Knowledge-Based Systems 100 (2016) 188–199

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

A deep learning approach for relationship extraction from interaction context in social manufacturing paradigm

Jiewu Leng, Pingyu Jiang*

State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China

ARTICLE INFO

Article history:
Received 12 September 2015
Revised 7 March 2016
Accepted 8 March 2016
Available online 14 March 2016

Keywords:
Manufacturing relationship extraction
Deep learning
Stacked denoising auto-encoder
Social manufacturing
Interaction context

ABSTRACT

There is an increasing unstructured text data produced in cross-enterprise social interaction media, forming a social interaction context that contains massive *manufacturing relationships*, which can be potentially used as decision support information for cross-enterprise manufacturing demand-capability matchmaking. How to enable decision-makers to capture these relationships remains a challenge. The text-based context contains high levels of noise and irrelevant information, causing both highly complexity and sparsity. Under this circumstance, instead of exploiting man-made features carefully optimized for the relationship extraction task, a *deep learning* model based on an improved *stacked denoising auto-encoder* on sentence-level features is proposed to extract manufacturing relationships among various *named entities* (*e.g.*, enterprises, products, demands, and capabilities) underlying the text-based context. Experiment results show that the proposed approach can achieve a comparable performance with the state-of-the-art learning models, as well as a good practicality of its' web-based implementation in social manufacturing interaction context. The ultimate goal of this study is to facilitate knowledge transferring and sharing in the context of enterprise social interaction, thereby supporting the integration of the resources and capabilities among different enterprises.

1. Introduction

Social manufacturing [1,2] has recently attracted significant attention, as a novel paradigm for supporting cross-enterprise manufacturing demand-capability matchmaking through extensive Information and Communication Technology (ICT) deployment. The successful deployment of ICT-enabled processes has motivated enterprises to reduce inter-organizational production processes inefficiencies, and also enable enterprises to match manufacturing demands, services, resources and capabilities in a wide range network for manufacturing insourcing or outsourcing [3]. Consequently, the growing use of ICT-enabled cross enterprise communication devices, media and platforms has resulted in explosion of the quantity of interaction text data forming *Manufacturing Service Interaction Contexts* (MSIC), which contain highly flexible, multi-dimensional, and cooperative *manufacturing relationships* among partners.

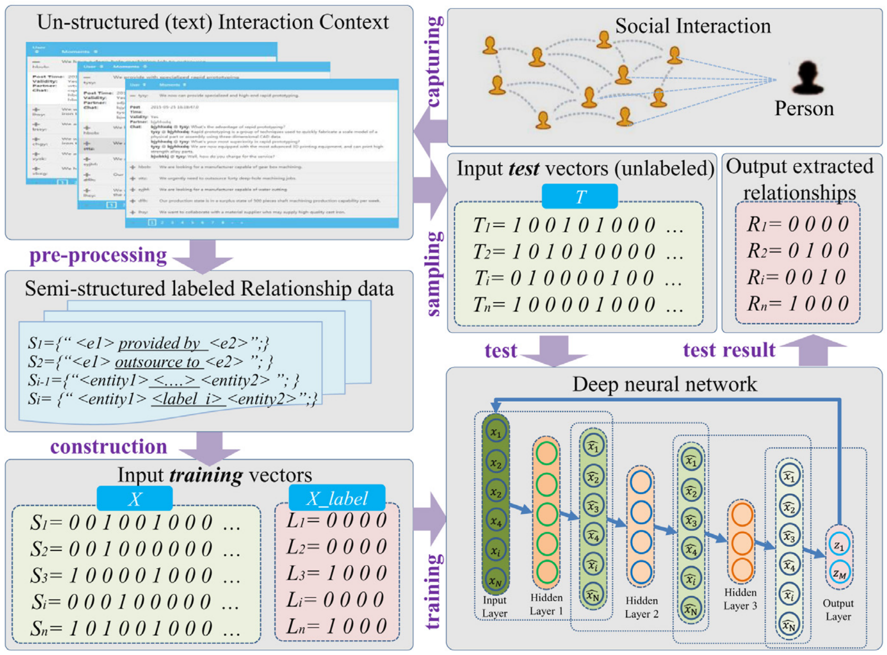
This situation consequently leaves a challenge for analysts and decision makers (DMs), which is the capture and maintenance of relationships and interactions that occur across enterprises. This information can be used by DMs to understand how manufacturing resources and capabilities are distributed in the network, and identify potential partners so as to outsource manufacturing operations to them. However, retrieving and processing this information is difficult due to the lack of formal structure in the natural language narrative MSIC.

The study bridges this gap by automatically mining and extracting manufacturing relationships from text-based MSIC, holding the promise of easily consolidating large amounts of underlying knowledge in computer-accessible form. The task of relationship extraction is to predict semantic relationships between pairs of *named entities* (*e.g.*, enterprises, product, demand, and capability) and can be defined as follows: given a sentence *S* with the annotated pairs of entities *e1* and *e2*, the goal is to identify the relationships between *e1* and *e2*.

To identify the relationships between pairs of entities, it is necessary to a skillfully use of sentence level clues from diverse syntactic and semantic structures in a sentence. For example, in the sentence "$\langle We\rangle_{e1}$ *are specialized in* $\langle rapid\,prototyping\rangle_{e2}$", to identify that an *enterprise* and a *manufacturing method* are in a *specialty* relationship, the marked entities and the meanings of the entire sentence should be leveraged. In this study, instead of exploiting man-made features carefully optimized for the relationship extraction task, a deep learning approach to extract sentence level features for relationships extraction is exploited. As shown in Fig. 1, the proposed approach takes all of the word tokens as input

* Corresponding author. Tel.: +86 2983395396.
E-mail address: pjiang@mail.xjtu.edu.cn (P. Jiang).

http://dx.doi.org/10.1016/j.knosys.2016.03.008
0950-7051/© 2016 Elsevier B.V. All rights reserved.



Un-structured (text) Interaction Context
capturing
Social Interaction
Person
pre-processing
Semi-structured labeled Relationship data
S1={" <e1> provided by <e2>";}
S2={"<e1> outsource to <e2> "; }
Si-1={"<entity1> <....> <entity2> "; }
Si= {" <entity1> <label_i> <entity2>";}
sampling
Input test vectors (unlabeled)
T
T1= 1 0 0 1 0 1 0 0 0 ...
T2= 1 0 1 0 1 0 0 0 0 ...
Ti= 0 1 0 0 0 0 1 0 0 ...
Tn= 1 0 0 0 0 1 0 0 0 ...
Output extracted relationships
R1= 0 0 0 0
R2= 0 1 0 0
Ri= 0 0 1 0
Rn= 1 0 0 0
construction
test
test result
Input training vectors
X
X_label
S1= 0 0 1 0 0 1 0 0 0 ...
S2= 0 0 1 0 0 0 0 0 0 ...
S3= 1 0 0 0 0 1 0 0 0 ...
Si= 0 0 0 1 0 0 0 0 0 ...
Sn= 1 0 1 0 0 1 0 0 0 ...
L1= 0 0 0 0
L2= 0 0 0 0
L3= 1 0 0 0
Li= 0 0 0 0
Ln= 1 0 0 0
training
Deep neural network
Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
Output Layer

Fig. 1. Logic flow of the deep learning approach based relationship extraction in the MSIC.

without complicated pre-processing, neither Part-of-Speech (POS) nor syntactic parsing. First, all the word tokens are transformed into training vectors by looking up word embeddings, which is a collection of all words in the context under study. Then, sentence level features are obtained using a *stacked denoising auto-encoder* based deep learning approach. These learned features are concatenated to form the final extracted feature vector. Finally, on the basis of the trained deep neural network, the extracted features can be viewed as the relationships between two marked entities. With proper further processing, these relationships results can finally enable the demand-capability matchmaking in the cross-enterprise outsourcing decision-making progress.

The rest of this paper is organized as follows. Section 2 reviews some of the related work in data mining in cross-enterprise manufacturing, and relationships extraction from text-based context. Section 3 discusses the deep learning methodology for manufacturing relationship extraction from the context. Section 4 gives the comparisons between the proposed approach and the state-of-the-art learning models, and also evaluates the practicality of a web-based software implementation based on the proposed approach in the MSIC. Section 5 concludes with the main contributions and future research directions.

2. Related work

This section is to survey the advancements in two research directions related to manufacturing relationship extraction, and identify the current hiatus that must be overcome.

2.1. Data mining in cross-enterprise manufacturing interaction context

Data mining in cross-enterprise manufacturing context is to extract useful information and knowledge for demand and capability management [4]. Most of existing research focused on the data mining from structured context database. For examples, Hui and Jha [5] integrated neural network, case-based and rule-based reasoning techniques to extract knowledge from the database to support service decision and fault diagnosis. Agard and Kusiak [6] used association rule-based clustering for customer functional requirements segmentation in the design of product families from customer response data. However, nowadays, in the paradigm of social and cloud manufacturing, large amount of context data from the social media were produced in un-structured plain text form rather than a predefined formal exchanging framework or model [7].

