Journal of Biomedical Informatics 61 (2016) 34–43

Contents lists available at ScienceDirect

Journal of Biomedical Informatics

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

A graph kernel based on context vectors for extracting drug–drug interactions

Wei Zheng [a,b,*], Hongfei Lin [a], Zhehuan Zhao [a], Bo Xu [a], Yijia Zhang [a], Zhihao Yang [a], Jian Wang [a]

[a] *School of Computer Science and Technology, Dalian University of Technology, Dalian, China*
[b] *College of Software, Dalian JiaoTong University, Dalian, China*

ARTICLE INFO

Article history:
Received 2 December 2015
Revised 14 March 2016
Accepted 18 March 2016
Available online 21 March 2016

Keywords:
Context vector
Graph kernel
Equivalent class
Drug–drug interactions

ABSTRACT

The clinical recognition of drug–drug interactions (DDIs) is a crucial issue for both patient safety and health care cost control. Thus there is an urgent need that DDIs be extracted automatically from biomedical literature by text-mining techniques. Although the top-ranking DDIs systems explore various features of texts, these features can't yet adequately express long and complicated sentences. In this paper, we present an effective graph kernel which makes full use of different types of contexts to identify DDIs from biomedical literature. In our approach, the relations among long-range words, in addition to close-range words, are obtained by the graph representation of a parsed sentence. Context vectors of a vertex, an iterative vectorial representation of all labeled nodes adjacent and nonadjacent to it, adequately capture the direct and indirect substructures' information. Furthermore, the graph kernel considering the distance between context vectors is used to detect DDIs. Experimental results on the DDIExtraction 2013 corpus show that our system achieves the best detection and classification performance (*F*-score) of DDIs (81.8 and 68.4, respectively). Especially for the Medline-2013 dataset, our system outperforms the top-ranking DDIs systems by *F*-scores of 10.7 and 12.2 in detection and classification, respectively.

1. Introduction

Drug–drug interactions (DDIs) occur when one administered drug has an influence on the level or activity of another drug. The DDI problem is one of the most important causes of medical errors. Knowing all potential interactions is very important for physicians prescribing varying combinations of drugs for their patients.

Because it is becoming more common that multiple drugs are used simultaneously, DDIs are causing a great threat to public health [1]. The interactions among drugs are affected by many factors, such as the dose dependence of many DDIs, natural genetic and demographic variation. Therefore, the recognition of DDIs in the clinic is difficult even if a drug has been tested by stages. Today, most DDIs are discovered by accident in the clinic or during phases IV clinic trials that take place once a drug is already on the market [2]. Only a few drug combinations can be analyzed at a time by laboratory studies, which is not enough to recognize, understand and predict DDIs on a large scale. However, huge amounts of data relevant to DDIs are preserved in specialized databases and scientific documents. On the one hand, existing databases with structured data are generally incomplete because of their long update periods. On the other hand, the amount of biomedical literature with unstructured data, such as MEDLINE, is increasing exponentially. As a result, there is an urgent need that DDIs should be extracted automatically and effectively from biomedical literature. This contributes significantly not only to detecting known DDIs, but also to automating the database curation process, which is currently performed manually [3].

To promote the development of text-mining techniques and improve the performance for detecting DDIs from biomedical texts, Segura-Bedmar et al. [4,5] developed valuable gold standard sets in the DDIExtraction challenges 2011 (DDI-2011) and 2013 (DDI-2013), which provided an opportunity to address the problem of DDI extraction for the community. While DDI-2011 corpus only focused on the identification of all possible pairs of interacting drugs, DDI-2013 corpus [6] also proposed, in addition to detection, a more fine-grained classification of each true DDI. In both challenges, the detection of DDIs was seen as a binary classification problem, where each candidate drug pair is judged as having an interacting relation or not. In particular, the DDI-2013 corpus

* Corresponding author at: School of Computer Science and Technology, Dalian University of Technology, Dalian, China.

E-mail addresses: weizheng@mail.dlut.edu.cn (W. Zheng), hflin@dlut.edu.cn (H. Lin).

http://dx.doi.org/10.1016/j.jbi.2016.03.014
1532-0464/© 2016 Elsevier Inc. All rights reserved.

includes the DrugBank dataset (DB-2013) and the Medline dataset (ML-2013). However, it is worth noting that, for the detection and classification tasks, there are relatively low F-measures on the ML-2013 dataset compared with the DB-2013 dataset. Therefore, what this paper is concerned with is how to improve the performance on the ML-2013 dataset.

The support vector machine (SVM), one machine-learning-based approach, has been prevalent in systems with state-of-the-art performance [7–13] in both challenges. Kernel methods, the key technology of SVM, can be regarded as a generalization of feature-based methods by replacing the dot-product with a similarity function between two vectors [14]. Composite kernels [8,9,13,15,16], which combine different characteristics of several kernels, are frequently used in SVM systems to obtain heterogeneous and complementary features. The composite-kernel-based systems achieved the best results in both the DDI-2011 and DDI-2013 challenges. However, Segura-Bedmar et al. [17] demonstrate that it is rarely possible to further enhance the performance of DDI tasks by building different ensemble systems after the DDI-2013 challenge. In addition, a composite-kernel-based system is generally an intricate system. Composite kernels require more computational cost because of the accumulative complexity of the underlying kernels. Furthermore, additional learning is required to optimize the weights of individual kernels [18]. In fact, a recent study [19] suggests that the performance improvement of relation extraction systems depends on more expressive feature sets rather than on more complex kernel functions.

Therefore, many researchers have focused recently on how to build a DDIs system with a single kernel [18,20], which should be a practicable alternative for DDI tasks if it performs as well as composite kernels. Compared with a linear kernel with only word-level features in an *n*-dimensional space, a nonlinear kernel can map a structured representation of a sentence into a high-dimensional feature space where instances can be divided linearly into two or more groups. A graph kernel is a nonlinear kernel which builds on a graph structure with labels on nodes and/or edges. A graph is a powerful tool for modeling relations and events, because the graph provides a flexible structure to represent a network and to naturally describe the relations between its components [21]. Systems based on a graph kernel have shown an advantage in relation extraction tasks [12,22,23]. Hence, it is good to explore the use of a graph kernel to extract the drug–drug interactions.

Contexts are any information that can be used to characterize the situation of a token in a sentence. Machine-learning-based approaches can statistically capture the characteristics of contexts and arguments relating to syntactic structures. However, existing graph kernels generally explore limited contextual information of graphs. The walk-weighted subsequence kernel [24] only matches the e-walk and v-walk on the shortest path of the dependency graph. The all-path graph kernel [12] labels only the token itself, computes the infinite sum of the weights of all paths of any label pairs, and neglects the adjacent structure of a node. The hash subgraph pair kernel (HSP) [25] computes a neighborhood hash of a node. However, the bit operation and unavoidable inherent conflict of hash operations lead to failures to identify coordinate structures, negation expressions and appositions. The extended graph kernel [26] adds the integer sum of the neighborhood of each vertex into its labels by using the Morgan algorithm. It is obvious that extracted contextual features are not expressive enough to capture overall information around a node in the above two systems.

The appropriate choice and effective representation of contexts affect the properties of a feature space. The Shallow Linguistic kernel [27] is possibly the best feature-based kernel. It is the combination of the local contexts and global contexts. However, this system only counts the frequency or co-occurrence times of words encountered in the same context without considering the grammatical relations holding between the two entities. The syntactical parse of the sentential structure can capture the relation of long-distance entities, which is especially important for the ML-2013 dataset, which contains long and subordinated sentences characterized by very scientific language from abstracts in the MEDLINE database. In addition, recent advances in parsing technology [28,29] allow parsing of syntactic dependencies with very high speed and near state-of-the-art accuracy. UTurku [30], one of the best participating systems, develops an open source linear kernel system, TEES, where dependency chains starting from a token of interest and dependency path N-grams on the shortest path are used. Therefore, syntactical and other contexts should be utilized adequately by graph-kernel-based systems to extract DDIs. Furthermore, the holistic and expressive representation of contextual information enriches and subdivides features around a vertex and makes it possible to improve the performance of DDI extraction systems.

