Journal of Biomedical Informatics 44 (2011) 1086–1092

Contents lists available at SciVerse ScienceDirect

Journal of Biomedical Informatics

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

Neighborhood hash graph kernel for protein–protein interaction extraction

Yijia Zhang *, Hongfei Lin, Zhihao Yang, Yanpeng Li

School of Electronics and Information Engineering, Dalian University of Technology, Dalian, Liaoning 116023, China

ARTICLE INFO

Article history:
Received 15 December 2010
Accepted 17 August 2011
Available online 23 August 2011

Keywords:
Interaction extraction
Hash
Graph kernel
Biomedical literature

ABSTRACT

Automated extraction of protein–protein interactions (PPIs) from biomedical literatures is an important topic of biomedical text mining. In this paper, we propose an approach based on neighborhood hash graph kernel for this task. In contrast to the existing graph kernel-based approaches for PPI extraction, the proposed approach not only has the capability to make use of full dependency graphs to represent the sentence structure but also effectively control the computational complexity. We evaluate the proposed approach on five publicly available PPI corpora and perform detailed comparisons with other approaches. The experimental result shows that our approach is comparable to the state-of-the-art PPI extraction system and much faster than all-path graph kernel approach on all five PPI corpora.

1. Introduction

With the exponential explosion of biomedical literature, information extraction from biomedical literature has been a topic of intense research. Automated protein–protein interaction (PPI) extraction from biomedical literature is an important task in biomedical text mining, which contributes to PPI network analysis and discovery of new functions of proteins. A lot of research interests [1–10] have been reported for this task during recent years. The existing PPI methodologies rely on different approaches broadly divided into three main categories: co-occurrences engineering, pattern engineering and machine learning. With the gradual maturity of the kernel method, machine learning based approach has got the advantage of the performance over the others. In particular, the recent studies [6–10] show most state-of-the-art systems are in the framework of machine learning.

Machine learning based approach for PPI extraction usually tackles the task as a classification problem. A major challenge is how to supply the learner with the semantic/syntactic information needed to distinguish between interactions and non-interactions [6]. Therefore, kernel methods are required by PPI extraction system, which can learn rich structural data such as syntactic parse tree or dependency graph.

However, the existing kernel methods exploit only limited information of the syntactic parse tree or dependency graph and are still computationally expensive. The walk-weighted subsequence kernels [7] match the e-walk and v-walk on the shortest path of the full dependency graph, which can only represent the semantic/syntactic information of the shortest path. The tree kernels [1] can represent more complex structures, but the tree representations are still not enough to completely represent all semantic/syntactic information of the dependency graph. All-paths graph kernel [6] maps the dependency graph into the label pairs feature space, but it cannot match the label sequence on every path of dependency graph. Furthermore, these methods are computationally expensive, particularly when computing the complex dependency graphs.

In this paper, we proposed a neighborhood hash kernel based method for PPI extraction. The framework of neighborhood hash kernel is proposed by Hido [11]. To the best of our knowledge, we first apply the neighborhood hash kernel to the task of PPI extraction. Firstly, we use a mapping function to transform each node label of dependency graph into a bit label which is represented as a binary array of fixed length. Secondly, we replace the bit label of node by a new bit label produced by order-independent logical operations on the bit labels of the node and the neighboring nodes. Updating the node label with the new bit label by the specific logical operation, such as *XOR*, allow us to combine the neighborhood structure into the updated label. We apply this procedure to all of the nodes in dependency graph to exchange the semantic/syntactic information between connected nodes, and repeat several times to propagate the features of the high order substructures over the dependency graph. Finally, we can efficiently compute the similarity of the two dependency graphs based on the intersection ratio of the updated label sets. As the neighborhood hash kernel can calculate the full dependency graphs, our method can reduce the risk of missing important features. Furthermore, the neighborhood hash kernel is a linear time and the overall computational complexity of our method is only $O(n^2 + DR\bar{d}n)$ which is significantly lower than other kernel methods.

* Corresponding author.
E-mail address: zhyj@dlut.edu.cn (Y. Zhang).

1532-0464/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.jbi.2011.08.011

The rest of the paper is organized as follows: we first give a brief introduction of neighborhood hash graph kernel, and then describe in details the neighborhood hash graph kernel approach for protein–protein interaction extraction (PPIE). Next, we investigate the performance of the neighborhood hash graph kernel approach on five publicly available PPI corpora. Finally, we conclude and present our future plan.

2. Method

2.1. Neighborhood hash graph kernel

In recent years, the kernel-based methods, such as support vector machine (SVM), have been successfully applied to almost every task in machine learning, which can be transformed to feature vectors. Graph is a widely used tool for modeling structured data in computer science, but it is generally difficult to transform graphs to feature vectors without loss of the structural information contained in the graphs. However, graph kernels can provide a way to apply the standard kernel-based algorithms such as SVM to graph learning.

We next introduce neighborhood hash graph kernel in brief. This kernel can be considered as a practical instantiation which applies the theoretical graph kernel framework [11] to directed vertex-labeled graph.

Let V be a set of vertices (or nodes) and E be a set of edges (or links). Then, a graph $G = (V,E)$ is called a directed graph if E is a set of directed edges $E \subset V \times V$.

Definition 1 (vertex-labeled graph) Let L be a set of labels (or attributes) and $M \subset V \times L$ be label allocations. Then, $G = (V,E,L)$ is called a vertex-labeled graph.