Text-based context data is the main knowledge source from cross-enterprise social interaction context [8,9]. Many text mining studies have been conducted in text classification [10,11], terminology extraction [12], entity recognition [13], relationship extraction [14], hypothesis generation [15], and opinion mining [16]. Two strategies have been reported in the literature for these text mining applications. The first strategy is a two-phase process. It proposes the pre-processing phase of noise elimination from text, and then proceeds with analysis on cleaned or formulated form such as ontology [12,14–16]. The second strategy processes the noisy text directly using machine learning techniques which can learn patterns from the underlying text [10,11,13]. The adopted strategies usually depend on the ultimate information extraction task. This study focuses on the manufacturing relationship extraction from the text-based context data, and adopts the second strategy to deal with the big-data and high sparsity nature of the MSIC.

2.2. Techniques for features and relationships extraction in text data

Significant progress of text mining applications has been made in the relationship extraction aspect, which is to detect instances of pre-specified types of relationships between a pair of entities of given types. Several approaches have been reported in the literature: (1) *statistical methods* identify relationships that predicted by chance or probability [17], (2) *template-based methods* define patterns manually by domain experts to extract relationships [18], (3) *NLP (i.e., Natural Language Processing) based methods* perform sentence parsing to decompose the text into layered-structure from which relationships can be extracted [19], and (4) *automatic or supervised methods* create similar templates by learning patterns of interest [20]. Supervised methods can be further categorized into feature-based methods and kernel-based methods. Feature-based methods utilize a set of features that are obtained by performing textual analysis. They transform these features into a vector or symbolic IDs using a paradigm that is similar to the bag-of-words model [21]. Feature-based methods suffer from the issue of selecting an efficient feature set when transforming the structured representation into feature vectors. Conversely, kernel-based methods enable the use of a large set of features without explicitly extracting the features. Various kernels, such as the dependency tree kernel [22], subsequence kernel [23], and convolution tree kernel [24], have been proposed for the relationship extraction. These supervised methods yield relatively high performances. However, their performances strongly depend on the quality of the elaborately designed features or kernels, which lead to the propagation of the errors in the existing extraction processes and thus hinder the performances of such methods. In addition, the methods mentioned above suffer from a lack of sufficient labeled training data, and are also vulnerable to the wrong labels [25,26].

To extracting features dependent from existing NLP tools as less as possible, recently, many researchers have introduced Deep Learning technique to directly extract features in large-scale coarse data. For instances, Zeng et al. [27] presented a *convolutional deep neural network* that learns lexical and sentence level feature vectors for relation classification. Ebrahimi and Dou [28] proposed a *recursive neural network* for relation classification based on the shortest path between two entities in a dependency graph. The studies of deep learning were originated from artificial neural network research, in which usually suffers from the probability of falling into local optima due to the increment of network layers before the year of 2006. The breakthrough to effective training strategies for deep learning architectures came with the unsupervised greedy layer-wise pre-training algorithm followed by supervised fine-tuning [29]. Various deep learning models such as *convolutional deep neural network* [30], *deep belief network* [31,32], and *stacked auto-encoder network* [33,34] were proposed. Some models have been introduced to solve the relationship extraction issue [27,28]. These models are more effective than conventional machine learning methods such as *Support Vector Machines* and other shallow neural networks, because they possess a higher generalization ability and can leverage a larger body of linguistic information [27,29,34,35]. In this study, to effectively alleviate the shortcomings of traditional pre-specified features, a superior deep learning approach that based on *stacked denoising auto-encoder* model is introduced to extract sentence level features for manufacturing relationships extraction.

3. Deep learning methodology for relationship extraction

3.1. Problem descriptions and assumptions

The *distributional hypothesis theory* [36] states that words used in the same context tend to have similar meanings. Accordingly, it is assumed that the pairs of entities (*i.e.*, words or phrases) occurred in similar contexts tend to have similar relationships. Then the *relationship extraction* task is transferred as assigning relationship labels to pairs of entities. It is thus necessary to specify which relationship labels should be assigned to the given pairs of entities.

The input of the model is the MSIC. As shown in Figs. 1 and 10, the MSIC may include any feature of social interaction context (*e.g.*, specialty, supply connections, or service interactions). The first assumption is that the context record moments and events such that (1) each moment refers to a manufacturing demand or capability, (2) each event refers to an activity (*i.e.*, a well-defined step in the process), (3) each event refers to a case (*i.e.*, a process instance), and (4) each event includes two or more entities (*e.g.*, the enterprise executing or initiating the activity). Any social enabled platform such as cloud ERP [1,2] will offer this information in some form. The last assumption is that it is possible to collect logs with event data.

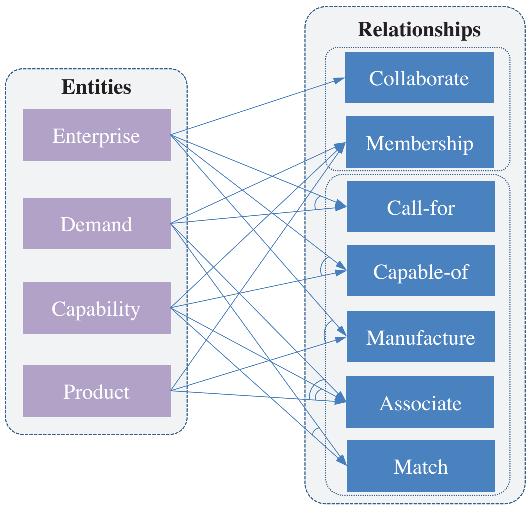


Fig. 2. Type of manufacturing entities and relationships in the MSIC.

Different categories of relationships usually imply different layout and syntactic of context sentence, and thus latterly the *experiments and result discussions* section will do the deep learning for each relationship category independently to achieve higher extraction performance. In the MSIC, the enterprise DMs usually focus on different decision variables (entities), among which have complex correlations. A *Principal Component Analysis* can be conducted to identify a smaller set of key variables. Based on the empirical investigation data obtained from DMs in a National High-New Technology Zone of China, in this study, four types of *manufacturing entities* in the MSIC are identified as shown in Fig. 2. This analysis was carried out by using the SPSS software package, and illustrated in the Appendices A–E. Appendix D shows that principal components 1, 2, 3, and 4 account for approximately 80% of the total variance for the ten original variables. The loadings of each variable on each principal component are reported in Appendix E: component 1 mainly loads on *Technology, Price, Service*, and *Quality*; component 2 mainly loads on *Productivity, Delivery, Equipment*; component 3 mainly loads on *Manufacturer* and *Supplier*; and component 4 mainly loads on *Demand*. Finally, these four principal components are identified and named as *Product, Capability, Enterprise*, and *Demand*, respectively.

These four identified components may not include all the information in the MSIC, but can obtain the key information that the DMs attend to capture. In the MSIC, integrating *Demand* and *Capability* helps *Enterprises* prioritize and ensure *Products* fulfillment based upon the shared generation, dissemination, interpretation and application of real-time customer/partner demand as well as ongoing capacity constraints [37,38]. Specifically, the *Demand* entity refers to various level (including process-, part-, and product-) manufacturing activities that an enterprise wants to outsource to other enterprise; while the *Capability* entity is a phrase that describes what manufacturing capability an enterprise can provide, including various manufacturing methods, equipment, and services.

Seven categories of *relationships* among the four types of *manufacturing entities* are identified as shown in Fig. 2, and they could be divided into two categories. The first category captures the relationship inside a single type of entities, including the *Collaborate* relationships inside *Enterprises*, and the *Membership* relationships inside *Demand, Capability*, or *Product* entities. The second category relationship describes the relationship between

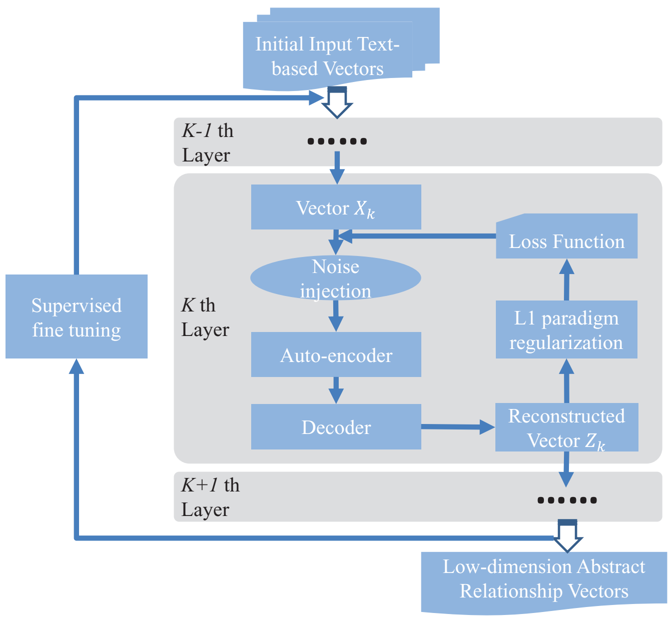

Initial Input Text-based Vectors
K-1 th Layer
Vector X_k
Loss Function
Noise injection
Supervised fine tuning
K th Layer
L1 paradigm regularization
Auto-encoder
Decoder
Reconstructed Vector Z_k
K+1 th Layer
Low-dimension Abstract Relationship Vectors

Fig. 3. The deep learning architecture for relation extraction.

two types of entities, including *Call-for, Capable-of, Manufacture, Associate*, and *Match*. Note that the way that classifies these entities and relationships does not influence the methodology of the deep learning based relationship extraction. Therefore, the practitioners can make classifications different from Fig. 2 based on their specific goals of relationship extraction from the MSIC in implementation processes.

