Bioinformatics Advance Access published August 20, 2016

Bioinformatics, 2016, 1–10
doi: 10.1093/bioinformatics/btw486
Advance Access Publication Date: 27 July 2016
Original Paper

Data and text mining

Drug drug interaction extraction from biomedical literature using syntax convolutional neural network

Zhehuan Zhao[1], Zhihao Yang[1,*], Ling Luo[1], Hongfei Lin[1] and Jian Wang[1]

[1]College of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China

*To whom correspondence should be addressed.

Associate Editor: Alfonso Valencia

Received on November 25, 2015; revised on June 26, 2016; accepted on July 18, 2016

Abstract

Motivation: Detecting drug-drug interaction (DDI) has become a vital part of public health safety. Therefore, using text mining techniques to extract DDIs from biomedical literature has received great attentions. However, this research is still at an early stage and its performance has much room to improve.

Results: In this article, we present a syntax convolutional neural network (SCNN) based DDI extraction method. In this method, a novel word embedding, syntax word embedding, is proposed to employ the syntactic information of a sentence. Then the position and part of speech features are introduced to extend the embedding of each word. Later, auto-encoder is introduced to encode the traditional bag-of-words feature (sparse 0–1 vector) as the dense real value vector. Finally, a combination of embedding-based convolutional features and traditional features are fed to the softmax classifier to extract DDIs from biomedical literature. Experimental results on the DDIExtraction 2013 corpus show that SCNN obtains a better performance (an *F*-score of 0.686) than other state-of-the-art methods.

Availability and Implementation: The source code is available for academic use at http://202.118.75.18:8080/DDI/SCNN-DDI.zip.

Contact: yangzh@dlut.edu.cn

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Adverse drug reactions (ADRs) lead to 300 000 deaths per year in the USA and Europe (Businaro, 2013). Drug-drug interaction (DDI), which is broadly described as a change in the effect of one drug by the presence of another drug (Baxter and Preston, 2010), is an important subset of ADRs. Landau reported that about 2.2 million people in USA, aged 57–85, faced potentially dangerous drug combinations (Landau, 2009). As a result, detecting DDIs has become a vital part of public health safety. With rich DDIs knowledge, the patients can be prevented from taking two or more drugs simultaneously which will cause harmful interactions.

Currently, some drug related databases, like DrugBank (Knox *et al.*, 2011) and Stockley's Drug Interactions (Baxter and Preston, 2010), have been created to help detect DDIs. However, since the volume of biomedical literature is growing rapidly, a large number of valuable DDIs remain hidden in the unstructured biomedical texts. Thus, the automatic extraction of DDIs information from biomedical literature has become an important research area.

In recent years, DDIExtraction 2011 (Segura Bedmar *et al.*, 2011) and 2013 challenges (Segura Bedmar *et al.*, 2013) have been held successfully. DDIExtraction 2011 only focuses on the identification of all possible DDIs while the 2013 challenge requires, in addition to DDI detection, the classification of each DDI, i.e. the DDIs need to be classified into four predefined DDI types: ADVICE, EFFECT, INT and MECHANISM (Herrero-Zazo *et al.*, 2013). ADVICE is assigned when a recommendation or advice regarding concomitant use of two drugs involved is described; EFFECT is assigned when the effect of the DDI is described; INT is assigned when a DDI appears in the text without any additional information provided; MECHANISM is assigned when a DDI is described by its pharmacokinetic mechanism.

Existing DDI extraction methods can be roughly divided into two categories: the one-stage and two-stage methods. The one-stage method accomplishes DDI detection and classification simultaneously by training a multiclass SVM. It directly classifies each candidate instance into one of the five DDI types (ADVICE, EFFECT, INT, MECHANISM and NEGATIVE for the negative instances). The two-stage method divides the learning problem into two stages: first, all the DDIs are detected and second, the detected DDIs are classified into one of the four specific DDI types (ADVICE, EFFECT, INT and MECHANISM).

In DDIExtraction 2013 challenge, FBK-irst team used a two-stage method (Chowdhury and Lavelli, 2013). It employs a hybrid kernel to detect DDIs and, then, assign them to one of the four DDI types by training four separate models (one-against-all), respectively. The method achieves an *F*-score of 0.651, which ranks top in the challenge. Later, Kim *et al.* (2015) also proposed a two-stage method based on linear SVM using rich features, and obtained a higher *F*-score of 0.670 on the same corpus. The method uses the word feature, word pair feature, parse tree feature and noun phrase-constrained coordination feature. In addition, its one-against-one multiclass classification strategy also contributes greatly to its performance while the other methods choose one-against-all strategy.

An example of the one-stage method is the one presented by UTurku team (Björne *et al.*, 2013) in DDIExtraction 2013 challenge. It accomplishes DDI detection and classification tasks simultaneously by training a multiclass SVM (Crammer and Singer, 2002) and classifies each candidate instance into one of the five DDI types (ADVICE, EFFECT, INT, MECHANISM and NEGATIVE). The method achieves the third best performance (an *F*-score of 0.594) in DDIExtraction 2013 challenge.

Although many methods have been proposed, DDI extraction research is still at an early stage and its performance has much room to improve. For example, in DDIExtraction 2013 challenge, the best performance achieved is 0.651 in *F*-score (Chowdhury and Lavelli, 2013). In addition, the state-of-the-art DDI extraction methods are all feature engineering based ones, i.e. they need to design effective features elaborately using various NLP tools and knowledge resources, which is still a labor-intensive and skill-dependent task. Therefore, the performance of these methods is heavily dependent on the choice of features. Finally, since DDIExtraction 2013 challenge is a multiclass classification problem, a multiclass classifier is needed. However, most methods with top performance train several binary SVMs to solve the multiclass classification problem (Chowdhury and Lavelli, 2013; Kim *et al.*, 2015). In fact, *K* or *K* (*K*-1)/2 (here *K* is the number of target classes) binary classifiers are needed when one-against-all or one-against-one strategies are chosen, respectively (Hsu and Lin, 2002). Although such methods can achieve better performances, they increase the complexity of the DDI extraction.

To solve these problems, we propose a syntax convolutional neural network (SCNN) based DDI extraction method. The method uses a novel word embedding (word embedding is a parameterized function that maps words to high-dimensional vectors and was first introduced by Bengio *et al.* (2003) to fight the curse of dimensionality in the process of learning language model using neural network), the syntax word embedding, to introduce the syntactic information of a sentence which has proven to be helpful in boosting the performance of relation extraction (Bunescu and Mooney, 2005). Then the syntax word embedding is extended by the position feature (Zeng *et al.*, 2014) and the part of speech (POS) feature to introduce the position and POS information. The latter is first introduced in our method to capture the POS information of each word. Later, auto-encoder (Hinton and Zemel, 1994) is introduced to transfer the traditional sparse 0–1 features to the dense real value features before they are combined with the embedding-based convolutional features. Finally, the combined features are passed to the softmax (Zeng *et al.*, 2014) to learn the DDI classifier.

2 Materials and methods

In this section, we first introduce a SCNN-based one-stage method (SCNN[1]) which directly classifies each candidate instance into one of the five DDI types (ADVICE, EFFECT, INT, MECHANISM and NEGATIVE). Then a SCNN-based two-stage method (SCNN[2]) is presented since the performances of the two-stage methods are usually better than those of the one-stage ones.

