A protein–protein interaction extraction approach based on deep neural network

Zhehuan Zhao, Zhihao Yang*, Hongfei Lin and Jian Wang

College of Computer Science and Technology,
Dalian University of Technology,
Dalian 116024, China
Email: Zhehuan@mail.dlut.edu.cn
Email: yangzh@dlut.edu.cn
Email: hflin@dlut.edu.cn
Email: wangjian@dlut.edu.cn
*Corresponding author

Song Gao

Department of pharmacy,
First Affiliated Hospital of Dalian Medical University,
Dalian 116023, China
Email: songgaodmu@gmail.com

Abstract: Protein–Protein Interactions (PPIs) information extraction from biomedical literature helps unveil the molecular mechanisms of biological processes. Machine learning methods have been the most popular ones in PPI extraction area. However, these methods are still feature engineering-based, which means that their performances are also heavily dependent on the appropriate feature selection which is still a skill-dependent task. This paper presents a deep neural network-based approach which can learn complex and abstract features automatically from unlabelled data by unsupervised representation learning methods. This approach first employs the training algorithm of auto-encoders to initialise the parameters of a deep multilayer neural network. Then the gradient descent method using back propagation is applied to train this deep multilayer neural network model. Experimental results on five public PPI corpora show that our method can achieve better performance than can a multilayer neural network: on two 'toughest handling' corpora AImed and BioInfer, the former outperforms the latter with the improvements of 3.10 and 2.89 percentage units in *F*-score, respectively. In addition, the performance comparison with APG also verifies the effectiveness of our method.

Keywords: deep learning; biomedical text mining; interaction extraction; neural network.

Reference to this paper should be made as follows: Zhao, Z., Yang, Z., Lin, H., Wang, J. and Gao, S. (2016) 'A protein–protein interaction extraction approach based on deep neural network', *Int. J. Data Mining and Bioinformatics*, Vol. 15, No. 2, pp.145–164.

Biographical notes: Zhehuan Zhao is currently working toward the PhD degree in Computer Science from Dalian University of Technology, Dalian, China. His research interests include machine learning and biomedical text mining.

Zhihao Yang received his PhD degree in Computer Science from Dalian University of Technology, Dalian, China, in 2008. He is currently a Professor in the College of Computer Science and Technology at Dalian University of Technology. He has published over 20 research papers on topics in biomedical literature data mining. His research interests include biomedical literature data mining and information retrieval. His research projects are funded by the Natural Science Foundation of China.

Hongfei Lin received the MS degree from Dalian University of Technology, China, in 1992 and the PhD degree from Northeastern University, China, in 2000. He is currently a Professor in the College of Computer Science and Technology at the Dalian University of Technology. His research interest includes text mining for biomedical literatures, learning to rank, sentimental analysis, and opinion mining.

Jian Wang received her PhD degree in Computer Science and Technology from the Dalian University of Technology, China, in 2014. She is working as a Professor in the School of Computer Science and Technology at the Dalian University of Technology. She has published more than 50 research papers in various journals, conferences, and books. In recent years, she has focused on machine learning, text mining for biomedical literatures, information extraction from huge biomedical resources, sentimental analysis.

Song Gao received her BSc degree in College of Life Sciences from the Jilin University, China, in 2004. She graduated at the Pharmacy from the Anshan Medical School, China, in 1988. Currently, she is working as a Pharmacist in the First Affiliated Hospital of Dalian Medical University. She has published some articles in various journals. In recent years, she has focused on pharmacy, drug-disease relationship, and preparation detection research.

This paper is a revised and expanded version of a paper entitled 'Deep neural network based protein-protein interaction extraction from biomedical literature' presented at the '2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)', Washington DC, 9–12 November 2015.

1 Introduction

Protein–Protein Interactions (PPIs) are essential for understanding the molecular mechanisms of biological processes. A number of databases such as MINT (Zanzoni et al., 2002), BIND (Bader et al., 2001), and DIP (Xenarios et al., 2000) have been created to store protein interaction information in structured and standard formats. Many databases, such as MINT, manually detect and curate protein interactions from biomedical literature. However, with the rapidly expanding biomedical literature containing PPIs, it becomes difficult for database curators to manually curate PPI information from this literature. Thus, the automatic extraction of such information from biomedical literature has become an important research area.

Existing PPI works can be roughly divided into two main categories: manual pattern engineering-based approaches and machine learning-based approaches. Manual pattern engineering approaches define a set of rules for possible textual relationships, called patterns, which encode similar structures in expressing relationships. The Hinton et al. used regular expressions, with probabilities that reflect the experimental accuracy of each pattern to extract interactions into predefined frame structures (Blaschke and Valencia, 2002). Ono et al. (2001) defined a set of rules based on syntactic features to preprocess complex sentences, making further use of negation structures. Corney et al. (2004) applied manually engineered templates that combine lexical and semantic information to identify protein interactions. Such manual pattern engineering approaches for information extraction are very hard to scale up to large document collections since they require labour-intensive and skill-dependent pattern engineering.

Currently, the machine learning approaches have become the popular ones in PPI extraction research due to the public availability of large annotated PPI corpora, such as AImed (Bunescu et al., 2005), BioInfer (Pyysalo et al., 2007) , and IEPA (Ding et al., 2002). In these approaches, the PPI extraction is performed by a decision function that determines for each unordered candidate pair of protein names occurring together in a sentence whether the two proteins interact or not. Xiao et al. (2005) used Maximum Entropy models to combine diverse lexical, syntactic and semantic features for PPI extraction. Zhou et al. (2006) employed a semantic parser using the Hidden Vector State (HVS) model for PPIs which can be trained using only lightly annotated data while simultaneously retaining sufficient ability to capture the hierarchical structure.

Among others, kernel-based methods are an effective alternative to explicit feature extraction (Cristianini and Shawe-Taylor, 2000). They retain the original representation of objects and use the object only via computing a kernel function between a pair of objects. Formally, a kernel function is a mapping $K: X \times X \rightarrow [0, \infty)$ from input space X to a similarity score $K(x,y) = \phi(x)\cdot\phi(y) = \sum_i \phi_i(x)\phi_i(y)$, where $\phi_i(x)$ is a function that maps X to a higher dimensional space with no need to know its explicit representation. Such a kernel function makes it possible to compute the similarity between objects without enumerating all the features. Several kernels have been proposed to extract PPIs from biomedical literature, such as subsequence kernels (Mooney and Bunescu, 2005), tree kernels (Moschitti, 2006; Sætre et al., 2007), shortest path kernels (Bunescu and Mooney, 2005), and graph kernels (Airola et al., 2008).