3.2. The deep learning architecture for relationships extraction

An extraction strategy of utilizing all of the local features and predicting a relationship globally is adopted. Fig. 3 describes the architecture of the *stacked denoising auto-encoder* (SDAE) [33,34] based deep learn neural network model that used for merging all the local features for relationship extraction. Starting with the sentences with two marked entities, the model does not need any complicated syntactic or semantic preprocessing. Then, the word tokens are transformed into vectors by looking up the word embeddings that aggregated from the MSIC. The neural network has multiple hidden layers. Each layer K is densely connected to layer $K+1$. The deep learning model first induces parameters in an unsupervised way for each layer, and discovers multiple levels of feature extraction, where higher levels represent more abstract aspects of the inputs. After a pre-specified epochs of unsupervised pre-training layer by layer, the model then do a supervised back-propagate fine tuning. After a training phase, the deep neural network has learned and memorized hidden patterns/features between mappings of input sentences and labels globally.

This trained neural network can be used as a tool to extract the most representative features vector (key words) from input sentences collected in a same MSIC. The automatically extracted sentence level features could be viewed as relationships to build a manufacturing network that comprised of numbers of named entities, which can be the basis of information retrieval systems supporting the integration of the resources and capabilities across enterprises.

3.3. Input data representation

The goal of deep learning is to obtain a trained neural network that memorizes the mapping between the input sentence data (***X***) and corresponding pre-identified relationship labels (***X_labels***) from training examples. The representation of these two parts of input data is the first issue, and a good representation is to retain a significant amount of information about the input.

In the very beginning of representation of input words, each input word token is transformed into a vector by looking up word embeddings. The word embeddings are supposed to contain far more syntactic and semantic information than the randomly initialized embeddings [39], and should be obtained using significant amounts of context data. Although it is a tough task, there are many trained word embeddings that are freely available, such as Collobert and Weston [40] embeddings, and HLBL [41] embeddings. Here, for concise reason, the training algorithm provided by Turian et al. [42] is directly utilized to aggregate the initial word embeddings from the MSIC.

Note that all tokens are represented as word vectors, which have been demonstrated to correlate well with human judgments of word similarity. However, single word vector models are of limited effectiveness because they do not capture distance features and semantic compositionality, which hinders us to understand the meanings of a longer expression. In this study, the deep learning model automatically extracts sentence level features to enable the relationships extraction, as shown in Fig. 4. Each token in sentence level feature is further captured by Word Features (WF) and Position Features (PF).

The *distributional hypothesis theory* [36] states that words occurred in the same context tend to have similar meanings. Therefore, the WF is designed as the combination of a word's vector representation together with the vector representations of the word in its neighbor context to capture this characteristic. Take the following sentence as an example:

$$x :< We>_0 \, are_1 \, specialized_2 \, in_3 \, < drilling>_4. \tag{1}$$

All of the word tokens of the sentence $\boldsymbol{x}$ are represented as a list of vectors (x_0, x_1, x_2, x_3, x_4), where x_i corresponds to the word embedding (*i.e.*, an index in the word embeddings) of the *i*th word in the sentence. To use a context size of *w*, the size *w* windows of word vectors are aggregated into a richer feature. For example, given $w = 3$, the WF of word x_i in the sentence $\boldsymbol{x}$ is expressed as

$$WF_i = (x_{i-1}, x_i, x_{i+1}). \tag{2}$$

where x_{i-1} refers to left token of word, and x_{i+1} denotes right token of word. Similarly, considering the whole sentence, the WF can be represented as follows:

$$\boldsymbol{x} = \{(x_s, x_0, x_1), (x_0, x_1, x_2), \ldots, (x_3, x_4, x_e)\}. \tag{3}$$

Besides the basic word features, structure features also serve as important cues for deciding relationships. To further capture the structure information (*e.g.*, the shortest dependency path between entities), PF are proposed, which is the combination of the relative distances of the current word to entities e_1 and e_2. For instance, the relative distances of "*specialized*" in sentence $\boldsymbol{x}$ to entities "*we*" and "*drilling*" are 2 and −2, respectively. These two relative distances also are mapped to a vector of dimension corresponding to the WF. Then, the distance vectors d_1 and d_2 with respect to the relative distances of the current word to entities e_1 and e_2 are obtained, and $PF_i = [d_1, d_2]_i$. The word is represented as $[WF_i, PF_i]^T$ by combining the WF_i and PF_i,. Finally, the whole sentence can be denoted as follows

$$\boldsymbol{x} = \left\{[WF_0, PF_0]^T, [WF_1, PF_1]^T, \ldots, [WF_4, PF_4]^T\right\}, \tag{4}$$

and all observed sentences of this form are subsequently aggregated as $\boldsymbol{X}$, *i.e.*, the first part of training examples. Note that vectors are written in lowercase bold and matrices are in uppercase bold throughout this article.

With respect to the second part of input training examples (*i.e.*, labels) that pre-defines the relationships between two named

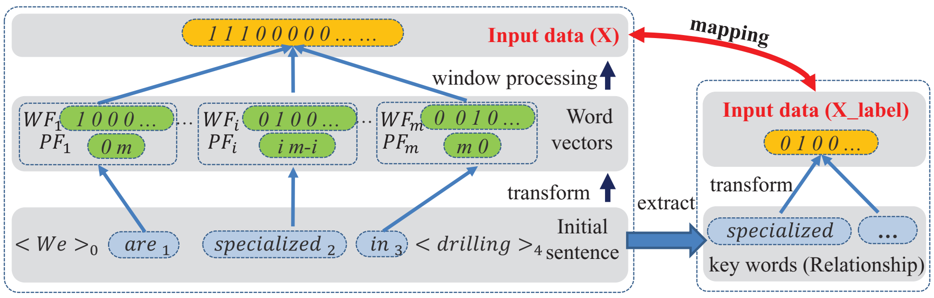


Fig. 4. Illustration of sentence level input data formulation.

entities, all of the words in the labels are represented as a series of WFs:

$$\boldsymbol{x\_label} = \left\{ [WF_0]^T, [WF_1]^T, \ldots, [WF_j]^T \right\}, \tag{5}$$

where j denotes to the words sequence number in the pre-defined labels. Similarly, all corresponding labels are subsequently aggregated as the second part of training examples, *i.e.*, ***X_labels***.

3.4. Modeling of stacked denoising auto-encoder

The word representation technique presented above can capture contextual information through combinations of vectors in a window. However, it only produces local features around each word of the sentence. It is necessary to utilize all of the local features and predict a manufacturing relationship globally. Therefore, a deep neural network approach is introduced as a natural method to merge all of the features. In the following, *stacked denoising auto-encoders* [33,34] are presented as a building block to train the deep neural network, and the framework and terminology of the model including activation units, regularization methods, denoising strategy, and parameters are detailed.

Neural networks give a way of defining a complex, non-linear form of hypotheses. Neuron is a computational unit. Let us denote $\boldsymbol{x} \in \boldsymbol{R}^N$ as the input of a neuron, and $\boldsymbol{z} \in \boldsymbol{R}^M$ as the code. Note that N and M denote the data dimension of input and output, respectively. Let the weights in encoder and decoder be the columns of matrix $\boldsymbol{W}_e \in \boldsymbol{R}^M$ and $\boldsymbol{W}_d \in \boldsymbol{R}^N$, which means tied weights are used. And let $\boldsymbol{b}_e \in \boldsymbol{R}^M$ and $\boldsymbol{b}_d \in \boldsymbol{R}^N$ be the biases in the encoder and decoder, respectively. Collectively, $\boldsymbol{\theta}_d = \{\boldsymbol{W}_d, \boldsymbol{b}_d\}$ denotes for the decoder, and $\boldsymbol{\theta}_e = \{\boldsymbol{W}_e, \boldsymbol{b}_e\}$ for the encoder, Then, the encoder and decoder, respectively, compute as $\boldsymbol{y} = f_{\boldsymbol{\theta}_e}(\boldsymbol{x}) = s(\boldsymbol{W}_e{}^*\boldsymbol{x} + \boldsymbol{b}_e)$ and $\boldsymbol{z} = g_{\boldsymbol{\theta}_d}(\boldsymbol{y}) = s(\boldsymbol{W}_d{}^*\boldsymbol{y} + \boldsymbol{b}_d)$, where $f$is called the activation function. Here, the sigmoid function is adopted:

$$f(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}} . \tag{6}$$

This encoding-decoding process yields an associated reconstruction error to be optimized, and training this auto-encoder to minimize reconstruction error amounts to maximizing a lower bound on the *mutual information* between input and learnt representation. A *cross-entropy* function is introduced to measure the loss. Given all training examples $(\boldsymbol{x}^{(i)}, \boldsymbol{x\_label}^{(i)})$, the optimization can be denoted as follows:

$$\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{z}) = -\sum_{\boldsymbol{i}} \left[\boldsymbol{x}^{(\mathbf{i})} \log \boldsymbol{z}^{(\mathbf{i})} + \left(1 - \boldsymbol{x}^{(\mathbf{i})}\right) \log \left(1 - \boldsymbol{z}^{(\mathbf{i})}\right) \right], \tag{7}$$

$$\boldsymbol{\theta}_e, \boldsymbol{\theta}_d = \arg\min_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d} \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{z}) . \tag{8}$$