Our one-stage method SCNN[1] contains six processing steps as shown in Figure 1:

1. Negative instance filtering step that rebalances the datasets' class distribution by removing possible negative instances.
2. Preprocessing step that constructs an easily understood corpus for classifiers.
3. Learning word embedding step that generates the syntax word embedding using Enju parser (Miyao and Tsujii, 2008) and word2vec (Mikolov *et al.*, 2013).
4. Feature extraction step that extracts the convolutional and traditional features.
5. Training the classifier[5] step that learns a five-class classifier based on the extracted features.
6. DDI detection and classification step that classifies each instance of the test set into one of the five DDI types using the convolutional neural network (CNN) model.

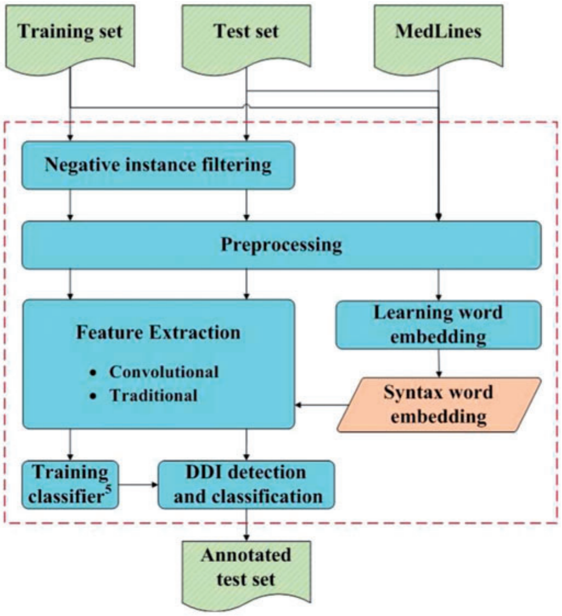

Fig. 1. The processing flow of our one-stage method SCNN[1]

The details are described in the following sections.

2.1 Negative instance filtering

Classification of the data with imbalanced class distribution has encountered a significant drawback of the performance attainable by most standard classifier learning algorithms which assume a relatively balanced class distribution and equal misclassification costs (Sun *et al.*, 2009). DDIExtraction 2013 challenge also suffers from the imbalanced class distribution problem (e.g. the ratio of the positive instances to the negative instances in the training set is 1:5.91). To alleviate this problem, we construct a less imbalanced corpus by removing possible negative examples with the following two rules:

Rule 1: The instances in which two candidate drugs refer to the same drug are removed as any drug is unlikely to interact with itself (Chowdhury and Lavelli, 2013). More specifically, the following two cases are considered: (i) two drugs have the same name; (ii) one drug is the abbreviation of the other drug (the method of finding the abbreviation is described in the Supplementary Material: the method of finding the abbreviation). Two examples are given as follows.

> Same name: Interactions for Vitamin B2 (Riboflavin$_{\text{drug1}}$): Alcohol impairs the intestinal absorption of riboflavin$_{\text{drug2}}$
>
> Abbreviation: *Methyldopa does not interfere with measurement of vanillylmandelic_acid$_{\text{drug1}}$ (vanillylmandelic (VMA)$_{\text{drug2}}$), a test for pheochromocytoma, by those methods which convert VMA to vanillin*

Rule 2: The instances in which two candidate drugs are in coordinate relations are filtered out since they are prone to false positives (Segura Bedmar *et al.*, 2014). For example, the following instance will be removed with rule 2.

*Methscopolamine may interact with antidepressants (tricyclic type), monoamine oxidase (MAO) inhibitors (*e.g. *phenelzine$_{\text{drug1}}$, linezolid$_{\text{drug2}}$, tranylcypromine, isocarboxazid, selegiline, furazolidone*).

2.2 Preprocessing

Appropriate preprocessing can boost the final performance significantly. In our method, two preprocessing operations are conducted, i.e. tokenization and transforming the numbers to two uniform forms.

2.2.1 Tokenization

In our method, tokenization process is performed since it is one of the standard preprocessing steps. We employ the tokenization tool developed specifically for biomedical literature by Jiang and Zhai (2007). Besides the normal tokenization strategies, the tool uses the rules to handle the complicated biomedical entities (e.g. Macrophage inflammatory protein (MIP)-1alpha, 1, 2, 3, 4-TeCDD, 2', 5'-linked 3'-deoxyribonucleotides).

2.2.2 Transforming the numbers to two uniform forms

Numbers (integers and decimals) occur frequently in the DDI corpus. For example, in the sentence 'The serum concentration of phenytoin$_{\text{drug1}}$ increased dramatically from 16.6 to 49.1 microg/mL when fluvoxamine was coadministered, although the daily dosage of phenytoin$_{\text{drug2}}$ and other drugs had not changed', there are two decimals ('16.6' and '49.1'). Transforming the two decimals to a uniform form ('float') won't change the DDI's semantic expression. Therefore, the sentence becomes 'The serum concentration of phenytoin$_{\text{drug1}}$ increased dramatically from float to float microg/mL when fluvoxamine was coadministered, although the daily dosage of phenytoin$_{\text{drug2}}$ and other drugs had not changed'. Then we train a word embedding on the processed sentences with word2vec (Mikolov *et al.*, 2013) (a widely used tool for learning word embedding) and it will generate an embedding for 'float' instead of for '16.6' and '49.1'. Since word2vec trains a sentence based on sliding window mechanism, the 'float' will be trained twice while '16.6' and '49.1' are trained once only. As more training times will generate more accurate embedding, replacing all the integers and decimals with 'num' and 'float', respectively, will provide more powerful embedding for 'num' and 'float'. In addition, it will significantly reduce the size of the vocabulary and make the embedding more compact.

2.3 Syntax word embedding

The syntactic information plays a key role in the sentence level relation classification problems (Bunescu and Mooney, 2005; Fundel *et al.*, 2007). Therefore, it is also employed by deep learning methods to solve the relation classification problems. Xu *et al.* (2015) proposed a CNN-based relation classification model that utilizes the shortest dependency paths information. It takes the word sequence on the shortest path order as the input instead of the original sentence order. Yan *et al.* (2015) solved the relation classification problem with long short term memory networks along the shortest dependency path, whose information is used to generate the new ordered input sequence. However, these methods use the syntactic information to generate the new ordered input sequence instead of training the word embedding.

A word embedding is a parameterized function that maps words to high-dimensional vectors. Word embedding was first introduced by Bengio *et al.* (2003) to fight the curse of dimensionality in the process of learning language model using neural network. Since then, various word embeddings were proposed for learning language models (Collobert *et al.*, 2011; Huang *et al.*, 2012; Mikolov *et al.*, 2013). In addition, word embeddings were also widely used in various NLP tasks. Collobert *et al.* (2011) achieved the state-of-the-art performance on POS tagging, chunking, Named Entity Recognition and Semantic Role Labeling (SRL) using CNN with the word embedding as the input. Zeng *et al.* (2014) proposed a word embedding based convolutional neural network to solve the relation classification problem.

However, the word embeddings mentioned above are all based on the linear contexts (i.e. the surrounding words in linear order of a sentence). They ignore the syntactic information that plays a key role in the sentence level classification problems like DDI extraction.

