BIOINFORMATICS ORIGINAL PAPER

Vol. 24 no. 1 2008, pages 118–126
doi:10.1093/bioinformatics/btm544

Data and text mining

Kernel approaches for genic interaction extraction

Seonho Kim[1,*], Juntae Yoon[2,*] and Jihoon Yang[1,*]

[1]Department of Computer Science, Sogang University and [2]Daumsoft Inc., Se-Ah Venture Tower, Seoul, Korea

Received on May 14, 2007; revised on September 21, 2007; accepted on October 25, 2007

Advance Access publication November 14, 2007

Associate Editor: John Quackenbush

ABSTRACT

Motivation: Automatic knowledge discovery and efficient information access such as named entity recognition and relation extraction between entities have recently become critical issues in the biomedical literature. However, the inherent difficulty of the relation extraction task, mainly caused by the diversity of natural language, is further compounded in the biomedical domain because biomedical sentences are commonly long and complex. In addition, relation extraction often involves modeling long range dependencies, discontiguous word patterns and semantic relations for which the pattern-based methodology is not directly applicable.

Results: In this article, we shift the focus of biomedical relation extraction from the problem of pattern extraction to the problem of kernel construction. We suggest four kernels: predicate, walk, dependency and hybrid kernels to adequately encapsulate information required for a relation prediction based on the sentential structures involved in two entities. For this purpose, we view the dependency structure of a sentence as a graph, which allows the system to deal with an essential one from the complex syntactic structure by finding the shortest path between entities. The kernels we suggest are augmented gradually from the flat features descriptions to the structural descriptions of the shortest paths. As a result, we obtain a very promising result, a 77.5 *F*-score with the walk kernel on the Language Learning in Logic (LLL) 05 genic interaction shared task.

Availability: The used algorithms are free for use for academic research and are available from our Web site http://mllab.sogang.ac.kr/~shkim/LLL05.tar.gz.

Contact: shkim@lex.yonsei.ac.kr

1 INTRODUCTION

Due to the dynamic progress in biomedical technology, a huge amount of new information and research results have been constantly published. This tendency makes it difficult to keep track of newly provided information, thus requiring automatic knowledge discovery from biomedical text such as biomedical named entity (NE) recognition and relation extraction among the named entities. In particular, successful results have been reported from recent research on biomedical NE recognition, while biomedical relation extraction is still a challenge.

In general, biomedical information extraction systems aim to extract pre-defined types of facts, such as relationships/interactions between biomedical entities. For example, we can consider several relations such as 'cure', 'prevent', 'vague', 'side effect' and 'no cure' between two entities, *treatment* and *disease*.

In this work, we identify genic (protein/gene) interactions between biomedical entities on the LLL 05 shared task (Aubin, 2005). The goal of the challenge is to learn rules for extracting protein/gene interactions from Medline abstracts, discriminating between agent NEs and target NEs of the interactions. Such interactions are fundamental in functional genomics because they form regulation networks that are very informative for determining the functions of genes.

However, a considerable portion of descriptions for such interactions is available not in a structured biomedical database, but in scientific papers with raw text format. Moreover, much useful information is actually scattered across multiple papers. For this reason, various approaches have been applied to relation extraction in the biomedical domain. We can broadly categorize prior biomedical relation extraction systems into two methods: co-occurrence-based and pattern-based approaches.

In the simple co-occurrence-based work, two entities are assumed to have a relationship if they are only mentioned together without being necessarily related in a specific way. That is, the relationship implies that two entities repeatedly occur together or by the presence of some linguistic expressions. However, relations between entities are less predictable by pure co-occurrences of terms in sentences. Thus, we need a specificity measure to ensure that the extracted relations are not too general. For instance, 6 of genes and proteins appear in Figure 1a, only 5 pairs among 30 possible ordered NE pairs have real interactions: (GerE, cotD), (GerE, cotA), (sigma K, cotA), (GerE, SigK) and (sigK, sigma K). In order to extract the correct interaction pairs, deep-level linguistic processing such as syntactic and semantic analysis is required. We first have to recognize that the NE of a protein name is the subject of interaction verbs such as 'stimulate' or 'inhibit', and the NE at the object position is a gene name or gene expression. In addition, in order to identify 'stimulates' and 'inhibits' that share the same subject, 'GerE', a coordination processing should be performed.

The other dominating method is the pattern-based approach that utilizes a set of words, phrases, sentence patterns/templates or longest common subsequences concerning special verb/noun keywords such as 'interact', 'bind', 'associate' and 'complex' used to represent biomedical interactions. For instance, a keyword 'interact' is associated with the following

*To whom correspondence should be addressed.

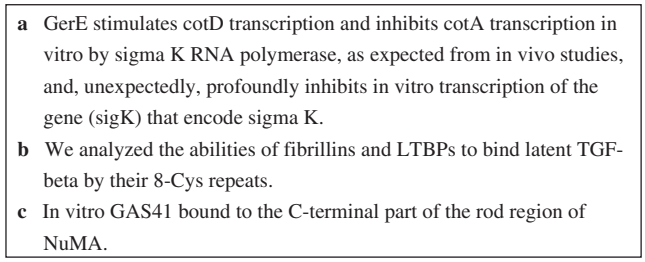

Fig. 1. Data representation of kernels.

patterns: 'NE_A interacts with NE_B', 'interaction of NE_A (with|and) NE_B', 'interaction (between|among) NE_A and NE_B', 'NE_A-NE_B interaction', and 'NE_A and NE_B interact' (Ono *et al*., 2001). Such patterns/templates can be extracted by using a set of syntactic tags or part-of-speech (POS) tags as rules, which are constructed automatically or manually. For example, 'protein + VB + TO + protein'[1] is a POS tag rule for the 'bind to' pattern and 'protein + VB + IN + protein', for the 'interact/associate with' pattern.

There has been much work on pattern-based relation extraction as it provides an intuitively easy and rather accurate framework. Blaschke *et al*. (1999) have extracted interactions based on a set of manually developed matching rules, where each rule is simply a sequence of words or POS tags anchored on two protein entities. Fundel *et al*. (2005), Ono *et al*. (2001) and Jang *et al*. (2006) have also suggested a rule-based pattern extraction from parse trees. Huang *et al*. (2004) extracted generalized POS rule patterns from a biomedical POS tagged corpus. They used a dynamic programming algorithm to align relevant sentences for each keyword and compute distinguishing POS tag patterns. Hao *et al*. (2005) extended Huang *et al*.'s model to efficiently reduce and merge the POS patterns using the minimum description length (MDL).