As shown in Fig. 5, a deep neural network is put together by hooking together multiple neurons, so that the output of a neuron can be the input of another. *Denoising Auto-Encoders* (DAEs) [43] train one-layer neural networks to reconstruct input data from *partial random corruption* (*i.e.*, denoising, which will be detailed in next section). The DAEs are stacked into deep learning architectures (called as stacked denoising auto-encoders) where the weights are fine-tuned with back-propagation [34]. After training a single former-level DAE in the left in Fig. 5, its learned encoding function $f_{\boldsymbol{\theta}_e}^{(i)}$ is used on clean input. The resulting representation is used to train the next-level DAE to learn next encoding function $f_{\boldsymbol{\theta}_e}^{(i+1)}$. From there, the procedure can be repeated and stacked as in the middle of Fig. 5. This is a feed forward neural network, since the connectivity graph does not have any directed loops or cycles.

Back propagation training is a process of supervised fining tuning that can avoid the over-fitting problem [44,45]. It is to measure how much that node encoding and decoding was responsible for any errors in the mapping between *learned* and *pre-defined* manufacturing relationships, and then make some parameters adjustment in the next learning epoch. For a neural node, the difference between the network's activation and the true target value can be directly measured by Eq. (7) . To decrease the magnitude of the weights and help prevent over-fitting, a regularization term (also called weight decay) is introduced as

$$\boldsymbol{J}\left(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d\right) = \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{z}) + \delta \sum_{j=0}^{|\boldsymbol{\theta}_e, \boldsymbol{\theta}_d|} \left| \boldsymbol{\theta}_{e\,j}, \boldsymbol{\theta}_{d\,j} \right|, \tag{9}$$

where $\boldsymbol{\delta}$ refers to the weight decay parameter that controls the relative importance of the two terms. The goal is to minimize $\boldsymbol{J}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d)$ as a function of $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$. Each parameter $\boldsymbol{\theta}_e$ and $\boldsymbol{\theta}_d$ will be initialized to a small random value near zero, and then adjusted by an optimized technique to train the neural network. Since $\boldsymbol{J}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d)$ is a non-convex function, gradient descent is susceptible to local optima; however, in practice gradient descent usually works fairly well. The log likelihood $\boldsymbol{J}(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d)$ can be maximized using a simple optimization technique called *Stochastic Gradient Descent* (SGD). Because the parameters are in different layers of the neural network, a back propagation strategy of differentiation chain rule is implemented through the network until the input layer is reached by iteratively selecting an example $(\mathrm{x}^{(\mathrm{i})}, \mathrm{x\_label}^{(\mathrm{i})})$ and applying the following update rule.

$$\boldsymbol{\theta}_e \leftarrow \boldsymbol{\theta}_e - l * \frac{\partial\, J\left(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d\right)}{\partial \boldsymbol{\theta}_e}, \quad \boldsymbol{\theta}_d \leftarrow \boldsymbol{\theta}_d - l * \frac{\partial\, J\left(\boldsymbol{\theta}_e, \boldsymbol{\theta}_d\right)}{\partial \boldsymbol{\theta}_d}, \tag{10}$$

where l is the learning rate that preset by users.

3.5. Denoising strategy

A crucial problem on back propagation is its generalization capability. A network successfully trained for given samples is not guaranteed to provide desired associations for untrained inputs as well. On one hand, according to the research of Bengio et al. [32], when the output layer dimension in the automatic encoder is

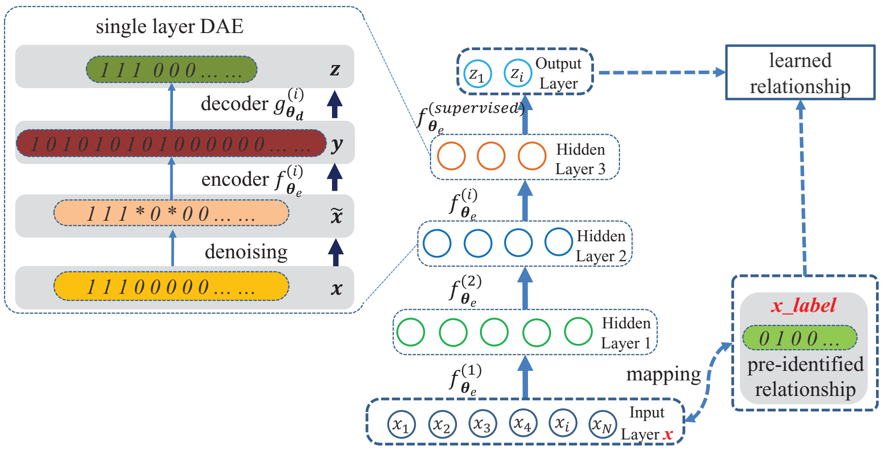


Fig. 5. Stacked denoising auto-encoders for relationship extraction.

greater than or equal to the input layer dimension, a good result can be achieved for feature extraction. But short text (sentence) based context input vector is very sparse, the auto-encoder is likely to do linear transformation of learning, and cannot achieve the purpose of feature abstracting or extracting. On the other hand, the input of text usually contains some personalized symbols and language, and thus the trained model must have better robustness.

To enhance the generalization capability in this study, the network is trained with additional noise injected inputs. Vincent et al. [33] only replace a percentage of the input data to 0 in the image classification issues. In contrast, corresponding to above two situations in the sentence-level text feature extraction, part of the input data are selected and replaced into 0, and also part of the data into 1, which is known as *salt-and-pepper noise*. The former is given that there may be some high dimension of input vector data is missing, and the latter is to avoid affected by personalization input. After the noise injection procedure, the input vector x is transferred to $\tilde{\boldsymbol{x}}$ with a certain *noise level*, and thus the SGD based optimizing algorithm was reformulated as follows

$$\boldsymbol{\theta}_e, \boldsymbol{\theta}_d = arg \min_{\boldsymbol{\theta}_e, \boldsymbol{\theta}_d} \boldsymbol{J}\left(\boldsymbol{x}, g_{\boldsymbol{\theta}_d}\left(f_{\boldsymbol{\theta}_e}(\tilde{\boldsymbol{x}})\right)\right) . \tag{11}$$

Note that SDAEs are still minimizing the same reconstruction loss between a clean $\boldsymbol{x}$ and the reconstruction from $\tilde{\boldsymbol{x}}$.

4. Experiments and result discussions

In this section, two set of experiments are conducted so as to test the efficiency and practicality of the proposed approach. The first set of sensitivity experiments is to test several variants via cross validation to gain understanding of how the choice of parameters impacts upon the algorithm performance. The second set of experiments is to make comparison of the performance with other approaches. Note that the input data of these experiments were collected from actual MSIC (*e.g.*, Fig. 10) in a social manufacturing prototype system, which contains interactions of 1000 around Printing Machine manufacturing enterprises that mainly from a National High-New Technology Zone of China.

4.1. Input data and parameter settings

Totally 6950 samples (sentences) of manufacturing relationships are collected from the MSIC in a social manufacturing prototype system, with known ground truth about the labels to conduct computational experiments to assess the accuracy of the algorithm. Each relationship category contains 1000 around samples. Table 1 gives some labeled input examples for each category.

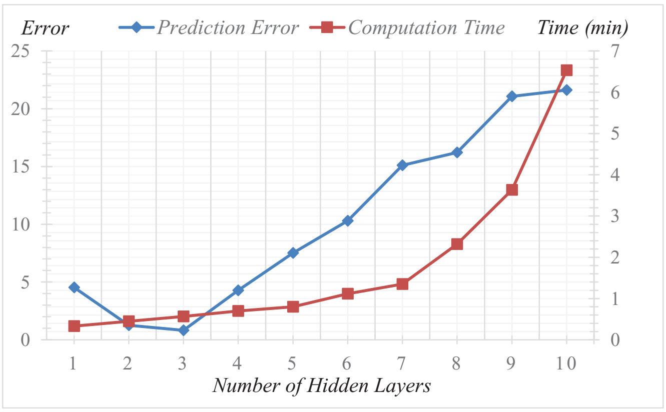


Fig. 6. Prediction error and computation time variation with number of hidden layers.

Table 2 offers an overview of the scale information used in the each category of manufacturing relationship. Note that the "*All*" category refers to aggregating all the identified seven categories of manufacturing relationships as an independent experiment, and thus its number of samples equals the sum of other seven experiments. However, its data dimension of input and output does not equal to the sum of other seven experiments because there are coincident input embedding words and pre-identified output labels.