We denote a label as a binary array consisting of D-bits (0 or 1), such as $s = \{b_1, b_2, \ldots, b_D\}$. Where the constant D satisfies $2^D - 1 \gg |\sum|$. It can represent an unsigned integer value up to $2^D - 1$, and the node labels $\sum$ are in a finite set of discrete values. In our experiments, we denote all labels as 24-bit labels ($D = 24$). Let $XOR(s_i, s_j) = s_i \oplus s_j$ denote the XOR operation between two bit label s_i and s_j that produces another binary array with each bit representing the XOR value for each digit, and let $ROT_o(s) = \{b_{o+1}, b_{o+2}, \ldots, b_D, b_1, \ldots, b_o\}$ denote the ROT_o operation for $s = \{b_1, b_2, \ldots, b_D\}$ shifts the last $D - o$ bits to the left by o bits, and moves the first o bits to the right end.

We compute a neighborhood hash of a vertex-labeled graph using XOR and ROT. Firstly, we obtain the set of adjacent nodes $V^{adj}(v)$ of every node v. Secondly, we calculate a neighborhood hash for the node v by the following Eq. (1). Where $l(v)$ denote the bit label of node v. If the edge v_1 v is an out-going edge of node v, let $ROT_{edge1} = ROT_2$, and if the edge v_1 v is an in-coming edge of node v, let $ROT_{edge1} = ROT_3$.

$$NH(v) = ROT_1(l(v)) \oplus \left(ROT_{edge1}\left(l\left(v_1^{adj}\right)\right) \oplus \cdots \oplus ROT_{edged}\left(l\left(v_d^{adj}\right)\right)\right) \quad (1)$$

Fig. 1 shows an example of how the neighborhood hash works for a node given a set of directed connected nodes and labels. For simplicity we denote that their labels as 4-bit arrays.

We replace the label of a node v with the neighborhood hash value $NH(v)$, and obtain a new graph $G^1 = \{V,E,l^1(\cdot)\}$, where $l^1(v) = NH(v)$ for all nodes. This update aggregates the information of the neighborhood nodes to each node. We denote this operation as the neighborhood hash function to a graph, that is, $G^1 = NH(G^0)$, where G^0 is G. The neighborhood hash can be applied iteratively as $G^{r+1} = NH(G^r)$, where $l^{r+1}(v) = NH(v^r)$. Since the updated node label $l^r(v)$ represents the label distribution of the r-neighbors of node v on G, two node v_i and v_j in different graphs will have the same label distribution of the r-neighbors if $l^r(v_i) = l^r(v_j)$. Therefore, we can efficiently compute the similarity of two graphs G_a and G_b by comparing the set of bit labels of $\{G_a^0, \ldots, G_a^r\}$ and $\{G_b^0, \ldots, G_b^r\}$.

2.2. PPI extraction based on neighborhood hash kernel

In most recent work on machine learning for PPIE [6–10], PPI extraction has been targeted the task as the extraction of binary interactions, where the system identifies whether each unordered candidate protein pair in a sentence has a biologically relevant relationship. In the following, we first give our graph representation, and then describe the neighborhood hash graph kernel approach in detail.

2.2.1. Graph encoding of sentence structure

For representing an interaction candidate pair, we first give the graph representation. We assume that the sentences have been parsed by dependency parser tools and the candidate protein pairs have been marked. Based on this, we represent each sentence as a directed vertex-labeled graph that consists of two unconnected subgraphs. One represents the dependency structure of the sentence and the other represents the linear order of the sentence. In contrast to the work [6], both dependency subgraph and linear subgraph are unweighted.

Fig. 2 is an example of graph representation generated from a sentence (LLL.d18.s0). We represent a sentence with dependency subgraph and linear subgraph. Dependency subgraph represents the dependency structure of the sentence which is built from the dependency analysis, and linear subgraph represents the linear order of the sentence. In the graph, every node has a label for each token and dependency. In Fig. 2, PROT1, PROT2 and PROT denote protein names respectively, where PROT1 and PROT2 are the pair

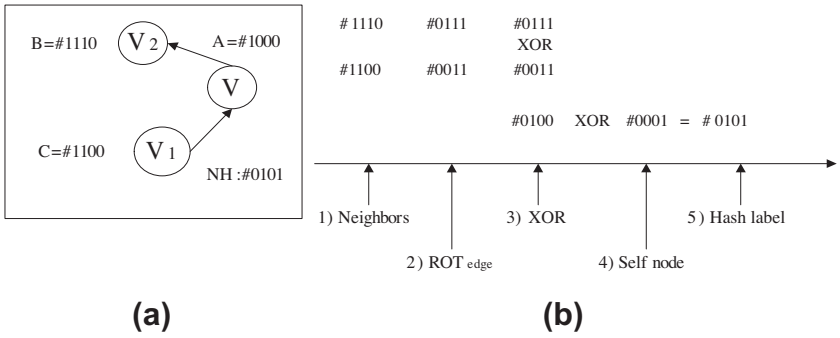

Fig. 1. An example of neighborhood hash. (a) A node v having two neighbors. (b) The procedure to compute the neighborhood hash for v using XOR and ROT.