However, these prior works have mainly focused on syntactic aspects, which often fail to correctly account for relations between entities. As shown in Figure 2c, 'GerE' and 'SigK' have a long-distance dependency relation that spans several clauses. In the biomedical domain, such long range relations or discontiguous word patterns are very common since biomedical sentences are long and complex. In addition, syntactic tags or POS tags rules are not enough to indicate semantic relations. For example, as shown in Figure 1b and c, pattern-based approaches cannot appropriately handle the semantic aspects, such that 'the ability of fibrillins to bind' conveys the meaning of 'fibrillins bind' and 'bound to the C-terminal part of the rod region of NuMA' conveys the meaning of 'bound to NuMA' (Jang *et al*., 2006). That is, various syntactic realizations with the same meaning cannot be accounted for using only the patterns extracted by syntactic or morphological tags. As a result, the pattern-based approaches have shown excessively low recall rates. In addition, the type and the direction of a relation cannot be easily identified using only the syntactic patterns.

[1]VB, TO and IN are POS tags for verb, to and preposition that are defined in Penn Treebank.

In this article, we address the problem of genic interaction extraction by using kernel-based machine learning. The kernel is a kind of similarity function for features derived from a pair of objects. In our work, an object corresponds to the shortest path between two NEs on the syntactic graph for a sentence.

Actually, syntactic dependency information provided by the LLL shared task is hard to represent by a tree form, thereby making effective feature extraction difficult. As an alternative for the tree-based representation for sentence structures, we adopt a graph-based method. That is, we represent each interaction example given by the shared task in the form of a directed graph, where an edge corresponds to the dependency relation between two vertices, a head and its dependent. Based on the graph structure, the path between two entities is found by the shortest path algorithm and then the scope of structure for interaction learning is confined to the directed shortest dependency path. As a consequence, the shortest path between two entities in a dependency graph can provide a more concise representation of information needed to assess their relation prediction by restricting learning features to elements inside the path.

With the data representation, we propose four kernels for interaction learning, namely, predicate kernel, walk kernel, dependency kernel and hybrid kernel, which can explore a variety of aspects in syntactic and semantic information on shortest dependency paths. The kernels are classified into two distinct methods: feature-based kernel and structure-based kernel. The feature-based kernel computes the similarity based on feature sets derived from the graph. On the other hand, the structure-based kernel is based on the structural isomorphism between two graphs. The predicate and walk kernels belong to the feature-based kernel and the dependency and hybrid kernels, to the structure-based kernel.

In short, we focus our research on the following issues: (1) what kind of data representation is efficient for retrieving interactions? (2) What lexical and syntactic features are useful for the identification of genic interactions and how can such useful features be incorporated into kernels? (3) How can structured data like a graph or a tree be processed with a kernel?

As a result, our kernels efficiently deal with structured data like our graph as the learning features. In the experiments, we achieve the best result on the LLL 05 shared task with the walk kernel, a quite promising *F*-score of 77.5.

This article is organized as follows: In Section 2, we describe the data representation and introduce the kernels that we propose. In Section 3, we present the data set for experiments and support vector machine (SVM) learning. Finally, we discuss some experimental results and end with conclusion remarks.

2 METHODS

The main task of prior information extraction (IE) systems in natural language processing (NLP) literature is to recognize names such as people, organizations and locations, and relations between them by applying various machine-learning (ML) methods. However, there have been few attempts to develop ML techniques for extracting relations in the biomedical domain (Bunescu *et al*., 2005; Riedel and Klein, 2005).

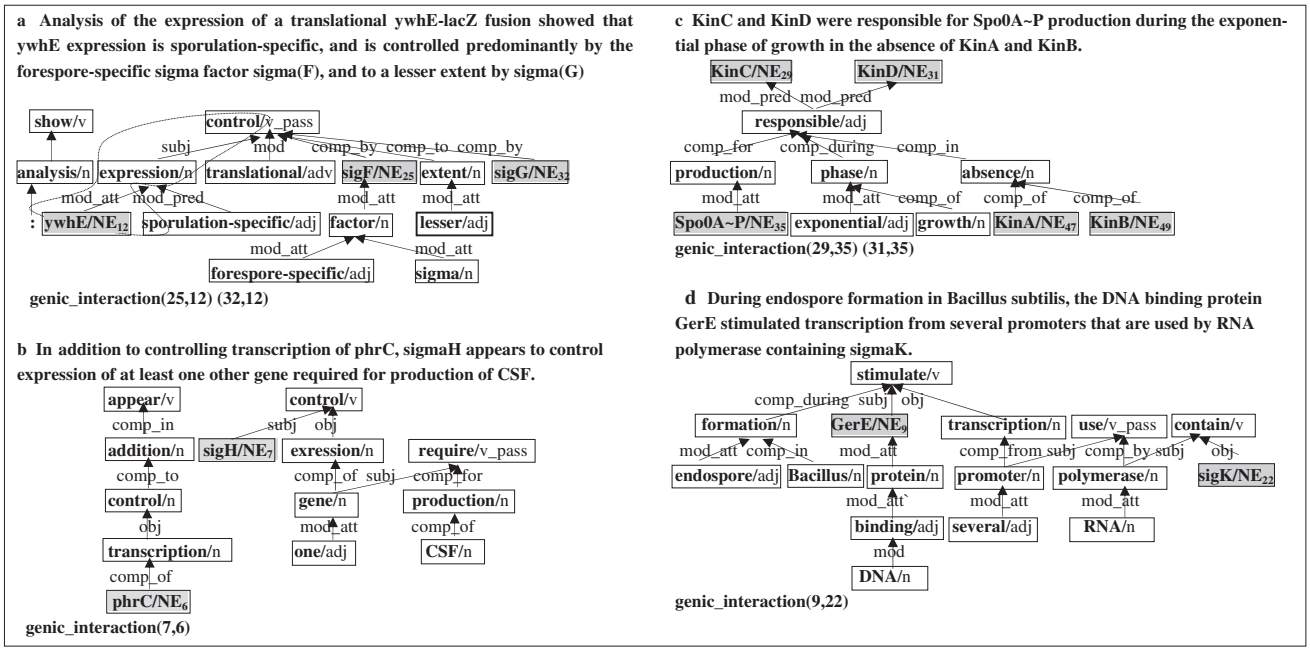

Fig. 2. Examples of biomedical parsed sentences.

