3. Generally it is an odd number. The advantage is that the center of the convolution kernel corresponds to the center of the convolution result.

Stride: The stride size when the convolution kernel traverses the image, the default setting is 1.

Padding: A trick to deal with sample boundaries. There are two advantages: (1) The image size of the convolution result can be made the same as the image size of the input result. (2) The area on the image boundary is relatively fair, and the number of convolutions is increased.

Input and output channels: When constructing a convolutional layer, the number of input channels I and the number of output channels O need to be defined. The parameter amount of each network layer is $I \times O \times K$ (K is the number of parameters of the convolution kernel). The color image has 3 channels. The output channel is the thickness of the image.

Pooling layer: It can be said that it is a special form of convolution.

Fully connected layer: The upper neuron and the lower neuron are all connected, and the amount of calculation parameters is relatively large, so it is used in the connection output layer.

Dropout: The neurons in the fully connected layer are inactivated with a certain probability, and the inactivated neurons no longer participate in training. The reference to Dropout effectively alleviates the overfitting of the model. Generally, it is only used in a complex structure such as a fully connected layer.

Pre-Training: First train a part of the small network to ensure stability, and then gradually deepen the network on this basis. Because the number of layers is very large, it may not be very effective to directly train a large network. You can train a small network by layer-by-layer pre-training, and then deepen it layer by layer.

The convolutional layer includes two parts: convolution calculation and nonlinear activation function. Among them, the role of nonlinear activation function in neural network is irreplaceable. Convolution calculation itself is a linear operation, and multiple consecutive linear convolutions are equivalent to one linear convolution. In this way, the computing power of the network will not be improved. Moreover, multiple convolutions will additionally introduce learnable parameters, which makes the network more prone to overfitting, for example: ReLU nonlinear activation function. When the input is greater than 0, the output value is equal to the input value; when it is less than 0, the input is 0; when it is greater than 0, the gradient value is equal to 1, so the gradient will not increase or decrease in the error back propagation. Therefore, the problems of disappearance and gradient explosion can be solved; when the input value is less than 0, the output value is also less than 0. At this time, the neuron is not involved in training, so the trainable parameters in the entire network are reduced. In this way, the overfitting problem can be alleviated. Due to its reduced calculation, alleviation of gradient disappearance, and overfitting, the ReLU activation function has now become the most common activation function in neural networks.

The function of the hidden layer is to perform linear conversion and nonlinear activation of the transferred value, and then transfer it to the next layer. Its essence is to transform the output data of the previous layer into another vector space. The features of traditional CNN can be concluded as follows:

- the neural layer adopts a three-dimensional form (width×height×depth).
- neuron local area connection (local perception), non-global connection Neurons share parameters (the same characteristics of different spatial locations can share the same sensory neuron, which greatly reduces the parameters that need to be trained), not completely independent.
- the dimensions of the output end can be adapted and changed according to the needs of the task (increasing the application range and capabilities of convolutional neural networks).

B. Connected to ontology setting

In ontology setting, all concepts (vertices) are structurally expressed by an ontology graph. In other words, the ontology graph structure is a kind of structured data, and traditional convolutional neural networks cannot handle structured data like ontology graphs. In order to apply the convolutional neural network to the ontology graph structure, we need to use graph convolution to replace the traditional convolution.

The classic Fourier transform can be expressed as

$$x(t) = \frac{1}{n} \sum_{w=0}^{n-1} e^{\frac{2\pi i}{n} tw} X(w)$$

and the corresponding formula in the ontology graph setting becomes

$$x(i) = \sum_{l=1}^{n} \widehat{x}(\lambda_l) u_l(i).$$

We only consider undirected ontology graphs, $\mathbf{W} \in \mathbb{R}^{n \times n}$ is the adjacency matrix ($W_{ij} = 0$ indicating that the vertices corresponding to i and j are not adjacent, otherwise W_{ij} is the weight of the edges), $\mathbf{D} \in \mathbb{R}^{n \times n}$ is the degree matrix (diagonal matrix), $D_{ii} = \sum_j W_{ij}$. The Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is a symmetric positive semi-definite matrix with n linearly independent characteristic vectors (they are a set of orthogonal basis in n-dimensional space with a modulus of 1). The characteristic vectors corresponding to different eigenvalues are orthogonal to each other. The matrix formed by these orthogonal eigenvectors is an orthogonal matrix, and the eigenvectors are non-negative real numbers.

The normalized form of the graph Laplacian matrix is stated by $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}}$.

The Eigen decomposition of Laplace matrix, also known as Spectral decomposition, is a method of decomposing a matrix into the product of its eigenvalues and eigenvectors:

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \mathbf{U}\begin{bmatrix}\lambda_1 & & \\ & \ddots & \\ & & \lambda_n\end{bmatrix}\mathbf{U}^{-1},$$

where n is the number of ontology vertices, $\mathbf{U} = (u_1, u_2, \cdots, u_n) \in \mathbb{R}^{n\times n}$ is an orthogonal matrix satisfying $\mathbf{U}\mathbf{U}^{-1} = \mathbf{I}$ (hence $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$), $u_i \in \mathbb{R}^n$ is the eigenvector corresponding to eigenvalue λ_i (here $i \in \{1, , n\}$).