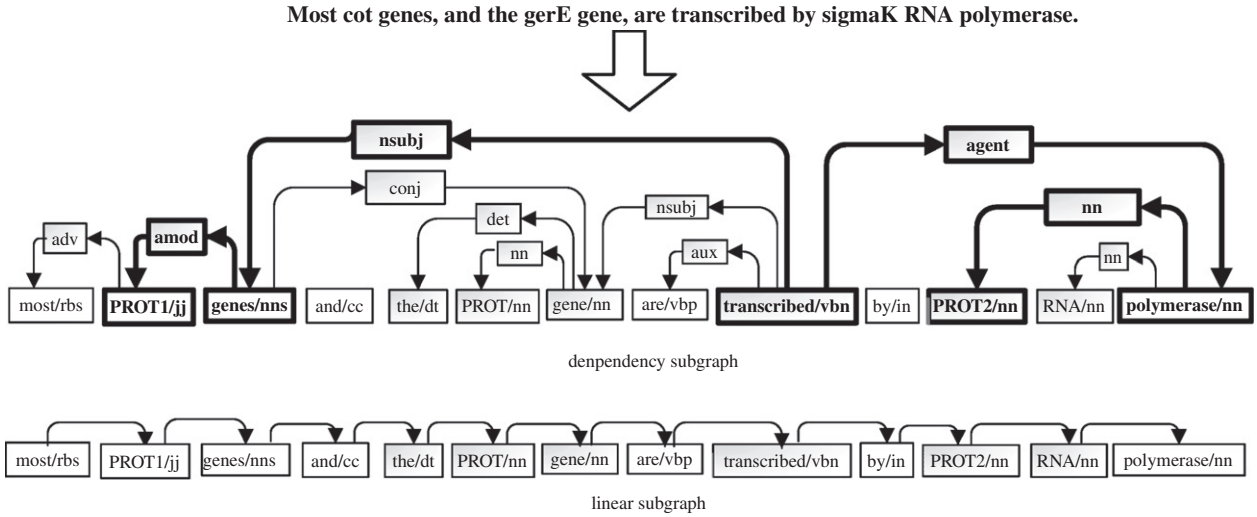

Fig. 2. An example of graph representation.

of interest. In the dependency subgraph, the shortest path between the proteins is shown in bold.

2.2.2. Neighborhood hash of graph representation

We handle dependency subgraph and linear subgraph using neighborhood hash kernel. Firstly, we replace the label of every node in dependency subgraph and linear subgraph with a binary array consisting of D-bits (0 or 1) using a one-to-one mapping function between the original labels and the random binary arrays of D length. For the task of PPI extraction, the quantity of labels is very large. To reduce the problematic hash collisions, we choose $D = 24$ in our experiments. We denote the bit label set as $l(\cdot)$. Secondly we compute the neighborhood hash value $NH(v)$ of every node v by the Eq. (1), and replace the bit label $l(v)$ with $NH(v)$. Thirdly, we iteratively compute G^{r+1} as $G^{r+1} = G^r$, where $l^{r+1}(v) = NH(v^r)$.

Fig. 3 shows that how compute G^r from G^0. Note that the bit labels are represented by hex forms and the initial bit labels of G^0 is generated by a random function.

2.2.3. Calculation of the kernel matrix

We next calculate the kernel matrix using the set of the graphs updated by the neighborhood hash. Let Γ be the set of the graph and R be the maximum order of the neighborhood hash. Further let $K^r \in \Re^{|\Gamma|\times|\Gamma|}$ $(0 \leqslant r \leqslant R)$ denote the similarity matrix of r iterations which is initialized as I (a unit matrix) and K^r_{ij} is the similarity of two graphs G^r_i and G^r_j. The similarity of two graphs (G^r_i and G^r_j) depends on the number of same labels. For the task of PPI extraction, the studies [6,7] show that the shortest path of candidate proteins in dependency subgraph has more important distinguishing information. Therefore, we see the dependency subgraph and linear subgraph as a whole graph and set the different similarity weight for the shortest path nodes and the normal nodes. Note that the normal nodes include the linear subgraph nodes and the rest nodes of dependency subgraph which removes the shortest path nodes. We define the factor of similarity weight as $\lambda = \frac{weight_{sp\_node}}{weight_{nor\_node}}$, where $weight_{sp\_node}$ and $weight_{nor\_node}$ are the similarity weight of shortest path nodes and the normal nodes respectively. The values of $weight_{sp\_node}$ and $weight_{nor\_node}$ are positive integer, for example, we set $weight_{sp\_node} = 2$ and $weight_{nor\_node} = 1$ and we get $\lambda = 2$. Let η be the similarity of two graphs G^r_i and G^r_j. Thus, we use the following equation to calculate the similarity of two graphs. Eq. (2) is derived from the Jaccard–Tanimoto coefficient which is commonly used to compute the similarity metric between two sets of discrete values.

$$\eta = \frac{n + \lambda s}{n_i + n_j - n + \lambda(s_i + s_j - s)} \tag{2}$$

In Eq. (2), s_i and s_j respectively are the number of nodes which are on the shortest dependency paths of two graphs, n_i and n_j are the number of nodes which are on the normal paths of two graph, s is the number of nodes with the same label which are on the shortest dependency paths, and n is the number of nodes with the same label which are on the normal paths. λ is the factor of similarity weight and we set $\lambda = 2$ in our experiments. Based on the set $\{K^1, \ldots, K^R\}$, we calculate the similarity matrix K with the Eq. (3).

$$K = \frac{1}{R}\sum_{r=1}^{R} K^r \tag{3}$$

In our experiment, the kernel is combined with SVM. We use the SVM-light package which allows user-defined kernels and set the parameter c value as 4 for the best F-value.