The main reason is the lack of a good quality data set that meets the requirements of the NLP and ML fields at the same time. Recently, works on biomedical relation extraction using ML techniques have been attempted as the linguistically well-annotated data set such as the LLL 05 was constructed. In this study, we also test our method on the LLL 05 shared task, which is tailored to reflect the requirements of deep-level analysis for learning interactions. In this section, we first introduce the LLL task and its annotated linguistic information and then describe the data representation and kernel approaches we suggest.

2.1 LLL shared task

The LLL 05 challenge task is to learn rules for identifying protein/gene interactions between two NEs and their roles, agent or target. The task focuses on information extraction for 'transcription' in '*Bacillus subtilis*', which has been used as a model bacterium in genetic and molecular biological studies. The 'transcription' is also a central phenomenon in functional genomics involved in gene interaction, which is a popular IE problem.

The data set consists of Medline biology abstracts retrieved by the query '*Bacillus subtilis and transcription*'. The training data includes three distinct relations: an explicit action, a binding of the protein on the promoter of the target gene and a membership to a regulon family. The challenge data contains various types of linguistic information such as words, their lemma and the syntactic dependencies among words. The interactive NE pairs in the training sentences are annotated indicating the agent and target of an interaction.

For the syntactic analysis, the Link Grammar Parser was used (Sleator and Temperly, 1993). The analysis contains morpho-syntactic tags and syntactic dependency functions. The Link Grammar formalism is closely related to dependency grammar and builds a set of labeled links between pairs of words, rather than constructing phrase structures in a tree form. A link represents the grammatical dependency relation between a head and its dependent in the form of 'F:A-B', where F is a function of the relation that links two words, A is a morpho-syntactic head and B is the morpho-syntactic dependent for the head. The dependency is represented by five different morpho-syntactic categories, V (verb), V_PASS (passive verb), N (noun), ADJ (adjective) and ADV (adverb) and seven functions of relations, APPOS (apposition), COMP_prep (prepositional complement), MOD (modifier), MOD_ATT (attributive modifier), NEG (negation), OBJ (object) and SUBJ (subject) (Sophie Aubin, 2005). We can consider the relation between 'controlled' and 'expression' marked with labels such as ('subj:V_PASS-N', 'control', 'word') in the sentence in Figure 2a.

Because Link Grammar parsing is computationally efficient and convenient for representing relationships between words, it has been adopted in many biomedical relation extraction systems (Ding *et al.*, 2003; Tsivtsivadze *et al.*, 2006).

The LLL task assumes that all candidate NEs for genic interaction can be recognized in advance by the NE dictionary. The dictionary contains all NEs, their variants and synonyms. However, the category of each NE such as gene and protein is not specified in the dictionary, which can be important for determining its role, agent or target.

2.2 Data representation

Figure 2 exhibits some examples of training sentences annotated with dependency relations. In the figure, each directed edge corresponds to the syntactic dependency relation between two nodes, the head and its dependent. Dependency relations are shown as arrows from the dependent to its head. The predicate 'genic_interaction(id_NE_1, id_NE_2)' refers to an interaction between id_NE_1 and id_NE_2, where id_NE_1 and id_NE_2 are positions of the agent NE_1 and the target NE_2, respectively.

There are several characteristics in the Link Grammar's structural annotations. First, separate trees can be constructed for subordinate clauses as in Figure 2b, since there is no explicit notion of a root word for a sentence in the Link Grammar. Second, one child (dependent) can be shared by two parents (heads), as shown the Figure 2c. Due to this property, the structure of a sentence can be adequately expressed by a directed graph rather than by a tree.

In this study, we regard a parsed sentence as a directed graph and restrict the structure of interest to the shortest syntactic path of two NEs instead of the whole graph. In general, it is important to strip out

unnecessary features from the input data as much as possible for accurate learning. Actually, many words in a dependency graph of a sentence have no direct influence on relation learning. Thus, we first narrow down the dependency graph into the shortest syntactic path between a pair of NEs, which can be defined as a sequence of words connected by dependency relations. The shortest syntactic path can be found by using the Dijkstra's algorithm (Cormen *et al*., 2001). Figure 3 shows the data representations related to the shortest path between two NEs, 'ywhE' and 'sigF' in Figure 2a.

However, we cannot find the path from the dependency graphs in many cases because every syntactic relation is toward the syntactic head, as shown in Figure 3a. Thus, we allow edges of the Directed Acyclic Graph (DAG) to be traversed in any direction when finding the path, so for each original dependent-head edge labeled with an 'UP' direction, we add the corresponding head-to-dependent edge marked with 'DN' direction. As a consequence, the original directed graph like in Figure 3a is treated as an un-directed graph like in Figure 3b. In this case, we can find the shortest path from 'ywhE' to 'sigF' by traversing the edge of 'comb_by' in reverse. The dotted arrows in Figure 3b display the shortest path between the two NEs. Also, all the nodes and edges on the path can be represented by the string as shown in Figure 3c.

Furthermore, we add predicate information, called 'PRED', to the path string to indicate the direction change. The presence of the 'PRED' at a word on the path means that the directions of the left or right edges connected to the word are changed. This often occurs in predicative words. In the example, the node of 'control' is identified as a predicate by the connected edges, 'subj(UP)' and 'comp_by(DN)'.

The shortest paths we suggest contain linguistic information such as lexical information of words, their POS, typed dependencies between words and directions of the dependencies, which provide contextual and structural information for relation learning.

In the case of 'sigF' and 'ywhE' in Figure 2a, because the interactive entities serve as arguments for the predicate, 'control', they will be connected directly on the shortest path. If two entities belong to different predicate-argument structures like 'GerE' and 'sigK' of Figure 2d but the substructures are connected by common arguments, then the shortest path will pass the common arguments. In addition, there is a case that the shortest path does not exist for an interactive NE pair like 'sigH' and 'phrC' in Figure 2b.

We use the graph representation of the shortest path to train the predicate and walk kernels for relation learning (Fig. 3c). The dependency graph can be represented by the dependency lists as presented in Figure 3d. The lists consist of the dependency list for word and POS. For each node *w*, the word dependency list contains the word at the node and a set of (*relation*, *dependent_word*) pairs, which are the direct dependents of *w*. The POS dependency list contains the morpho-syntactic information of *w* and a set of (*relation*, *dependent_pos*) pairs, which are the direct dependents' POS of *w*. For example, the word '*expression*' has the relation set with its direct children as follows: {(subj, expression), (comp_by, NE2)}. We use the list structures for the other kernel methods, the dependency and hybrid kernels (Fig. 3d).