The Laplacian operator Δ on the ontology graph is defined as the divergence of the gradient, i.e.,

$$\Delta f = div(grad(f)).$$

From this point of view, the Laplacian matrix is a class of Laplacian on the ontology graph. More generally, the Laplacian operator on the graph can be defined as follows:

$$\Delta f_i = \sum_{ij\in E(G)} (f_i - f_j),$$

where $f = (f_1, \cdots, f_n)$ represents the signal of each vertex on n vertices.

In most cases, the ontology graph is a weighted graph, and domain experts will define the edge types and weights according to the characteristics of the ontology in the particular application domain. In weighted graph setting, we have

$$\begin{aligned}\Delta f_i &= \sum_{ij\in E(G)} W_{ij}(f_i - f_j)\\ &= D_{ii}f_i - \sum_{j=1}^{n} W_j f_j,\end{aligned}$$

where it can be understood as the central vertex minus the surrounding vertices in turn, multiplied by the weight, and then summed up together. Hence, we infer

$$\begin{aligned}\Delta\mathbf{f} &= \begin{bmatrix}\Delta f_1\\ \vdots\\ \Delta f_n\end{bmatrix}\\ &= \begin{bmatrix}D_{11}f_1 - \sum_{j=1}^{n} W_{1j}f_j\\ \vdots\\ D_{nn}f_n - \sum_{j=1}^{n} W_{nj}f_j\end{bmatrix}\\ &= \begin{bmatrix}D_{11} & & \\ & \ddots & \\ & & D_{nn}\end{bmatrix}\mathbf{f} - \mathbf{W}\mathbf{f}\\ &= \mathbf{D}\mathbf{f} - \mathbf{W}\mathbf{f} = \mathbf{L}\mathbf{f}.\end{aligned}$$

Here, f is considered as the ontology graph value, and the above derivation can be understood as follows: the result of Laplacian on the value f on the ontology graph is equivalent to the ontology graph value f multiplied by the Laplacian matrix.

For the value $\mathbf{x} \in \mathbb{R}^n$ on the ontology graph, if we need to perform a Fourier transform, we need to find a set of orthogonal bases, where $x \in \mathbb{R}^n$ is expressed by linear combinations of these orthogonal bases. The graph Fourier transform actually uses the eigenvector of the Laplacian matrix $\mathbf{U} = (u_1, \cdots, u_n)$ as the basis function of the graph Fourier transform. Using Laplace's eigenvector as the basis function, any value on ontology graph can be expressed by

$$\mathbf{x} = \widehat{x}(\lambda_1)u_1 + \widehat{x}(\lambda_2)u_2 + \cdots + \widehat{x}(\lambda_n)u_n.$$

Hence, ontology graph inverse Fourier transform is stated as

$$\mathbf{x} = \widehat{x}(\lambda_1)u_1 + \widehat{x}(\lambda_2)u_2 + \cdots + \widehat{x}(\lambda_n)u_n$$

and its summation form is

$$x(i) = \sum_{l=1}^{n} \widehat{x}(\lambda_l)u_l(i).$$

Moreover, its matrix form can be denoted by

$$\begin{aligned}\begin{bmatrix}x(1)\\ x(2)\\ \vdots\\ x(n)\end{bmatrix} &= \begin{bmatrix}u_1(1) & u_2(1) & \cdots u_n(1)\\ u_1(2) & u_2(2) & \cdots u_n(2)\\ \vdots & \vdots & \ddots \quad \vdots\\ u_1(n) & u_2(n) & \cdots u_n(n)\end{bmatrix}\begin{bmatrix}\widehat{x}(\lambda_1)\\ \widehat{x}(\lambda_2)\\ \vdots\\ \widehat{x}(\lambda_n)\end{bmatrix}\\ &= (\overline{u}_1, \overline{u}_2, \cdots, \overline{u}_n)\begin{bmatrix}\widehat{x}(\lambda_1)\\ \widehat{x}(\lambda_2)\\ \vdots\\ \widehat{x}(\lambda_n)\end{bmatrix}.\end{aligned}$$

Written in matrix form as $\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}}$, where $\widehat{x}(\lambda_i)$ $(1 \le i \le n)$ represents the Fourier coefficient before the basis function (i.e., the amplitude of the basis function), $x(i)$ $(1 \le i \le n)$ on the left side of the matrix form denotes the signal of the i-th vertex, matrix column vector $\overline{u}_i = \begin{bmatrix}u_i(1)\\ u_i(2)\\ \vdots\\ u_i(n)\end{bmatrix}$ for $1 \le i \le n$.

In expression $\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}}$, $\mathbf{x}$ is the original signal, $\mathbf{U}$ is the matrix formed by the eigenvectors of the graph Laplacian, $\widehat{x}$ the signal in the spectral domain in which each element corresponds to the amplitude of the basis function.