The last two rows in Table 2 show the remarks of each relationship category from the perspective of *dimension* [46] and *sparsity* [47] of input text data, which are evaluated by *N* and *(N – M)/N* in this study, respectively. Note that an input data with high dimension refers to a large *N*, and also an input data with high sparsity implies a large *(N – M)/N*. For example, the "*Capable-of*" category has a high dimension and high sparsity, evidenced by $N = 475$ and $(N-M)/N = 0.94;$ in relative terms, the "*Call-for*" category has a low dimension and low sparsity, evidenced by $N = 219$ and $(N-M)/N = 0.81$. These differences will result in different learning performances, which will be discussed latterly.

With respect to the number of hidden layer, the number is experimentally varied from 1 to 10 and compute the perform result, as illustrated in Fig. 6. The results show that it does not improve the performance when the layer number is greater than 3. Moreover, the computation time increase exponentially with the increase of layer number. Because the computation scale of the training dataset is limited, the deep neural network is prone to over-fitting when using large hidden layers. Thus, the deep learning network has three layers. Note that the number of

Table 1
Examples of input labeled manufacturing relationships.

Category	Examples	Labeled relationships
Collaborate	*<e1>We</e1> will outsource a deep-hole machining job to <e2>Qinya</e2>.*	*<e1> Outsource machining job to <e2>*
Membership	*This <e1>gravure printing machine</e1> has a major component named <e2>compression roller</e2>.*	*<e1> Has component <e2>*
Call-for	*<e1>We</e1>have a gear box <e2>grinding</e2> task to outsource.*	*<e1> Have <e2> to outsource*
Capable-of	*<e1>We</e1> can provide with specialized computer-controlled <e2>glass milling</e2> capability.*	*<e1> Provide <e2> capability*
Manufacture	*These components are the up-grade version of <e1>coating machine</e1> which are already produced by <e2>Beiren</e2>.*	*<e1> Produced by <e2>*
Associate	*We have an <e1>assembly</e1> demand from the part of <e2>gravure printer</e2>to outsource.*	*<e1> From part of <e2>*
Match	*Our assembly apart has the <e1>capacity</e1> to cover the requirement from <e2>Junye</e2>.*	*<e1> Cover requirement from <e2>*

Table 2
Sample information of each relationship category.

Items	All	Collaborate	Membership	Call-for	Capable-of	Manufacture	Associate	Match
Samples (S)	6950	1140	830	1005	960	1005	980	1030
N	1426	298	301	219	475	567	343	282
M	97	37	28	42	28	40	35	35
$(N-M)/N$	0.93	0.87	0.91	0.81	0.94	0.93	0.90	0.88
Dimension	High	Low	Relatively low	Low	High	High	Relatively high	Low
Sparsity	High	Relatively low	Relatively high	Low	High	High	Relatively high	Relatively low

Table 3
Deep neural network parameters used for each relationship category.

Items	All	Collaborate	Membership	Call-for	Capable-of	Manufacture	Associate	Match
L_1	1500	320	320	240	500	580	360	300
L_2	800	150	150	120	240	280	180	150
L_3	300	60	50	60	80	90	60	60

hidden layers is problem dependent and generally tuned based on performance on validation dataset (also called a trial-and-error approach) [33,34]. With respect to the neural number in each hidden layer (L_1, L_2, and L_3), the strategy is in accordance with in Collobert et al. [39], which is increasing dimension in the first layer (*i.e., over-complete*) to capture the complexity and then decreasing dimension layer by layer (*i.e., under-complete*) to extract key features. Finally, the optimal parameters of the key deep learning parameters used in the each category of manufacturing relationship are experimentally obtained as show in Table 3.

As shown in Fig. 7, the numbers of the three parameters (*i.e., weight decay*, *learning rate*, and *noise level*) are experimentally varied for the relationship extraction. In the training model, selection of learning rate for stochastic gradient descent process is very important. This study tested 10 different parameters, and eventually found the optimal learning rate is set as 10^{-1}. The results show that the model converges to local optimal solution when the learning rate is too large, while too small learning rate will make the training time is too long. With respect to the weight decay for regularization penalty in back propagation training, the best result is obtained when it is set as 10^{-4}. In terms of noise level, when parameter is too large, the algorithm will produce high error because of too much information missing. In addition, through the experiment, it can be found that when the noise level decrease layer by layer can achieve a more satisfied result. The selected noise levels in three hidden layers are 0.1, 0.05, and 0.01, respectively.

4.2. Comparative experiments and discussions

To evaluate the performance of the proposed algorithm, three benchmark algorithms are introduced in Table 4. The first competitor employs *Back Propagation* (BP) artificial neural network as the classifier [48]. The second competitor is a naïve *Denoising Auto-Encoder* (DAE) without any supervised fine tuning [33,34,43]. The third competitor is a *Stacked Auto-Encoder* (SAE) that fine-tune by gradient descent but without denoising [35,49]. These three competitors are good choices for us to try test learning performance of the proposed approach on real-world data while spending minimal efforts on preprocessing and formatting. These learning algorithms are evaluated using two standard evaluation metrics: *prediction error* and *computation time* (min). Note that the prediction error is evaluated by *Mean Squared Error* (MSE) that measures the average of squares of the difference between the predicted and the real value (*i.e.,* $\boldsymbol{x\_label}$). Each experiment is performed via repeated 5-times 10-fold cross-validation to make sure the result more persuasive. Table 4 illustrates the measure results including *average error* and *Standard Deviation* (*SD*) for these competing methods. Based on these results, the following detailed observations can be made.

With respect to the *prediction error*, the proposed SDAE method generally achieves the best performance (including the *average error* and *SD*) among all of the compared methods, and a strict ordering can also be figured: SAE being better than BP being better than DAE, as shown in Fig. 8. For example, in the experiment of category "*Call-for*", the SDAE method achieves improvements 93%, 98%, and 83%, respectively, as evidenced by decreases of 1.53, 6.83, 0.59 $\rightarrow$ 0.10. Compared to DAE, the SDAE method achieves a large improvement as it has a back propagation fine tuning process while DAE does not have one, which can avoid the over-fitting problem. Compared to SAE, the result of SDAE shows that pre-training with a non-zero noise level is better than that without denoising.

As stated in the former Section 4.1, the dimension and sparsity of input data are two main factors that affect the learning result. It could be concluded that the learning performance of all compared methods (including SDAE) for data with high dimension and high sparsity are relatively poor in terms of the prediction error, as evidenced by three highest errors obtained in the "*All*" experiment: 7.85, 16.87, 4.67 and 2.18, respectively. However, from the viewpoint of performance improvement, the SDAE method has an obvious performance improvement than BP DAE, and SAE, as respectively evidenced by a large decrement of 72%, 87%, and 53% (7.85, 16.87, 4.67 $\rightarrow$ 2.18). Moreover, as shown in Fig. 8, when

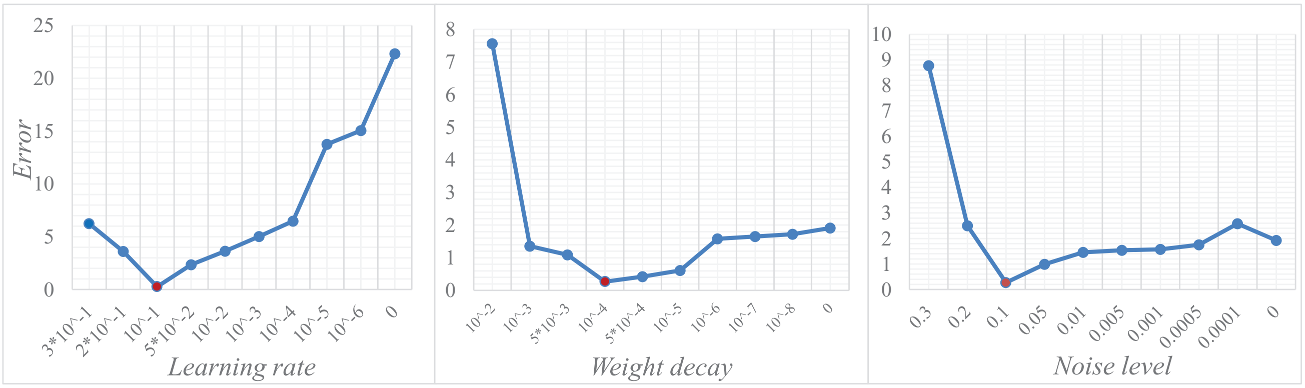


Fig. 7. Algorithm parameters setting based on the sensitivity experiments.

Table 4
Comparison of four approaches for eight categories.