In recent years, researchers have proposed the combination of multiple kernels to extract PPIs. Kim et al. (2008) suggested four kernels – predicate kernel, walk kernel, dependency kernel and hybrid kernel – to adequately encapsulate information required for a relation prediction based on the sentential structures involved in two entities. Miwa et al. (2009) proposed a method that combines bag-of-words (BOW) kernel, subset tree kernel, and graph kernel, based on several syntactic parsers, in order to retrieve the widest possible range of important information from a given sentence. Yang et al. (2011) combined the following kernels: feature-based kernel, tree kernel, APG kernel and part-of-speech path kernel. The combination of multiple kernels can retrieve the widest range of important information in a given sentence and achieves better performance. However, like other kernel-based methods, these methods are still feature engineering-based and, therefore, their performances are heavily dependent on the feature selection. In addition, they only make use of the limited labelled data but ignore the huge amount of biomedical texts readily available.

To solve the problem, we introduce a deep learning-based approach into PPI extraction from biomedical literature, which can learn complex and abstract features automatically from unlabelled data by unsupervised representation learning methods (Bengio, 2009; Bengio et al., 2013; Hinton et al., 2006; Hinton and Salakhutdinov, 2006; Bengio, 2013). As described in detail previously (Bengio, 2013), a deep learning algorithm is a particular kind of representation learning procedure that discovers multiple levels of representation, with higher-level features representing more abstract aspects of the data, which has had important successes in many traditional AI applications such as computer vision (Hinton et al., 2012b; Wan et al., 2013; Krizhevsky et al., 2012), speech recognition (Graves et al., 2013; Hinton et al., 2012a), and natural language processing (Collobert and Weston, 2008; Collobert et al., 2011). However, few studies about its application in the research field of information extraction (IE) from biomedical literature have been reported.

In this paper, we present a deep learning-based approach to extract PPIs from biomedical literature. This approach employs greedy layer-wise unsupervised learning procedure relying on the training algorithm of auto-encoders to initialise the parameters of a deep multilayer neural network. Then it applies the gradient descent method using back propagation to train this deep multilayer neural network model. In standard learning strategies of deep multilayer neural networks, weights are randomly initialised, which is empirically known to find poor solutions for networks with three or more hidden layers (Larochelle et al., 2009). Instead, our approach uses greedy layer-wise unsupervised learning to initialise the parameters which not only makes learning deep multilayer neural network possible, but also learns much information from large unlabelled data automatically. Experimental results on five public PPI corpora show that our approach can achieve better performance than multilayer neural network (NN).

2 Methods

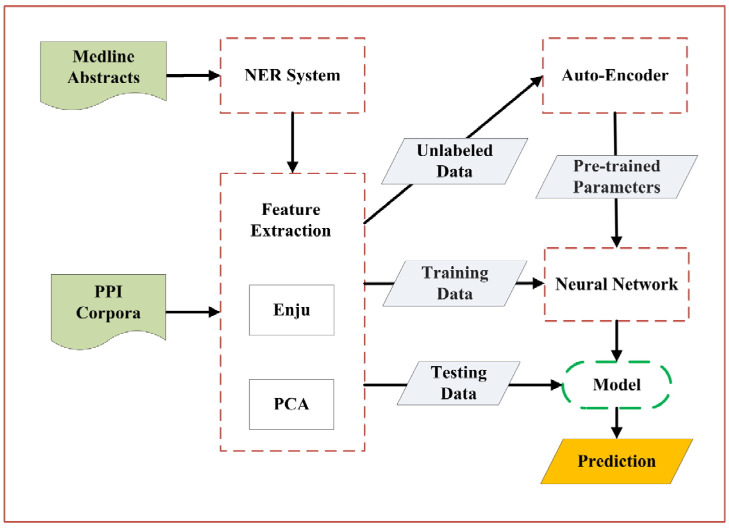
Our method contains four processing steps, as shown in Figure 1: (1) Named Entity Recognition (NER), which aims to identify the protein names in the biomedical text; (2) feature extraction, which generates raw features using Enju parser (Miyao and Tsujii, 2008) and Principal Component Analysis (PCA) (Jolliffe, 2005); (3) auto-encoder, which learns complex and abstract features by unsupervised greedy layer-wise training on the raw features; and (4) neural network, which trains a supervised neural network using the features learned by the auto-encoder.

2.1 NER

In our method, an NER module based on the semi-supervised learning strategy, FCG (Li et al., 2009), is employed to recognise the protein names in the unlabelled data set. The performance of FCG is examined in a named entity classification task, which is designed to remove non-gene entries in a large dictionary derived from online resources. The results show that new features generated by FCG outperform lexical features by 5.97 percentage units in *F*-score. Also in this framework each extension yields significant improvements and the sparse lexical features can be transformed into both a lower dimensional and more informative representation. A forward maximum match method based on the refined dictionary produces an *F*-score of 86.2% on BioCreative II Gene

Mention test set. Then the dictionary is combined with a CRFs-based gene mention tagger, achieving an *F*-score of 89.05%. More details about FCG are presented in Li et al. (2009).

Figure 1 The processing flow of our method (see online version for colours)



2.2 *Original raw feature extraction*

To ensure the generalisation of the learned model, the protein names recognised in the text are replaced with PROT1, PROT2, or PROT, where PROT1 and PROT2 are the pair of interest, and the rest of the proteins are marked as PROT. An example is given as follows:

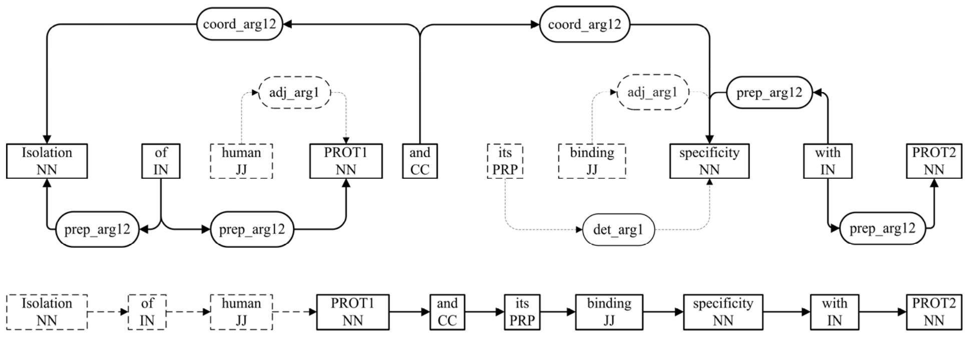
"Isolation of human PROT1 and its binding specificity with PROT2"

In our method, the syntactic and word information of the sentence used in the method of Airola et al. (2008) are extracted as our original raw features (the structural representation of the example sentence is shown in Figure 2). Enju is applied to parse the sentences and provide us with phrase structures and predicate-argument structures (Miyao and Tsujii, 2008).

Word features used in our method are all the words of this sentence, such as 'human', 'its', 'binding', and syntactic features are the parts of speech of the words and the semantic relationships of the predicate-argument pairs (e.g. 'NN', 'JJ', and 'coord_arg12').

The words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding their relationships (Bunescu et al., 2005). Therefore, we use the same weight assignment mechanism as the one in Airola et al. (2008) to distinguish these features. The features in the shortest paths and the features between two entities in the linear order are initialised to 0.9, while the rest are initialised to 0.3. Once a repeated feature is found in one sentence, the weight will be multiplied by a coefficient (which is set to 1.1 in our method) since, instinctively, a feature which occurs two or more times in a sentence is more important than a feature that occurs only once.