The Fourier transform of the ontology graph can be expressed as

$$\widehat{x}(\lambda_l) =< \mathbf{x}, \overline{u}_l >= \sum_{i=1}^{n} x(i)u_l(i),$$

which implies

$$\begin{bmatrix}\widehat{x}(\lambda_1)\\ \widehat{x}(\lambda_2)\\ \vdots\\ \widehat{x}(\lambda_n)\end{bmatrix} = \begin{bmatrix}u_1(1) & u_2(1) & \cdots u_n(1)\\ u_1(2) & u_2(2) & \cdots u_n(2)\\ \vdots & \vdots & \ddots \quad \vdots\\ u_1(n) & u_2(n) & \cdots u_n(n)\end{bmatrix}\begin{bmatrix}x(1)\\ x(2)\\ \vdots\\ x(n)\end{bmatrix}$$

and equals to

$$\widehat{\mathbf{x}} = \mathbf{U}^{\mathrm{T}}\mathbf{x}.$$

The classic Fourier transform and the Fourier transform on the ontology graph can be compared as follows. In classic Fourier transform, we have

$$f(t) = \frac{1}{n}\sum_{w=1}^{n} F(w)e^{i\frac{2\pi}{n}wt}$$

$$F(w) = \sum_{t=1}^{n} f(t)e^{-i\frac{2\pi}{n}wt}.$$

When it comes to Fourier transform on the ontology graph, we deduce

$$x(i) = \sum_{l=1}^{n} \widehat{x}(\lambda_l) u_l(i)$$

$$\widehat{x}(\lambda_l) = \sum_{i=1}^{n} x(i) u_l(i).$$

The two are very similar. The classical Fourier transform uses $e^{i\frac{2\pi}{n}wt}$ as the base, while the ontology graph Fourier transform applies the Laplacian matrix eigenvector as its base. In the classical Fourier transform, the basis used by the original transform and the inverse transform is a conjugate relationship, namely $e^{i\frac{2\pi}{n}wt}$ and $e^{-i\frac{2\pi}{n}wt}$; while in the ontology graph Laplace transform, since the eigenvector of the real Laplacian matrix must be a real number. Therefore, after conjugation, it remains itself.

Here, we also want to specifically explain that the essence of the inverse Fourier transform is to express the original function as a linear combination of basis functions, but why does the graph Fourier transform use the Laplacian matrix eigenvectors as the basis? In theory, is there an array basis for n-dimensional space? Why not use other basis functions on ontology graph?

We simply answer these questions. The classic Fourier transform has the following law: the basis function of the Fourier transform is the eigenfunction of the Laplacian operator (eigenfunctions, also known as the characteristic function, for operators). Since the Laplacian matrix is the Laplacian on the graph, similarly, the basis function of the graph Fourier transform is the eigenvector of the graph Laplacian matrix. The eigenvalues of the Laplacian matrix assume a position similar to the frequency. The eigenvalues of Laplace are all non-negative. And the minimum characteristic value is 0. It is similar to the constant value in the classical Fourier transform (which can be regarded as frequency 0). The eigenvector of the Laplacian matrix takes the position of the basis function. The 0 eigenvalue corresponds to a constant eigenvector, which is similar to the constant term in the classical Fourier transform. The eigenvectors corresponding to low eigenvalues are relatively smooth, and the eigenvectors corresponding to high eigenvalues are transformed more intensely. The two correspond to the low-frequency basis function and the high-frequency basis function (in the classical Fourier transform, the low-frequency sine wave signal transformation is relatively flat, while the high-frequency sine wave signal transformation is more intense).

The eigenvector basis can be described by the zero crossing function:

$$Z_G(\mathbf{x}) = |\{e = (i,j) \in E(G) : x(i)x(j) < 0\}|, \quad (1)$$

where $\mathbf{x}$ in the left hand denotes the graph structure data of G. It records the number of the edges which have reversed signal values at the vertices at both ends. The more such edges, the stronger the signal oscillation and the more intense the transformation. The figure on the right also shows that as the feature value increases, the value of the zero crossing function increases. Hence, small eigenvalues correspond to low-frequency signals, and large eigenvalues correspond to high-frequency signals.

Define the smoothness of the signal by defining the graph Laplacian quadratic form. It represents the sum of the squared difference of the two node signals connected by the edge multiplied by the weight of the edge. The smaller the value, the smoother the signal:

$$\mathbf{x}^T\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{n} W_{ij}(x(i) - x(j))^2. \quad (2)$$

In (2), if $\mathbf{x}$ takes eigenvector, the smooth value of the quadratic form obtained is the eigenvalue corresponding to the eigenvector:

$$\overline{u}_l^T\mathbf{L}\overline{u}_l = \lambda_l, \quad (3)$$

which is exactly in line with the frequency setting in the classic Fourier transform: the higher the frequency, the steeper the basis function (cosine function) changes.