2.2.4. Computational complexity

For completely computing the dependency graphs, existing PPI extraction systems based on graph kernel easily lead high computational complexity. In the work [12], Gärtner has proved that it is still NP-hard to compute an all-substructure-based kernel on a graph. All-paths kernel approach proposed by Airola et al. [6] is based on the graph kernel method proposed by Gärtner which computational complexity is $O(n^3)$ [13].Therefore, it is important for an efficient system based on graph kernel to control the complexity of graph kernel. Our approach involves accessing to the d neighboring nodes of a node v and the bit operations such as *XOR* and *ROT* for the $d + 1$ bit labels. If the fixed length D is no more than the bit size of the processor architecture (32 or 64), all bit operations can be done in one clock. In the work [11], Hido proves that the computation of the neighborhood hash requires $O(D\bar{d}n)$ for all of the nodes in a graph, where $\bar{d}$ is the average number of neighbors and n is the number of nodes of the graph. The computing similarity matrix is also can be done in linear time with the number of nodes. To calculate the shortest dependency path of graph using Dijkstra algorithm can be done in $O(n^2)$. Therefore, if the maximum iterations is R, the overall computational complexity

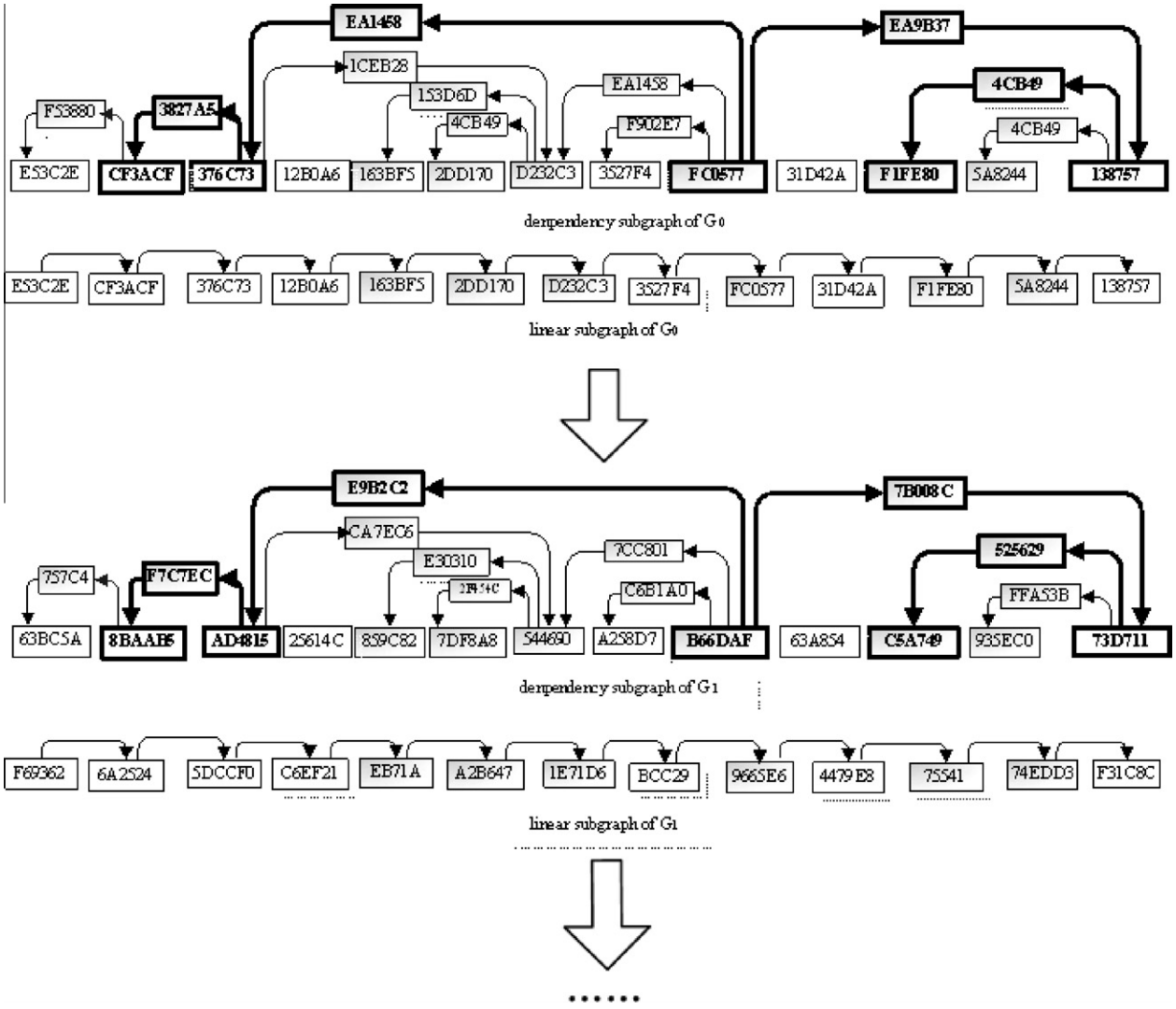

Fig. 3. An example that shows how compute G^r form G^0.

for the approach based on neighborhood hash kernel is only $O(n^2 + DR\bar{d}n)$.