2.3 Kernels

A kernel can be thought of as a similarity function for pairs of objects. In our work, an object corresponds to the shortest path between two NEs on the syntactic graph for a sentence. In this work, we suggest two distinct types of kernels to extract relations: feature-based kernels and structure-based kernels. The feature-based kernels use a well-known polynomial kernel on feature sets derived from the graph for computing the similarity, whereas the structure-based kernels calculate the graph similarity directly. In our models, the predicate and walk kernels belong to feature-based kernels and the dependency and hybrid kernels belong to structure-based kernels.

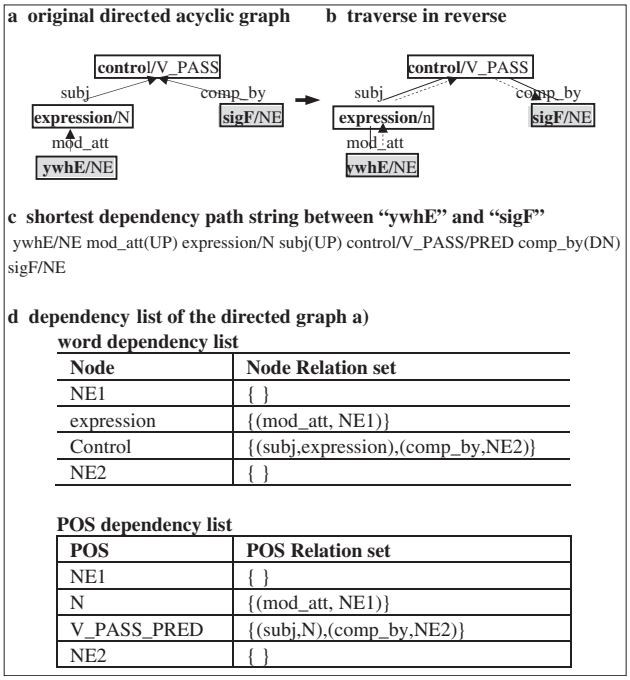

Fig. 3. Data representation of the interaction ('ywhE' and 'sigF') in Figure 2a.

An object in the feature-based kernels is represented by a traditional feature enumeration vector. That is, the predicate and walk kernels are just discriminated not by the kernel function but by the feature extraction methods. Actually, the two kernels operate on the same kernel, the so-called 'polynomial kernel', with different feature vector representations. However, we confer the names of predicate and walk kernel on the two models in order to emphasize the difference of the feature extraction framework.

In contrast, we design the kernel functions for the structure-based kernels such as dependency and hybrid kernels. An object in the structure-based kernels is represented with a vector in which each element contains a structural similarity value with other objects. The structure-based kernels can provide a good formalism to learn the structured data like a graph or a tree by using structural isomorphism.

The kernels we propose are learned and tested by SVM. SVM uses the kernel functions to compare the test object with those derived from training sentences. Since the feature-based kernels use the polynomial kernel as a kernel function, descriptions of the predicate and walk kernels in Sections 3.2.1 and 3.2.2 are mostly related to the feature sets used in the polynomial kernel. In Sections 3.2.3 and 3.2.4, the kernel functions for computing the structural similarity are introduced.

All NE pair objects are represented in word order and the SVM classifies a test object for a pair of NEs into 'TA', 'AT' or 'O'. 'TA' denotes the target-agent relation, where the former NE is a target and the latter NE is an agent. 'AT' stands for the agent-target relation and 'O' means that two NEs have no genic relation. With the kernels, we gradually increase the internal information on the shortest path from links related to predicate and walk to the whole dependency structure on the path.

2.3.1 Predicate kernel In this model, we focus on the central role of predicates in relation extraction. We assume that a predicate and its arguments are crucial for relation extraction. That is, a pair of NEs is regarded to have a genic relationship if an interaction predicate exists

on the shortest path between two NEs and if each NE serves as a meaningful argument of the interaction predicate in a certain syntactic configuration. This problem is analogous to semantic role labeling that assigns semantic roles of 'agent' and 'target' to NEs (Xue and Palmer, 2004).

Similar to semantic role labeling features, the predicate kernel is developed with features related to predicates and their immediate child nodes on the dependency path. As mentioned before, we identify the node as a predicate when the directions of the edges connected to the node are changed. The features consist of basic features and conjunctions of the basic features as shown in Table 1. As basic features, predicates, their POS, their immediate child nodes, and dependencies with the predicates and the immediate child nodes are used. That is, this kernel is designed to consider the topmost structures rooted at predicate nodes by allowing only features related to predicates and their arguments. Besides the features, SAME_NE, SAME_ROLE and LINK_TYPE features are additionally used. The binary feature, SAME_NE indicates if the strings of two NEs are the same or not, and the SAME_ROLE feature indicates if a coordinate conjunction like 'and/or' joins an NE pair. The LINK_TYPE feature shows how many predicates on the shortest dependency path connecting two NEs there are. The feature has the value of No_Link when there is no path between an NE pair. The values, Link_OnePred and Link_Preds, denote that a path includes one predicate and more than one predicate, respectively. In addition, conjunctions of these basic features are considered. Figure 4a shows the features for the predicate kernel with respect to the interaction (25, 12), i.e. 'sigF' and 'ywhE' in Figure 2a. Here, we normalize all NEs to 'NE' to reduce the data sparseness.

In the case of the predicate kernel and the walk kernel, x_i, the training feature vector of pattern i, is passed to the polynomial kernel for SVM classification. The kernel function is represented as follows:

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$$

Here, x_i is the feature vector of pattern j, and γ, r and d are kernel parameters (Shawe-Taylor and Cristianini, 2000). The SVM empirically looks for an optimal separating hyperplane that maximizes the distance (margin) between the hyperplane and the support vectors on each class. Support vectors are the nearest training vectors to the hyperplane. The polynomial kernel maps the training vector into a higher dimensional space in order that SVM can find a linear separating hyperplane in the higher dimensional space.

2.3.2 Walk kernel In Figure 3a, the NE, 'ywhE' does not have any direct syntactic relation with the predicate, 'control', as it operates as the modifier of the noun 'expression'. That is, 'ywhE' that is not directly governed by the predicate has an interaction relation with 'sigF'. In order to capture the contextual information, we consider the walk information in this kernel. In our path-based syntactic structure, it is rather easy to give structural information to the learning scheme because a path reflects the dependency relation map for words on the path between two NEs. In the walk kernel, the structural information is defined by a walk in a graph.