In this article, we present an effective and scalable context vector graph kernel that exploits contexts adequately to extract DDIs from biomedical texts. Our method includes four critical steps: first, build the graph representation to obtain the relations among long-range words as well as close-range words for each candidate drug pair. Next, calculate iteratively context vectors of a vertex by using all labeled nodes adjacent and nonadjacent to this vertex, which sufficiently capture the direct and indirect substructure information. Then, the graph kernel considering the distance between context vectors is derived to detect DDIs from texts. Finally, the one-versus-all strategy is adopted to classify each true DDI into one of four fine-gained types. The experimental results on the DDIExtraction 2013 corpus indicate that our system achieves the best *F*-scores of 81.8 for detecting DDIs and 68.4 for classification. Especially for the Medline-2013 dataset, our system outperforms the top-ranking DDIs systems' *F*-scores by 10.7 and 12.2, respectively.

2. Methods

In this section, we describe our approach to extracting drug–drug interactions from biomedical texts.

2.1. Text preprocessing

All sentences of the corpus are cleaned and normalized by following text preprocessing, which enhances the speed and performance of a relation extraction system.

(1) Sentences for only one entity or two entities with same tokens are filtered out.
(2) To highlight syntactic relations around an entity mention when dependency parsing, rules are defined to mark their different positions for the following entities: (I) Entity mentions sharing prefix or suffix. (II) Entity mentions which consist of multiple words. All words for above each an entity mentions are connected to form a special string without spaces. Then the corresponding mention is replaced by this string. For example, the phrase "*inactivated ND and AI vaccines*" is annotated as two drug mentions (*inactivated ND vaccines and inactivated AI vaccines*, respectively), and it may be transformed into "*inactivated@ND@vaccines*" and "*inactivated@AI@vaccines*". Experiments indicate that this method promotes obviously the performance of our systems.
(3) In order to reduce the sparsity of feature space, a special token is used to replace the digit string which is not a substring of a drug entity. For the following sentence "*We*

evaluated the effects of calcium doses between 200 and 1500 mg on absorption of 5 mg nonheme iron", the semantic expression of this sentence should not change if the three digit string (200, 150 and 5) are transformed into a uniform form ("*NUM*"), respectively.

After the above process, the number of entity mentions in an input sentence is equal to that of the original. For example, a sentence with three annotated entities shown in bold is as follow: ***L-histidine****, via its metabolism to histamine, might decrease the efficacy of* ***H1 and H2 blockers*** (from DDI-DrugBank.d365.s1 in L-Histidine_ddi.xml). Its simplified sentence is as follows: ***L-histidine****, via its metabolism to histamine, might decrease the efficacy of* ***H1@blockers*** *and* ***H2@blockers.*** Furthermore, dependency parsing will be performed on the simplified sentence by the Stanford parser [31], and then syntactic and dependency parsing results are obtained.

2.2. The structured representation of sentences

Dependency subgraphs and linear subgraphs are two kinds of important graph representations in the tasks of biomedical relation extraction. The dependency subgraph represents long-range governor-dependent relations among sentential constituents. The linear subgraph embodies the position sequence relations among tokens in a sentence. A sentence in our approach is represented by the two subgraphs, which contain complementary information (see Fig. 1). The combination of the dependency subgraph and the linear subgraph reflects the relation among different distance entities in a sentence. Fig. 1 shows an example for the structured representation of a sentence.

The corresponding sentence for each a candidate drug pair is represented as a directed vertex-labeled weighted graph. The dependency subgraph (Fig. 1(A)) and the linear subgraph (Fig. 1(B)) are similar to previous studies [12]. One vertex and an associated set of labels are created for each token and for each dependency. In the dependency subgraph, each token vertex uses the text and part-of-speech (POS) of the token as labels, and each dependency vertex is labeled with the dependency type. For example, the label "*effects/NNs*" denotes that the text of the token node is "*effects*" and the POS is "*NNs*", and the label "*amod*" is the dependency type of token nodes "*additive*" and "*effects*". For the generalization of our approach, all entity mentions are marked with DRUG* (* denotes 0, 1, 2). DRUG1 and DRUG2 are the candidate entities and DRUG0 replaces other entities in the input sentences. The shortest path between the candidate entities is shown in bold in the dependency subgraph. Moreover, all nodes on a shortest path are specialized by using a prefix-tag (sp). In the linear subgraph, a second vertex with labels is created for each token. Besides the text and POS-tag of the token, the labels of each word are specialized by using the position prefix-tag (B, M, or A), which denotes whether the word appears before, in-between, or after the candidate drug pair. A token dictionary preserves all labels of graphs. In the dependency subgraph, the edges are assigned weights according to the weight scheme of Airola et al. [12] where all edges on the shortest path receive a weight of 0.9 and other edges receive a weight of 0.3. In the linear subgraph, the weight 0.9 is assigned to edges from the three words before DRUG1 to after DRUG2 and the weight 0.3 is assigned to other edges.

2.3. Constructing context-based vectors

The strength of the relationship between candidate entities might be affected by different kinds of contexts and their representations. Levy and Goldberg [29] demonstrate that linear contexts reflect topic similarities, and dependency-based contexts or syntax-based contexts reflect more functional similarities of syntactic structures. In their experiments, the top word and syntactic contexts for the word "batman" are those phrases such as "superman/conj", "spider-man/conj" and "robin/conj". Therefore, it is important for relation extraction that the contextual representation has enough expressive power. The following vectorial representation, based on the combination of the above two contexts, provides an effective way to take full advantage of contexts.

Some essential terminologies are introduced before our method is described. Consider a graph $G = (V, \varepsilon)$ where V denotes a finite set

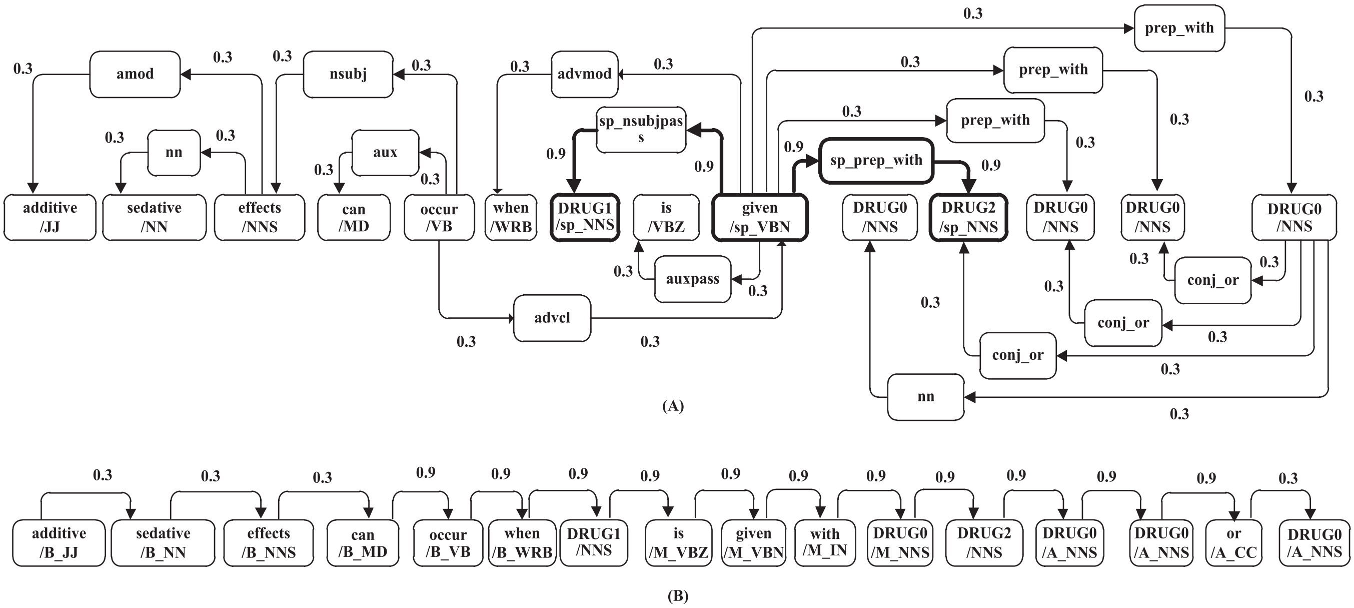


Fig. 1. The graph representation of the corresponding sentence for a candidate pair. The candidate pair is marked as DRUG1 and DRUG2 and the other drugs are marked as DRUG0. (A) The dependency subgraph where the labels of all nodes on the shortest path between the drugs shown in bold are specialized by using a prefix-tag (sp). (B) The linear subgraph where the labels of each node are specialized by using the position prefix-tag (B, M, or A), which denotes whether the word appears before, in-between, or after the candidate drug pair.