Category	Scale ($S*N$)	Average error (SD)				Computation time			
		BP	DAE	SAE	SDAE	BP	DAE	SAE	SDAE
All	9,910,700	7.85(0.42)	16.87(0.93)	4.67(0.02)	**2.18**(0.04)	69.4	176.3	53.7	53.9
Collaborate	339,720	3.54(0.36)	5.85(0.43)	1.64(0.03)	**0.55**(0.03)	6.9	2.2	13.3	14.4
Membership	249,830	2.61(0.46)	4.64(0.36)	0.82(0.02)	**0.27**(0.01)	4.1	1.5	9.6	10.3
Call-for	220,095	1.53(0.24)	6.83(0.41)	0.59(0.03)	**0.10**(0.01)	3.5	1.1	7.8	8.7
Capable-of	456,000	3.19(0.34)	9.58(0.54)	2.53(0.01)	**1.13**(0.02)	8.7	3.1	17.9	18.2
Manufacture	569,835	4.23(0.31)	7.36(0.55)	2.16(0.03)	**0.87**(0.03)	11.6	4.7	20.2	20.4
Associate	336,140	2.47(0.30)	4.46(0.40)	1.34(0.03)	**0.82**(0.02)	5.3	1.6	13.3	13.8
Match	290,460	3.32(0.28)	6.17(0.46)	0.89(0.04)	**0.41**(0.02)	4.6	1.5	10.4	10.6

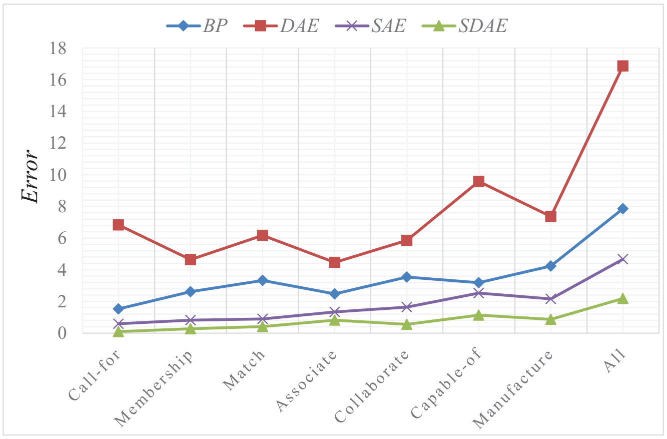


Fig. 8. Prediction error of four approaches for eight categories.

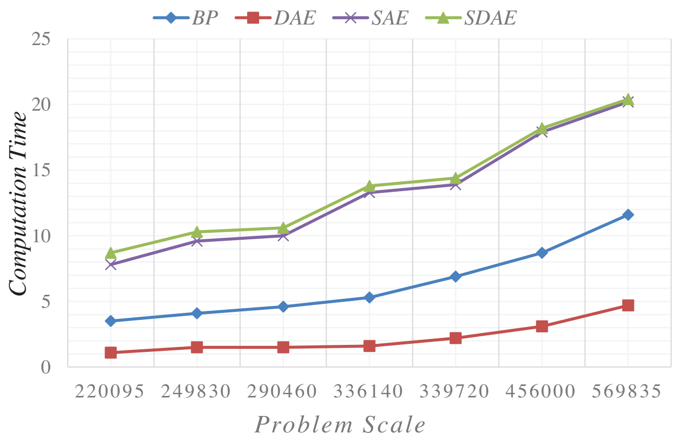


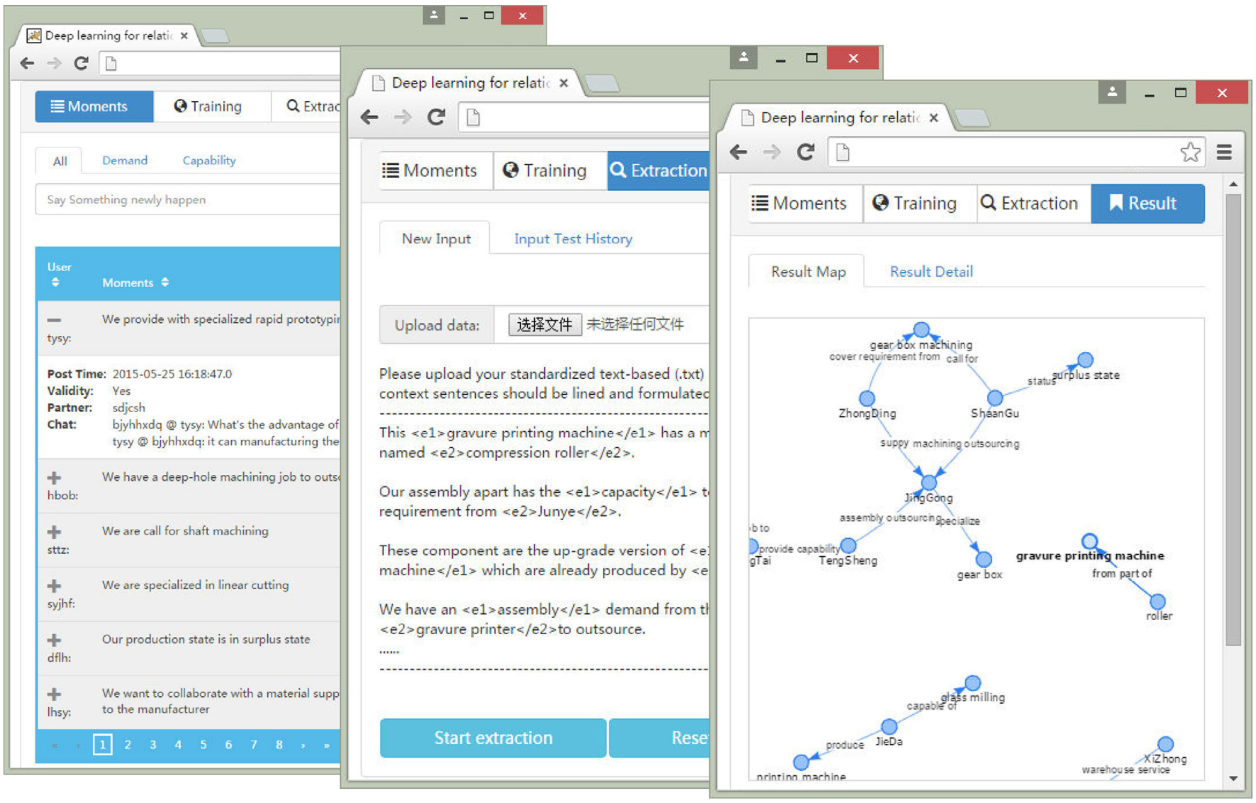
Fig. 9. Computation time variation with the problem scale.

the dimension range from 475 (category "*Capable-of*") to 1426 (category "*All*"), the results achieved by SDAE increase from 1.13 to 2.18, which is in a relatively smaller range than BP (4.19 → 7.85), DAE (9.58 → 16.87), and SAE (2.53 → 4.67). In another word, the SDAE method is relatively more reliable and effective when the dimension of input data increases. Especially, the "*Capable-of*" category is a test dataset that with high sparsity as shown in Table 2; and the results show that the SDAE method achieves a relatively small performance improvement than BP, as evidenced by a small decrement of 64% (3.19 → 1.13). It implies that the SDAE are not efficient enough when the input data are of high sparsity although it achieves a better comparative result, which will be the focus of future work.

With respect to the *computation time*, as shown in Fig. 9, it can be inferred that the computation times of all four methods are increase with the increase of problem scale, which is defined as the product of number of samples times the dimension of input data. And the SDAE is the slowest algorithm compared to other three methods for the seven single-category experiments. On the surface, the SDAE is supposed to be slower than others, as it has a back propagation fine tuning process. However, when the problem scale increases from 569835 (*i.e.*, category "*Capable-of*") to 9910700 (*i.e.*, category "*All*"), the SDAE has the smallest increase of 164% (20.4 → 53.9), as BP is 498% (11.6 → 69.4), DAE is 3651% (4.7 → 176.3), and SAE is 166% (20.2 → 53.7). This is because the SDAE trains the neural network layer by layer, and thus its computation time is linear to the problem scale. It implies that the proposed SDAE methods are relatively more efficient when the dimension of input data increases.

4.3. An implementation of manufacturing relationships extraction via a prototype system

The goal of deep learning is to select features that is the most representative and coherent with the rest of the sentence, and



tysy: We provide with specialized rapid prototypi

Post Time: 2015-05-25 16:18:47.0
Validity: Yes
Partner: sdjcsh
Chat: bjyhhxdq @ tysy: What's the advantage of
tysy @ bjyhhxdq: it can manufacturing the

hbob: We have a deep-hole machining job to outs

sttz: We are call for shaft machining

syjhf: We are specialized in linear cutting

dflh: Our production state is in surplus state

lhsy: We want to collaborate with a material supp to the manufacturer

« ‹ 1 2 3 4 5 6 7 8 › »

Please upload your standardized text-based (.txt)
context sentences should be lined and formulatec

This <e1>gravure printing machine</e1> has a m
named <e2>compression roller</e2>.

Our assembly apart has the <e1>capacity</e1> t
requirement from <e2>Junye</e2>.

These component are the up-grade version of <e
machine</e1> which are already produced by <e

We have an <e1>assembly</e1> demand from th
<e2>gravure printer</e2>to outsource.
......