Figure 2 Structural representation of the example sentence. The top sub-graph is the predicate-argument structure of the example sentence and the bottom sub-graph shows the linear order of the sentence. The nodes and edges in the shortest path connecting two proteins in predicate-argument structure and the nodes and edges between two proteins in linear order graph are shown in bold and the rest are shown in dotted lines. The rectangular nodes represent words and corresponding parts of speech. The ellipse vertices represent specific relationships between the predicates and their arguments.



Initially, a feature set with a dimension size larger than three million is obtained. However, considering either time complexity or space complexity, it's hard to train a deep multilayer neural network with such an input layer size. Therefore, the following two steps are taken to reduce the input space size. First, document frequency-based feature selection is performed (Liu et al., 2003) and 8582 features are retained. Second, PCA is applied, in which 96% of the variance are preserved. Finally, a feature space with a size of 965 is obtained.

2.3 Auto-encoder

In the next step of our method, auto-encoder, a non-linear unsupervised learning model, is applied to learn more rich features by unsupervised greedy layer-wise training on the raw features. Auto-encoder is a neural network trained to compute a representation of the input from which it can be reconstructed with as much accuracy as possible (Hinton and Zemel, 1994). It can also be seen as the parameter initialisation of the Deep Neural Networks (DNNs). As described in detail previously (Larochelle et al., 2009), though auto-encoder is designed with the goal of dimensionality reduction, the new representation's dimensionality does not necessarily need to be lower than the input's dimensionality in practice. We will consider the auto-encoder network with only one hidden layer, in which the hidden representation of *x* is a code *h(x)* obtained from the encoding function:

$$h_j(x) = f\left(a_j(x)\right) \text{ where } a_j(x) = b_j + \sum\nolimits_i W_{ji} x_i \tag{1}$$

The input's reconstruction is obtained from a decoding function, here a linear transformation of the hidden representation with the weight matrix w^*, possibly followed by a non-linear activation function:

$$x_k^* = g(a_k) \text{ where } a_k = c_k + \sum\nolimits_j W_{kj}^* h_j(x) \tag{2}$$

The sigmoid activation function is used for both *f*( ) and *g*( ), and b_j and c_k are the bias terms. We set $W^T = W^*$ in all of our experiments to avoid the network learning a trivial identity function. The network with small weights W_{ji} between the input and hidden layers and large weights W_{kj}^* between the hidden and output layers could encode such an unwished identity function (Larochelle et al., 2009).

Learning an auto-encoder network is almost the same as training a standard artificial neural network. When the loss function is given, the back-propagation algorithm is used (described in the following section). As the target of the auto-encoder is minimising the reconstruction error, the loss function can be defined as Formula 3 where weight decay term is used to penalise large weights:

$$L\left(x^*, x\right) = \sum_n \left(x^{(n)^*} - x^{(n)}\right)^2 + \frac{\beta}{2} W^2 \tag{3}$$

2.4 Back-propagation neural networks

In our method, back-propagation method is used to train both auto-encoders and the whole DNN. Back propagation is a common method of training artificial neural networks used in conjunction with an optimisation method such as gradient descent (Rumelhart et al., 1988; Werbos, 1994). Learning a back-propagation neural network is implemented in two phases (i.e. feed forward propagation and error back propagation, as shown in Figure 3), which is the general case of a multilayer neural network in which auto-encoder contains one hidden layer and the output unit size is the same as the input unit size. For convenience, the bias parameters associated with each hidden and output units are absorbed into the set of weight parameters by defining additional variables x_0, $z_0^{(1)}$ and $z_0^{(2)}$ whose values are clamped at 1 (Bishop, 2006).

2.4.1 Feed forward propagation

Firstly, each hidden and output unit value is calculated layer by layer with the function given by Formula 4.

$$\mathrm{z_k} = h\left(a_k\right) \text{ where } a_k = \sum\nolimits_j W_{kj} z_j \tag{4}$$

Where *h*( ) is an activation function. Logistic sigmoid function, defined as Formula 5, is used as the activation function throughout our experiments.

$$Sigmoid\left(x\right) = \frac{1}{1 + \exp\left(-x\right)} \tag{5}$$

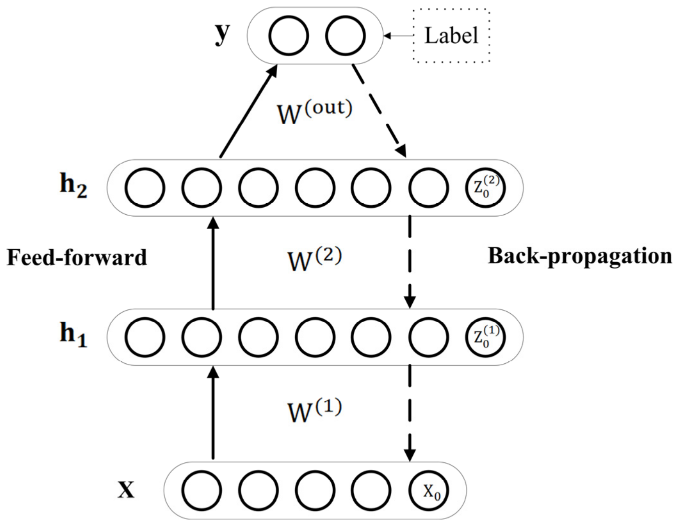
Then Formulas 4 and 5 are combined to give the overall network function which takes the form as follows:

$$\mathrm{y}_k\left(x, W\right) = \sigma\left(\sum_m W_{km}^{(out)} h\left(\sum_j W_{mj}^{(2)} h\left(\sum_i W_{ji}^{(1)} x_i\right)\right)\right) \tag{6}$$

where $\sigma(\ )$, the activation function of the output unit, is also the sigmoid function in our work, while it can be an identity function for standard regression problems. Then loss function of this neural network is derived from Formula 6. We choose the cross-entropy, defined as Formula 7, as our loss function since we treat PPI extraction as a classification task. Similar to Formula 3, the weight decay penalty is also used in Formula 7.

$$L(W) = \sum_{n} \left\{ y^{(n)} \ln\left(y\left(x^{(n)}, W\right)\right) + \left(1 - y^{(n)}\right) \ln\left(1 - y\left(x^{(n)}, W\right)\right) \right\} + \frac{\beta}{2} W^2 \tag{7}$$

Figure 3 Back-propagation algorithm with early stopping criteria. A back-propagation neural network with two hidden layers and two output units, where x is the input of the network, h_1 and h_2 are the hidden layers, $W^{(1)}$, $W^{(2)}$, and $W^{(\text{out})}$ are the parameters between the corresponding two layers, and y is the output of the network. $Z_0^{(1)}$ and $Z_0^{(2)}$ are the bias parameters associated with each hidden and output units.