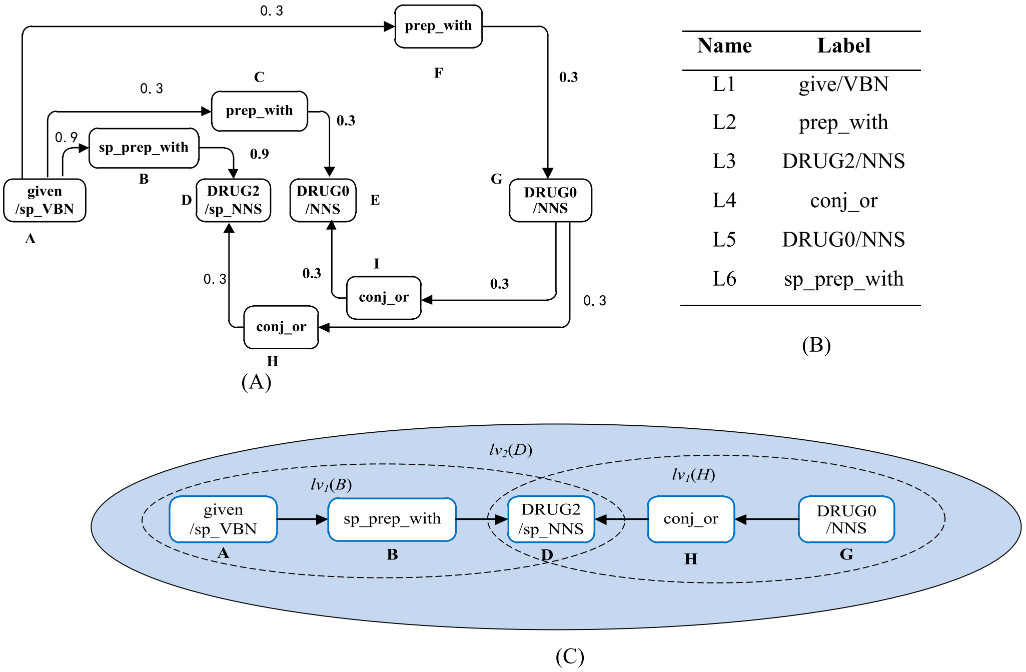
of vertices (or nodes) and $\varepsilon \subseteq V \times V$ denotes a finite set of edges (or links). A graph $G = (V, \varepsilon, \mathcal{L})$ is called a vertex-labeled graph where labels are only assigned to vertices and $\mathcal{L}$ is a finite set of labels (or attributes) along with a label assigning function ℓ.

Definition 1. Context Vector: Given a vertex $v_x \in V$ and its adjacent nodes $adj(v_x) = \{v_{adj\_1}, v_{adj\_2}, \ldots, v_{adj\_m}\}$, the label of the vertex v_x is denoted as an ordered vector $lv_i(v_x)$ in the i-th iteration of its adjacent nodes, which is defined as

$$lv_i(v_x) = \begin{cases} (l(v_x)) & i = 0 \\ \cup_{k=1}^{m} lv_{i-1}(v_{adj\_k}) \quad (v_x, v_{adj\_k}) \in \varepsilon \text{ and } & i >= 1 \end{cases} \qquad (1)$$

where $l(v_x)$ denotes the labels of the vertex v_x. Fig. 2 shows an example for the calculation of context vectors in a vertex-labeled graph G (Fig. 2(A)), which is a part of Fig. 1. The column name of Fig. 2(B) is the alias of the label of each node in the token dictionary. The alias will replace the corresponding label on a node to simplify the following representation in the computational process. The context vectors of each vertex are computed in Fig. 2(D). At the beginning, the elements of $lv_0(v_x)$ consist of elements of the labels $l(v_x)$ itself. Then, Eq. (1) is used to calculate iteratively the context vector $lv_i(v_x)$ whose elements consist of elements of the context vector $lv_{i-1}(v_{adj\_k})$ ($v_{adj\_k} \in adj(v_x)$) for all its adjacent nodes. Finally, each context vector is added to a feature dictionary. In particular, all elements of $lv_i(v_x)$ are sorted by dictionary sequence. Take a node D as an example. Its label "*DRUG2/NNS*" may be replaced by the alias $L3$ and then its context vector $lv_0(D) = [L3]$. Furthermore, because the context vector $lv_1(D)$ is made up of the 0-th context vectors ($lv_0(B)$, $lv_0(H)$) of its all adjacent nodes, the vector $lv_1(D) = [L4, L6]$. The iteration of vectors for other vertices on the dependency and linear subgraphs may be performed in the same way. Thus, the vector $lv_2(D) = [L1, L3, L3, L5]$, which contains not only the whole information for nodes within path length 2 of the node D, but also embodies sub-structure information (see Fig. 2(C)). If there are many identical labels in a vector, effective storage means might be chosen to enhance data access efficiency.

The definition of Eq. (1) indicates that the context vectors of a vertex, as iterative sequences of all its neighboring nodes, imply



(A)

Name	Label
L1	give/VBN
L2	prep_with
L3	DRUG2/NNS
L4	conj_or
L5	DRUG0/NNS
L6	sp_prep_with

(B)

(C)

	Nodes	A	B	C	D	E	F	G	H	I
	lv_0 (v$_x$)	L1	L6	L2	L3	L5	L2	L5	L4	L4
G0	$\mathcal{L}_0$	{ [L1], [L6], [L2], [L3], [L5], [L2], [L5], [L4], [L4] }								
	group	{A} {B} {C, F} {D } {E, G} {H, I}								
	lv_1(v$_x$)	L2, L2, L6	L1, L3	L1, L5	L6, L4	L2, L4	L1, L5	L2, L4, L4	L3, L5	L5, L5
G1	$\mathcal{L}_1$	{ [L2, L2, L6], [L1, L3], [L1, L5], [L6, L4], [L2, L4], [L1, L5], [L2, L4, L4], [L3, L5], [L5, L5] }								
	group	{A} {B } {C, F } {D} { E} {G} {H} {I}								

(D)

Fig. 2. An example for the computational process of context vectors of nodes in a graph. (A) An instance of a graph G. (B) The column name is the alias of the label of each node in the token dictionary. The alias will replace the corresponding label on a node to simplify the following representation. (C) The implication of the vector $lv_2(D)$. (D) The calculation of context vectors and equivalent class.

topology structure information and pass the information of non-adjacent nodes (e.g., lv_2 (D) passes indirectly the information of nodes A, G)). In addition, while our vectors mean a whole functional group and sub-groups of structures on a dependency subgraph, they embody token pairs before and after a word on a linear subgraph. Thus, our vectors integrate many kinds of syntactic and lexical information around nodes.

As a result, the labels of vertex v_x ($v_x \in V$) form a vector sequence $lv(v_x) = (lv_0(v_x), lv_1(v_x), lv_2(v_x), \ldots)$ by applying Eq. (1) iteratively. This method subdivides a node into multiple vectors, which allows the contextual information of each vertex to be utilized fully.