Levy and Goldberg, (2014) proposed the dependency-based syntactic contexts to learn the word embedding. Hashimoto *et al.* (2014) learned the word embedding using predicate-argument structure contexts and used it to measure semantic similarity between short phrases. In their methods, the syntactic information is introduced by constructing the syntactic contexts instead of the normal linear contexts (i.e. the surrounding words in linear order of a sentence) for the training process. Compared with the latter, the syntactic context yields more inclusive and more focused embeddings (Levy and Goldberg, 2014).

In our method, we propose a novel word embedding that contains the syntactic information, syntax word embedding, by changing the input of the word2vec tool (Mikolov *et al.*, 2013), i.e. the word sequences on the shortest path in the predicate-argument structure instead of the original linear order word sequences are inputted into the word2vec. Take the sentence 'Trimethoprim may inhibit the hepatic metabolism of phenytoin' as an example. We first use Enju parser (Miyao and Tsujii, 2008) to parse the sentence and generate the predicate-argument structure of the sentence as shown in Figure 2. Then the word sequence, 'Trimethoprim inhibit metabolism of phenytoin', is obtained by combining the words on the shortest path connecting the first and last words. It retains the backbone of the

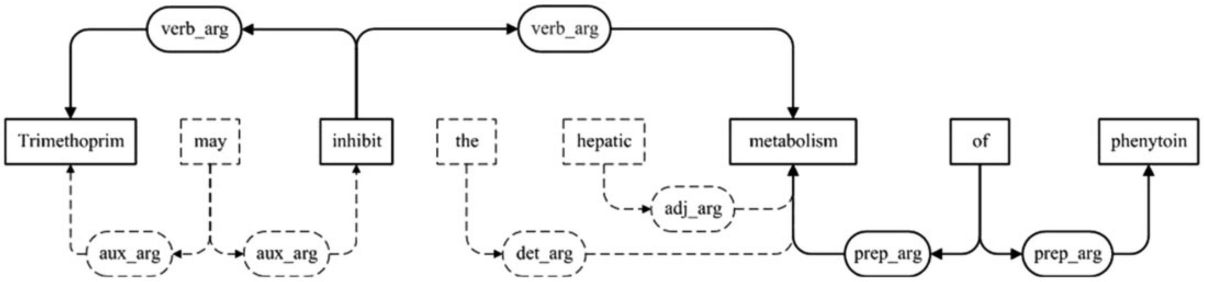

Fig. 2. The predicate-argument structure of the example sentence. The nodes and edges on shortest path connecting the first and last words in the predicate-argument structure are shown in bold and the rest in dotted line. The rectangular nodes represent words. The ellipse vertices represent specific relationships between the predicates and their arguments

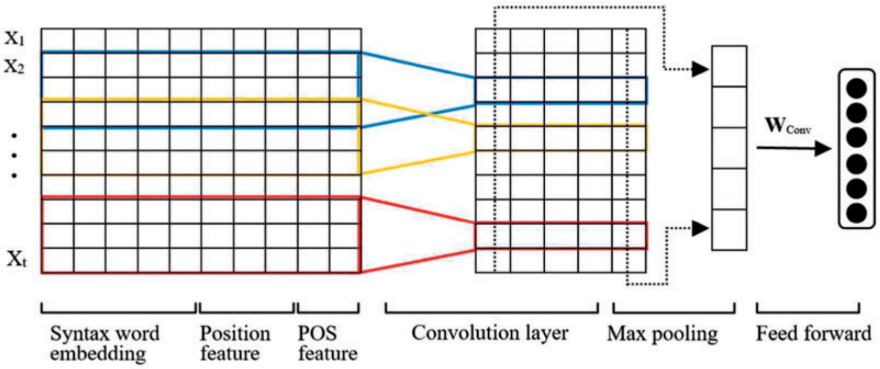

Fig. 3. Convolutional feature extraction. X_i ($i = 0, 1, \ldots, t$) represents ith word in a t length sentence

sentence which is vital for representing a sentence's syntactic structure while filtering out the less important adjunct words (e.g. 'may', 'the' and 'hepatic'). In this way, the syntax word sequence becomes more concise. Then these shortest path order sequences are inputted into word2vec to generate the syntax word embedding. We use $\mathbf{E}^{\text{word}} \in \mathbf{R}^{m \times n}$ to represent the syntax word embedding, where m is the size of the vocabulary and n is the dimension size of the syntax word embedding.

In contrast to previous syntactic context word embeddings (Hashimoto *et al.*, 2014; Levy and Goldberg, 2014), our embedding is learned only based on the concise syntax word sequence, which represents a sentence's syntactic structure while discards the less important words. It is simple but proven to be effective for DDI extraction by our experiments.

2.4 Feature set

Collobert *et al.* (2011) proposed a CNN-based architecture to solve sentence level tasks like SRL. Then Zeng *et al.* (2014) applied this architecture to relation classification problems. In this article, we also propose a CNN-based method to extract DDIs from biomedical literature. The method uses both the embedding-based convolutional features (dense real value vectors) and the traditional bag-of-words features (sparse 0–1 vectors). Then auto-encoder is utilized to narrow down their difference. Thus, we can integrate two kinds of features more effectively.

2.4.1 Convolutional features

2.4.1.1 Word representation. Collobert *et al.*'s architecture enables each word's feature vector to be extended to any discrete features which are helpful to the task of interest. Zeng *et al.* (2014) extended the position feature to specify two target nouns in a relation extraction problem. In our method, as shown in Figure 3, a word is represented with the syntax word embedding, position feature and POS feature.

Syntax word embedding. As described in Section 2.3, the shortest path order sequences are inputted into word2vec to generate the syntax word embedding.

Position feature. DDI extraction task considers the relationship of two candidate drugs in a sentence. But the syntax word embedding itself cannot capture the position information of two drugs which is important for a relation extraction problem. Therefore, the position feature presented by Zeng *et al.* (2014) is also used in our method. The position feature consists of two relative distances, [d_1, d_2], where d_1 and d_2 represent relative distances of the current word to drug1 and drug2, respectively. Take the following sentence as an example.

Hyaluronan_lyase$_{\text{drug1}}$ had a limited effect and collagenase$_{\text{drug2}}$ was ineffective. The relative distances of the word 'effect' to drug1 (*Hyaluronan lyase*) and drug2 (*collagenase*) are −4 and 2, respectively. Then the relative distances are mapped to a ten bit binary vector, where the first bit stands for the sign and the remaining bits for the distance. The position feature ($\mathbf{E}^{p} \in \mathbf{R}^{19 \times 10}$) is shown in Supplementary Table S4 due to space limitation.

As can be seen from the table, more attention is paid to the words near the two drugs, especially the ten surrounding words. On the other hand, the words whose relative distances exceed thirty are all treated the same. It is consistent with the intuition that the closer words contain more information than the more distant words do for the current word.

POS feature. In our method, besides the position feature, the POS feature is also proposed to extend the syntax word embedding. Intuitively, the POS information of the words will be helpful to DDI extraction. Take the following instance as an example.