2.4.2 Error back propagation

After the loss function $L(W)$ is defined in the previous step, we calculate the gradient of $L(W)$ with respect to the weight parameters W and propagate it back layer by layer.

First of all, error information of the output units $\delta_k^{(out)}$ is evaluated by Formula 8 where y_k is the label information, where $k = 1, 2$. Later, this error signal will be back propagated using Formula 9, where $\delta_i^{(l)}$ stands for the error information of the ith hidden unit in the lth layer.

$$\delta_k^{(out)} = \sigma'\left(a_k^{(out)}\right)\left(y_k(x, W) - y_k\right) \tag{8}$$

$$\delta_i^{(l)} = h'\left(a_i^{(l)}\right) \sum_{j} W_{ji}^{(l+1)} \delta_j^{(l+1)} \tag{9}$$

where $\sigma'(x) = h'(x) = sigmoid(x)(1 - sigmoid(x))$ since $\sigma(\cdot) = h(\cdot) = sigmoid(\cdot)$. Then the required gradient is obtained simply by multiplying the value of δ for the unit at the output end of the weight by the value of *z* for the unit at the input end of the weight as shown in Formula 10.

$$\frac{\partial L(W)}{\partial W_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \tag{10}$$

where the superscript *l* represents the layer in the network. At last, weight parameters are updated by Formula 11.

$$W_{ji}^{(l)}(\tau+1) = W_{ji}^{(l)}(\tau) - \eta \frac{\partial L(W)}{\partial W_{ji}^{(l)}} \tag{11}$$

where η is the learning rate which can adjust the converge speed. Forward propagation and backward propagation will be executed in turn until the early stopping condition is satisfied as described in Figure 4.

Figure 4 Back-propagation algorithm with early stopping criteria

```
Algorithm 1. Back-propagation algorithm with early stopping criteria

Input: train data set T; validation data set D.
T={$(x_t^{(1)}, y_t^{(1)}), (x_t^{(2)}, y_t^{(2)}), \ldots, (x_t^{(n)}, y_t^{(n)})$}; D={$(x_v^{(1)}, y_v^{(1)}), (x_v^{(2)}, y_v^{(2)}), \ldots, (x_v^{(m)}, y_v^{(m)})$};
$x_t^{(i)}, x_v^{(j)} \in R^s$ and $y_t^{(i)}, y_v^{(j)} \in \{0,1\}$ where $i = 1,2,\ldots,n$; $j = 1,2,\ldots,m$; $s = 965$

1) Initialize the parameters $W^{(1)}, W^{(2)}, \ldots, W^{(out)}$ to the small random values near zero.
2) Apply an input vector $\mathbf{x}_t$ to the network and forward propagate through the
   network using (4) and (5) to find the activations of all the hidden and output units.
3) Evaluate the $\delta_k^{(out)}$ for all the output units using (8).
4) Back-propagate the $\delta^{(out)}$ using (9) to obtain $\delta_i^{(l)}$ for each hidden unit in the network.
5) Use (10) to evaluate the required derivatives.
6) Update $W^{(1)}, W^{(2)}, \ldots, W^{(out)}$ by means of formula (11).
7) Test the model on D. If the validation error begins to rise, then stop training,
   otherwise, if the validation error keeps going down, then go to step 2.

Output: well trained neural network model {$W^{(1)}, W^{(2)}, \ldots, W^{(out)}$}
```

2.4.3 Back-propagation algorithm with early stopping criteria

Back-propagation algorithm is the fundamental algorithm in our approach since it is used in the process of greedy layer-wise training of auto-encoders and the fine-turning phase. Early stopping is applied to prevent the neural networks from over-fitting.

2.5 Deep neural network algorithm

The detailed DNN algorithm is shown in Figure 5 which contains unsupervised pre-training phase and supervised fine-tuning phase.

Figure 5 Deep neural network algorithm

Algorithm 2. Deep neural network algorithm

Input: train data set *T*; validation data set *D*; unlabeled data set *U*.

T = $\{ (x_t^{(1)}, y_t^{(1)}), (x_t^{(2)}, y_t^{(2)}), \ldots, (x_t^{(n)}, y_t^{(n)}) \}$; D = $\{ (x_v^{(1)}, y_v^{(1)}), (x_v^{(2)}, y_v^{(2)}), \ldots, (x_v^{(m)}, y_v^{(m)}) \}$

U = $\{ x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{r} \}$; $x_t^{(i)}, x_v^{(j)}, x_u^{k} \in R^s$ and $y_t^{(i)}, y_v^{(j)} \in \{0,1\}$ where $i = 1,2,\ldots,n$; $j = 1,2,\ldots,m$; $k = 1,2,\ldots,r$; $s = 965$

1) Train an auto-encoder on *U* as shown in Figure 6-a.
2) Train an auto-encoder based on h_{i-1} to gain the higher level code h_i, *i*={2,3,…,L} where *L* is the top hidden layer number (as shown in Figure 6-b).
3) Put a logistic regression classifier over the stacked auto-encoders as the output layer as shown in Figure 6-c.
4) Train this back-propagation neural network based on *T* and *D*. Where the weight parameters are initialized by $\{W^{(1)}, W^{(2)}, \ldots\}$ which are the results of step 1 and 2.

Output: fine-turned neural network model $\{ W_{fine}^{(1)}, W_{fine}^{2}, \ldots W_{fine}^{(out)} \}$

3 Experimental results and discussion

3.1 Experimental data sets and settings

Table 1 The details of five PPI corpora

Corpus	*#Positive*	*#Negative*	*#Example*	*I/EP*
AImed	1000	4655	5655	0.18
BioInfer	2473	6367	8840	0.28
IEPA	335	476	811	0.41
HPRD50	163	270	433	0.38
LLL	164	166	330	0.5

Our method was evaluated on five publicly available corpora: AImed, BioInfer, IEPA, HPRD50 (Fundel et al., 2007), and LLL (Nédellec et al., 2005). The statistics of the five PPI corpora is shown in Table 1. The sentences which can't be correctly parsed by Enju are removed from the corpora and thus not considered in evaluation. Here, it is noted that the I/EP denotes the average number of interactions per sentence divided by the average number of entity pairs per sentence. According to Pyysalo et al. (2008), the I/EP may serve as a rough, easily computable baseline for determining the comparative difficulty of PPI extraction tasks across corpora (which is inversely related to the I/EP value). Therefore, it is more difficult to achieve good performance on AImed (with an I/EP of 0.18) and BioInfer (with an I/EP of 0.28) corpora than on the other three corpora.

For the unlabelled data used in the pre-training process, we combined the five PPI corpora and removed the label information to obtain the unlabelled data (16,909 instances in total). We also downloaded all the Medline abstracts, ranging from 1994 to 2009, to generate additional unlabelled data. Protein/gene names in these abstracts are tagged by our NER system. Then the sentences containing at least two protein names are retained, since we are concerned with the interactions between two proteins. Finally, original raw features are extracted from the unlabelled data.