3. Results and discussion

3.1. Corpora and evaluation criteria

We evaluate neighborhood hash graph kernel approach using five publicly available corpora that contain PPI interaction annotation: AIMED [2], BioInfer [4], HPRD50 [3], IEPA [14] and LLL [15]. In our experiment, we choose the converted version by [16] which is parsed with Charniak and Lease parser [17] and contains full automated dependency parsing results. The statistics of the five PPI corpora is listed in Table 1.

Table 1
Statistics of the five PPI corpora.

Corpus	Positive	Negative	All examples
AIMED	1000	4834	5834
BioInfer	2479	7174	9653
IEPA	335	482	817
HPRD50	163	270	433
LLL	164	166	330

To keep our evaluation metrics as the same as the recent works [6–10], we test our method with 10-fold document-level cross-validation on all of the corpora, where documents are divided into 10 groups, and one is used for testing and the others for training in each round. We use *F–M* as the primary evaluation measure, which is defined as $F = (2PR)/(P + R)$, where P denotes precision and R recall. In addition, we also report AUC which is invariant to the class distribution of the used dataset.

3.2. Performance on five PPI corpora

To evaluate the effect of λ in Eq. (2), we first fix $R = 1$ and test our approach on LLL and AIMED. Table 2 shows that the *F–M* is

Table 2
Evaluation results. R takes the fixed value 1 and λ takes the value from 0.5 to 4.

λ	LLL				AIMED			
	P	*R*	*F–M*	AUC	*P*	*R*	*F–M*	AUC
$\lambda = 0.5$	80.9	87.8	84.2	86.5	50.7	63.8	54.9	80.9
$\lambda = 1$	**85.4**	89.2	86.5	88.7	52.4	64.7	56.9	82.6
$\lambda = 2$	85.2	91.3	**88.1**	**89.9**	54.7	66.5	**59.0**	**84.5**
$\lambda = 3$	84.1	**91.4**	86.9	89.2	54.1	**67.4**	58.9	83.9
$\lambda = 4$	85.2	91.1	87.4	89.3	**55.6**	64.2	58.7	83.7

Table 3
Evaluation results. λ takes the fixed value and R takes the value from 1 to 4.

R	AIMED		BioInfer		IEPA		HPRD50		LLL	
	F–M	AUC	*F–M*	AUC	*F–M*	AUC	*F–M*	AUC	*F–M*	AUC
$R=1$	58.9	84.2	62.7	81.2	73.3	79.8	73.6	74.5	**89.4**	90.3
$R=2$	**60.2**	85.3	**63.4**	**82.7**	**75.3**	82.2	74.6	**77.8**	89.1	90.2
$R=3$	59.9	85.0	62.9	81.9	74.1	**82.4**	**75.0**	77.6	88.9	**90.4**
$R=4$	59.4	**85.7**	61.9	81.6	73.8	81.2	74.4	76.5	88.5	90.1

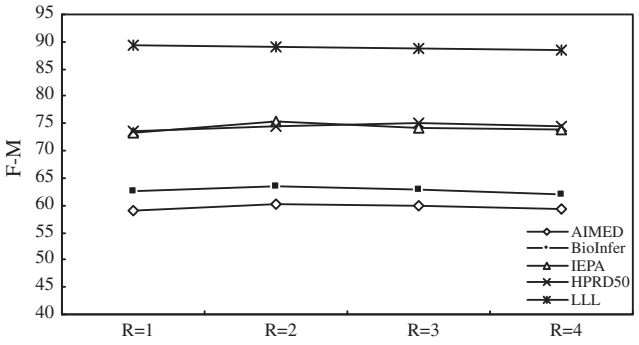

Fig. 4. Relation between the performance of neighborhood hash kernel approach and the value of *R*.

Table 4
Results on five corpora at $\lambda = 2$ and $R = 2$.

Corpus	Positive	Negative	*P*	*R*	*F–M*	AUC	Time (s)
AIMED	1000	4834	54.9	68.5	60.2	85.3	47
BioInfer	2479	7174	59.3	68.1	63.4	82.7	103
IEPA	335	482	72.4	79.8	75.3	82.2	8
HPRD50	163	270	67.8	85.3	74.6	77.8	3
LLL	164	166	86.2	92.1	89.1	90.2	3

the lowest on both LLL and AIMED, when $\lambda = 0.5$. When $\lambda = 2$, that is, the similarity weight of shortest dependency path nodes is twice as much as the normal nodes, our approach gets the best *F–M* on both LLL and AIMED. That shows the shortest dependency path nodes have more important distinguishing information than the normal nodes, but when $\lambda > 2$, the performance of our approach gradually reduces which is probably caused by excessively neglecting of the distinguishing information in the normal nodes.

Therefore, we next fix $\lambda = 2$ and evaluate the effect of R in Eq. (3). The results on the five corpora are listed in Table 3. We also use *F–M* as the primary evaluation measure. Table 3 shows that when $R = 2$, our approach gets the best *F–M* on AIMED, BioInfer and IEPA. But for HPRD50 and LLL, our approach gets the best *F–M* respectively at $R = 3$ and $R = 1$. Furthermore, for all of the five corpora, the maximum *F–M* change does not exceed 3% when R takes the value from 1 to 4.