> *Fluvoxamine$_{drug1}$ [inhibits] the CYP2C9 catalyzed biotransformation of tolbutamide$_{drug2}$.*

Drug1 (*Fluvoxamine*) is more likely to interact with drug2 (*tolbutamide*) since there is a verb 'inhibits' (annotated with a POS tag 'vb') between them. Therefore, the POS feature is informative for DDI extraction and is introduced in our method. The similar POSs generated by Enju parser are assigned to the same group and all 37 POSs (as shown in Supplementary Table S5) are divided into eight groups. Then each group is mapped to an eight bit binary vector using POS feature. In this way, the dimension of the POS feature is reduced.

The POS feature is represented with the traditional bag-of-words feature (sparse 0–1 vector). Intuitively, concatenating the embedding based features (dense real value vectors) and sparse 0–1 vectors directly is not elegant enough. Therefore, we introduce the auto-encoder (Hinton and Zemel, 1994) to solve this problem as discussed in the following section.

2.4.1.2 Auto-encoder. Auto-encoder is a non-linear unsupervised learning model that is widely used as one of the building blocks in the greed layer-wise training process of deep learning (Bengio *et al.*, 2013). It is a neural network (NN) trained to compute a representation of the input from which it can be reconstructed with as much accuracy as possible (Hinton and Zemel, 1994). Coding and decoding processes of the input are obtained from Equations (1) and (2), respectively.

$$\mathbf{h}_j(\mathbf{x}) = f(\mathbf{a}_j(\mathbf{x})) \text{ where } \mathbf{a}_j(x) = \mathbf{b}_j + \sum_i \mathbf{W}_{ji}\mathbf{x}_i \tag{1}$$

$$\mathbf{x}^*_k = g(\mathbf{a}_k) \text{ where } \mathbf{a}_k = \mathbf{c}_k + \sum_j \mathbf{W}^*_{kj}\mathbf{h}_j(\mathbf{x}) \tag{2}$$

where $\mathbf{x}$ is the input, $\mathbf{h}(\mathrm{x})$ is the hidden representation that we need and $\mathbf{x}^*$ is the reconstruction result. We utilize **tanh** and **sigmoid** as the activation functions for $\mathbf{f}()$ and $\mathbf{g}()$, respectively, as the input values range from 0 to 1 and the hidden representation values should be real numbers which can be negatives. To avoid the network learning a trivial identity function, we set $\mathbf{W}^T = \mathbf{W}^*$ as did in the work of Larochelle *et al.* (2009). As the goal is minimizing the reconstruction error, the loss function of auto-encoder can be defined as Equation (3):

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

Then, back-propagation algorithm is used to learn the auto-encoder model with the training data which is represented with our POS feature vector. Using the learned auto-encoder model, we encode the sparse POS feature vector as the dense real value feature vector ($\mathbf{E}^{\text{pos}} \in \mathbf{R}^{8\times8}$).

Finally, given a sentence ($X_1, X_2, \ldots, X_i, \ldots, X_t$) of length t, the ith word X_i will be represented as $\mathbf{V}_i = [\mathbf{E}_i^{\text{word}}, \mathbf{E}^p_{i-\text{drug1}}, \mathbf{E}^p_{i-\text{drug2}}, \mathbf{E}_i^{\text{pos}}] \in \mathbf{R}^{n_0}$, where each element of $\mathbf{V}_i$ represents the syntax word embedding, position feature for drug1, position feature for drug2, and POS feature of the word X_i, respectively, and $n_0 = n + 10 + 10 + 8$ where n is the dimension size of the syntax word embedding.

2.4.1.3 Window approach. After representing each word with a n_0-dimensional vector, we use the window approach to capture each word's context information. Given a word, we will combine a fixed size (*win*) of its surrounding words' vectors. And a new vector $\mathbf{WinV}_i = [\mathbf{V}_{i\text{-}win/2}, \ldots, \mathbf{V}_{i\text{-}1}, \mathbf{V}_i, \mathbf{V}_{i+1}, \ldots, \mathbf{V}_{i+win/2}] \in \mathbf{R}^{n1}$ is obtained with the window approach, where i is the word's index in a sentence and $n_1 = \text{win} \times n_0$. Take the following sentence as an example.

> *Hyaluronan_lyase[1] had[2] a[3] limited[4] effect[5] and[6] collagenase[7] was[8] ineffective[9]*

After the window process, the word 'limited', whose index in the sentence is 4, will be represented as $\mathbf{WinV_4} = [\mathbf{V_3}, \mathbf{V_4}, \mathbf{V_5}]$ where win = 3.

2.4.1.4 Convolution. DDI extraction task is a sentence level relation classification problem which predicts relation types for each sentence that is marked with two target nouns. Therefore, it is necessary to utilize all the local features of the sentence (Zeng *et al.*, 2014). In our method the convolutional approach (Collobert *et al.*, 2011; Zeng *et al.*, 2014) as expressed by Equation (4) is employed to merge all the local information obtained with the window approach.

$$\mathbf{ConS} = \mathbf{WinS} \cdot \mathbf{M} \tag{4}$$

where $\mathbf{WinS} = \begin{bmatrix} \mathbf{WinV_1} \\ \mathbf{WinV_2} \\ \cdots \\ \mathbf{WinV_t} \end{bmatrix} \in R^{t\times n_1}$, t is the number of words in a sentence and $\mathbf{M}^{n1 \times n2}$ is the transformation matrix that is the same across all local features of t in the sentence. $\mathbf{ConS} \in R^{t\times n_2}$ is the transforming result of **WinS** using **M**, where n_2 is a hyper-parameter. Then the max pooling operation is conducted over times on **ConS**. Suppose **ConS**(. ,i) is the ith column of **ConS** and $MaxS_i$ is the maximum value of **ConS**(. ,i). These $MaxS_i$ are found out over time on **ConS** with Equation (5) and are regarded as the most useful features in each dimension of features.

$$\text{MaxS}_i = \max(\mathbf{ConS}(\cdot, i)), 1 < i < n_2 \tag{5}$$

Finally, the hyperbolic tanh activation function (Equation 6) is used to learn the complicated non-linear features.

$$\mathbf{MaxF} = \tanh(\mathbf{MaxS}) \tag{6}$$

where $\mathbf{MaxF} \in \mathbf{R}^{\mathbf{n}_2}$. We can find that the varying length sentences will be represented by the same dimensional (n_2) feature vectors after the convolution process. Thus standard NN can be applied easily to the max feature vectors (**MaxF**).

2.4.1.5 Convolutional feature. After the preceding process, max feature vector (**MaxF**) is obtained and then a feed forward layer is stacked over **MaxF**, using Equation (7), to learn the higher level convolutional features.

$$\mathbf{ConvF} = \tanh(\mathbf{MaxF} \cdot \mathbf{W_{conv}}) \tag{7}$$

where $\mathbf{W_{conv}} \in \mathbf{R}^{n_2 \times n_3}$, n_3 is the dimension of the convolutional feature vector (**ConvF**).

2.4.2 Traditional features

Some traditional features, such as the context of two target entities and information on the shortest path connecting them, can provide important cues for predicting two entities' relationship in a sentence. Therefore, in our method, the context and shortest path features are used.

2.4.2.1 Context features. Our context features include two drug names, their predicate words, their surrounding words, and their surrounding words' predicate words.

One word's predicate word is found in the predicate-argument structure of a sentence. As shown in Figure 2, the predicate of *Trimethoprim* is *inhibit*. Thus, the contexts of both linear and syntactic structure are considered. Finally, all these words' syntax word embeddings are concatenated to generate the context feature vectors ($\mathbf{ContF} \in \mathbf{R}^{n_4}$ where n_4 is the dimension of the context feature vector).