Results are reported in the document-level tenfold cross-validation scheme, which is the de facto standard in PPI extraction. To apply early stopping criteria in our experiments, we took onefold from the ninefold train set as the validation set. Therefore, the proportion of train, validation and test sets is 8:1:1.

Training deep architectures requires determining appropriate hyper-parameters. For the raw neural network (NN), the hyper-parameters are the number of units per layer, the learning rate and the epochs of iteration. The DNN has an additional hyper-parameter, the learning rate for training auto-encoder. Larochelle et al. (2009) suggested that the same size for all the hidden layers worked generally better or the same as using a decreasing size or increasing size. And they also recommended that the number of units in first hidden layer should be more than units in input layer. Considering both accuracy and efficiency, we chose the topology structure of 965-1000-1000-…-1000-1. The number of epochs of supervised phase, NN and fine-tuning in DNN is chosen by early stopping which is based on the progression of classification error on the validation set. We take 10,000 epochs for the first four layers and 20,000 epochs for the remaining four layers when training auto-encoders. The learning rate is determined by the form $\eta_t = \frac{\eta_0 \tau}{\max(t, \tau)}$ (Bengio, 2012) which keeps the learning rate constant for the first τ steps and then decreases it in $O(1/t)$. We set $\eta_0 = 0.001$ and $\tau = \infty$, constant learning rate, when training auto-encoders and set $\eta_0 = 0.001$ and $\tau = 70$ for the supervised phase. All these hyper-parameters are determined in the light of the performance on the validation set.

3.2 Experimental results

In this section, we provide a comprehensive evaluation of our method, and compare our results with those of other methods. Like the majority of PPI extraction system evaluations, we use the balanced F-score measure for quantifying the performance of the systems. This metric is defined as $F = (2PR)/(P + R)$, where P denotes precision and R recall.

3.2.1 Performance comparison with NN

In our experiments, performance comparison between DNN and NN with the same architectures is made. The main difference between DNN and NN is the parameter initialisation. Instead of assigning random small values to the weights in NN, the DNN weights are generated by pre-training with auto-encoder. The network layers are limited to five since the performance of NN drops sharply when the depth exceeds four. The reason is that training deep architectures involves optimisation problems that are more difficult than those involved in training shallow architectures (Bengio, 2009). This is proved by the results shown in Table 2. With the increase of the network layers, the performance of NN gradually degrades on all the corpora. The ones of NN5 become poor compared with those of other layers. By contrast, the performance of DNN increases in most cases when the network layer increases from three to four. When the network layer further increases to five, the performance remains stable. The reason is that deep architectures yield an error surface that is non-convex and hard to optimise, with the suspected presence of many local minima. A gradient-based optimisation should thus end in the local minimum of whatever basin of attraction we started from. However, pre-training procedure will put us in a region of parameter space where basins of attraction run deeper than when picking starting parameters at random (Erhan et al., 2009). It’s the reason that pre-training can help to learn a deep architecture.

NN-3 denotes the NN with three layers; DNN-3 denotes the DNN with three layers, and so on. Δ denotes the average performance improvement of DNN-4 over NN-3 (percentage unit); the highest F-scores are shown in bold. In addition, the increase of the network depth will lead to the increase of the network capacity (Hoekstra and Duin, 1995), which roughly corresponds to the ability to model any given function. Therefore, in most cases, DNN performs better when the network layer increases from three to four.

By comparing the best performance of NN-3 (with three layers) and DNN-4 (with four layers), we found that, on two ‘toughest handling’ (with the least I/EP values) corpora AImed and BioInfer, DNN-4 outperforms NN-3 with the improvements of 3.10 and 2.89 percentage units in F-score, respectively. On the IEPA and HPRD50, DNN-4 and NN-3 achieve almost equal performance. Only on the ‘easiest handling’ corpus LLL, the performance of DNN-4 is inferior to that of NN-3 by 1.23 percentage units in F-score. The reason is that the pre-training procedure with auto-encoder can improve the performance by learning more rich features automatically when the problems, like the PPI extraction on AImed and BioInfer corpora, are complex enough (Larochelle et al., 2007). However, on the simple problem like the PPI extraction on IEPA, HPRD50 and LLL, DNN does not work well since it may over-fit when a complicated model is used to learn an easy problem (Wan et al., 2013) as will be discussed in the following section.

3.2.2 The effect of the network depth on performance

A theoretical motivation for deep architectures comes from the complexity theory: when a function can be represented compactly with an architecture of depth *k*, representing it with an architecture of depth *k*-1 might require an exponential size architecture (Håstad and Goldmann, 1991). In other words, deep architectures can be much more efficient (sometimes exponentially) than shallow architectures, in terms of computational elements required to represent some functions. Therefore, we test the DNN depths from three to ten to explore the effect of the network depth on our method's performance. The results on five different corpora are shown in Figure 6. The *F*-score curves on all corpora except LLL will ascend in the beginning and then decline after they reach the peaks, while, for LLL, the curve fluctuates all the time. To explain the reason, we introduce the concept of discrimination capacity (DC) (Hoekstra and Duin, 1995), the maximum number of regions a network can discriminate. DC roughly corresponds to the ability to model any given function. It is related to the amount of information that can be stored in the network and, therefore, will increase along with the number of the hidden units or the depths. Analysing Figure 5 from the view of DC, we can find that the performance of DNN improves along with the increase of network depth, since the network with larger DC can learn more complicated problems. However, it will over-fit easily when a complicated model is used to learn an easy problem (Wan et al., 2013). The reason that the performance of the DNN model begins to decrease when the network depth exceeds a certain threshold is that it may over-fit after that threshold. For LLL, the 'easiest handling' corpus, according to I/EP theory, it might over-fit from the very beginning.

Table 2 Performance comparison with NN

Corpus		*NN-3*	*NN-4*	*NN-5*	*DNN-3*	*DNN-4*	*DNN-5*	*Δ*
AImed	*P*	48.21	43.37	20.26	50.27	51.51	50.2	
	R	64.12	66.91	89.79	62.05	63.38	65.83	
	F	54.43	51.81	32.59	54.77	**56.12**	56.08	3.10
	σ_F	5.31	5.66	3.82	5.23	4.9	5.3	
BioInfer	*P*	52.09	48.78	28.46	55.6	55.66	53.89	
	R	70.22	69.14	99.86	65.56	68.71	72.9	
	F	59.54	56.54	44.06	59.5	61.26	**61.63**	2.89
	σ_F	3.06	5.36	6.31	3.61	3.26	3.53	
IEPA	*P*	66.95	43.37	45.01	71.84	68.7	68.74	
	R	83.98	97.04	95.27	79.37	83.48	83.86	
	F	**74.23**	59.81	60.03	**74.22**	74.19	74.14	–0.05
	σ_F	6.0	3.73	5.07	5.93	5.61	5.67	
HPRD50	*P*	61.44	68.25	40.75	66.27	58.72	58.93	
	R	88.53	79.03	93.75	81.1	92.37	92.02	
	F	71.11	70.15	55.73	70.16	**71.28**	70.99	0.24
	σ_F	9.45	10.91	11.98	9.38	9.19	8.75	
LLL	*P*	75.84	72.08	55.47	80.57	80.65	75.96	
	R	91.81	95.32	100	83.48	84.36	90.97	
	F	**82**	80.72	70.15	81.34	80.99	**81.36**	–1.23
	σ_F	13.51	13.87	13.19	11.7	12.62	12.51	