From Fig. 4, it can be also seen that the inflection point occurs at $R = 2$ for AIMED, BioInfer and IEPA. Compare to HPRD50 and LLL, AIMED, BioInfer and IEPA are much larger and more sufficient for training and reliably testing our method. This result shows that neighborhood hash kernel can achieve the best performance by calculate the 2-neighborhood hash for the automatic PPI extraction.

The performance of our approach on five corpora is listed in Table 4 when we set $\lambda = 2$ and $R = 2$. To evaluate the efficiency of our approach, we list the computation time on five corpora. We run all of the experiments on a PC equipped with Intel Core 2 Duo 2.93 GHz and 4 GB main memory. All the numbers in Table 4 are averages taken over the 10 folds.

Table 4 shows that our approach achieves an average *F–M* about 72.5 and an average AUC about 83.6 on five corpora. In particular, our approach achieves state-of-the-art performance on LLL, which *F–M* and AUC are respectively 89.1 and 90.2. Similar to the results of work [6–8], *F–M* over AIMED and BioInfer are evidently inferior to other corpora, which is respectively 60.2 and 63.4. This is mainly because of the two corpora include nested entities and their distributions of negative and positive examples are very unbalanced. It is well known AUC is more stably to measure the performance than *F–M* when the corpora are widely different in sizes or on distributions. Compared with *F–M*, AUC on AIMED and BioInfer is respectively 85.3 and 82.7. Furthermore, we observe all AUC of five corpora is over 80 except for HPRD50, which is 77.8. Thus, we come to the conclusion that the *F–M* performance varies strikingly among the different corpora, whereby for the distribution-invariant AUC measure, the performance is relatively stable.

We next evaluate the efficiency of our approach. Although we run the experiments on a PC equipped with Intel Core 2 Duo 2.93 GHz and 4 GB main memory, the computation time of LLL, HPRD50 and IEPA is in 10 s. Since AIMED and BioInfer is much larger than other corpora, the computation time is 47 s and 103 s.

3.3. Cross-corpus performance

Cross-corpus experiments for PPI extraction derive from the work [5] and aim to study the fundamental question for practical PPI extraction using our approach: which corpora will be chose to be the training data? The cross-corpus results are listed in Table 5, which rows correspond to training corpora and columns to test corpora. We observe that all *F–M* achieved by cross-corpus is evidently lower than *F–M* achieved by single-corpus. For instance, the best *F–M* on AIMED is 45.1 which is achieved by using BioInfer to train, but the *F–M* on AIMED is 60.2 which is achieved by single-corpus. The main reason may be the large differences in the distributions of the five corpora, which breaks the basic assumption of machine learning theory: the training and test examples are identically distributed.

To choose the best training corpora from a generalization perspective, we first rank the training corpora separately according to the results on each of the other corpora and then compute the average rank of each training corpora. Comparing the average rank,

Table 5
Cross-corpus results on five corpora.

	AIMED		BioInfer		IEPA		HPRD50		LLL		Avg. rank
	F–M	Rank	*F–M*	Rank	*F–M*	Rank	*F–M*	Rank	*F–M*	Rank	
AIMED	–	–	48.7	2	68.4	3	66.9	2	76.1	3	2.5
BioInfer	**45.1**	1	–	–	**71.1**	1	**69.0**	1	**82.3**	1	1
IEPA	38.4	3	**51.0**	1	–	–	65.5	3	81.8	2	2.25
HPRD50	40.3	2	45.7	3	65.4	4	–	–	73.4	4	3.25
LLL	35.9	4	45.5	4	69.1	2	61.8	4	–	–	3.5

Table 6
Performance compared to all-paths kernel approach.

Corpus	Neighborhood hash kernel approach					All-paths kernel approach				
	P	*R*	*F–M*	AUC	Time (s)	*P*	*R*	*F–M*	AUC	Time (s)
AIMED	54.9	68.5	60.2	85.3	47	52.9	61.8	56.4	84.8	162
BioInfer	59.3	68.1	63.4	82.7	103	56.7	67.2	61.3	81.9	373
IEPA	72.4	79.8	75.3	82.2	8	69.6	82.7	75.1	85.1	21
HPRD50	67.8	85.3	74.6	77.8	3	64.3	65.8	63.4	79.7	12
LLL	86.2	92.1	89.1	90.2	3	72.5	87.2	76.8	83.4	11

BioInfer unsurprisingly performs best, which is the largest one of five corpora. This is similar to the results in work [6]. However, a surprising result is the second average rank is IEPA, which is much smaller than AIMED. One of the reasons for this maybe inaccurate dependency representations on AIMED reduce the performance of our method, which includes multiple dependency categories, unnecessary syntactic relations according to grouping named entity words, and cycle relations [7].

Overall, we also find the similar results to the work [6], which the performance is related to the size of the training data available. The large corpus, such as BioInfer, AIMED and IEPA, perform superior to the small corpus, such as HPRD50 and LLL.

3.4. Performance compared to all-paths kernel approach

The all-paths kernel approach for PPI extraction is proposed by Airola et al. in work [6], which achieves the state-of-the-art performance. We next discuss the performance of our approach compared to all-paths kernel approach. Table 6 shows comparisons with all-paths kernel approach. Our approach achieves markedly superior performance on all five corpora, especially improves the *F–M* 76.8–89.1 on LLL. But we have to note that AUC on IEPA and HPRD50 by our approach is evidently lower than by all-paths kernel approach. The performance comparison with all-paths kernel approach should be more reliable on AIMED and BioInfer than other small corpora, since the AIMED corpora and BioInfer corpora are much larger to show an advance and robustness of a method. We observe our approach outperforms all-path kernel method in *F–M*, AUC, precision and recall on AIMED and BioInfer.