Fig. 10. Relationship extraction on a prototype system based on the proposed approach.

these selected features can represent the desired manufacturing relationships from interaction context. Fig. 10 shows an implementation of manufacturing relationships extraction on a *social manufacturing prototype system* based on the proposed approach. This prototype system is a browser/server architecture that developed through Java and Spring–Struts–Hibernate (SSH) framework on the server side and visit with Chrome on the browser side.

Firstly, the text-based MSIC (as shown in the left window of Fig. 10) is manually selected, and then a pre-processing is conducted to get standardized input sentence vectors, each of which contains two named entities and a relationship label just as Table 1 illustrated. Secondly, the deep neural network is trained with these high dimensional sentence level vectors. The resulting trained neural network parameters imply the underlying patterns between input sentence vectors and labels, which can be used to extract very subtle manufacturing relationships between named entities from the same MSIC. Thirdly, as shown in the middle window of Fig. 10, the treated and semi-structured text input data is uploaded, and the information extraction procedure is implemented on these data. Finally, a manufacturing-relationships-interconnected sparse network is extracted in the right window of Fig. 10, and it contains many relationships, such as *Capability* and the *Enterprise* it belongs to. In the implementation process, some observations and insights can be drawn as follows:

(1) The presented system focuses on the learning of hidden patterns for extracting relationships from text, and thus facilitates knowledge transferring in the MSIC. For example, in terms of a relationship edge in the right window of Fig. 10: *<e1>JingGong</e1> assembly outsourcing <e2>TengSheng</e2>*, although it is easy to identify the words "*assembly outsourcing*" as a key feature for extracting the manufacturing relationships, it is much more challenging when subjecting this feature in a more complex input text. Actually, these learned features are supposed to have a smallest distance to as much related samples as possible, and then can be act as labels for capturing the key information. Thus word features trained on even larger datasets with larger dimensionality will significantly improve the learn ability.

(2) As a manufacturing entity may appear multiple times in the input data, a sparse manufacturing relationships network that aggregated from all extracted manufacturing entities and relationships can be obtained, as shown in right window of Fig. 10. As this relationship network has captured relationships and interactions that occur across enterprises in the MSIC, it can be used by DMs to understand how manufacturing resources and capabilities are distributed in the network, and identify potential partners so as to outsource manufacturing operations to them. For instance, in this network, the majority of manufacturing resources and activities surround enterprise *<e>JingGong</e>*, as this node has the largest in-degree from the graph theory perspective. It suggests that this enterprise has stable productivity and credibility, and thus the DMs can consider outsourcing machining task to this enterprise.

(3) The obtained manufacturing relationships network was sparse and heterogeneous in its structure, the DMs can manually find out the desired type of relationships among enterprises, products, demands, and capabilities from the captured context. For instance, if the DMs has the specialty of *gear box machining* and seek manufacturing orders, he can find the potential patter *<e>ShaanGu</e>*, as this enterprise "call-for" *gear box machining*. However, when the DMs input a large amount of MSIC data, the system will output a very large relationships network, which cannot be utilized directly or manually as decision-support knowledge. Under this circumstances, further methods for processing

this structured network-form knowledge, such as graph matching algorithm [3] and complex network analysis, are supposed to be introduced to support the cross-enterprise decision.

5. Conclusion

This study proposes a deep learning approach based on *stacked denoising auto-encoder* to capture sentence level features for the manufacturing relationships extraction issue in social manufacturing paradigm. The contributions of this study includes following aspects: (1) Instead of exploiting man-made features carefully optimized for the relationships extraction task, the proposed approach directly learns internal representations on the basis of deep learning from a large collection of coarse plain-text data. (2) The automatically learned features can also replace the elaborately designed features that are based on the outputs of existing NLP tools. (3) The experiments proved that the proposed deep learning model is efficient for manufacturing relationship extraction and scales well to large data than other three latest methods.

However, the proposed approach has some limitations for real-world applications: (1) The proposed approach needs *Named Entity Recognition* as preprocessing in the very beginning. Future work will focus on reducing the reliance on valid labeled data and addressing the manufacturing relationship extraction task on large scale unlabeled interaction context data. Also, the adaption ability of the proposed approach needs further research. (2) The contribution of different features such as lexical and type features need further study, and features with high effect should be identified to improve the performance of manufacturing relationship extraction. (3) Due to the high sparsity and large-scale nature in the interaction context of social manufacturing paradigm, extracting and detecting of abnormal relationships is a challenge, and the optimization techniques such as swarm intelligent methods [50] and biogeography-based optimization [51] could be introduced as complements for current extraction model. These limitations should be taken into consideration in future research.

Acknowledgment

The research work presented in this study is under the support of NSFC with Grant no. 71571142.

Appendices:

Appendix A. Decision makers' evaluations on ten original variables in the MSIC

	Manufacturer	Productivity	Technology	Demand	Delivery	Equipment	Supplier	Price	Service	Quality
DM_1	5	5	4	10	2	8	6	8	7	3
DM_2	2	5	3	2	2	7	5	7	5	1
DM_3	5	2	6	9	10	5	7	7	5	6
DM_4	2	2	4	2	3	5	7	8	6	2
DM_5	5	2	2	1	4	4	8	8	7	1
DM_6	2	5	9	2	7	5	7	7	7	6
DM_7	7	4	4	9	3	4	7	7	6	2
DM_8	2	5	6	9	4	4	7	7	5	4
DM_9	8	10	8	4	1	9	7	7	6	5
DM_{10}	7	4	10	7	5	6	7	7	9	10
DM_{11}	5	4	5	9	4	7	7	7	6	7
DM_{12}	7	2	4	10	5	5	7	6	7	4
DM_{13}	3	5	2	6	3	6	7	7	5	4
DM_{14}	2	2	3	9	2	4	8	8	5	2
DM_{15}	6	3	10	5	6	5	6	7	7	10
DM_{16}	3	2	6	3	8	4	5	7	8	6
DM_{17}	6	3	4	10	4	5	9	8	6	5
DM_{18}	9	3	7	2	5	5	8	7	10	4
DM_{19}	8	5	3	1	3	8	7	6	9	10
DM_{20}	5	3	1	10	3	5	8	8	4	3

Appendix B1. Correlation matrix[a]

	Manufacturer	Productivity	Technology	Demand	Delivery	
Correlation	Manufacturer	1.000	.189	.216	.039	−.039
	Productivity	.189	1.000	.242	−.132	−.476
	Technology	.216	.242	1.000	−.152	.447
	Demand	.039	−.132	−.152	1.000	−.034
	Delivery	−.039	−.476	.447	−.034	1.000
	Equipment	.327	.737	.059	−.129	−.446
	Supplier	.282	−.170	−.225	.260	−.146
	Price	−.364	−.233	−.297	.169	−.263
	Service	.559	−.053	.449	−.463	.227
	Quality	.386	.127	.628	−.063	.412

Appendix B2. Correlation matrix[a]

		Equipment	Supplier	Price	Service	Quality
Correlation	Manufacturer	.327	.282	−.364	.559	.386
	Productivity	.737	−.170	−.233	−.053	.127
	Technology	.059	−.225	−.297	.449	.628
	Demand	−.129	.260	.169	−.463	−.063
	Delivery	−.446	−.146	−.263	.227	.412
	Equipment	1.000	−.252	−.239	.122	.269
	Supplier	−.252	1.000	.351	−.069	−.134
	Price	−.239	.351	1.000	−.381	−.514
	Service	.122	−.069	−.381	1.000	.491
	Quality	.269	−.134	−.514	.491	1.000

a. Determinant = 0.004.

Appendix C. Communalities

	Initial	Extraction
Manufacturer	1.000	.858
Productivity	1.000	.832
Technology	1.000	.634
Demand	1.000	.897
Delivery	1.000	.856
Equipment	1.000	.844
Supplier	1.000	.798
Price	1.000	.543
Service	1.000	.880
Quality	1.000	.784

Appendix D. Total variance explained

Component	Initial eigenvalues			Extraction sums of squared loadings		
	Total	% of variance	Cumulative %	Total	% of variance	Cumulative %
1	3.180	31.802	31.802	3.180	31.802	31.802
2	2.177	21.766	53.568	2.177	21.766	53.568
3	1.464	14.640	68.208	1.464	14.640	68.208
4	1.106	11.060	79.268	1.106	11.060	79.268
5	.764	7.636	86.904			
6	.451	4.511	91.415			
7	.387	3.871	95.286			
8	.260	2.598	97.884			
9	.134	1.342	99.225			
10	.077	.775	100.000			

Extraction method: principal component analysis.

Appendix E. Component matrix[a]

	Component			
	1(Product)	2(Capability)	3(Enterprise)	4(Demand)
Manufacturer	.563	.145	.716	−.088
Productivity	.364	.810	−.111	.179
Technology	.711	−.258	−.093	.233
Demand	−.366	−.035	.401	.775
Delivery	.308	−.837	−.147	.195
Equipment	.442	.801	−.027	.081
Supplier	−.332	−.069	.820	−.103
Price	−.714	.014	.107	−.145
Service	.733	−.213	.221	−.499
Quality	.803	−.198	.118	.293

Extraction method: principal component analysis.[a]
a. 4 components extracted.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.knosys.2016.03.008.