2.4.2.2 The shortest path features. Our shortest path features include the words, the dependency types and biomedical semantic types of the words on the shortest path connecting two drug names.

As shown in Figure 2, the dependency types on the shortest path connecting *Trimethoprim* and *phenytoin* are *verb_arg* and *prep_arg*. And the biomedical semantic types of the words are generated by MetaMap (Aronson, 2001), which is a tool that maps biomedical texts to corresponding concepts of UMLS Metathesaurus (Bodenreider, 2004). MetaMap can generate 104 kinds of semantic types, including ftcn (functional concept), chem (chemical), anim (Animal), etc.

Like the POS feature, the shortest path information is represented with traditional bag-of-words feature (sparse 0–1 vector). Therefore, in the similar way as discussed in Section 2.4.1.2, the auto-encoder is utilized to encode it as the dense real value feature vector ($\mathbf{ShortF} \in \mathbf{R}^{n_5}$ where n_5 is the dimension of the encoded shortest path feature vector) so that it can be concatenated with the embedding based features (dense real value vectors) more effectively.

2.4.2.3 Traditional feature generation. Then the context feature (**ContF**) and encoded shortest path feature (**ShortF**) are concatenated to generate the context and shortest path features ($\mathbf{CSF} = [\mathbf{ContF}, \mathbf{ShortF}] \in \mathbf{R}^{n_6}$, where $n_6 = n_4 + n_5$). Similar to the convolutional features, a feed forward layer is stacked over **CSF**, using Equation (8), to learn the higher level traditional feature (**TradF**)

$$\mathbf{TradF} = \tanh(\mathbf{CSF} \cdot \mathbf{W_{trad}}) \tag{8}$$

where $\mathbf{W_{trad}} \in \mathbf{R}^{n_6 \times n_7}$, n_7 is the size of the traditional feature vector.

2.5 Classifier training

The convolutional features and the traditional features are concatenated into one feature vector, $\mathbf{OutF} = [\mathbf{ConvF}, \mathbf{TradF}] \in \mathbf{R}^{n_8}$, where $n_8 = n_3 + n_7$. Then the **OutF** is fed into the output layer:

$$\mathbf{out} = \mathbf{OutF} \cdot \mathbf{W_{out}} \tag{9}$$

$\mathbf{W_{out}} \in R^{n_8 \times n_9}$, n_9 is the size of the output layer that is equal to the number of the DDI types in DDI classification problem. The output can be represented as $\mathbf{out} = [out_1, out_2, \ldots, out_i, \ldots, out_{n9}]$, where out_i is interpreted as the confidence score of the corresponding DDI type i.

The parameters of the model can be stated as a quad $\theta = (\mathbf{M}, \mathbf{Wconv}, \mathbf{W_{trad}}, \mathbf{W_{out}})$. The probability value of each DDI type is obtained through the following softmax operation over all DDI types using Equation(10):

$$p(i|x, \theta) = \frac{e^{\mathrm{out}_i}}{\sum_{j=1}^{n_9} e^{\mathrm{out}_j}} \tag{10}$$

Then the log likelihood of the parameters is calculated using Equation (11) when all training instances ($T = \{(\mathrm{x}^{(i)}, \mathrm{y}^{(i)})\}$) are given:

$$J(\theta) = \sum_i \log(p(y^{(i)}|x^{(i)}, \theta)) \tag{11}$$

As in the work of Zeng *et al.* (2014), the stochastic gradient descent technique is used in our method to maximize the log likelihood.

2.6 Two-stage method

As shown in Figure 1 of Supplementary Materials, our SCNN-based two-stage method (SCNN^2) includes the following processing steps:

1. Negative instance filtering step that rebalances the datasets' class distribution by removing possible negative instances.
2. Preprocessing step that constructs an easily understood corpus for classifiers.
3. Learning word embedding step that generates the syntax word embedding using Enju parser and word2vec.
4. Feature extraction step that extracts the convolutional and traditional features.
5. Training the classifier^2 step that trains a binary classifier based on the extracted features.
6. DDI detection step that detects the DDIs from the test set using the classifier^2.
7. Training the classifier^4 step that trains a four-class classifier based on the extracted features.
8. DDI classification step that classifies the extracted DDIs in step 6 into four specific DDI types using the classifier^4.

Like other two-stage methods, our two-stage method divides the DDI extraction task into DDI detection stage and DDI classification stage. The only difference between our one-stage and two-stage methods is that the latter needs training two SCNN classifiers (the output layer sizes are two and four, respectively) while the former only needs training one SCNN classifier with the output layer size five.

3 Experimental results and discussions

3.1 Experimental settings

Our SCNN model is coded with Python and trained using Numbapro (http://docs.continuum.io/numbapro/index-archived), a Python compiler from Continuum Analytics (https://www.continuum.io/), which can compile Python code for execution on CUDA-capable GPUs or multicore Central Processing Units. With a Graphics Processing Unit (GPU) of Nvidia Tesla k20, it takes only a few hours to train our model. However, the syntax word embedding learning will take almost one month and most of the time is spent on parsing the massive amount of texts with Enju parser to generate the syntactic information. Since the larger corpus will generate the better embedding (Lai *et al.*, 2015), besides the original DDI corpus, a total of 4 653 097 Medline abstracts were downloaded from PubMed website (http://www.ncbi.nlm.nih.gov/pubmed/) to learn the syntax word embedding with a query string 'drug'.

Our method was evaluated on the DDIExtraction 2013 corpus which consists of 1017 texts (784 DrugBank texts and 233 MedLine abstracts) and was manually annotated with a total of 18 491 pharmacological substances (drug names) and 5021 DDIs (Herrero-Zazo *et al.*, 2013; Segura Bedmar *et al.*, 2013). The corpus contains four different DDI types: ADVICE, EFFECT, INT and MECHANISM. As mentioned in Section 2.1, we use the negative instance filtering to

Table 1. The statistics of the DDI corpus

Corpus	Positives	Negatives	Total	Ratio
OriginalTraining Set	4020	23 772	27 792	1:5.9
NewTrainingSet	3840	8989	12 829	1:2.3
OriginalTestSet	979	4782	5761	1:4.9
NewTestSet	971	2084	3055	1:2.2

Notes. Ratio denotes the ratio of the positives to the negatives in the corpus. OriginalTrainingSet and OriginalTestSet denote the original training and test sets, respectively. NewTrainingSet and NewTestSet denote the new training and test sets obtained after possible negatives are removed, respectively. It should be noted that 22 interactions in the training set whose corresponding sentences can't be parsed correctly by the Enju parser are removed and, therefore, the number of positives (4020) in OriginalTrainingSet is slightly different with that (4042) of the DDIExtraction 2013 corpus.

construct a more balanced corpus. Then the ratios of the positive instances to the negative instances increase from 1:5.9 and 1:4.9 to 1:2.3 and 1:2.2 for the training and test sets, respectively. In addition, as in Chowdhury and Lavelli's (2013) work, any positive instance removed from the test set is automatically considered a false negative during the calculation of *F*-score. Table 1 shows the statistics of the DDI corpus before and after the negative instance filtering process.