To formalize, let $P = (V, E)$ be a graph, where V and E are a set of vertices and edges, respectively. Given $v \in V$ and $e \in E$, a walk is defined as alternating sequences of vertices and edges, v_i, $e_{i,i+1}$, v_{i+1}, $e_{i+1,i+2}$, …, v_{i+n-1}, beginning with a vertex and ending with a vertex. The length of a walk is the number of edges that it uses. We take into consideration walks of length 1, namely, v_i, $e_{i,i+1}$, v_{i+1}, among all possible subsets of walks on a path between two arbitrary NEs. We call it v-walk in this article.

Besides v-walk, we define e-walk that starts and ends with an edge (e.g. $e_{i,\ i+1}$, v_{i+1}, $e_{i+1,\ i+2}$). It is actually not a walk defined in the graph theory but an *ad hoc* concept to provide a syntactic structure for the learning model because contexts by syntactic relations are crucial.

a feature for Predicate kernel
class=TA SAME_NE=0 pword=control ppos=V_PASS L_srole=subj R_srole =comp_by L_head=expression R_head=NE pword+ppos=control+V_PASS pword+ L_srole=control+subj pword+R_srole=control+comp_by pword+L_srole+R_srole= control+subj+comp_by ppos+L_srole=V_PASS+subj ppos+R_srole=V_PASS+comp_ by ppos+L_srole+R_srole=V_PASS+subj+comp_by pword/pos+L_srole=control/V_ PASS+subj pword/ppos+R_srole=control/V_PASS+comp_by pword/ppos+L_srole+R _srole=control/V_PASS+subj+comp_by SAME_ROLE=0 LINK_TYPE=Link_One PRED
b feature for walk kernel
class=TA S_W=NE+mod_att(UP)+N L_W=NE+mod_att(UP)+expression S_W=mod _att(UP)+N+subj(UP) L_W=mod_att(UP)+expression+subj(UP) S_W=N+subj(UP)+V_ PASS_PRED L_W=expression+subj(UP)+control S_W=subj(UP)+V_PASS_PRED+ comp_by(DN) L_W=subj(UP)+control+comp_by(DN) S_W=V_PASS_PRED+comp_by (DN)+NE L_W=control+ comp_by(DN)+NE
c dependency kernel
$K_D(d1,d1)$ $=C_w(NE_1,NE_1)+C_w$(expression,expression)$+C_w$(control,control)$+C_w$ (NE_2, NE_2)+ $C_p(NE_1,NE_1)+C_p$(V_PASS_PRED,V_PASS_PRED)$+C_p$(n,n)$+C_p$ (NE_2, NE_2) $=0+(C_w(NE_1,NE_1)+2-1)+((C_w$(expression,expression)$+2)(C_w(NE_2,NE_2)+2))-1)+0+$ $0+(C_p(NE_1,NE_1)+2-1)+((C_p(n,n)+2)$ $(C_p(NE_2,NE_2)+2))-1)+0+$ $=0+1+((C_w(NE_1,NE_1)+2-1)+2)(2))-1)+0+0+1+((C_p(NE_1,NE_1)+2-1)+2)(2))-1)+0$ $=12$
d hybrid kernel
$K_H(d_1,d_1) = 12+10$

Fig. 4. Data representation of kernels.

We thus consider the e-walk of $e_{i,i+1}$, v_{i+1}, $e_{i+1,\ i+2}$, too. For each of the v-walk and e-walk, we consider lexical walks and syntactic walks. The lexical walk consists of lexical words and dependency relations, and the syntactic walk, of POS and dependency relations, respectively.

Consequently, this walk kernel can look more closely into the information inside the shortest path as compared to the predicate kernel that is minimal to some extent, since all possible contiguous subpaths of restricted length are considered. Figure 4b shows the features for the walk kernel with respect to the interaction ('sigF', 'ywhE'). L_W and S_W denote lexical walk and syntactic walk, respectively. We add the direction of 'UP' or 'DN' to each edge and 'PRED' to the POS of the identified predicates. Also, all NEs are used as 'NE' instead of their name.

2.3.3 Dependency kernel In the previous two kernels, we encoded structure properties related to nodes and edges on the shortest path with feature vectors. As features, we used links related to predicates and their direct children on the shortest path for the predicate kernel, and syntactic and semantic walks for the walk kernel. However, the feature-based approaches can sometimes fail to identify similar relations because the dependency path we consider is sensitive to small changes of parse-trees. In addition, structured data like a tree or a graph is not often represented properly by flat features.

Thus, we do not explicitly generate feature vectors in the following kernels but, instead, directly calculate the similarity between two shortest path graph structures by investigating common subgraphs. The kernel functions between a pair of objects measure how similar two graphs are by how many common subgraphs they share. The isomorphism between two graphs is established in terms of common word dependencies and common POS dependencies. As a result, this kernel can implicitly explore a much larger feature space than feature approaches by using the structural similarity considering all sub-structures without enumerating features.

This kind of kernel methodology has been actively applied to many areas such as parsing (Collins and Duffy, 2001), semantic role labeling (Moschitti, 2004) and relation extraction (Culotta and Sorensen, 2004;

Table 1. Features for the predicate kernel

Basic feature	
Feature	Description
Pword	Predicate word(s)
Ppos	Pos of the predicate word(s)
L_srole	Syntactic relation of immediate left child and predicate
R_srole	Syntactic relation of immediate right child and predicate
L_head	Word of predicate's immediate left child
R_head	Word of predicate's immediate right child
SAME_NE	Are the NEs the same?
SAME_ROLE	Are L_srole and R_srole the same?
LINK_TYPE	No_Link, Link_OnePred, Link_Preds
Feature conjunction	
pword + ppos, pword + L_srole, pword + R_srole, pword + L_srole + R_srole, ppos + L_srole, ppos + R_srol, ppos + L_srole + R_srole, pword/pos + L_srole pword/ppos + R_srole, pword/ppos + L_srole + R_srole	

Bunescu and Mooney, 2005; Zelenko *et al.*, 2003). In fact, the design of kernels in the structural domain is one of rich research areas in NLP. Also, kernel approaches have been used for a lot of relation extraction systems on the ACE[2] corpus, the main goal of which is to find target relations such as person-affiliation and organization-location. On the other hand, the approaches are still rare in the biomedical relation extraction area (Bunescu *et al.*, 2005).