Let's discuss the specific implementation of the algorithm below. Regarding the two signals as the input signal and the convolution kernel respectively, the convolution operation can be defined in this way:
1) It converts the spatial domain signal to the frequency domain and then multiplies;
2) It converts the result of the multiplication to spatial domain.
It can be expressed by

$$\begin{aligned} \mathbf{x} *_G \mathbf{g} &= \mathcal{F}^{-1}(\mathcal{F}(\mathbf{x}) \odot \mathcal{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}), \end{aligned}$$

where $\odot$ denotes harmand product. Set

$$\widehat{\mathbf{x}} = \mathbf{U}^T\mathbf{x} = \begin{bmatrix} \widehat{x}(\lambda_1) \\ \widehat{x}(\lambda_2) \\ \vdots \\ \widehat{x}(\lambda_n) \end{bmatrix}$$

and

$$\widehat{\mathbf{g}} = \mathbf{U}^T\mathbf{g}) = \begin{bmatrix} \widehat{g}(\lambda_1) \\ \widehat{g}(\lambda_2) \\ \vdots \\ \widehat{g}(\lambda_n) \end{bmatrix}.$$

If this formula is expressed in the form of matrix multiplication, removing the harmand product. At the same time, we usually neglect what the filter signal g looks like in the spatial domain, but only care about its condition in the frequency domain. Let $\mathbf{g}_\theta = diag(\mathbf{U}^T\mathbf{g})$, the equation equivalently transforms into the following formula:

$$\begin{aligned} \mathbf{x} *_G \mathbf{g}_\theta &= \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x} \\ &= \mathbf{U}\begin{bmatrix} \widehat{g}(\lambda_1) & & \\ & \ddots & \\ & & \widehat{g}(\lambda_n) \end{bmatrix}\begin{bmatrix} \widehat{x}(\lambda_1) \\ \widehat{x}(\lambda_2) \\ \vdots \\ \widehat{x}(\lambda_n) \end{bmatrix} \\ &= \mathbf{U}\begin{bmatrix} \widehat{g}(\lambda_1) & & \\ & \ddots & \\ & & \widehat{g}(\lambda_n) \end{bmatrix}\mathbf{U}^T\mathbf{x}. \end{aligned}$$

That is: the product of the diagonal matrix and the column vector $\widehat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ are equal to the harmonic product of the two column vectors $\widehat{\mathbf{x}} = \mathbf{U}^T\mathbf{x}$ and $\widehat{\mathbf{g}} = \mathbf{U}^T\mathbf{g}$. The meaning of the above formula: the form of the filter signal

(convolution kernel signal) $\widehat{\mathbf{g}} = \mathbf{U}^T\mathbf{g}$ in the spectral domain is also an n-dimensional vector. Its function is to enlarge or reduce the components of the original signal, and the value on the component.

Another express:

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T,$$

$$\begin{aligned}\mathbf{y} &= \mathbf{g}_\theta(\mathbf{L})\mathbf{x} = \mathbf{U}\mathbf{g}_\theta(\mathbf{\Lambda})\mathbf{U}^T\mathbf{x} \\ &= \mathbf{g}_\theta * \mathbf{x} = \mathbf{U}\widehat{y}.\end{aligned}$$

Disadvantages: (1) Computational complexity, parameter complexity $O(n^3)$ (mainly reflected in the filter $\mathbf{g}_\theta$, it is a diagonal matrix, the number of learnable parameters is the same as the number of vertices) (2) Spatial convolution locality cannot be guaranteed (Convolution should convolve the local first-order domain of vertices, but spectral domain graph convolution cannot be done, because the feature transformation matrix U performs matrix multiplication ($\widehat{x} = \mathbf{U}^T\mathbf{x}$) on all vertices, so when convolution is performed on a vertex, the vertices far away are also taken into account, which is an undesirable result at this time. Its local computation cannot be guaranteed).

C. Ontology algorithm description

The signal of the input layer can be expressed as a $n \times C$ matrix, where n is the number of vertices in the graph, and C is the number of channels. The signal on the ontology graph can be decomposed into the signal on each vertex, $X \in \mathbb{R}^{n\times C}$ represents the signal of the entire ontology graph, where the vector $X_i \in \mathbb{R}^{1\times C}$ represents the signal corresponding to the i-th vertex. In the output layer, you can see that the graph structure is unchanged, the number of channels or the signal value of each vertex is changed. For example, the first vertex in the input layer corresponds to $1 \times C$, and the output layer corresponds to the vector of $1\times F$, and both the value and the number of channels change.

A learnable diagonal matrix can be used to replace the convolution kernel of the spectral domain to realize the graph convolution operation:

$$g_\theta = diag(\mathbf{U}^T g) \longrightarrow g_\theta = diag(\theta),$$

$$g_\theta = \begin{bmatrix}\widehat{g}(\lambda_1) & & \\ & \ddots & \\ & & \widehat{g}(\lambda_n)\end{bmatrix} \longrightarrow \begin{bmatrix}\theta_1 & & \\ & \ddots & \\ & & \theta_n)\end{bmatrix}.$$

That is, the original diagonal matrix g_θ only has n values on the diagonal, and the other n values are learned to form a new diagonal matrix to replace the original g_θ. The specific formula is defined as follows:

$$x_{k+1,j} = h(\mathbf{U}\sum_{i=1}^{k-1} F_{k,i,j}\mathbf{U}^T x_{k,i}) \tag{4}$$

where $j = 1, \cdots, C_k$, $x_k \in \mathbb{R}^{n\times C_k}$, $x_{k+1} \in \mathbb{R}^{n\times C_{k+1}}$,

$$F_{k,i,j} = \begin{bmatrix}\theta_1 & & \\ & \ddots & \\ & & \theta_n\end{bmatrix},$$

where C_k represents the number of channels in the k-th layer, $x_{k,i} \in \mathbb{R}^n$ represents the feature map of the i-th channel in the k-th layer, and $F_{k,i,j} \in \mathbb{R}^{n\times n}$ represents the convolution kernel matrix of the parameterized spectral domain (it is a diagonal matrix containing n learnable parameters), $h(\cdot)$ is the activation function.