3.2.3 The effect of the unlabelled data size on performance

Since the pre-training of DNN plays an important role and works in an unsupervised fashion, it is necessary to explore the effects of reinforcing pre-training with the introduction of more unlabelled data. The pre-training can be viewed as an unsupervised feature learning approach. When the neural network becomes deeper, the feature level becomes higher (Cai et al., 2013). So a large amount of unlabelled data is expected to contribute to the feature detector. To explore the effect of the unlabelled data set size on our method's performance, the corresponding experiments are conducted. As discussed in the previous section, the initial size of the unlabelled data we used is 16,909 instances (achieved by removing the label information of the five PPI corpora), and we added additional unlabelled data of 16,000, 32,000, and 320,000 instances (obtained from downloaded Medline abstracts), respectively. The experiments are performed on all five PPI corpora with various network depths and the results are shown in Table 3. In most cases, the introduction of additional unlabelled data can improve the performance: when 320,000 new instances are added, an average *F*-score improvement of about one percentage unit over all corpora can be achieved from network depth three to five. However, the performance improvements on these corpora are different. For example, the performance on HPRD50 keeps improving with the introduction of more unlabelled data, while those on other corpora may reach the peak before all the unlabelled data (320,000 instances) are added. This may be due to the different similarity between the unlabelled data and the PPI corpora. In fact, the properties among the five PPI corpora are quite different not only from the statistics but also from the entity types, interaction types, etc. (Pyysalo et al., 2008). For example, approximately 5% of interactions in the AImed corpus are self-interactions (a single protein interacting with itself), and essentially no self-interactions occur in the other corpora.

Figure 6 Performance with different network depths. The horizontal axe represents the network depth and vertical axe for *F*-score (see online version for colours)

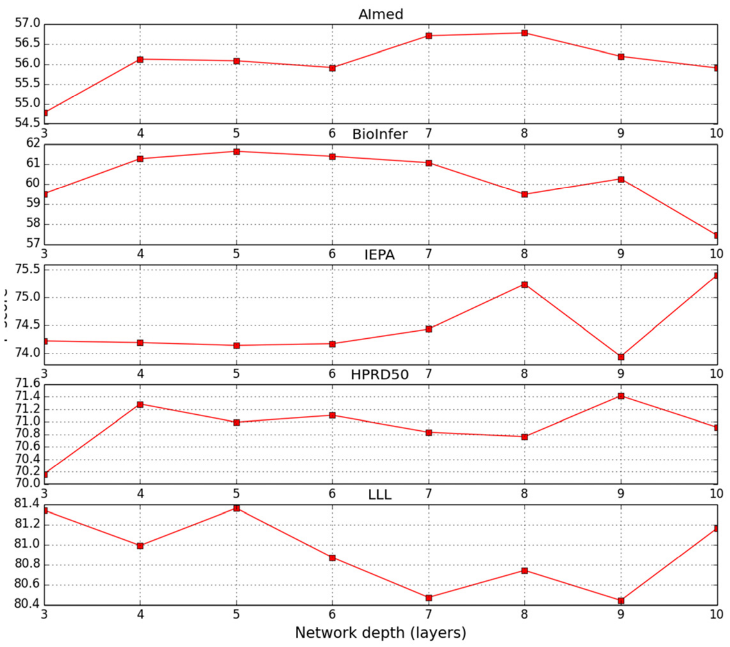


Table 3 Performance with different unlabelled data set sizes

Corpus	*DNN-3*				*DNN-4*				*DNN-5*			
	16,909	*+16,000*	*+32,000*	*+320,000*	*16,909*	*+16,000*	*+32,000*	*+320,000*	*16,909*	*+16,000*	*+32,000*	*+320,000*
AImed	54.77	55.12	55.17	**55.34**	56.12	55.97	**56.32**	55.91	56.08	55.74	55.71	**56.33**
Bioinfer	59.5	60.51	**60.77**	60.58	61.26	**61.5**	61.23	61.11	61.63	**61.98**	61.37	61.31
IEPA	74.22	73.85	**74.5**	74.29	74.19	75.41	75.18	**75.8**	74.14	75.53	75.18	**76.02**
HPRD50	70.16	71.36	**71.77**	71.49	71.28	71.36	71.33	**73.53**	70.99	72.85	72.77	**73.15**
LLL	81.34	**81.7**	81.43	80.66	80.99	80.91	81.11	**81.52**	81.36	81.12	80.23	**81.69**
Avg.△				**0.786**				**1.14**				**1.18**

Note: Avg. Δ denotes the average F-score improvement on five PPI corpora when 320,000 additional unlabelled instances are added; the highest F-scores achieved with a certain number of additional unlabelled instances are shown in bold.

In addition, HPRD50 annotates direct physical interactions, regulatory relations and physical modifications, while AImed has no specific types. According to Pyysalo et al. (2008), there is over 10% average PPI extraction performance difference between corpora.

Instinctively, more similarly, the properties of the unlabelled data set are to those of a corpus, and more likely it will help improve the DNN's performance on the corpus. The reason is that when the amount of the training data is increased, the training error converges to the generalisation error. In the extreme case, the training error is just the generalisation error when we trained on the complete data set in the world. Therefore, adding additional unlabelled data improves the generalisation capability of auto-encoder and this leads to better performance of DNN (Cai et al., 2013). From this point of view, it can be concluded that the additional unlabelled data may have the highest similarity with HPRD50, since its performance improves all the way with the introduction of more unlabelled data and its *F*-score value is boosted most among the five corpora. By contrast, in some cases, the introduction of more unlabelled data will no longer help improve the PPI extraction performance on the other corpora.

3.2.4 Performance comparison with all-paths graph (APG) kernel approach

In our method, we use the features similar to those used in APG (Airola et al., 2008). Therefore, the performance comparison with APG is made. The similarities and differences between these two methods are listed as follows:

1 Both methods use the parsed graph information as features and assign larger weight (0.9) to the features in the shortest paths connecting two proteins.

2 Both methods extract the linear order graph information and assign larger weight (0.9) to the features between the two proteins.

3 The method of Airola et al. (2008) considers all the paths connecting two proteins but our method only uses the shortest path.

4 Our method applies dimension compressing with PCA and by filtering low-frequency features, while that of Airola et al. (2008) does not.