In addition, consider that it is inevitable that many vertices, such as dependency type or blinded entities, may have the same context vectors in the iterative process. For example, many texts on drug information, such as the DB-13 dataset, have a long drug list. Multiple identical vectors in a sentence cannot play a more important part in enhancing the expression of the feature space.

Definition 2. Equivalent class based on context vectors: If V is the set of all vertices in a vertex-labeled graph, and "has the same context vector as" is an equivalent relation $\sim$ on V, the equivalent class of an element ℓ ($\ell \in V$) is denoted as $[\ell]$, whose definition is the set of elements that are related to ℓ by $\sim$.

$$[\ell] = \{x \in V | \ell \sim x\} \tag{2}$$

After labels of vertex v_x are replaced with the context vector $lv_i(v_x)$ every time, the set of vertices in graph G conforms to the equivalent relation $\sim$. Furthermore, equivalent classes of all nodes are calculated by using Eq. (2), and then equivalent classes of nodes which have the same elements in their sets are partitioned into a group to minimize the representation of a graph. Fig. 2(D) shows the partition process of vertices. Because the two nodes (C, E) in $G0$ have the same vector $lv_0(v_x)$, their corresponding equivalent classes $[C] = [E] = \{C, E\}$. Thus, the two nodes are partitioned to a group. After the first partition, nine vertices are divided into six groups. The number ∂_h of groups is less than that of vertices. Furthermore, we define a new graph $G_h = (V, \varepsilon, \mathcal{L}_h, C_h)$ after each iteration, where h is the number of iterations, C_h is the set of groups, and $\mathcal{L}_h$ is the set of context vectors of all nodes after the h-th iteration; it is defined as shown in Eq. (3)

$$\mathcal{L}_h = \{lv_h(v_1), lv_h(v_2), \ldots, lv_h(v_j), \ldots, lv_h(v_n)\} v_j \in V \tag{3}$$

As can be seen from Fig. 2, $lv_0(v_x)$ only represents the basic information of vertex v_x, and $lv_h(v_x)$ covers the structure information and more associated contexts of nodes which have a path length h to the vertex v_x. The path length h is also called the context window. In other words, the vectorial representation captures the direct and indirect contextual information of nodes more accurately from the vicinity to the distance as the iteration proceeds. Thereby, a graph forms a hierarchical graph sequence $G = (G_0, G_1, \ldots)$.

Table 1 gives the algorithm **CCV** to compute the context vectors of all vertices in graph G. When the context vectors of a node are calculated, if the number of the iteration $h \geqslant 1$, only adjacent nodes with edge weight 0.9 are selected to reduce noise and control the sparsity of the feature space; otherwise all adjacent nodes are selected. There are two termination conditions for the iterative computation of the context vectors. The computational process ends when the number of groups in G_i is equal to that in G_{i-1}. In the other case, we may set the number h of iterations. The parameter h^* of the algorithm **CCV** is the upper bound of the context window size h; it is determined by experiments.

2.4. The context-vector graph kernel based on equivalent classes

The distance between (all) pairs of vertices with labels has an effect on the performance of the classification based on the graph [32]. In our system, the union set of any vectors $lv_h(v_i)$ and $lv_h(v_j)$ in the same layer G_h after the partition of equivalent classes, if the vector exists, forms the context-vector pairs $Vp_h(v_i, v_j)$. For Example, the vector pair $Vp_0(A, D) = [L1, L3]$ is made up of two vectors $lv_0(A)$ and $lv_0(D)$ of the layer G_0 in Fig. 2(D). The similarity of two graphs G and G' depends on the summed weight of all paths of context-vector pairs of all layers.

The following terms on kernel definition follow Gärtner et al. [32]. All context vectors of the layer G_h in a sentence graph are presented as an allocation matrix $L_h \in R^{|k| \times |V|}$, where $|k|$ is the number of possible context vectors. The context-vector graph kernel is defined as in Eq. (4).

$$\begin{aligned} K(G,G') &= \sum_{h=0}^{h^*} \beta_h \left\langle L_h \left(\sum_{l=0}^{\infty} E^l \right) L_h^T, L'_h \left(\sum_{l=0}^{\infty} E'^l \right) L'^T_h \right\rangle \\ &= \sum_{h=0}^{h^*} \sum_{m=0}^{|k|} \sum_{n=0}^{|k|} \beta_h \left[L_h \left(\sum_{l=0}^{\infty} E^l \right) L_h^T \right]_{mn} \left[L'_h \left(\sum_{l=0}^{\infty} E'^l \right) L'^T_h \right]_{mn} \end{aligned} \tag{4}$$

where E and E' are adjacency matrixes of graph G and G', respectively. Each component $[E]_{ij}$ denotes the weight of the edge connecting vertex $v_i \in V$ and vertex $v_j \in V$. Replacing the adjacency matrix E by its n-th power ($n \in N$, $n \geqslant 0$), each component $[E^n]_{ij}$ gives the summed weight of the paths of length n from vertex v_i to v_j. Multiplying this by the context vector matrix L_h, we obtain the matrix $L_h E^n L_h^T$. Each component $[L_h E^n L_h^T]_{ij}$ corresponds to the summed weight of the paths of length n between the vertex labeled $lv_h(v_i)$ and the vertex labeled $lv_h(v_j)$ in graph G_h. For example, the vector pair $Vp_0(A, D)$ in Fig. 2(A) receives a weight 1.8 when the path length is 2. Because we are interested not only in paths of one specific length, the matrix $L_h \left(\sum_{l=0}^{\infty} E^l \right) L_h^T$ denotes the context-vector pair matrix of graph G_h with the summed weight of all paths. The matrix power series $\sum_{l=0}^{\infty} E^l$ combines the effect of label pairs with all lengths, and can be calculated efficiently by Eq. (5).

$$\sum_{l=0}^{\infty} E^l = (I - E)^{-1} - I \tag{5}$$

After much iteration, the matrix $G = \sum_{h=0}^{h^*} L_h \left(\sum_{l=0}^{\infty} E^l \right) L_h^T$ represents an instance of the sentence graph of a candidate interaction. The vector pair $Vp_0(A, D)$ with all path lengths in Fig. 2 (A) receives a weight 3.0 (path length 2 and 4, respectively). In addition, context vectors of different window sizes have a distinctive function in relation-extraction tasks. Therefore, the decay factor $\beta_h > 0$ is set to adjust the effect of context-vector pairs with different window sizes. If the window size $h = 0$, the context-vector graph kernel is equivalent to the all-path graph kernel. In particular, because context-vector pairs from different groups might form multiple same sequences of vector pairs, the minimum among their weight is assigned to the sequence. In general, if there are multiple vertices in an equivalent class, their corresponding context vectors have weak expression. In conclusion, the context-vector graph kernel achieves not only the structure and rich contextual information about the neighborhood of the vertices but also all paths' information between any context-vector pairs. Therefore, it has a more expressive feature space.