The above formula can be understood in this way, assuming that the input and output channels are 1, the simplified version of the ontology graph convolution formula is as follows

$$x_{k+1} = h(\mathbf{U}F_k\mathbf{U}^T x_k),$$

where $x_k \in \mathbb{R}^n$, $x_{k+1} \in \mathbb{R}^n$, $\mathbf{U}^T x_k$ is the Fourier transform of the ontology graph, which transforms the spatial domain into the spectral domain; F_k replacing the convolution kernel of the original spectral domain, each θ_i of the matrix F_k is to enlarge or reduce each frequency component, that is, to adjust the entire amplitude; Multiply $\mathbf{U}$ the adjusted signal $F_k\mathbf{U}^T x_k$ to convert the spectral domain to the spatial domain. The core idea is to turn the original convolution kernel into a learnable convolution kernel F_k in the spectral domain.

Clearly, this trick has its disadvantages: (1) It is very time-consuming to calculate the eigenvalue decomposition of the Laplacian matrix. The computational complexity is $O(n^3)$, where n is the number of vertices. When dealing with large-scale graph data (such as social network data, which usually has millions of vertices), it will face great challenges. (2) The parameter complexity of the model is relatively large. The computational complexity is $O(n)$, and it is easy to overfit when there are many nodes. (3) Local links cannot be guaranteed (traditional convolutional networks of 3×3 or 5×5 can guarantee local links, but the trick is a global link form, not a local link form, which is contrary to the nature of classic convolution).

In order to solve the shortcomings of the above trick, Chebyshev polynomial can be applied instead of the convolution kernel in the spectral domain (Chebyshev polynomials play an important role in approximation theory and can be used for polynomial interpolation. The corresponding interpolation polynomial can minimize the Runge phenomenon and provide the best uniform approximation of the polynomial on the continuous function):

$$g_\theta = diag(\mathbf{U}^T g) \overset{\widehat{\Lambda}=\frac{2}{\lambda_{\max}}-\mathbf{I}_n}{\longrightarrow} g_\theta(\Lambda) = \sum_{k=0}^{K}\beta_k T_k(\widehat{\Lambda}),$$

$$g_\theta = \begin{bmatrix}\widehat{g}(\lambda_1) & & \\ & \ddots & \\ & & \widehat{g}(\lambda_n)\end{bmatrix} \longrightarrow$$

$$\begin{bmatrix}\sum_{k=1}^{K}\beta_k T_k(\widehat{\lambda}_1) & & \\ & \ddots & \\ & & \sum_{k=1}^{K}\beta_k T_k(\widehat{\lambda}_n)\end{bmatrix}.$$

Here, the traditional Chebyshev polynomial can be formulated by

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x),$$

where $n = 1, 2, \cdots$, $T_0(x) = 1$ and $T_1(x) = x$. Hence, it can be computed that

$$T_2(x) = 2x^2 - 1,$$

$$T_3(x) = 4x^3 - 3x,$$

$$T_4(x) = 8x^4 - 8x^2 + 1,$$

$$T_5(x) = 16x^5 - 20x^3 + 5x,$$

$$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1,$$

$$\cdots$$

Obviously, the coefficient of the highest term of $T_n(x)$ is 2^{n+1}, and its matrix expression is denoted by

$$T_{n+1}(L) = 2LT_n(L) - T_{n-1}(L),$$

where $n = 1, 2, \cdots$, $T_0(L) = \mathbf{I}$ and $T_1(L) = L$. That is to say,

$$p(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_K x^K$$

$$\longrightarrow$$

$$p(x) = \beta_0 T_0(x) + \beta_1 T_1(x) + \beta_2 T_2(x) + \cdots + \beta_K T_K(x).$$

We need to learn the coefficients $\beta_0, \beta_1, \cdots$, when using Chebyshev polynomials to approximate the function, replace the basis $1, x, x^2, \cdots$ with $T_0(x), T_1(x), T_2(x), \cdots$. This trick considers that the value of the spectral domain convolution kernel is a function related to the eigenvalue, and then the Chebyshev polynomial can be used to approximate this function.

Specifically, g_θ is the convolution kernel of the spectral domain. A Chebyshev polynomial $\sum_{k=0}^{K} \beta_k T_k(\widehat{\Lambda})$ interpolation is used to approximate it. We only need to learn to get each β_k. The advantage of this is that U can be put into the polynomial summation part in the derivation process, and the above derivation can be carried out by using the spectral decomposition of the Laplacian matrix $\widehat{L} = \mathbf{U}\widehat{\Lambda}\mathbf{U}^T$ to get $\sum_{k=0}^{K} \beta_k T_k(\widehat{L})x$. This shows that we no longer need to solve $\mathbf{U}$, but can directly use the Laplace matrix. Therefore, the complexity of spectral decomposition is avoided by $O(n^3)$, and the computational complexity is reduced.