The existing DDI extraction methods use the balanced *F*-score measure for quantifying the performance (Segura Bedmar *et al.*, 2013). This metric is defined as *F*-score = (2PR)/(P + R), where P denotes the precision and R denotes the recall. To compare with these methods, we also use *F*-score to evaluate the performance.

3.2 Performance comparison with other methods

The performance comparison between our method and others is shown in Table 2. As can be seen from it, the two-stage methods usually outperform the one-stage methods. For example, the best two results in DDIExtraction 2013 challenge (FBK-irst (Chowdhury and Lavelli, 2013) and WBI (Thomas *et al.*, 2013)) are both achieved with the two-stage methods. Later, Kim *et al.* (2015) also employed a two-stage method to achieve an even better performance. This may clash with the intuition that the two-stage method cannot outperform the one-stage one since it has a drawback that the errors in the DDI detection stage will propagate to the DDI classification stage.

The reasons are as follows. First, dividing the DDI extraction problem into two stages decreases the imbalance degree of the corpus, which is denoted by the ratio of the sample size of the small class to that of the prevalent class (Sun *et al.*, 2009). The number of instances in the original training set for five DDI types (ADVICE, EFFECT, INT, MECHANISM and NEGATIVE) are 826, 1687, 188, 1319 and 23 772, respectively. For a one-stage method, the ratio of the small type (INT) to the prevalent type (NEGATIVE) is 1:126. However, for a two-stage method, in DDI detection, the ratio of the small type (the sum of ADVICE, EFFECT, INT and MECHANISM) to the prevalent type (NEGATIVE) is 1:5.9 and, in DDI classification, the ratio of the small type (INT) to the prevalent type (EFFECT) is 1:9, where the imbalance degree decreases significantly compared with the one-stage method. Since most classification methods, including neural networks, suffer from the class imbalanced corpus problem (Sun *et al.*, 2009), it is reasonable that the two-stage methods outperform the one-stage methods by alleviating the problem.

Another possible reason why the two-stage methods outperform the one-stage methods is that, intuitively, the difficulty degree of a classification problem will rise along with the increase of the target class number. For the one-stage methods, DDI Extraction 2013 challenge task is a five-class classification problem since it needs to classify each candidate instance into one of the five DDI types (ADVICE, EFFECT, INT, MECHANISM and NEGATIVE). But for the two-stage methods, the task is divided into two classification problems, i.e. the DDI detection and DDI classification problems. The former predicts whether an instance is a DDI or not (a binary classification problem), and then the latter classifies each DDI candidate into one of the four DDI types (ADVICE, EFFECT, INT and MECHANISM) (a four-class classification problem). It can be seen as decomposing a complicated problem into two simpler ones. Since the classifiers usually can solve a simple problem better than a complicated one, the two-stage methods usually outperform the one-stage ones.

In DDIExtraction 2013 challenge, the teams UTurku (Björne *et al.*, 2013) and NIL_UCM (Bokharaeian and Dıaz, 2013) used the one-stage methods. As shown in Table 2, our one-stage method (SCNN[1]) achieves an *F*-score of 0.670 which is much better than those of UTurku and NIL_UCM (0.594 and 0.517 in *F*-score, respectively). The improvements have been proven to be significant using MCNemar's test (Dietterich, 1998) at the 0.05 level. The performance is even the same with the known best result achieved with a two-stage method (Kim *et al.*, 2015). The reason is that CNN is a powerful multiclass classifier (Krizhevsky *et al.*, 2012). With the introduction of the syntax word embedding extended by the position and POS features and the utilization of auto-encoder for transferring the sparse traditional feature vector to the dense real value vector, the performance is further improved.

To compare with the existing two-stage methods, we propose a two-stage one (SCNN[2]) as well. In this method, we use the SCNN-based method for the DDI detection by setting n_9 (the size of the output layer, refer to Equation 9) to 2. Then another SCNN model (n_9 is set to 4) is applied to classify the detected DDIs into four specific DDI types, i.e. ADVICE, EFFECT, INT and MECHANISM. Compared with the two top-performing two-stage methods (Kim *et al.*, (2015) and FBK-irst), our method achieves almost equal or lower *F*-scores in DDI detection (0.772 Versus 0.775 and 0.800). The reason is that, as may happen when a complicated model is used to learn an easy problem (Wan *et al.*, 2013), SCNN may overfit the DDI detection, an easy binary classification problem. However, in the DDI classification, its performance exceeds those of other two methods (*F*-scores of 0.686 Versus 0.670 and 0.651). The improvement of SCNN[2] over FBK-irst has been proven to be significant using MCNemar's test at the 0.05 level. However, the significant test between SCNN[2] and Kim *et al.*'s (2015) method cannot be performed since their detailed classification results are not available. This verifies that SCNN is a large capacity model which is more powerful for a complicated problem (e.g. four-class or five-class DDI classification problems).

In addition, SCNN is based on CNN, a multiclass classifier inherently (Krizhevsky *et al.*, 2012). It can solve the multiclass classification problem better than the methods combining multiple SVMs do since SVM is originally designed for the binary classification problem. For example, compared with the best method in DDIExtraction 2013 challenge (FBK-irst, which trains four binary SVMs in the classification stage), the performance of SCNN[2] is much better (0.686 Versus 0.651 in *F*-score) though its DDI detection performance is inferior to that of the latter (0.772 Versus 0.800 in *F*-score).

In the meantime, as shown in Supplementary Table S7, SCNN can obtain the state-of-the-art performance with fewer classifiers.

Table 2. Performance comparison on DDIExtraction 2013 test set

	Method	Classification				Detection		
		P	*R*	*F*-score	△	*P*	*R*	*F*-score
One-stage	SCNN[1]	0.691	0.651	0.670	**7.6%**	0.747	0.768	0.757
	UTurku (Björne et al., 2013A)	0.732	0.499	0.594		0.858	0.585	0.696
	NIL_UCM (Bokharaeian and Dıaz, 2013)	0.535	0.501	0.517		0.608	0.569	0.588
Two-stage	SCNN[2]	0.725	0.651	**0.686**	1.6%	0.775	0.769	0.772
	Kim et al., (2015)	–	–	0.670		–	–	0.775
	FBK-irst (Chowdhury and Lavelli, 2013)	0.646	0.656	0.651		0.794	0.806	**0.800**
	WBI (Thomas et al., 2013)	0.642	0.579	0.609		0.801	0.722	0.759

Notes. SCNN[1] denotes our SCNN-based one-stage method and SCNN[2] denotes our SCNN-based two-stage method. Δ denotes the performance improvement of SCNN[1] over UTurku, and SCNN[2] over that of Kim et al. (2015). The boldfaced numerals are the highest values in the corresponding column.

Table 3. The effect of the strategies and features on performance

Strategy or feature removed	*P*	*R*	*F*-score	Δ
None	0.725	0.651	0.686	–
Negative instance filtering	0.685	0.610	0.645	−4.1%
Syntax	0.711	0.599	0.650	−3.6%
POS	0.707	0.623	0.662	−2.4%
POS Encoding	0.690	0.652	0.670	−1.6%
Shortest Path	0.671	0.586	0.626	−6.0%
Shortest Path Encoding	0.661	0.616	0.638	−4.8%
Position	0.680	0.636	0.657	−2.9%
Word Embedding	0.639	0.572	0.604	−8.2%
Context	0.657	0.599	0.627	−5.9%
ConvolutionLayer1	0.611	0.576	0.592	−9.4%
ConvolutionLayer2	0.577	0.648	0.611	−7.5%

Notes. Δ denotes the corresponding F-score decrease percentage when a strategy or feature is removed.