We next compare the efficiency with all-paths kernel approach. Fig. 5 shows the comparison of computational time between our approach and all-paths kernel approach. As discussed in Section 2.2.4, the computational time of our approach is much less than all-paths graph kernel on all five corpora. For example, it take 373 s and 162 s respectively for all-path kernel approach to deal with BioInfer and AIMED, but it only take 103 s and 47 s for our neighborhood hash kernel approach. Therefore, compared to all-path kernel approach, our approach has more advantage on large corpora such as BioInfer. Thus our approach is more suitable to scale up to larger corpus or document collections than all-path kernel approach.

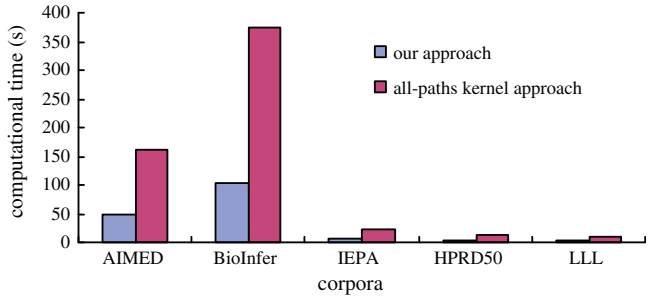

Fig. 5. Comparison of computational time between our approach and all-paths kernel approach.

3.5. Performance compared to other approaches

Finally we compare the performance of our approach with other successful approaches in recent years. We summarize the comparison on LLL and AIMED in Tables 7 and 8 respectively. We have to note that there are substantial differences according to the data set used for training and testing, including whether to preprocess, whether to include self interactions and how to split the data set used in cross-validation.

Table 7 shows comparisons with other approaches over LLL. Fundel et al. [3] create candidate relations from dependency parse trees by applying patterns and rules. Kim et al. [7] propose a walk-weighted subsequence kernel approach based on dependency graphs. Miwa et al. [8] propose a method that combines kernels based on several syntactic parsers, in order to retrieve the rich features from a given sentence. In addition, to improve the performance of their system, Miwa et al. [10] proposes a ruled-method to remove the unnecessary information of PPI corpora. So all approaches listed in Table 7 use the syntactic information of dependency graph. From Table 7, it can be seen that our approach significantly outperforms other approaches in *F–M*, precision and recall except for AUC which is slightly lower than by the approach of Miwa et al. [8]. It is worthwhile to note that though the approach of Miwa et al. [8] uses several syntactic parsers and combines three kernels, namely bag-of-words kernel, subset tree kernel and graph kernel, our approach outperforms in the major measure *F–M*. There are two possible reasons why this result occurs. One is our neighborhood hash kernel has more capability to express the syntactic information of dependency graph than bag-

Table 7
Performance compared to other approaches on LLL corpora.

Approach	*P*	*R*	*F–M*	AUC
Our approach	**86.2**	**92.1**	**89.1**	90.2
[8]	–	–	86.7	**90.8**
[10]	–	–	82.9	90.5
[7]	79.3	85.1	82.1	–
[6]	72.5	87.2	76.8	83.4
[3]	68	78	72	–

Table 8
Performance compared to other approaches on AIMED corpora.

Approach	*P*	*R*	*F–M*	AUC
[8]	60.0	**71.9**	**65.2**	**89.3**
[9]	60.5	68.4	63.5	87.2
Our approach	54.9	68.5	60.2	85.3
[7]	**61.4**	53.3	56.6	–
[6]	52.9	61.8	56.4	84.8
[10]	–	–	54.9	83.7

of-words kernel, shortest path kernel and all-paths kernel. The other possible reason is that LLL is more standard and accuracy than large corpora such as AIMED and the multiple syntactic parsers cannot remarkably improve the performance, though multiple syntactic parsers can reduce the noise of corpora.

Table 8 shows comparisons with other approaches over AIMED, which is much larger than LLL and more sufficient for training and testing machine learning approaches. Li et al. [9] uses a semi-supervised learning strategy to learn an enriched representation of local contexts from a large of unlabeled examples. Except for the approach of the work [9], all approaches use syntactic information by automated parsers. But it is interesting that the approach of the work [9] achieves very competitive results in *F–M* and AUC. The best performing approach, that of the approach of Miwa et al. [8], evidently outperforms other approaches in recall, *F–M* and AUC. As discussed above, one of the main causes for the high performance of the work [8] maybe AIMED corpora has amounts of noise which can be reduced by multiple syntactic parsers especially the deep syntactic parsers such as Enju 2.3.0. In addition, we note that the approach of Kim et al. [7] achieves the highest precision among all approaches, whereas the recall is far below the other methods.

Compared with other approaches over AIMED, *F–M* and AUC of our approach is lower than the approach of Miwa et al. [8] and the approach of Li et al. [9]. Furthermore we note that the precision of our approach is evidently lower than other methods except for the approach of Airola et al. [6]. However, the precision of our approach over LLL is highest among all approaches. This result shows that the noise of training data and testing data can cause a certain effect on the performance of neighborhood hash kernel approach. Therefore, we can improve the performance of our approach by using multiple parsers or the preprocessing methods proposed by Miwa et al. [10] to deal with the corpora. In addition, the rich information hiding the domain knowledge will further improve the performance of our method.