The expression can be concluded by

$$x \star_G g_\theta = \mathbf{U} g_\theta \mathbf{U}^T x = \sum_{k=1}^{K} \beta_k T_k(\widehat{L})x,$$

where β_k is the coefficient in front of the Chebyshev polynomial and is a learnable parameter. We summarize its characteristics are:

1) The convolution kernel has only $K + 1$ learnable parameters. Generally, K is much smaller than n, and the complexity of the parameters is greatly reduced. (In the specific experiment, K takes a number less than 10; and n in the social network data set is millions, on the order of tens of millions);

2) After using the Chebyshev polynomial to replace the convolution kernel in the spectral domain, after public deduction, ChebNet does not need to perform eigendecomposition on the Laplacian matrix. The most time-consuming steps are omitted;

3) The convolution kernel has strict spatial locality. At the same time, K is the "receptive field radius" of the convolution kernel. The K-order neighbor vertices of the central vertex are regarded as neighbor vertices.

Usually the pooling layer is connected to the back end of the convolutional layer, and the fully connected layer is connected to the last end of the entire neural network. The role of the pooling layer is to reduce model parameters, reduce training burden, and control the over-fitting process. The pooling operation runs independently on the feature map of each depth, and the resolution is reduced in space. There are multiple reduction methods, such as max, average, L_2-norm, etc. The choice of pooling method is also a hyperparameter of the convolutional neural network. There are two more hyperparameters about the pooling layer: spatial extent and stride. Spatial coverage indicates how large a neighborhood is to pool local image data, and the step size determines the moving distance of the previous neighborhood and the next neighborhood.

The weight connection method of the fully connected layer is the same as that of the traditional neural network. Each neuron in this layer is connected to all the neurons in the previous layer. The reason why the fully connected layer is the last segment of the entire convolutional neural network is to map the feature data extracted by the front end to a fixed dimension. At the same time, it can also prevent over-smoothing.

III. Experiment on PE ontology

We build the PE ontology called "Sports Item Ontology", which can be represented as a tree-like structure, and the top vertex is the virtual vertex "sports item". The following is divided into several branches as follows:

• dangerous sports: Parachuting, powered parachutes, towed parachutes, gliding (paragliders, gliders, hang gliding), hot air balloons, light aircraft; swimming (fin swimming), sailing (windsurfing), rafting, surfing, diving, airships, sports yachts, motor boats, ect.

• badminton, baseball, basketball, football, handball, hockey, softball, table tennis, tennis (including soft tennis), volleyball (including beach volleyball), bocce, sepak takraw, croquet, shuttlecock, squash, billiards, golf Ball, bowling, potball, rugby; gymnastics (including rhythmic gymnastics), trampoline, technique, fitness, aerobics, sports dance, Mulan boxing (including Mulan sword, Mulan fan, etc.), Shapin exercise, sports health massage; figure skating, Ice dance, ice hockey, speed skating, skiing; track and field, cross-country, skateboarding, roller skating, bicycles (including mountain bikes, unicycles, etc.), water skiing, boating, kayaking, life-saving (water-saving); horse racing, Weightlifting, fencing, boxing, judo, wrestling, martial arts (including Sanda), taekwondo, karate, health qigong; chess, Chinese chess, go, bridge; fishing, pigeon, dragon dance, lion dance, dragon boat, kite, tug of war, Swing, cockfighting, bullfighting; nautical model, aviation model, vehicle model, aerospace model, radio, orienteering, ect.

• petanque, sepak takraw, croquet, shuttlecock, squash, billiards, golf Ball, bowling, potball, rugby; gymnastics (including rhythmic gymnastics), trampoline, technique, aerobics, aerobics, sports dance, Mulan boxing (including Mulan sword, Mulan fan, etc.), Shapin archery, boxing, fencing, skin sliding Boat, trampoline, judo, rowing, swimming (fin swimming), life-saving (life-saving on water), shooting, wrestling, sailing (windsurfing), parachuting, powered parachuting, towing parachute, gliding (paraglider, glider, hang gliding), Hot air balloon, mountaineering, rock climbing, bungee jumping, adventure, car, motorcycle, motorboat,

crossbow, water skiing, martial arts (including Sanda), taekwondo, bodybuilding, aerobics, sports dance, rafting, surfing, diving, airship, Light aircraft, sports yachts, karate, Shapin, horse racing, adventure, golf, ect.