5 The method of Airola et al. (2008) constructs the graph kernel, while our method uses the flat features only.

Compared to APG, our features contain smaller chunks of information, as we consider only the shortest path connecting two proteins but APG uses all the paths. Our method also loses some information in dimension-reduction processes. In addition, APG is the kernel method, which can efficiently compute the similarity between structure data, such as a dependency graph or a syntactic parse tree, while ours uses the flat features only. It stands to reason that the performance of our method will be inferior to that of APG instinctively. However, as shown in Table 4, our DNN-4 model outperforms AGP on two 'toughest handling' corpora, i.e., AImed and BioInfer. This verifies the effectiveness of our method on tough handling problems once again.

Table 4 Performance comparison with APG

Corpus	*DNN-4*				*APG*			
	P	*R*	*F*	σ_F	*P*	*R*	*F*	σ_F
AImed	51.51	63.38	56.12	4.9	49.15	64.6	55.33	4.9
BioInfer	55.66	68.71	61.26	3.3	53.33	70.14	59.96	2.4
IEPA	68.7	83.48	74.19	5.6	70.57	82.21	74.99	5.1
HPRD50	58.72	92.37	71.28	9.2	73.98	80.85	76.5	5.8
LLL	80.65	84.39	80.99	12.6	79.88	88.23	82.66	11.1

4 Conclusions

In this paper, we present a deep learning-based approach to extract PPIs from biomedical literature. The approach can learn complex and abstract features at higher layers representation by greedy layer-wise training auto-encoders with large amount of unlabelled data. The unlabelled data, in real-world applications, is often much more abundant and cheaper compared to the labelled data.

The experimental results show that our approach can achieve better performance than NN. In addition, performance comparison with APG also verifies the effectiveness of our approach. We also found that the appropriate increase of the network depth can improve the performance of DNN, since the network with larger DC can learn more complicated problems. Besides, in most cases, introducing more unlabelled data into the unsupervised training phase case can improve the performance.

However, as the first attempt to introduce deep learning into the biomedical literature information extraction field, compared with the performance of deep learning in other areas, such as computer vision and speech recognition, there is still much room for improvement. For example, our original raw features are derived from the output of the NER system and Enju parser, which may lead to the error propagation of existing NLP systems and hinder the performance (Bach and Badaskar, 2007). Therefore, in the future, we will explore a word embedding-based deep learning method (Collobert et al., 2011; Bengio et al., 2003; Mikolov et al., 2013), since the word embeddings are learned automatically from large amounts of unlabelled data without using any NLP tools. In addition, our work focuses on the binary PPIs extraction from biomedical literature and biomedical event extraction (Ananiadou et al., 2010) can link pathways to literature evidence and aid pathway construction and enrichment (Dai et al., 2010). Therefore, we will introduce our DNN method into the biomedical event extraction task.

Acknowledgements

This work is supported by the grants from the Natural Science Foundation of China (Nos. 61070098, 61272373, 61340020, 61572102 and 61572098), Trans-Century Training Program Foundation for the Talents by the Ministry of Education of China (NCET-13-0084) and the Fundamental Research Funds for the Central Universities (Nos. DUT13JB09 and DUT14YQ213).

References

Airola, A., Pyysalo, S., Björne, J., Pahikkala, T., Ginter, F. and Salakoski, T. (2008) 'All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning', *BMC Bioinformatics*, Vol. 9, p.S2.

Ananiadou, S., Pyysalo, S., Tsujii, J. and Kell, D.B. (2010) 'Event extraction for systems biology by text mining the literature', *Trends in Biotechnology*, Vol. 28, No. 7, pp.381–390.

Bach, N. and Badaskar, S. (2007) 'A review of relation extraction', *Literature Review for Language and Statistics II*, Carnegie Mellon University, Pittsburgh, PA.

Bader, G.D., Donaldson, I., Wolting, C., Ouellette, B.F.F., Pawson, T. and Hogue, C.W.V. (2001) 'BIND – the biomolecular interaction network database', *Nucleic Acids Research*, Vol. 29, No. 1, pp.242–245.

Bengio, Y. (2009) 'Learning deep architectures for AI', *Foundations and Trends® in Machine Learning*, Vol. 2, No. 1, pp.1–127.

Bengio, Y. (2012) 'Practical recommendations for gradient-based training of deep architectures', *Neural Networks: Tricks of the Trade*, Springer, Berlin, pp.437–478.

Bengio, Y. (2013) 'Deep learning of representations: looking forward', *Lecture Notes in Computer Science*, Vol. 7978, pp.1–37.

Bengio, Y., Courville, A. and Vincent, P. (2013) 'Representation learning: a review and new perspectives', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 8, pp.1798–1828.

Bengio, Y., Ducharme, R., Vincent, P. and Janvin, C. (2003) 'A neural probabilistic language model', *The Journal of Machine Learning Research*, Vol. 3, pp.1137–1155.

Bishop, C.M. (2006) *Pattern Recognition and Machine Learning*, Vol. 1, Springer, New York.

Blaschke, C. and Valencia, A. (2002) 'The frame-based module of the SUISEKI information extraction system', *IEEE Intelligent Systems*, Vol. 17, No. 2, pp.14–20.

Bunescu, R.C., Ge, R., Kate, R.J., Marcotte, E.M., Mooney, R.J., Ramani, A.K. and Wong, Y.W. (2005) 'Comparative experiments on learning information extractors for proteins and their interactions', *Artificial Intelligence in Medicine*, Vol. 33, No. 2, pp.139–155.

Bunescu, R.C. and Mooney, R.J. (2005) 'A shortest path dependency kernel for relation extraction', *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Vancouver, BC, pp.724–731.

Cai, M., Zhang W-Q. and Liu, J. (2013) 'Improving deep neural network acoustic models using unlabeled data', *2013 IEEE China Summit & International Conference on Signal and Information Processing (ChinaSIP)*, 6–10 July, IEEE, Beijing, pp.137–141.

Collobert, R. and Weston, J. (2008) 'A unified architecture for natural language processing: deep neural networks with multitask learning', *Proceedings of the 25th International Conference on Machine Learning*, ACM, New York, pp.160–167.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011) 'Natural language processing (almost) from scratch', *The Journal of Machine Learning Research*, Vol. 12, pp.2493–2537.

Corney, D.P., Buxton, B.F., Langdon, W.B. and Jones, D.T. (2004) 'BioRAT: extracting biological information from full-length papers', *Bioinformatics*, Vol. 20, No. 17, pp.3206–3213.

Cristianini, N. and Shawe-Taylor, J. (2000) *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge.

Dai, H.J., Chang, Y.C., Tsai, R.T.H. and Hsu, W.L. (2010) 'New challenges for biological text-mining in the next decade, *Journal of Computer Science and Technology*, Vol. 25, No. 1, pp.169–179.

Ding, J., Berleant, D., Nettleton, D. and Wurtele, E. (2002) 'Mining MEDLINE: abstracts, sentences, or phrases', *Proceedings of the Pacific Symposium on Biocomputing*, World Scientific, Kauai, Hawaii, pp.326–337.

Erhan, D., Manzagol, P-A., Bengio, Y., Bengio, S. and Vincent, P. (2009) 'The difficulty of training deep architectures and the effect of unsupervised pre-training', *International Conference on Artificial Intelligence and Statistics*, Clearwater Beach, FL, USA, pp.153–160.