For the DDI classification, Kim *et al.* (2015) and FBK-irst team learned the multiclass classifiers by combining several binary SVMs. They used six ($n_9 \times (n_9-1)/2$, n_9 is four) and four (n_9) binary SVMs, respectively, as they chose one-against-one and one-against-all strategies (Hsu and Lin, 2002). SCNN[2] uses only one classifier, but its performance is better than those of other two methods. This advantage will be demonstrated more fully when there are more target classes. Along with the increase of the target class number, for the other two methods, the number of binary classifiers will increase rapidly while this won't happen to SCNN. In fact, there are many multiclass classification problems with large target classes. For example, SemEval-2010 Task 8 (Hendrickx *et al.*, 2009) is a relation classification problem of nineteen target classes. Large Scale Visual Recognition Challenge (Krizhevsky *et al.*, 2012) is an image classification problem of one thousand target classes. It would be a disaster to train a one-thousand-class classifier with one-against-one strategy, as it needs 499 500 binary SVMs (1000 × (1000−1)/2).

Furthermore, the DDIExtraction 2013 corpus consists of two parts, i.e. documents from the DrugBank database and MedLine abstracts. The performance comparisons of various methods on DrugBank and MedLine test sets are made in Supplementary Materials due to space limitation.

3.3 The effect of the strategies and features on performance

In addition, to evaluate the effectiveness of the strategies and features of our method, the corresponding experiments are conducted with SCNN[2]: we remove a feature or a strategy each time and then calculate the *F*-score and the corresponding decrease compared with the one before it is removed as shown in Table 3.

Negative instance filtering: after the negative instance filtering strategy is removed, the *F*-score decreases by 4.1%. By further analysis, we found that alleviating the imbalanced class distribution problem contributes to an *F*-score improvement of 1.4% and removing the false positives from the final result that otherwise will be generated by the classification model contributes the rest 2.7%.

In fact, some of our negative instance filtering rules are first introduced in our method. For example, the instances in which two candidate drugs are in coordinate relations are filtered out since they are prone to false positives. These rules will remove numerous possible negatives from both the training and test sets. On the one hand, the reduction of numerous negatives can improve the classification performance of our SCNN model by constructing a more balanced training set. But on the other hand, it leads to the loss of much information about the filtered negatives. The final performance depends on the trade-off between the above two factors. Therefore, our pre-filtering rules may be not necessarily effective for any classification model as it is for ours, which has been verified by the experimental results on the UTurku system (http://jbjorne.github.io/TEES/, the only DDI classification system we could successfully download, install and run). For UTurku, when our pre-filtering rules are only applied on the test set, the *F*-score is improved only a little (0.7 point in *F*-score from 0.593 to 0.6). The reason is that, trained with numerous negatives in the original training set, UTurku has achieved enough ability to recognize the negatives in the test set and, therefore, the pre-filtering rules cannot boost the performance significantly any more. When the rules are also applied on the training set, the *F*-score of UTurku even drops 2.6 points (from 0.6 to 0.574). The cause behind is that, for UTurku, the loss of much information about filtered negatives due to the pre-filtering overwhelms the performance improvement brought with a more balanced training set, leading to its worse final performance.

Therefore, the final effect of our negative instance filtering strategy on performance is closely related to the classification model and it cannot be regarded as a simple rule necessarily effective for any classification model by accurately filtering the negatives from the test set. Experimental results show that it is suitable for our SCNN model and significantly improves the performance. In some sense, it can be regarded as one part of our SCNN approach.

Syntax: replacing syntax word embedding with normal word embedding decreases the *F*-score by 3.6%. The syntax word sequence is concise since it only retains the backbone of the sentence by filtering out the adjunct words. Thus the syntax word embedding is learned based only on the core words, which are vital for representing a sentence's syntactic structure. It is the reason that the syntax word embedding works better than the common one does.

POS, shortest path and their encodings: removing the POS and shortest path features lead to the decreases of *F*-score by 2.4% and 6.0%, respectively. The reason is that the shortest path feature contains much more information than the POS feature since their feature dimensions are 1705 and 8, respectively. In addition, removing the encoding mechanism for the shortest path feature and POS feature leads to the decreases of *F*-scores by 4.8% and 1.6%, respectively. This shows that encoding the sparse POS and shortest path feature vectors (0–1 vectors) as the dense real value feature vectors can boost the DDI classification performance.

Position: the position feature is indispensable to the relation classification problem as the *F*-score decreases by 2.9% when it is removed. Without the position feature, the two target nouns' information, which is essential for the relation classification problem, will be missed.

Word embedding and context: removing the word embedding from the convolutional feature set (only the position and POS features are left) lowers the performance significantly (8.2% in *F*-score) while, when the context feature is removed from the traditional feature set, the *F*-score drops dramatically by 5.9%. This shows that the word information plays a key role in our feature sets.

Convolutionlayer: a CNN is a normal NN with additional convolutional layer. Therefore, we tested the influence of removing the convolutional layer on performance. However, different from CNN that allows the inputs with different sizes, NN requires the fixed input size (the number of words). Therefore, we used the following two methods to solve this problem: **Convolutionlayer1**, extending each sentence by a padding word to the size of the maximal sentence length (74 words in our corpus), where the padding word's embedding is a zero-vector; **Convolutionlayer2**, extracting two target drugs and their 10 surrounding words from a sentence. Both mechanisms lead to much worse performances: the *F*-scores drop by 9.4% and 7.5%, respectively. It shows that the convolution process can integrate all words' information more effectively than the method of simply concatenating them does. More detailed discussions are provided in Supplementary Material: *The effect of the strategies and feature on performance*.

In addition, the combination of the convolutional and traditional feature sets is explored in Supplementary Material: *Combinations of the convolutional and traditional features*.

4 Conclusions

In this article, we present a SCNN based DDI extraction approach. In this approach, a novel word embedding (syntax word embedding) is proposed to exploit the syntactic information of a sentence. Then the syntax word embedding is extended by the position and POS features to introduce the position and POS information. In addition, autoencoder is employed to transfer sparse bag-of-words feature vectors to dense real value feature vectors before they are combined with the convolutional features. Finally, their combination is passed to a softmax to learn the DDI classifier. Experimental results on the DDIExtraction 2013 corpus show that our method achieves an *F*-score of 0.686 which is superior to those of the state-of-the-art methods.

The main contributions of our work can be summarized as follows: (i) Utilizing concise syntax word sequence to learn the syntax word embedding. (ii) Applying POS feature to extend the syntax word embedding. (iii) Using auto-encoder to transfer the sparse bag-of-words features to the dense real value feature vectors. In addition, SCNN is designed for the multiclass problem, and, with the fewer classifiers, it outperforms other methods that implement the multiclass classifiers by combining several binary SVMs.

However, the performance of SCNN on DDI detection is not as satisfactory as on DDI multiclass classification since, as a large capacity model, it fits a complicated problem well, but may over-fit an easy problem (e.g. the two-class DDI detection problem). This is a problem to be further studied. In addition, considering the limited generalization ability of the rule-based negative instance filtering method, in the future, we will try to improve the SCNN method to make it be less influenced by the unbalanced class distribution.