Table 1
The algorithm to calculate context vectors.

```
Algorithm CCV (Calculate_Context_Vector)

Input: the adjacency matrix $E$ and label allocation matrix $L$ of a graph $G$
  $h^*$ is the upper bound on the number $h$ of hierarchies
Output: context vectors of all vertices in $G$
1: for $x = 1$ to $n$ do //$n$ is the number of vertices
2:   $lv_0(v_x) = [l(v_x)]$
3:   add $lv_0(v_x)$ to the feature dictionary $F$
4: equivalent classes of all vertices are calculated using Eq. (2), and
   the nodes are grouped by equivalent classes
   // last∂ is the number of groups last time
5: $last\partial \leftarrow 0$, $count \leftarrow 0$
6: $\partial \leftarrow$ the number of groups
7: while $\partial$ > last∂
8:   for $x = 1$ to $n$ do
9:     for $adj\_k = 1$ to $m$ do//m is the number of adjacency vertices of $v_x$
10:      if w(x,$v_{adj\_k}$) == 0.9 // w(x,$v_{adj\_k}$) is the weight between x and $v_{adj\_k}$
11:        add $lv_{i-1}(v_{adj\_k})$ into $lv_i(v_x)$
12:    sort $lv_i(v_x)$
13:    add $lv_i(v_x)$ to the feature dictionary $F$
14:  do so as the 4th step
15:  $\partial \leftarrow$ the number of groups
16:  $count = count + 1$
17:  if $count == h^*$
18:    break
19:  if $\partial == last\partial \leftarrow 0$
20:    break
21:  else
22:    $last\partial \leftarrow \partial$
```

3. Results and discussion

3.1. Datasets and evaluation settings

We train and evaluate the proposed approach on the DDI-2013 corpus. The DDI-2013 corpus includes documents selected randomly from the DrugBank database (DB-2013) and Medline abstracts (ML-2013); each of them is split into two parts, one of which is a training dataset and the other a test dataset. All drug mentions and drug pairs in the documents are annotated manually by sentences. Each drug pair is annotated as either no interaction (negative instance) or true interaction (positive instance). Moreover, one of four different types with more fine-gained annotations is assigned to a true interaction. Table 2 lists the statistics of this corpus.

For each test dataset, the performance of our system is evaluated by the standard evaluation measures (precision, recall and F-score). F-score is defined as $F = (2PR)/(P + R)$, which can play a balanced role between P and R. In particular, for the ML-2013 corpus, this system is trained on the combination of the DB-2013 and ML-2013 training datasets, as suggested by [8,13].

We conduct 5-fold sentence-level crossing validation on the corresponding training set to optimize the parameters of our approach. All features extracted are normalized. We use the least squares support vector machine to learn a model, as Airola et al. [12] do. The parameters ($C = 1.5$) of the SVM on the training datasets were obtained experimentally.

Table 2
Statistics for DDI-2013 corpus.

Instances	DDI type	DB-2013		ML-2013	
		Training set	Test set	Training set	Test set
Positive	Mechanism	1257	278	62	24
	Effect	1535	298	152	62
	Advice	818	214	8	7
	Int	178	94	10	2
Negative		22,217	4381	1555	401
Total		26,005	5265	1787	496

3.2. Parameter selection

To achieve better performance from our system, the parameters h^* and β_h in Eq. (4) need to be assigned values based on experiment.

The parameter h^* indicates the importance of the fit window size of contexts. For the parameter h^*, a smaller value means limited contextual information, which is insufficient to collect effective features, whereas a larger value signifies more extensive coverage. In other words, a larger subgraph leads to matching two sentences in a strict manner. Thus, over-fitting problems will occur in the process of machine learning. In our system, the best performance on the DB-2013 dataset is obtained when the parameter $h^* = 2$. A small performance improvement on the ML-2013 dataset will require more time cost with the increasing value of the parameter h^*. Therefore, the overall performance on the ML-2013 dataset is optimal when the parameter $h^* = 2$.

The window size of contexts has various influences on a feature space. In general, the closer the contexts are away from a token, the more significant the features are. The weight of the window size can be adjusted by the parameter β_h of Eq. (4). To simplify the selection of the parameter β_h, we set an original value β and assign the h-th power of β to the parameter β_h. Figs. 3 and 4 show the evaluation results on the two training sets for parameter β when the parameter $h^* = 2$. The system reaches the best performance when the parameter $\beta = 0.3$ on the ML-2013 dataset and $\beta = 0.8$ on the DB-2013 dataset. Fig. 5 shows that the best result is achieved when the parameter $\beta = 0.4$ on the union set of the ML-2013 and DB-2013 datasets.

3.3. Performance comparisons with other graph kernels

Our context-vector graph kernel is a special all-path graph kernel when the context window size $h = 0$. Table 3 shows that the context-vector graph kernel outperforms the all-path graph kernel. The all-path graph kernel only calculates attributes of a vertex itself and ignores contextual information (neighboring attributes of the vertex). On the contrary, our approach not only allows for contexts but also represents the contexts in a vector way. The relations among close-range tokens are embodied by linear contexts, which are sequences representing token pairs before and after a token, while the relations among long-range tokens are shown by dependency contexts, which are sequences representing either relations around a token node or tokens around a dependency node. Moreover, the iteration from context vectors of adjacent nodes obtains contextual information around a token within the window size h as well as farther distances. Vectorial representation of contexts provides more overall representation for the words and syntactic structures around tokens. Furthermore, the feature space of model learning is built on any context-vector pairs after the partition of equivalent classes, instead of being limited to context vectors alone. Table 4 gives the contribution of different context types to the performance of our system. The feature Spc reduces the irrelevant noise to helpful contexts to raise the F-score for long complicated sentences, whereas it does not have too much impact on the performance of the DDI extraction from short sentences.

Compared with the HSP graph kernel based on hash operation with high recall, the context vector-graph kernel is a more expressive and holistic approach. As a result, our precision on the ML-2013 and DB-2013 datasets in Table 3 is significantly higher than that of the HSP graph kernel, especially on the ML-2013 dataset. On the whole, the F-measure of both datasets is rising.

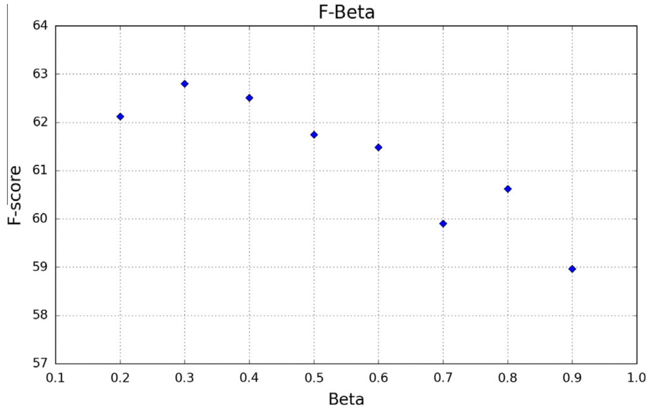


Fig. 3. Evaluation for the parameter β on the ML-2013 training dataset.

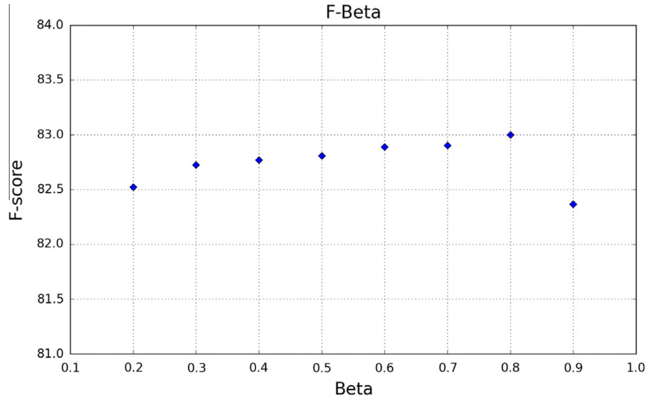


Fig. 4. Evaluation for the parameter β on the DB-2013 training dataset.

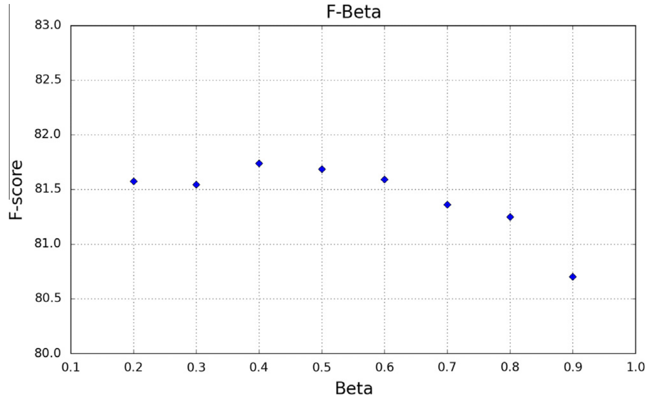


Fig. 5. Evaluation for the parameter β on the all training datasets which combine the ML-2013 dataset with the DB-2013 dataset.