Our dependency kernel is a modification of Collins and Duffy's convolution kernel for a dependency structure (Collins and Duffy, 2001). We define the dependency kernel to capture the isomorphism between two shortest path graph structures. For this, we make a matrix whose element contains the similarity value of two graphs evaluated by the number of common subgraphs. As mentioned earlier, the graph means the directed shortest dependency graph between a pair of NEs. It can be represented as a dependency list form that is composed of a set of nodes and a set of relations between a node and its child nodes, as shown in Figure 3d. The figure shows the dependency list corresponding to the interaction ('sigF', 'ywhE') in Figure 2a.

Before we describe how isomorphism is established between two graphs, we will introduce some notations. Let d_1 and d_2 be dependency graphs, and N_1 and N_2 be the sets of nodes in the dependency graphs, respectively. For each node x, $word(x)$ is the word at the node and $children_w(x)$ refers to the direct dependents of x represented by the set of (*relation*, *word*) pairs that consists of the words of direct dependents of x *and* the syntactic relations with them. POS(x) refer to the POS of node x and $children_p(x)$ denotes the set of (*relation*, *POS*) pairs that are direct dependents of x. Given two parent nodes n_1 and n_2, $sc_w(n_1, n_2)$ is the set of common word dependencies between two subgraphs rooted by n_1 and n_2, respectively. $sc_p(p_1, p_2)$ corresponds to the set of common POS dependencies between two subgraphs rooted by POS p_1 and p_2, respectively. If the direct child nodes of two parent nodes, x and y are the same word and have the same dependent relation with their parents n_1 and n_2, then the pair (x, y) is an element of $sc_w(n_1, n_2)$.

[2]http://www.nist.gov/speech/tests/ace.

Also, $C_w(n_1, n_2)$ is the number of common subgraphs between two graphs rooted at n_1 and n_2 nodes. $C_p(p_1, p_2)$ denotes the number of common subgraphs rooted at POS p_1 and p_2. As presented in Equation (2), C_w and C_p are computed recursively over all subgraphs. That is, if there is no child of n_1 or n_2, or if two nodes are different words, then $C_w(n_1, n_2)$ returns 0. Otherwise, it recursively calls C_w with respect to their common child pairs in the set $sc_w(n_1, n_2)$.

$$\begin{aligned} &SC_w(n_1, n_2). \\ &SC_w(n_1, n_2) = \{(x, y) | (relation, x) \in children(n_1), \\ &\qquad (relation, y) \in children(n_2), word(x) = word(y)\} \\ &SC_P(p_1, p_2) = \{(x, y) | (relation, x) \in children(p_1), \\ &\qquad (relation, y) \in children(p_2), POS(x) = POS(y)\} \end{aligned} \tag{1}$$

$$\begin{aligned} &\text{if } word(n_1) \neq word(n_2) \text{ or } children(n_1) \text{ is empty or } children(n_2) \text{ is empty}, \\ &\text{then } C_w(n_1, n_2) = 0 \\ &\text{else } C_w(n_1, n_2) = \prod_{(x,y) \in SC_w(n_1, n_2)} (C_w(x, y) + 2) - 1 \\ &\text{if } POS(p_1) \neq POS(p_2) \text{ or children } (p_1) \text{is empty or children } (p_2) \text{ is empty}, \\ &\text{then } C_p(p_1, p_2) = 0 \\ &\text{else } C_p(p_1, p_2) = \prod_{(x,y) \in SC_p(p_1, p_2)} (C_p(x, y) + 2) - 1 \end{aligned} \tag{2}$$

Then, we can define the dependency kernel function to evaluate the similarity of two graphs in terms of syntactic dependency as follows:

$$K_D(d_1, d_2) = \sum_{n_1 \in N_1, n_2 \in N_2} C_w(n_1, n_2) + \sum_{p_1 \in P_1, p_2 \in P_2} C_p(p_1, p_2)$$

It is a summation of common word dependency subgraphs and common POS dependency subgraphs between two graphs. For example, we can obtain the self-similarity value with respect to the dependency list of Figure 3c as Figure 4c. In the case of C_w('control','control'), it is computed by the common child nodes, 'NE_2' and 'expression', and further C_w ('expression', 'expression') is recursively computed by the common child 'NE_1', and C_w('NE_1','NE_1') returns 0 since there is no child node of NE_1. We normalize entities with distinguishing the preceding NE as NE_1 and the following NE, as NE_2.

2.3.4 Hybrid kernel The dependency kernel is computed using word and POS dependencies with parent nodes and all their children's nodes that two graphs have in common. Thus, the dependency kernel is a word-level model based on words and their POS in the dependency graph. In order to give more contextual information, we define the hybrid kernel that adds common walks between two graphs to word-level model. The hybrid kernel is a composite kernel that combines the dependency and walk kernels to take advantage of two models. In the hybrid kernel, the lexical and syntactic walks are combined with the dependency kernel by summing up the number of common walks as follows:

$$\begin{aligned} &K_H(d_1, d_2) = K_D(d_1, d_2) + K_W(d_1, d_2) \\ &K_W(d_1, d_2) = \sum_{i=1}^{|d_1|-s+1} \sum_{j=1}^{|d_2|-s+1} K_p(d_1(i : i+p), d_2(j : j+p)(p = 3) \\ &K_{p(s,t)} = \begin{cases} 1 \text{ if } s = t \\ 0 \text{ otherwise} \end{cases} \end{aligned} \tag{3}$$

In Equation (3), $K_D(d_1, d_2)$ denotes the value of dependency kernel and $K_W(d_1, d_2)$ denotes the number of common walk features between two shortest path strings, d_1, d_2. The string $d(i{:}i+p)$ means the length p substring $d_i \ldots d_{i+p}$ of d. In the case of interaction ('sigF', 'ywhE') in Figure 2a, the similarity value of a hybrid kernel is evaluated as in Figure 4d. It is the summation of Figure 4b) and c.In fact, both common subgraphs and common walks reflect the similarity of structures between two graphs in different ways. While the model by