References

[1] W. Cao, P.Y. Jiang, Cloud machining community for social manufacturing, Appl. Mech. Mater. 220–223 (2012) 61–64.
[2] D. Wu, D. Schaefer, D.W. Rosen, Cloud-based design and manufacturing systems: a social network analysis, in: Proceedings of the Nineteenth International Conference on Engineering Design, 2013, pp. 1–10.
[3] J. Leng, P. Jiang, K. Ding, Implementing of a three-phase integrated decision support model for parts machining outsourcing, Int. J. Prod. Res. 52 (12) (2014) 3614–3636.
[4] K. Wang, Applying data mining to manufacturing: The nature and implications, J. Intell. Manuf. 18 (4) (2007) 487–495.
[5] S.C. Hui, G. Jha, Data mining for customer service support, Inf. Manag. Amst. 38 (1) (2000) 1–13.
[6] B. Agard, A. Kusiak, Data-mining-based methodology for the design of product families, Int. J. Prod. Res. 42 (15) (2004) 2955–2969.
[7] O. Lopez-Ortega, M. Ramirez-Hernandez, A formal framework to integrate express data models in an extended enterprise context, J. Intell. Manuf. 18 (3) (2007) 371–381.
[8] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, From data mining to knowledge discovery in databases, AI Mag. 17 (3) (1996).
[9] R. Feldman, J. Sanger, The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data, Cambridge University Press, 2007.
[10] G. Forman, An extensive empirical study of feature selection metrics for text classification, J. Mach. Learn. Res. 3 (7-8) (2003) 1289–1305.
[11] C. Jiang, F. Coenen, R. Sanderson, M. Zito, Text classification using graph mining-based feature extraction, Knowl. Based Syst. 23 (4) (2010) 302–308.
[12] W. Zhang, T. Yoshida, X. Tang, Using ontology to improve precision of terminology extraction from documents, Expert Syst. Appl. 36 (5) (2009) 9333–9339.
[13] J. Nothman, N. Ringland, W. Radford, T. Murphy, J.R. Curran, Learning multilingual named entity recognition from Wikipedia, Artif. Intell. 194 (SI) (2013) 151–175.
[14] C. Zhang, W. Xu, Z. Ma, S. Gao, Q. Li, J. Guo, Construction of semantic bootstrapping models for relation extraction, Knowl. Based Syst. 83 (2015) 128–137.
[15] W. Jin, R.K. Srihari, H.H. Ho, A text mining model for hypothesis generation, in: Proceedings of the Nineteenth IEEE International Conference on Tools with Artificial Intelligence, 2007, pp. 156–162.
[16] A. Weichselbraun, A. Gindl, A. Scharl, Enriching semantic knowledge bases for opinion mining in big data applications, Knowl. Based Syst. 69 (SI) (2014) 78–85.
[17] R.K. Lindsay, M.D. Gordon, Literature-based discovery by lexical statistics, J. Am. Soc. Inf. Sci. 50 (7) (1999) 574–587.
[18] A. Gangemi, V. Presutti, S. Staab, R. Studer, Ontology design patterns, Handbook on Ontologies, Springer, Berlin Heidelberg, 2009, pp. 221–243.
[19] J. Naradowsky, S. Riedel, D. Smith, Improving NLP through marginalization of hidden syntactic structure, in: Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 810–820.
[20] A. Moh'D, Chi square feature extraction based Svms arabic language text categorization system, J. Comput. Sci. 3 (6) (2007) 430–435.
[21] F.M. Suchanek, G. Ifrim, G. Weikum, Combining linguistic and statistical analysis to extract relations from web documents, in: Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2006, pp. 712–717.
[22] A. Culotta, J. Sorensen, Dependency tree kernels for relation extraction, in: Proceedings of the Forty Second Annual Meeting of the Association for Computational Linguistics, 2004, p. 423.
[23] A.K. Seewald, F. Kleedorfer, Lambda pruning: an approximation of the string subsequence kernel for practical SVM classification and redundancy clustering, Adv. Data Anal. Classif. 1 (3) (2007) 221–239.
[24] G. Zhou, Q. Zhu, Kernel-based semantic relation detection and classification via enriched parse tree structure, J. Comput. Sci. Technol. Ch. 26 (1) (2011) 45–56.
[25] B. Min, R. Grishman, L. Wan, C. Wang, D. Gondek, Distant supervision for relation extraction with an incomplete knowledge base, in: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics, 2013, pp. 777–782.
[26] M. Mintz, S. Bills, R. Snow, D. Jurafsky, Distant supervision for relation extraction without labeled data, in: Proceedings of the Joint Conference of the Forty Seventh Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2009, pp. 1003–1011.
[27] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, Relation classification via convolutional deep neural network, in: Proceedings of the Twenty Fifth International Conference on Computational Linguistics, COLING 2014,, 2014, pp. 2335–2344.
[28] J. Ebrahimi, D. Dou, Chain based RNN for relation classification, in: Proceedings of the Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL: Association for Computational Linguistics, 2015, pp. 1244–1249.
[29] G.E. Hinton, S. Osindero, Y. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[30] H. Lee, R. Grosse, R. Ranganath, A.Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the Twenty Sixth Annual International Conference on Machine Learning, 2009, pp. 609–616.
[31] H. Larochelle, Y. Bengio, J. Louradour, P. Lamblin, Exploring strategies for training deep neural networks, J. Mach. Learn. Res. 10 (2009) 1–40.
[32] Y. Bengio, P. Lamblin, H. Larochelle, D. Popovici, U. Montreal, Greedy layer-wise training of deep networks, Adv. Neural Inf. Process. Syst. 19 (2007) 153–160.
[33] P. Vincent, H. Larochelle, Y. Bengio, P. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the Twenty Fifth International Conference on Machine Learning, 2008, pp. 1096–1103.
[34] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion, J. Mach. Learn. Res. 11 (12) (2010) 3371–3408.
[35] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.
[36] Z. Harris, Mathematical Structures of Language, Wiley, New York, 1968.
[37] T.L. Esper, E.E. Alexander, T.P. Stank, D.J. Flint, M. Moon, Demand and supply integration: a conceptual framework of value creation through knowledge management, J. Acad. Mark. Sci. 38 (1) (2010) 5–18.
[38] H.S. Jagdev, J. Browne, The extended enterprise-a context for manufacturing, Prod. Plan. Control. 9 (3) (1998) 216–229.
[39] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537.
[40] R. Collobert, J. Weston, A unified architecture for natural language processing: Deep neural networks with multitask learning, in: Proceedings of the Twenty Fifth International Conference on Machine Learning, 2008, pp. 160–167.
[41] A. Mnih, G. Hinton, A scalable hierarchical distributed language model, in: Proceedings of the 2008 Neural Information Processing Systems Foundation, 2008, pp. 1081–1088.
[42] J. Turian, L. Ratinov, Y. Bengio, Word Representations: A Simple and General Method for Semisupervised Learning, Association for Computational Linguistics, Uppsala, Sweden, 2012, pp. 384–394.
[43] M. Chen, Z.E. Xu, K.Q. Weinberger, F. Sha, Marginalized stacked denoising autoencoders, in: Proceedings of the Learning Workshop, 2012, pp. 1–2.
[44] R. Amalraj, M. Dharmalingam, A work point count system coupled with back-propagation for solving double dummy bridge problem, Neurocomputing 168 (2015) 160–178.
[45] Z. Yudong, W. Shuihua, J. Genlin, P. Phillips, Fruit classification using computer vision and feedforward neural network, J. Food Eng. 143 (2014) 167–177.
[46] L. Parsons, E. Haque, H. Liu, Subspace clustering for high dimensional data: a review, ACM SIGKDD Explor. Newsl. Spec. Issue Learn. Imbalanced Datasets 6 (1) (2004) 90–105.
[47] M. Grčar, D. Mladenič, B. Fortuna, M. Grobelnik, Data Sparsity Issues in the Collaborative Filtering Framework, Springer, Berlin Heidelberg, 2006.
[48] X. Yao, Evolutionary artificial neural networks, Int. J. Neural Syst. 4 (3) (1993) 203–222.
[49] M. Ranzato, F.J. Huang, Y.L. Boureau, Y. LeCun, Unsupervised learning of invariant feature hierarchies with applications to object recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2007, pp. 1–8.
[50] W. Shuihua, Z. Yudong, D. Zhengchao, D. Sidan, J. Genlin, Y. Jie, Y. Jiquan, W. Qiong, F. Chunmei, P. Phillips, Feed-forward neural network optimized by hybridization of PSO and ABC for abnormal brain detection, Int. J. Imaging Syst. Technol. 25 (2) (2015) 153–164.
[51] G. Yang, Z. Yudong, Y. Jiquan, J. Genlin, D. Zhengchao, W. Shuihua, F. Chunmei, W. Qiong, Automated classification of brain images using wavelet-energy and biogeography-based optimization, Multimed. Tools Appl. 74 (9) (2015) 1–17.