Table 3
Performance comparisons (F-score) with other graph kernels on the ML-2013 and DB-2013 test datasets. The highest score are highlighted in bold.

Dataset	Graph kernel	P (%)	R (%)	F (%)
ML-2013	All paths	46.8	**69.4**	55.9
	HSP	54.6	68.4	60.7
	Context Vector	**67.9**	60.0	**63.7**
DB-2013	All paths	75.6	**83.3**	79.3
	HSP	81.9	82.1	82.0
	Context Vector	**85.7**	80.7	**83.1**

In addition, we also consider the computational complexity of the context-vector graph kernel. In the algorithm CCV, the computation of context vectors requires the time $O(n^2)$ for all nodes of a graph. Furthermore, the runtime of this system is mainly spent on the computation of the similarity matrix, which is accomplishable within the time $O(n^3)$. Therefore, our approach has the same time complexity $O(n^3)$ as the all-path and the HSP graph kernel do. Nevertheless, because a vertex is labeled with context vectors, the runspeed of this system is a bit slower than theirs.

Recently a number of studies that compared the assessment of DDIs with an electronic database and a clinician's assessment showed a large discrepancy in the number and relevance of detected DDIs [33]. As a result, a system with high precision is especially advantageous when extracting knowledge from large corpora such as the MEDLINE database, because it avoids overloading end users with too many false positives [27]. Our graph kernel, with its characteristic of high precision, makes full use of tokens them self and their contexts and improves the performance of the DDI extraction system.

3.4. Performance comparisons with top-ranking systems

To evaluate our approach, we compare this system with top-ranking systems in the DDI task. Table 5 shows that our system achieves the best overall performance (81.8% F) on the union set of the ML-2013 and DB-2013 test datasets. To distinguish the performance on different types of documents, results on the ML-2013 and DB-2013 datasets are presented in Tables 6 and 7, respectively.

3.4.1. The performance on the ML-2013 dataset

Table 6 gives the performance comparisons between our system and the top-ranking systems on the ML-2013 test dataset. Our system outperforms other systems with an F-score increase ranging from 4.5 to 26.1 points. It should be noted that our performance has a significant improvement on the ML-2013 dataset. Texts of the ML-2013 dataset, which are characterized by very scientific language from the abstracts of the MEDLINE database, have relatively longer and more complex subordinated structural sentences than those of the DB-2013 dataset. Thus some associated contexts have longer spans for a token. Therefore, it is important for the ML-2013 dataset to capture the relations among long-range tokens, which hasn't been resolved well in the top-ranking systems. Our approach exploits the representation of the dependency relation and the iterative calculation of context vectors; thereby the relations among not only long-range tokens but also close-range tokens in the long sentences all are captured sufficiently. Moreover, there may be more nodes and relatively more complex structures in some corresponding sentence graphs on the ML-2013 dataset. Because vectorial representation of contexts is a more holistic expression that obtains all of the relevant information in a direct or indirect way in addition to the topological structure, our approach conveys the information of a graph elaborately.

Table 4
Performance changes by varying context types in the DDI detection task. The baseline feature includes all context vectors without context-vector pairs. The pairs feature denotes context-vector pair feature. The Spc feature denotes context vectors of vertices on the shortest path between a candidate drug pair.

Dataset	Feature	P (%)	R (%)	F (%)
ML-2013	(1): Baseline	46.2	64.2	53.7
	(2): (1)+pairs	61.0	64.2	62.5
	(3): Spc + pairs	67.9	60.0	63.7
DB-2013	(1): Baseline	81.6	80.0	80.8
	(2): (1)+pairs	83.7	82.8	83.2
	(3): Spc + pairs	85.7	80.7	83.1

Table 5
Performance comparisons (*F*-score) with top-ranking systems on all test datasets which combine the ML-2013 dataset with the DB-2013 dataset. The highest score are highlighted in bold.

Team	*P* (%)	*R* (%)	*F* (%)
Context Vector	**85.8**	78.2	**81.8**
FBK-irst	79.4	**80.6**	80.0
WBI	80.1	72.2	75.9
SCAI	74.8	66.6	70.4
UTurku	83.3	60.2	69.9

Table 7
Performance comparisons (*F*-score) with top-ranking systems on the DB-2013 test dataset. The highest score are highlighted in bold.

Team	*P* (%)	*R* (%)	*F* (%)
Context Vector	85.7	80.7	83.1
BioSem	**85.9**	81.2	**83.5**
FBK-irst	81.6	**83.8**	82.7
WBI	81.4	75.5	78.3
SCAI	79.6	68.1	73.4
UTurku	84.3	63.8	72.6

Furthermore, context-vector pairs based on the partition of equivalent classes enhance the expression of the feature space.

In contrast, contexts of tokens, especially among long-range tokens, haven't been used effectively by the other systems. In the BioSem system, for example, each candidate DDI pair is considered independently, which is taken out of context. They divide DDI pairs that span over more than one single clause into multiple single clauses. In fact, the number of sentences with subordinate clauses is more than 25% of the total 4023 interacting drug–drug pairs for the ML-2013 dataset. Although it is easy to learn a model from simplified sentences, this partition may result in difficulty detecting DDIs for candidate pairs spanning over the subordinate clauses. Thus, further performance improvement might be limited on DDI tasks. The FBK-irst system only uses the limited contexts of the Shallow Linguistic kernel. The UTurku system [11] uses sentence features, dependency features and informatics from external domains, such as the DrugBank and MetaMap model systems. But Table 4 shows that the feature of vector pairs, which isn't exploited by them, enhances the *F*-score 8.8 points in the ML-2013 test dataset. As a result, our experiments seem to manifest that the performance of DDI tasks can be improved significantly if contexts are appropriately exploited for long and complex sentences built on a better structured representation.

3.4.2. The performance on the DB-2013 dataset

The results in Table 7 indicate that our system achieves a comparable *F*-score (83.1%) on the DB-2013 test dataset. The performance has little improvement on the DB-2013 dataset. The possible reason may be because of the following factors. The DB-2013 dataset, which contains short and concise uncomplicated sentences well-curated manually, has limited room for performance improvements. In addition, even if contexts haven't been used sufficiently, it is also enough to obtain effective features to identify the relations between drug pairs by combining various text-mining techniques. The FBK-irst system and the WBI system [8] each benefit from a hybrid kernel system based on three kernels. This FBK-irst system still uses other techniques such as filtering techniques and parameter selection techniques. Thus this system seems to be a little intricate. The SCAI system [9] combines three machine learning techniques: linear SVM, Naive Bayes and Voting Perceptron. The BioSem system, based on chunks, uses lexical features, phrase and syntactic features besides some filtering techniques. However, the experiments of Segura-Bedmar et al. [17] demonstrate that a multiple-kernel combined system is difficult to design for further performance improvement on DDIs tasks.

Table 6
Performance comparisions (*F*-score) with top-ranking systems on the ML-2013 test dataset. The highest score are highlighted in bold.

Team	*P* (%)	*R* (%)	*F* (%)
Context Vector	**67.9**	60.0	**63.7**
BioSem	67.6	52.6	59.2
FBK-irst	55.8	50.5	53.0
WBI	62.5	42.1	50.3
SCAI	38.7	**63.0**	47.9
UTurku	65.8	26.3	37.6

In general, the top-ranking systems perform the tasks of relation extraction by exploiting either composite kernels or domain resources. In addition, these systems still use some filtering techniques to rule out irrelevant negative instances for the sake of the balance of the bias dataset. Considering that our features come exclusively from training data and the model is learned automatically from all candidate instances without external resources, our approach may be more generalizable to other relation extraction tasks.