Funding

This work was supported by the grants from the Natural Science Foundation of China (No. 61070098, 61272373, 61340020, 61572102 and 61572098), Trans-Century Training Program Foundation for the Talents by the Ministry of Education of China (NCET-13-0084), the Fundamental Research Funds for the Central Universities (No. DUT13JB09 and DUT14YQ213) and the Major State Research Development Program of China (No. 2016YFC0901902).

Conflict of Interest: none declared.

References

Aronson,A.R. (2001) Effective mapping of biomedical text to the UMLS Metathesaurus: the MetaMap program. In: *Proceedings of the AMIA Symposium*, American Medical Informatics Association, Washington, DC.

Baxter,K. and Preston,C.L. (2010) *Stockley's Drug Interactions*. Pharmaceutical Press, London.

Bengio,Y. *et al.* (2003) A neural probabilistic language model. *J. Mach. Learning Res.*, **3**, 1137–1155.

Bengio,Y. *et al.* (2013) Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, **35**, 1798–1828.

Björne,J. *et al.* (2013) UTurku: drug named entity recognition and drug-drug interaction extraction using SVM classification and domain knowledge. The North American Chapter of the Association for Computational Linguistics, *Atlanta, GA, USA*, p. 651.

Bodenreider,O. (2004) The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic Acids Res.*, **32**, D267–D270.

Bokharaeian,B. and Dıaz,A. (2013) *NIL UCM: Extracting Drug-Drug Interactions from Text Through Combination of Sequence and Tree Kernels*. The North American Chapter of the Association for Computational Linguistics, Atlanta, GA, USA, p. 644.

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

Businaro,R. (2013) Why we need an efficient and careful pharmacovigilance. *J. Pharmacovigilance*, **1**, 1000e1110,

Chowdhury,M.F.M. and Lavelli,A. (2013) *FBK-irst: A multi-phase kernel based approach for drug-drug interaction detection and classification that exploits linguistic information*. The North American Chapter of the Association for Computational Linguistics, Atlanta, GA, USA, p.53.

Collobert,R. *et al.* (2011) Natural language processing (almost) from scratch. *J. Mach. Learning Res.*, **12**, 2493–2537.

Crammer,K. and Singer,Y. (2002) On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learning Res.*, **2**, 265–292.

Dietterich,T.G. (1998) Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.*, **10**, 1895–1923.

Fundel,K. *et al.* (2007) RelEx-Relation extraction using dependency parse trees. *Bioinformatics*, **23**, 365–371.

Hashimoto,K. *et al.* (2014) Jointly learning word representations and composition functions using predicate-argument structures. *EMNLP*, **14**, 1544–1555.

Hendrickx,I. *et al.* (2009). Semeval-2010 task 8: multi-way classification of semantic relations between pairs of nominals. In: *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*. The Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 94–99.

Herrero-Zazo,M. *et al.* (2013) The DDI corpus: an annotated corpus with pharmacological substances and drug–drug interactions. *J. Biomed. Inform.*, **46**, 914–920.

Hinton,G.E. and Zemel,R.S. (1994) Autoencoders, minimum description length, and Helmholtz free energy. *Adv. Neural Inform. Processing Syst.*, **6**, 3–10.

Hsu,C.W. and Lin,C.J. (2002) A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Netw.*, **13**, 415–425.

Huang,E.H. *et al.* (2012) Improving word representations via global context and multiple word prototypes. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, The Association for Computational Linguistics, Jeju Island, Korea, pp. 873–882.

Jiang,J. and Zhai,C. (2007) An empirical study of tokenization strategies for biomedical information retrieval. *Inform. Retrieval*, **10**, 341–363.

Kim,S. *et al.* (2015) Extracting drug–drug interactions from literature using a rich feature-based linear kernel approach. *J. Biomed. Inform.*, **55**, 23–30.

Knox,C. *et al.* (2011) DrugBank 3.0: a comprehensive resource for ‘omics’ research on drugs. *Nucleic Acids Res.*, **39**, D1035–D1041.

Krizhevsky,A. *et al.* (2012) Imagenet classification with deep convolutional neural networks. *Adv. Neural Inform. Processing Syst.*, **25**, 1097–1105.

Lai, S., Liu, K., He, S., & Zhao, J. (2015). How to generate a good word embedding? *Intelligent Systems IEEE*, 1–1.

Landau,E. (2009) Jackson’s Death Raises Questions About Drug Interactions. CNN, June 26, 2009.

Larochelle,H. *et al.* (2009) Exploring strategies for training deep neural networks. *J. Mach. Learning Res.*, **10**, 1–40.

Levy,O. and Goldberg,Y. (2014) Dependency-based word embeddings. *ACL*, **2**, 302–308.

Mikolov,T. *et al.* (2013) Linguistic regularities in continuous space word representations. *HLT-NAACL*, **13**, 746–751.

Miyao,Y. and Tsujii,J. (2008) Feature forest models for probabilistic HPSG parsing. *Comput. Linguistics*, **34**, 35–80.

Segura Bedmar,I. Martinez, P. and Sánchez Cisneros, D. (2011) The 1st DDIExtraction-2011 challenge task: Extraction of Drug-Drug Interactions from biomedical texts. In: *Proceedings of the 1st Challenge Task on Drug-Drug Interaction Extraction*, CEUR-WS, Huelva, Spain, pp. 1-9.

Segura Bedmar,I. *et al.* (2013) Semeval-2013 task 9: extraction of drug-drug interactions from biomedical texts. In: *Proceedings of the 7th International Workshop On Semantic Evaluation*, The North American Chapter of the Association for Computational Linguistics, Atlanta, GA, USA, pp. 341–350.

Segura Bedmar,I. *et al.* (2014) Lessons learnt from the DDIExtraction-2013 shared task. *J. Biomed. Inform.*, **51**, 152–164.

Sun,Y. *et al.* (2009) Classification of imbalanced data: a review. *Int. J. Pattern Recognit. Artificial Intell.*, **23**, 687–719.

Thomas,P. *et al.* (2013) WBI-DDI: drug-drug interaction extraction using majority voting. In: *Proceedings of the Second Joint Conference On Lexical and Computational Semantics*, The North American Chapter of the Association for Computational Linguistics, Atlanta, GA, USA, pp. 628–635.

Wan,L. *et al.* (2013) Regularization of neural networks using dropconnect. In: *Proceedings of the 30th International Conference On Machine Learning (ICML-13)*, The North American Chapter of the Association for Computational Linguistics, Atlanta, GA, USA, pp. 1058–1066.

Xu, K. et al. (2015). Semantic relation classification via convolutional neural networks with simple negative sampling.In: *Proceedings of Empirical Methods in Natural Language Processing*, SIGDAT, Lisbon, Portugal, pp. 536–540.

Yan, X. et al. (2015). Classifying Relations via Long Short Term Memory Networks along Shortest Dependency Path. In: *Proceedings of Empirical Methods in Natural Language Processing*. SIGDAT, Lisbon, Portugal, pp. 1785–1794.

Zeng,D. *et al.* (2014) Relation classification via convolutional deep neural network. In: *Proceedings of COLING*, the Association for Computational Linguistics, Dublin, Ireland, pp. 2335–2344.