Table 2. Extraction performance on the LLL data

Kernel	Action	Bind	Regulon	No interaction	All	
Predicate	COR = 28	COR = 8	COR = 3	COR = 0	COR = 39	PRE = 70.9
	MIS = 8/36	MIS = 6/14	MIS = 1/4	MIS = 0/0	MIS = 15/54	REC = 72.2
	SPU = 1/29	SPU = 6/14	SPU = 0/3	SPU = 9/9	SPU = 16/55	F-M = 71.5
Walk	COR = 29	COR = 13	COR = 3	COR = 0	COR = 45	PRE = 72.5
	MIS = 7/36	MIS = 1/14	MIS = 1/4	MIS = 0/0	MIS = 9/54	REC = 83.3
	SPU = 3/32	SPU = 7/20	SPU = 0/3	SPU = 7/7	SPU = 17/62	F-M = 77.5
Dependency	COR = 20	COR = 11	COR = 3	COR = 0	COR = 34	PRE = 58.6
	MIS = 16/36	MIS = 3/14	MIS = 1/4	MIS = 0/0	MIS = 20/54	REC = 62.9
	SPU = 2/22	SPU = 7/18	SPU = 0/3	SPU = 15/15	SPU = 24/58	F-M = 60.7
Hybrid	COR = 25	COR = 12	COR = 3	COR = 0	COR = 40	PRE = 65.5
	MIS = 8/36	MIS = 2/14	MIS = 1/4	MIS = 0/0	MIS = 14/54	REC = 74.0
	SPU = 1/29	SPU = 8/20	SPU = 0/3	SPU = 10/10	SPU = 21/61	F-M = 69.5

common sub-structures gives a more comprehensive comparison of two graphs, common walks provide simplified context information that helps alleviate the data sparseness problem.

3 RESULTS

3.1 Experimental results and discussions

In the experiments, we evaluate the proposed four relation kernels using SVM on the LLL 05 challenge task. As a preprocessing step for genic interaction extraction, candidate biomedical NEs (genes/proteins) are first recognized using the provided NE dictionary.

We conducted learning for the kernels with 464 interacting NE pairs including 300 negative NE pairs, and classified 330 NE pairs in the test set. The training data of the LLL shared task does not explicitly describe negative examples, so any pairs for which interaction is not specified are used as negative examples for effective learning. At present, our system assigns the 'O' class to the NE pairs if any link path from source NE to destination NE is not found. Most of the NE pairs actually have no genic interaction but some pairs have genic relations.

As mentioned earlier, we tested our kernels with a SVM learner. Because SVM robustly handles a large-sized feature set and provides a high generalization performance even on unseen examples, it has been successfully applied to many NLP tasks. For the SVM learning, we used the LIBSVM 2.84[3] package wherein we can utilize our own kernel with the pre-computed kernel option as well as other well-known kernels such as polynomial, radial basis and sigmoid functions (Hsu *et al.*, 2003). It also supports multiclass classificationcc. In the case of the predicate kernel and the walk kernel, feature vectors are passed to the polynomial kernel with parameters, $\gamma = 1$, $d = 2$, $r = 0$, $c = 1000$ for C-SVM classification.[4] In contrast, the dependency and the hybrid kernel were directly used with the pre-computed kernel option.

Table 2 shows the performance of each kernel over the LLL data. The LLL task requires directed interactions, so the agent NE and the target NE for an interaction should be identified. The correct answer for each sentence is computed in a strict manner. For a given sentence ID, an interaction in the prediction file must be exactly the same as that in the key file of the evaluation system. In this task, we can evaluate the performance of each kernel only on the Web program[5] and the correct answers for the test data are hidden. The program computes the *F*-score for a given prediction file based on the following scores: 'COR' (correct), 'MIS' (missing) and 'SPU' (spurious). COR denotes the number of interactions in the prediction file that exists in the key file. MIS is the number of interactions in the key file that is missing in the prediction file. SPU is the number of interactions in the prediction file that is wrong, i.e. is not in the key file. The formula for the *F*-score is

$$\begin{aligned} PRE(precision) &= \frac{COR}{(COR + SPU)} \\ REC(recall) &= \frac{COR}{COR + MIS} \\ F{-}score &= \frac{2*(PRE*REC)}{(PRE + REC)} \end{aligned} \tag{4}$$

As shown in Table 2, the walk kernel outperforms the other kernels. It seems that the predicate kernel is less informative than the walk kernel since only predicates and their direct child nodes on the shortest paths of NE pairs are considered. In fact, the predicate kernel considers the most minimal subparts of the shortest path compared with other kernels. As we can expect, the walk kernel was better than the predicate kernel.

The performance of the dependency kernel was the worst among the four kernels. In general, structure-based kernels are increasingly appealing for learning rich structural data without extensive feature engineering and selection process. However, they produced more erroneous results in our experiments. One of the reasons might be that the kernel operates on the comparatively concise structures, the shortest graph paths between NE

[3]http://www.csie.ntu.edu.tw/~cjlin/libsvm/

[4]SVM employs an iterative training algorithm, which is used to minimize an error function. SVM models can be variously classified according to the form of the error function. C-SVM is one of them and c is a penalty parameter of error terms.

[5]http://genome.jouy.inra.fr/texte/LLLchallenge/scoringService.php

pairs, unlike other structure-based approaches. Additional constituents not on the path can play an important role in computing the similarity between two structures. Another reason is that the dependency kernel would not be optimal since all common subgraphs are counted equally regardless of importance of each subgraph. In other words, some subgraphs can be more useful structures than others for learning. In addition, C_w and C_p have different properties in different feature spaces. That is, C_w is related to lexical and C_p, to morpho-syntactic subgraphs. However, the current kernel function for the dependency kernel makes no distinction between structures with different properties, but focuses only on the common subgraph counts. In contrast, the feature-based models learn different weights for different features. More considerable work remains to be done to extract the full potential of structure-based kernels since the equally weighted counts of all subgraphs are not much effective for relation learning.

The hybrid kernel was better than the dependency kernel but was just comparable to the predicate kernel. This is because walk provides more concrete information, which shows the effectiveness of structural information in relation extraction.

In conclusion, it turns out that the feature-based kernels outperform the structure-based kernels in our experiments. Also, the simplified structural information provided by walks helps to identify the relation of a pair of NEs. It is interesting to note that the walk in three elements of (vertex, edge, vertex) or (edge, vertex, edge) provides good evidence for relation extraction, in that it is similar in other NLP applications such as language modeling and POS tagging. The use of the lexical and syntactic walks shows the best performance for genic interaction extraction.

According to the evaluation results, our system often failed to handle the negative examples. As shown in Table 2, it tended to misidentify the NE pairs with no interaction as interactive pairs. In particular, the predicate kernel was weak for the decision of 'bind' interaction.