3.4.3. The analysis of different performances on the ML-2013 and DB-2013 datasets

The results in Tables 6 and 7 indicate that there is almost 19 points *F*-score visible difference between the DB-2013 dataset and the ML-2013 dataset. One reason may be the different size of the two corpora. Table 2 shows that the ML-2013 training dataset is approximately 14 times smaller than the DB-2013 dataset. Learning a model from a small corpus is one of the challenges to machine-learning-based approaches. Another possible reason that might affect the system performance is the above-mentioned different document style of the two datasets. Their distinction can be seen obviously from Table 4, and this system achieves diverse evaluation results on the two datasets in spite of the identical feature-type Spc.

However, because the literature of the MEDLINE database, as one of major data sources for biomedical text-mining, has its particular linguistic style, further research for upcoming relation extraction tasks on it must continue to be carried out. Our system promotes the performance improvement of detecting DDIs on the ML-2013 dataset. Therefore, it might be a significant advance for biomedical text-mining on scientific literature.

3.5. DDI type classification

The classification of DDIs is a multiclass problem. Our approach takes two steps to conduct the task. One of four DDI types (*mechanism*, *effect*, *advice*, *int*) is assigned to the interacting drug pairs extracted.

Two popular ways to address a multiclass issue using binary SVM classifiers are the one-versus-all and one-versus-one strategies. The one-versus-all constructs k SVM classifiers, where k is the number of classes. The ith SVM is trained on the instances in the ith class with positive labels, and all other instances with negative labels. The one-versus-one builds a classifier for each pair of classes and all together $k(k-1)/2$ binary classifiers are constructed. For DDI type classification, FBK-irst uses a binary SVM with a one-versus-all method. WBI and UTurku use a multi-class SVM. Kim et al. [18] uses the one-versus-one strategy.

In our approach, the one-versus-all method is used to classify the extracted DDIs into different categories. We train 4 separate models. An extracted DDI instance receives the class label of the model which has the highest confidence score for this instance. If none of models assign a class label to a predicted DDI instance, a default label (one of four types) is assigned to it.

Tables 8 and 9 show the best multiclass performance on the two datasets when the default label is "*mechanism*" for the DB-13 dataset and "*effect*" for the ML-13 dataset, respectively. It is seen from Table 10 that our approach achieves the best *F*-score for the detection and classification task of DDIs on the union of the two test datasets.

3.6. Error analysis

To provide a road map for future work in the extraction of DDIs from texts, we analyze the main sources of errors produced by our system. There are not errors such as negation expressions errors, coordinate structures errors and appositions errors in our system. These errors remain currently one of the principal error sources in DDI tasks. Our system rules out these sentences by learning a model instead of using a filtering technique. The expressive representation of contextual information may be the possible reason.

Most of the other errors are non-trivial. Our errors can be categorized into five groups. Table 11 gives the main causes and examples for the false negatives in the DB-2013 dataset and ML-2013 dataset. One of the important factors contributing to false negatives on the DB-2013 dataset (nearly a quarter of the errors) is caused by complex sentences with the anaphora and cataphora description. But this error doesn't appear in false negatives on the ML-2013 dataset. A possible reason may be the difference in the linguistic structures of texts on the two datasets. Many texts of the DB-2013 corpus include a list of drugs or linguistic phenomena used as anaphora and cataphora.

The following reasons result in the second group of errors which exist in some long, complex and compound sentences. Some unusual text patterns are included in the test dataset, but they haven't been learned in the training dataset. In addition, the dependency parser needs to further improve the parsing accuracy for long and complicated sentences. The third error group comes from the errors of dependency parsing for sentences. Therefore, these two error sources pose a serious challenge to the performance of the dependency parser.

Table 8
Performance comparisons (*F*-score) of DDIs classification with top-ranking systems on ML-2013 test dataset. "DEC" indicates detection performance. "CLA" indicates detection and classification performance for all classes. "MEC", "EFF", "ADV" and "INT" are for mechanism, effect, advice and int types, respectively. The highest score are highlighted in bold. The symbols of the following tables have the same meaning.

Team	DEC	CLA	MEC	EFF	ADV	INT
ContextVector	**63.7**	**52.0**	40.0	**53.6**	**57.1**	50.0
FBK-irst	53.0	39.8	38.3	43.6	28.6	**57.1**
Kim	47.1	38.2	**45.5**	35.2	42.9	25.0

Table 9
Performance comparisons (*F*-score) of DDIs classification with top-ranking systems on DB-2013 test dataset.

Team	DEC	CLA	MEC	EFF	ADV	INT
ContextVector	**83.1**	**72.4**	69.2	**76.4**	**75.4**	**59.5**
Kim	80.4	69.8	**71.4**	70.6	73.6	49.7
FBK-irst	82.7	67.6	70.5	69.9	70.5	54.5

Table 10
Performance comparisons (*F*-score) of DDIs classification with top-ranking systems on all test datasets which combine the ML-2013 test dataset with the DB-2013 test dataset.

Team	DEC	CLA	MEC	EFF	ADV	INT
Context-Vector	**81.8**	**68.4**	66.9	**71.3**	71.4	51.6
Kim	77.5	67.0	**69.3**	66.2	**72.5**	48.3
FBK-irst	80.0	65.1	67.9	62.8	69.2	**54.7**
WBI	75.9	60.9	61.8	61.0	63.2	51.0
UTurku	69.9	59.4	58.2	60.0	63.0	50.7

Table 11
Analysis and examples of false negative instances in the DB-2013 and ML-2013 datasets.

ID	Error cause	Example	DDIs not detected
E1	Complex sentences with anaphora and cataphora	Drugs such as **erythromycin**$_{e0}$, **diltiazem**$_{e1}$, **verapamil**$_{e2}$, **ketoconazole**$_{e3}$, **fluconazole**$_{e4}$ and **itraconazole**$_{e5}$ were shown to significantly increase the C max and AUC of orally administered **midazolam**$_{e6}$	(e0,e6) (e1,e6) ... (e5,e6)
E2	Complex and compound sentences	During maintenance of anesthesia or sedation, the rate of **DIPRIVAN**$_{e0}$ injectable emulsion administration should be adjusted according to the desired level of anesthesia or sedation and may be reduced in the presence of supplemental **analgesic agents**$_{e1}$ (e.g., **nitrous oxide**$_{e2}$ or **opioids**$_{e3}$)	(e0,e1) (e0,e2) (e0,e3)
E3	Dependency parsing error	Interactions for **Vitamin B1**$_{e0}$ (**Thiamine**$_{e1}$): **Loop Diuretics**$_{e2}$, Oral **Contraceptives**$_{e3}$, **Stavudine**$_{e4}$, **Tricyclic Antidepressants**$_{e5}$	(e0,e2) (e0,e3) ... (e0,e5)
E4	The error of the short distance between entities	HEY cells treated with **dasatinib**$_{e0}$ plus **paclitaxel**$_{e1}$ formed fewer colonies than did cells treated with either agent alone	(e0,e1)

The fourth error source of false negatives only appears in the ML-2013 dataset and accounts for around one-third of 38 false negative instances. The wrong detection of DDIs occurs in sentences where two entities are described in the following ways, such as DRUG1/DRUG2, DRUG1 and DRUG2, DRUG1 or DRUG2, and DRUG1 plus DRUG2 (see E4 in Table 11). The distance between the two entities is too small to obtain enough information on detecting DDIs. Most systems solve the problem by rules-based methods or the expansion of the sentential structure at present. The last group of errors consists of cases where interesting DDI pairs syntactically seem to be true DDI pairs.

4. Conclusion

In this work, we apply context vectors to a graph kernel for detecting and classifying DDIs from biomedical texts. The proposed method focuses on the effective use of the different types of contexts and the relations among words with different distances. Based on the graph representation of a sentence for the relation among the close-range and long-range words, the context vectors calculated iteratively from adjacent labeled nodes obtain rich and expressive surrounding features of tokens of sentences. Each word of a sentence is represented as multiple vectors according to the subgraph type, its neighbor, different window sizes and the weight of an edge. Based on equivalent classes with the same context vectors, the grouping of vertices minimizes the representation of a graph. Furthermore, the diverse distance between context vectors in the same layer are exploited to embody the connection strength. Applied to the DDIExtraction 2013 corpus, our system without multiple kernels and external resources achieves the best detection and classification performance in existing top-ranking DDIs systems.