Fundel, K., Küffner, R. and Zimmer, R (2007) 'RelEx – relation extraction using dependency parse trees', *Bioinformatics*, Vol. 23, No. 3, pp.365–371.

Graves, A., Mohamed, A-R. and Hinton, G. (2013) 'Speech recognition with deep recurrent neural networks', *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, pp.6645–6649.

Håstad, J. and Goldmann, M. (1991) 'On the power of small-depth threshold circuits', *Computational Complexity*, Vol. 1, No. 2, pp.113–129.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A-R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P. and Sainath, T.N. (2012a) 'Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups', *Signal Processing Magazine*, Vol. 29, No. 6, pp.82–97.

Hinton, G.E., Osindero, S. and the, Y-W. (2006) 'A fast learning algorithm for deep belief nets', *Neural Computation*, Vol. 18, No. 7, pp.1527–1554.

Hinton, G.E. and Salakhutdinov, R.R. (2006) 'Reducing the dimensionality of data with neural networks', *Science*, Vol. 313, No. 5786, pp.504–507.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R.R. (2012b) 'Improving neural networks by preventing co-adaptation of feature detectors', *arXiv preprint arXiv:12070580*, Cornell University Library, Ithaca, NY.

Hinton, G.E. and Zemel, R.S. (1994) 'Autoencoders, minimum description length, and Helmholtz free energy', In Cowan, J., Tesauro, G. and Alspector, J. (Eds): *Advances in Neural Information Processing Systems*, Morgan Kaufmann Publishers, San Francisco, CA.

Hoekstra, A. and Duin, R. (1995) 'Exploring the capacity of simple neural networks', *Proceedings of the First Annual Conference of the ASCI*, Citeseer, 16–18 May, Heijen, NL, pp.56–62.

Jolliffe, I. (2005) *Principal Component Analysis*, Wiley Online Library, New York.

Kim, S., Yoon, J. and Yang, J. (2008) 'Kernel approaches for genic interaction extraction', *Bioinformatics*, Vol. 24, No. 1, pp.118–126.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) 'ImageNet classification with deep convolutional neural networks', *Neural Information Processing Systems*, Lake Tahoe, Nevada.

Larochelle, H., Bengio, Y., Louradour, J. and Lamblin, P. (2009) 'Exploring strategies for training deep neural networks', *The Journal of Machine Learning Research*, Vol. 10, pp.1–40.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J. and Bengio, Y. (2007) 'An empirical evaluation of deep architectures on problems with many factors of variation', *Proceedings of the 24th International Conference on Machine Learning*, ACM, New York, pp.473–480.

Li, Y., Lin, H. and Yang, Z. (2009) 'Incorporating rich background knowledge for gene named entity classification and recognition', *BMC Bioinformatics*, Vol. 10, No. 1, p.223.

Liu, T., Liu, S., Chen, Z. and Ma, W-Y. (2003) 'An evaluation on feature selection for text clustering', *ICML*, Atlanta, Georgia, USA, pp.488–495.

Mikolov, T., Yih, W-T. and Zweig, G. (2013) 'Linguistic regularities in continuous space word representations', *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (*HLT-NAACL)*, Association for Computational Linguistics, Atlanta, GA, pp.746–751.

Miwa, M., Sætre, R., Miyao, Y. and Tsujii, J. (2009) 'Protein–protein interaction extraction by leveraging multiple kernels and parsers', *International Journal of Medical Informatics*, Vol. 78, No. 12, pp.e39–e46.

Miyao, Y. and Tsujii, J. (2008) 'Feature forest models for probabilistic HPSG parsing', *Computational Linguistics*, Vol. 34, No. 1, pp.35–80.

Mooney, R.J. and Bunescu, R.C. (2005) 'Subsequence kernels for relation extraction', *Advances in Neural Information Processing Systems*, Vol. 18, pp.171–178.

Moschitti, A. (2006) 'Making tree kernels practical for natural language learning', *EACL*, pp.113–120. Available online at: http://www.aclweb.org/anthology/E06-1015

Nédellec, C. (2005) 'Learning language in logic – genic interaction extraction challenge', *Proceedings of the 4th Learning Language in Logic Workshop (LLL05)*, Bonn, Germany, pp.31–37.

Ono, T., Hishigaki, H., Tanigami, A. and Takagi, T. (2001) 'Automated extraction of information on protein–protein interactions from the biological literature', *Bioinformatics*, Vol. 17, No. 2, pp.155–161.

Pyysalo, S., Airola, A., Heimonen, J., Björne, J., Ginter, F. and Salakoski, T. (2008) 'Comparative analysis of five protein-protein interaction corpora', *BMC Bioinformatics*, Vol. 9, p.S6.

Pyysalo, S., Ginter, F., Heimonen, J., Björne, J., Boberg, J., Järvinen, J. and Salakoski, T. (2007) 'BioInfer: a corpus for information extraction in the biomedical domain', *BMC Bioinformatics*, Vol. 8, No. 1, p.50.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1988) *Learning Representations by Back-Propagating Errors*, MIT Press, Cambridge, MA.

Sætre, R., Sagae, K. and Tsujii, J. (2007) 'Syntactic features for protein-protein interaction extraction', *Journal of Chinese Information Processing*, Vol. 27, No. 1, pp.86–85.

Wan, L., Zeiler, M., Zhang, S., Cun, Y.L. and Fergus, R. (2013) 'Regularization of neural networks using dropConnect', *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, Atlanta, Georgia, USA, pp.1058–1066.

Werbos, P.J. (1994) *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, Vol. 1, John Wiley & Sons, New York.

Xenarios, I., Rice, D.W., Salwinski, L., Baron, M.K., Marcotte, E.M. and Eisenberg, D. (2000) 'DIP: the database of interacting proteins', *Nucleic Acids Research*, Vol. 28, No. 1, pp.289–291.

Xiao, J., Su, J., Zhou, G.D. and Tan, C. (2005) 'Protein–protein interaction extraction: a supervised learning approach', *Proceedings of the Symposium on Semantic Mining in Biomedicine*, European Bioinformatics Institute, Hinxton, UK, pp.51–59.

Yang, Z., Tang, N., Zhang, X., Lin, H., Li, Y. and Yang, Z. (2011) 'Multiple kernel learning in protein–protein interaction extraction from biomedical literature', *Artificial Intelligence in Medicine*, Vol. 51, No. 3, pp.163–173.

Zanzoni, A., Montecchi-Palazzi, L., Quondam, M., Ausiello, G., Helmer-Citterich, M. and Cesareni, G. (2002) 'MINT: a molecular INTeraction database', *FEBS Letters*, Vol. 513, No. 2, pp.135–140.

Zhou, D., He, Y. and Kwoh, C.K. (2006) 'Extracting protein–protein interactions from the literature using the hidden vector state mode', *Computational Science–ICCS 2006*, Springer, Berlin, pp.718–725.