4. Conclusions

In this paper, we propose a neighborhood hash graph kernel approach for PPI extraction. This approach presents the syntactic analysis and the linear order of the sentences as directed vertex-labeled graphs, and uses the neighborhood hash kernel efficiently to deal with the graph representation. Compared to existing graph kernel-based approach, the neighborhood hash kernel approach can make use of full information of the graph representation with a low computational complexity. The results on five PPI corpora demonstrate our approach is comparable to the state-of-the-art PPI extraction system. Especially, our approach is much faster than all-paths kernel approach on all five corpora. The results of cross-corpus evaluation show BioInfer and IEPA is better training corpora for our approach, when the extraction system works beyond the corpus.

As future work, we are planning to use multiple syntactic parsers and other preprocessing methods to deal with the corpora to reduce the effect of the noise of corpora. In addition, we will try to find an appropriate way of employing domain knowledge into PPI extraction. Due to the neighborhood hash kernel has the ability to deal with more complicated structures than dependency graph, we also attempt to apply neighborhood hash kernel to PPI network analysis.

Acknowledgments

This work is supported in part by grant from the Natural Science Foundation of China (Nos. 60673039 and 61070098), the National High Tech Research and Development Plan of China (No. 2006AA01Z151), Doctoral Program Foundation of Institutions of Higher Education of China (No. DUT10JS09) and Liaoning Province Doctor Startup Fund (No. 20091015).

References

[1] Zelenko D, Aone C, Richardella A. Kernel methods for relation extraction. J Machine Learn Res 2003;3:1083–106.
[2] Bunescu R, Ge R, Kate R, Marcotte E, Mooney R, Ramani A, et al. Comparative experiments on learning information extractors for proteins and their interactions. Artif Intell Med 2005;33(2):139–55.
[3] Fundel K, Kuffner R, Zimmer R. RelEx-relation extraction using dependency parse trees. Bioinformatics 2007;23(3):365–71.
[4] Pyysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Järvinen J, et al. BioInfer: a corpus for information extraction in the biomedical domain. BMC Bioinformatics 2007;8(50).
[5] Van Landeghem S, Saeys Y, Van de Peer Y, De Baets B. Extracting protein–protein interactions from text using rich feature vectors and feature selection. In: Proceedings of the third international symposium on semantic mining in biomedicine; 2008. p. 77–84.
[6] Airola A, Pyysalo S, Björne J, Pahikkala T, Ginter F, Salakoski T. All-paths graph kernel for protein–protein interaction extraction with evaluation of cross-corpus learning. BMC Bioinformatics 2008;9(Suppl. 11):S2.
[7] Kim Seonho, Yoon Juntae, Yang Jihoon, Park Seog. Walk-weighted subsequence kernels for protein–protein interaction extraction. BMC Bioinformatics 2010;11:107.
[8] Miwa M, Sætre R, Miyao Y, Tsujii J. A rich feature vector for protein–protein interaction extraction from multiple corpora. In: Proceedings of the 2009 conference on empirical methods in natural language processing, vol. 1, no. 1; 2009. p. 121–30.
[9] Li Yanpeng, Hu Xiaohua, Lin Hongfei, Yang Zhihao. Learning an enriched representation from unlabeled data for protein–protein interaction extraction. BMC Bioinformatics 2010;11(s2):s7.
[10] Miwa Makoto, Miyao Yusuke, Sætre Rune, Tsujii Jun'ichi. Entity-focused sentence simplification for relation extraction. In: Proceedings of the 23rd international conference on computational linguistics; August 2010. p. 788–96.
[11] Hido Shohei, Kashima Hisashi. A linear-time graph kernel. In: Proceeding of the 9th IEEE international conference on data mining 2009, Miami, Florida; 2009. p. 179–88.
[12] Gärtner T, Lach PA, Wrobel S. On graph kernels: hardness results and efficient alternatives. In: Proceedings of the sixteenth annual conference on learning theory and seventh annual workshop on kernel machines, Washington, DC. Lecture notes in artificial intelligence, Springer; 2003. p. 129–43.
[13] Vishwanathan S, Borgwardt KM, Schraudolph NN. Fast computation of graph kernels. Adv Neural Inf Process Syst 2007;19:1449–56.
[14] Ding J, Berleant D, Nettleton D, Wurtele E. Mining MEDLINE: abstracts, sentences, or phrases? In: Proceedings of the pacific symposium on biocomputing, Kaua'i, Hawaii; 2002. p. 326–37.
[15] Nédellec C. Learning language in logic–genic interaction extraction challenge. In: Proceedings of the 4th learning language in logic workshop, Bonn, Germany; 2005. p. 31–7.
[16] Pyysalo S, Airola A, Heimonen J, Björne J, Ginter F, Salakoski T. Comparative analysis of five protein–protein interaction corpora. BMC Bioinformatics 2008;9(Suppl. 3):S6.
[17] Lease M, Charniak E. Parsing biomedical literature. In: Proceedings of the 2nd international joint conference on natural language processing, Jeju Island. Korea, Lecture notes in computer science, Springer; 2005. p. 58–69.