The proposed method provides a valid description of how elaborate and expressive contextual information in a sentence are obtained effectively. Our approach captures the relations among not only close-range tokens but also long-range tokens. Moreover, the relations involve not only the direct but also various indirect syntactic contexts. Experiments indicate that our approach plays a role in the tasks of relation extraction from scientific literature containing long and complex sentences. In addition, it is also an effective alternative for classification tasks with the characteristic of more nodes and attributions, especially for such systems with higher precision requirements.

As for future work, the following factors may be considered. Structured representation may be extended to further embody the syntactic characteristic of complicated sentences. In addition, word embedding and the knowledge base may be integrated into the vectors.

Conflict of interest

There is no conflict of interest.

Acknowledgment

Authors gratefully acknowledge the financial support provided by the Natural Science Foundation of China under Nos. 61572102, 61340020, 61272004, 61272373, 61300088 and 61502071.

References

[1] S. Eltyeb, N. Salim, Chemical named entities recognition: a review on approaches and applications, J. Cheminf. 6 (2014) 6–17.
[2] B. Percha, R.B. Altman, Informatics confronts drug–drug interactions, Trends Pharmacol. Sci. 34 (2013) 178–184.
[3] B. Percha, Y. Garten, R.B. Altman, Discovery and explanation of drug–drug interactions via text mining, in: Pacific Symp. Biocomput., 2012, pp. 410–421.
[4] I. Segura-Bedmar, P. Martınez, D. Sánchez-Cisneros, 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, 761, 2011, pp. 1–9.
[5] I. Segura Bedmar, P. Martínez, M. Herrero Zazo, Semeval-2013 task 9 extraction of drug–drug interactions from biomedical texts (ddiextraction 2013), in: Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013
[6] María Herrero Zazo, Isabel Segura-Bedmar, Paloma Martínez, Thierry Declerck, The DDI corpus: an annotated corpus with pharmacological substances and drug–drug interactions, J. Biomed. Inform. 46 (5) (2013) 914–920.
[7] A. Ben Abacha, M.F.M. Chowdhury, A. Karanasiou, Y. Mrabet, A. Lavelli, P. Zweigenbaum, Text mining for pharmacovigilance: using machine learning for drug name recognition and drug–drug interaction extraction and classification, J. Biomed. Inform. 58 (2015) 122–132.
[8] P. Thomas, M. Neves, T. Rocktäschel, U. Leser, WBI-DDI: drug–drug interaction extraction using majority voting, in: Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013, pp. 628–635.
[9] P. Thomas, M. Neves, T. Rocktäschel, U. Leser, WBI-DDI: drug–drug interaction extraction using majority voting, in: Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013, pp. 675–83.
[10] J. Björne, A. Airola, T. Pahikkala, T. Salakoski, Drug–drug interaction extraction from biomedical texts with svm and rls classifiers, in: Proceedings of DDIExtraction-2011 Challenge Task, 2011, pp. 35–42.
[11] J. Björne, S. Kaewphan, T. Salakoski, UTurku: drug named entity recognition and drug–drug interaction extraction using SVM classification and domain knowledge, in: Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013, pp. 651–659.
[12] A. Airola, S. Pyysalo, J. Bjorne, T. Pahikkala, F. Ginter, T. Salakoski, All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning, BMC Bioinf. 9 (Suppl 11) (2008) S2.
[13] M.F.M. Chowdhury, A. Lavelli, FBK-irst: a multi-phase kernel based approach for drug–drug interaction detection and classification that exploits linguistic information, in: Seventh International Workshop on Semantic Evaluation (SemEval 2013), 2013, pp. 351–355.
[14] T. Gärtner, A survey of kernels for structured data, ACM SIGKDD Explorations Newsletter 5 (2003) 49–58.
[15] M. Miwa, R. Sætre, Y. Miyao, J.I. Tsujii, Protein–protein interaction extraction by leveraging multiple kernels and parsers, Int. J. Med. Inf. 78 (2009) e39–e46.
[16] Z. Yang, N. Tang, X. Zhang, H. Lin, Y. Li, Z. Yang, Multiple kernel learning in protein–protein interaction extraction from biomedical literature, Artif. Intell. Med. 51 (2011) 163–173.
[17] I. Segura-Bedmar, P. Martínez, M. Herrero-Zazo, Lessons learnt from the DDIExtraction-2013 shared task, J. Biomed. Inform. 51 (2014) 152–164.
[18] S. Kim, H. Liu, L. Yeganova, W.J. Wilbur, Extracting drug–drug interactions from literature using a rich feature-based linear kernel approach, J. Biomed. Inform. 55 (2015) 23–30.
[19] D. Tikk, I. Solt, P. Thomas, U. Leser, A detailed error analysis of 13 kernel methods for protein-protein interaction extraction, BMC Bioinf. 14 (2013).
[20] Q.-C. Bui, P.M. Sloot, E.M. van Mulligen, J.A. Kors, A novel feature-based approach to extract drug–drug interactions from biomedical text, Bioinformatics (2014). btu557.
[21] Y. Miyao, K. Sagae, R. Sætre, T. Matsuzaki, J.i. Tsujii, Evaluating contributions of natural language parsers to protein–protein interaction extraction, Bioinformatics 25 (2009) 394–400.
[22] L. Bai, P. Ren, X. Bai, E.R. Hancock, A Graph Kernel from the Depth-Based Representation, in: Structural, Syntactic, and Statistical Pattern Recognition, Springer, 2014, pp. 1–11.
[23] V.C. Ostuni, T. Di Noia, R. Mirizzi, E. Di Sciascio, A linked data recommender system using a neighborhood-based graph kernel, in: E-Commerce and Web Technologies, Springer, 2014, pp. 89–100.
[24] S. Kim, J. Yoon, J. Yang, S. Park, Walk-weighted subsequence kernels for protein-protein interaction extraction, BMC Bioinf. 11 (2010) 107.
[25] Z. Yijia, L. Hongfei, Y. Zhihao, W. Jian, L. Yanpeng, Hash subgraph pairwise kernel for protein-protein interaction extraction, IEEE/ACM Trans. Comput. Biol. Bioinf. 9 (2012) 1190–1202.
[26] P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, J.-P. Vert, Extensions of marginalized graph kernels, in: Proceedings of the Twenty-first International Conference on Machine Learning, 2004, pp. 70–77.
[27] I. Segura-Bedmar, P. Martinez, C. de Pablo-Sánchez, Using a shallow linguistic kernel for drug–drug interaction extraction, J. Biomed. Inform. 44 (2011) 789–804.
[28] Y. Goldberg, J. Nivre, Training deterministic parsers with non-deterministic oracles, Trans. Assoc. Comput. Linguistics 1 (2013) 403–414.
[29] O. Levy, Y. Goldberg, Dependency-based word embeddings, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 302–308.
[30] J. Björne, T. Salakoski, Generalizing biomedical event extraction, in: Proceedings of the BioNLP Shared Task 2011 Workshop, 2011, pp. 183–191.
[31] M.C. De Marneffe, B. MacCartney, C.D. Manning, Generating typed dependency parses from phrase structure parses, in: Proceedings of LREC; 2006, pp. 449–454.
[32] T. Gärtner, P. Flach, S. Wrobel, On graph kernels: hardness results and efficient alternatives, in: Learning Theory and Kernel Machines, Springer, 2003, pp. 129–143.
[33] T. Roblek, T. Vaupotic, A. Mrhar, M. Lainscak, Drug–drug interaction software in clinical practice: a systematic review, Eur. J. Clin. Pharmacol. 71 (2015) 131–142.