• competitive events: acrobatic gymnastics, athletics/track & field, beach, boat race, bobsleigh, bobsled, boxing, canoe slalom, chess, cricket, cycling, diving, downhill race, dragon-boat racing, dressage, equestrian, fencing, figure skating, football/soccer, freestyle, gliding, sailplaning, golf, Greece-Roman wrestling, gymnastic apparatus, gymnastics, hand-ball, hockey, hold, lock, horizontal bar, hurdles/hurdle race, shuttlecock kicking, ice skating, item Archery, judo, jumping, kayak, mat exercises, modern pentathlon, mountain bike, parallel bars, polo, relative work, relay race/relay, rings, roller skating, rowing, rugby, sailing, shooting, side horse, pommelled horse, ski jump, ski jumping competition, skiing, slalom, softball, surfing, swimming, table tennis, taekwondo, tennis, toxophily, trampoline, trapeze, triathlon, tug-of-war, volleyball, badminton, baseball, basketball, walking/walking race, wall bars, water polo, weightlifting, winter sports, wrestling, yacht, vault,
Men's 10m Platform
Women's Taekwondo Over 67kg
Women's Athletics 20km Walk
Men's Diving Synchronized 3m Springboard
Women's Diving 3m Springboard
Women's Diving Synchronized 10m Platform
Men's Wrestling Greco-Roman 58kg
Men's Diving 3m Springboard
Men's Artistic Gymnastics Parallel Bars
Women's Artistic Gymnastics Beam
Men's Table Tennis Singles
Women's Diving 10m Platform
Women's Artistic Gymnastics Uneven Bars
Women's Table Tennis Singles
Men's Badminton Singles
Women's Badminton Doubles
Men's Diving Synchronized 10m Platform
Women's Diving Synchronized 3m Springboard
Men's Table Tennis Doubles
Women's Badminton Singles
Men's Fencing Team Foil
Women's Judo Heavyweight +78kg
Men's Shooting 10m Running Target
Women's Shooting 25m Pistol
Women's Table Tennis Doubles
Men's Weightlifting 77kg
Women's Weightlifting 75+ kg
Mixed Badminton Doubles
Women's Artistic Gymnastics All-Around Finals
Women's Judo Half-Heavywt 78kg
Men's Artistic Gymnastics All-Around Finals
Women's Fencing Team Epee
Women's Artistic Gymnastics Team Finals
Women's Judo Half-Middlewt 63kg
Women's Weightlifting 63kg
Women's Weightlifting 69kg
Men's Artistic Gymnastics Team Finals
Men's Shooting 10m Air Rifle
Women's Shooting Trap
Women's Weightlifting 53kg
Women's Judo Half-Lightwt 52kg
Women's Shooting 10m Air Pistol
Women's Cycling Track 500m Time Trial
Men's Shooting 10m Air Pistol
Women's Shooting 10m Air Rifle
Men's Weightlifting 56kg
etc.

One thing to note is that the structure of ontology graph is not a tree, only tree-like graphs and their branches have the common vertices. We use the ontology learning algorithm to get the dimensional signal (a real number) to each vertex, and compute the similarity of ontology vertices by calculating the distance between their corresponding real numbers. Then, we compare the results with the standard answer given by experts, and the compared conclusions are measured by $P@N$ average precision ratio. In order to reflect the efficiency of the algorithm, we use the same $P@N$ standard, apply the ontology learning algorithm in Gao et al. [24], [25] [26] to the "Sports Item Ontology", and compare the corresponding accuracy. Some results are shown in Table I.

It can be seen that the efficiency of the algorithm in this paper is significantly higher than that of Gao et al. [24], [25] [26]. Especially when N increases, this advantage is more obvious.

IV. Conclusion

The purpose of this article is to create a PE ontology, and use graph convolutional neural network to calculate the similarity between the concepts of the ontology, so as to find out the sports items that are highly related to each other. Through the analysis of experimental data, we can see that the algorithm in this paper is feasible and effective for PE data.

The ontology is a real-time transforming database. The concepts in the ontology will be continuously added and deleted, so the corresponding ontology graph structure will also change accordingly. This requires the graph convolutional neural network algorithm to adapt the changes in the ontology graph structure. Such problems can be used as topics for future research.

References

[1] I. Huitzil, F. Alegre, F. Bobillo, "GimmeHop: A recommender system for mobile devices using ontology reasoners and fuzzy logic," *Fuzzy Sets and Systems*, vol. 401, pp. 55–77, 2020.

[2] I. Huitzil, F. Bobillo, J. Gomez-Romero, "Fudge: Fuzzy ontology building with consensuated fuzzy datatypes," *Fuzzy Sets and Systems*, vol. 401, pp. 91–112, 2020.

[3] J. A. Benitez-Andrades, I. Garcia-Rodriguez, C. Benavides, et al. "An ontology-based multi-domain model in social network analysis: Experimental validation and case study," *Information Sciences*, vol. 540, pp. 390–413, 2020.

[4] J. A. Garcia-Diaz, M. Canovas-Garcia, R. Valencia-Garcia, "Ontology-driven aspect-based sentiment analysis classification: An infodemiological case study regarding infectious diseases in Latin America," *Future Generation Computer Systems-The International Journal of Escience*, vol. 112, pp. 641–657, 2020.

[5] V. Nembaware, G. K. Mazandu, J. Hotchkiss, et al. "The sickle cell disease ontology: Enabling collaborative research and co-designing of new planetary health applications," *Omics-A Journal of Integrative Biology*, vol. 24, no. 10, pp. 559–567, 2020.

[6] A. J. C. Trappey, C. V. Trappey, A. C. Chang, "Intelligent extraction of a knowledge ontology from global patents: The case of smart retailing technology mining," *International Journal on Semantic Web and Information Systems*, vol. 16, no. 4, pp. 61–80, 2020.