Next, we compared our system with other systems tested on the LLL data. Table 3 shows the comparison results. Hakenberg *et al*. (2005) applied sequence alignment and finite state automata to generate syntactic patterns for identifying genic interactions. Riedel and Klein (2005) suggested a Markov Logic model to create a set of weighted clauses on a discourse representation structure that can classify pairs of NEs as genic interactions. Fundel *et al*. (2006) created candidate relations from dependency parse trees by applying a small number of rules. To our knowledge, our walk kernel showed the best performance on the LLL data set. In particular, the recall rate, which has been pointed out as a drawback of pattern approaches, was high, although it was impossible to analyze errors in detail because we could not get the set of correct interactions for the test data.

Table 3. comparison with other systems

IE system	Precision	Recall	F-M
Riedel and Klein (2005)	65.0	72.2	68.4
Hakenberg *et al*. (2005)	50.0	53.8	51.8
Fundel *et al*. (2006)	68	78	72
Our system	72.5	83.3	77.5

3.2 Conclusions and future work

In this article, we suggested four genic relation extraction kernels defined on the shortest dependency path between two NEs. We gradually augmented structural information on the shortest dependency path from the predicate kernel to the hybrid kernel. We dealt with the interaction extraction problem in terms of data representation, semantic role and syntactic aspects. As a result, we achieved a very promising result on the LLL data set.

One of the objectives for future works is to investigate how much influence words that are not on the shortest dependency path have on the interaction extraction decision, or that the words have nothing to do with the decision. Also, we will test and evaluate our kernels on the BioInfer[6] corpus that is based on Link Grammar dependency graphs. The LLL task data is the manually verified parsing output. Thus, we need to check how unfiltered dependency graphs found in real-world data affect performance.

In addition, we expect that our kernel can be extended with other kernels, which have been used for relation extraction, such as the contiguous kernel, sparse kernel, dependency tree kernel and subsequence kernel (Bunescu *et al*., 2005; Culotta and Sorensen, 2004; Zelenko *et al*., 2003).

ACKNOWLEDGEMENT

This work was supported by the BK21 program of the Ministry of Education, Republic of Korea.

Conflict of Interest: none declared.

REFERENCES

Aubin,S. (2005) Challenge LLL syntactic analysis guidelines. *Technical reports*. Université Paris Nord.

Blaschke,C. *et al*. (1999) Automatic extraction of biological information from scientific text: protein-protein interactions. In Proceedings of American Association for Artificial Intelligence, Orlando, USA, pp. 60–67.

Bunescu,R. and Mooney,R. (2005) Subsequence kernels for relation extraction. In Proceedings of the 19th Conference on Neural Information Processing Systems, Whistler, Canada.

Bunescu,R. *et al*. (2005) Comparative experiments on learning information extractors for proteins and their interactions. *Artif. Intell. Med.*, **33**, 135–155.

Carreras,X. and Màrquez,L. (2005) Introduction to the CoNLL-2005 shared task: semantic role labeling. In Proceedings of the 9th Conference on Computational Natural Language Learning (CoNLL), Ann Arbor, USA, pp. 152–164.

Collins,M. and Duffy,N. (2001) Parsing with a single neuron: convolution kernels for natural language problems. *Technical reports UCSC-CRL-01-01*. University of California at Santa Cruz.

Cormen,T.H. *et al*. (2001) *Introduction to Algorithms*. 2nd edn. MIT Press and McGraw-Hill, ISBN 0-262-03293-7. Section 24.3: Dijkstra's algorithm, Cambridge, USA, pp. 595–601.

Culotta,A. and Sorensen,J. (2004) Dependency tree kernels for relation extraction. In Proceedings of ACL 2004, Barcelona, Spain, pp. 423–429.

Ding,J. et al. (2003) Extracting biochemical interactions from MEDLINE using a link grammar parser. In Proceedings of the 15th

[6]http://www.it.utu.fi/BioInfer/

IEEE International Conference on Tools with Artificial Intelligence, Sacramento, USA, pp. 467–471.

Fundel,K. *et al.* (2006) RelEx – relation extraction sing dependency parse trees. *Bioinformatics*, **23**, 365–371.

Grinberg,D. *et al.* (1995) A robust parsing algorithm for link grammars. *Technical report CMU-CS-95-125*. Carnegie Mellon University Computer Science, and In Proceedings of the 4th International Workshop on Parsing Technologies, Prague, Czech, pp. 111–125.

Grishman,R. (1995) Message Understanding Conference 6, Columbia, USA. http://cs.nyu.edu/faculty/grishman/muc6.html.

Hakenberg,J. *et al.* (2005) LLL05 challenge: genic interaction extraction – identification of language patterns based on alignment and finite state automata. In Proceedings of the 4th Learning Language in Logic workshop (LLL05), Edinburgh, UK, pp. 38–45.

Hao,Y. *et al.* (2005) Discovering patterns to extract protein-protein interactions from the literature: Part II. *Bioinformatics*, **21**, 3294–3300.

Hsu,C. et al. (2003) A practical guide to vector classification. http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Huang,M. *et al.* (2004) Discovering patterns to extract protein-protein interactions from full texts. *Bioinformatics*, **20**, 3604–3612.

Jang,H. *et al.* (2006) Finding the evidence for protein-protein interactions from PubMed abstracts. *Bioinformatics*, **22**, 220–226.

Moschitti,A. (2004) A study on convolution kernels for shallow semantic parsing. In Proceedings of ACL 2004, Barcelona, Spain, pp. 335–342.

Ono,T. *et al.* (2001) Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, **17**, 155–161.

Riedel,S. and Klein,E. (2005) Genic interaction extraction with semantic and syntactic chains. In Proceedings of the 4th Learning Language in Logic workshop (LLL05), Edinburgh, UK, pp. 69–74.

Shawe-Taylor,J. and Cristianini,N. (2000) *Support Vector Machines and Other Kernel-based Methods*. Cambridge University Press, Cambridge, UK.

Sleator,D. and Temperly,D. (1993) Parsing English with a link grammar. In Third International Workshop on Parsing Technologies, Tilburg, Netherlands, pp. 277–291.

Tsivtsivadze,E. *et al.* (2006) Locality-convolution kernel and its application to dependency parse ranking. In Proceedings of Advances in Applied Artificial Intelligence, 19th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE, Annecy, France, pp. 610–618.

Xue,N. and Palmer,M. (2004) Calibrating features for semantic role labeling. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Barcelona, Spain, pp. 88–94.

Zelenko,D. *et al.* (2003) Kernel methods for relation extraction. *J. Mach. Learn. Res.*, **3**, 1083–1106.