TABLE I
THE EXPERIMENT DATA FOR PE ONTOLOGY

	P@3 average precision ratio	P@5 average precision ratio	P@10 average precision ratio	P@20 average precision ratio
Algorithm in our paper	0.07	0.12	0.29	0.73
Algorithm in Gao et al. [24]	0.05	0.07	0.16	0.39
Algorithm in Gao et al. [25]	0.06	0.08	0.18	0.41
Algorithm in Gao et al. [26]	0.04	0.05	0.13	0.33

[7] E. Ong, L. L. Wang, J. Schaub, "Modelling kidney disease using ontology: insights from the Kidney Precision Medicine Project," *Nature Reviews Nephrology*, vol. 16, no. 11, pp. 686–696, 2020.

[8] J. Liu, Z. Qu, M. Yang, et al. "Jointly integrating VCF-based variants and OWL-based biomedical ontologies in MongoDB," *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, vol. 17, no. 5, pp. 1504–1515, 2020.

[9] A. Hafeez, S. Abbas, Aqeel-ur-Rehman, "Ontology-based transformation and verification of UML class model," *International Arab Journal of Information Technology*, vol. 17, no. 5, pp. 758–768, 2020.

[10] H. Azevedo, J. P. R. Belo, R. A. F. Romero, "Using ontology as a strategy for modeling the interface between the cognitive and robotic systems," *Journal of Intelligent & Robotic Systems*, vol. 99, no. 3-4, pp. 431–449, 2020.

[11] C. H. Qiu, "Ontology mapping constructing by means of low rank distance matrix optimization," *Engineering Letters*, vol. 28, no. 3, pp. 724–732, 2020.

[12] W. Gao and Y. J. Chen, "Approximation analysis of ontology learning algorithm in linear combination setting," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 9, no. 1, 2020, 29, https://doi.org/10.1186/s13677-020-00173-y.

[13] W. Gao, Y. Q. Zhang, J. L. G. Guirao, Y. J. Chen, "A discrete dynamics approach to sparse calculation and applied in ontology science," *Journal of Difference Equations and Applications*, vol. 25, no. 9-10, pp. 1239–1254, 2019.

[14] J. Z. Wu, X. Yu, W. Gao, "Disequilibrium multi dividing ontology learning algorithm," *Communications in Statistics-Theory and Methods*, vol. 46, no. 18, pp. 8925–8942, 2017.

[15] W. Gao, A. Q. Baig, H. Ali, et al., "Margin based ontology sparse vector learning algorithm and applied in biology science," *Saudi Journal of Biological Sciences*, vol. 24, no. 1, pp. 132–138, 2017.

[16] M. X. Chen, B. X. Liu, D. S. Zeng, et al., "A framework for ontology-driven similarity measuring using vector learning tricks," *Engineering Letters*, vol. 27, no. 3, pp. 549–558, 2019.

[17] M. H. Lan, J. Xu, W. Gao, "Ontology similarity computation and ontology mapping using distance matrix learning approach," *IAENG International Journal of Computer Science*, vol. 45, no. 1, pp. 164–176, 2018.

[18] X. G. He, Y. Y. Wang, W. Gao, "Ontology similarity measure algorithm based on KPCA and application in biology science," *Journal of Chemical and Pharmaceutical Research*, vol. 5, no. 12, pp. 196–200, 2013.

[19] J. Z. Wu, X. Yu, W. Gao, "Similarity matrix learning for ontology application," International Journal of Information Technology and Management, vol. 15, no. 1, pp. 1–13, 2016.

[20] L. L. Zhu, Y. Pan, M. K. Jamil, et al., "Boosting based ontology sparse vector computation approach," *Engineering Letters*, vol. 25, no. 4, pp. 406-415, 2017.

[21] M. Lan, J. Xu, W. Gao, "Ontology feature extraction via vector learning algorithm and applied to similarity measuring and ontology mapping," *IAENG International Journal of Computer Science*, vol. 43, no. 1, pp. 10–19, 2016.

[22] L. Zhu, W. Tao, X. Min, et al. "Theoretical characteristics of ontology learning algorithm in multi-dividing setting," *IAENG International Journal of Computer Science*, vol. 43, no. 2, pp. 184–191, 2016.

[23] L. Zhu, Y. Pan, M. R. Farahani, et al. "Theoretical characteristics on scoring function in multi-dividing setting," *IAENG International Journal of Applied Mathematics*, vol. 47, no. 1, pp. 28–36, 2017.

[24] W. Gao, Y. Guo, K. Y. Wang, "Ontology algorithm using singular value decomposition and applied in multidisciplinary," *Cluster Computing-The Journal of Networks, Software Tools and Applications*, vol. 19, no. 4, pp. 2201–2210, 2016.

[25] W. Gao, A. Q. Baig, H. Ali, et al., "Margin based ontology sparse vector learning algorithm and applied in biology science," *Saudi Journal of Biological Sciences*, vol. 24, no. 1, pp. 132–138, 2017.

[26] W. Gao, L. L. Zhu, K. Y. Wang, "Ranking based ontology scheming using eigenpair computation," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 4, pp. 2411–2419, 2016.

Guan Li is a Lecturer of Department of Computer Science, Guangdong University Science and Technology, City of Dongguan, Guangdong Province, China. His research fields cover PE, Big Data, Artificial Intelligent, ect.