Kernel approaches for genic interaction extraction

Seonho Kim1,*, Juntae Yoon2,* and Jihoon Yang1,*

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

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, discontinuous 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
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).

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

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’ 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 discontinuous 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.
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 ‘transcription’ is also a central phenomenon in functional genomics involved in gene interaction, which is a popular IE problem.

The analysis of the expression of a translational ywhE-lacZ fusion showed that ywhE expression is sporulation-specific, and is controlled predominantly by the forespore-specific sigma factor sigma(F), and to a lesser extent by sigma(G).

b In addition to controlling transcription of phrC, sigmaII appears to control expression of at least one other gene required for production of CSF.

c KinC and KinD were responsible for Spo0A-P production during the exponential phase of growth in the absence of KinA and KmB.

d During endospore formation in Bacillus subtilis, the DNA binding protein GerE stimulated transcription from several promoters that are used by RNA polymerase containing sigmaK.

Fig. 2. Examples of biomedical parsed sentences.
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 undirected 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.

#### 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 syntactic graph for a sentence. In this work, we suggest two distinct types of kernels to extract relations: feature-based kernels and the dependency and hybrid kernels belonging to the morpho-syntactic information of the word at the node and a set of $(relation, dependent_word)$ pairs. We use the list structures for the other kernel methods, the dependency and hybrid kernels (Fig. 3d).
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:

$$K(x_i, x_j) = (y_i^T y_j + r)^d$$

Here, $y_i$ is the feature vector of pattern $i$, and $y$, $r$, 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 on the adjacent words ‘agent’ and ‘target’ of the predicate, ‘control’. 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 contexts by syntactic relations are crucial.

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 the 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 substructures 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;
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 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’, ‘whyE’) 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 \( children\_x(x) \) refers to the direct dependents of \( x \) represented by the set of (relation, \( \text{word} \)) pairs that consists of the words of direct dependents of \( x \) and the syntactic relations with them. \( \text{POS}(x) \) refers to the POS of node \( x \) and \( \text{children}_{\text{POS}}(x) \) denotes the set of (relation, \( \text{POS} \)) pairs that are direct dependents of \( x \). Given two parent nodes \( n_1 \) and \( n_2 \), \( \text{sc}_p(n_1, n_2) \) is the set of common word dependencies between two subgraphs rooted by \( n_1 \) and \( n_2 \), respectively. \( \text{sc}_p(p_1, p_2) \) corresponds to the set of common POS dependencies between two subgraphs rooted by \( \text{POS}(p_1) \) and \( \text{POS}(p_2) \), 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 \( \text{sc}_p(n_1, n_2) \).

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 \( \text{POS}(p_1) \) and \( \text{POS}(p_2) \). 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 \( \text{sc}_p(n_1, n_2) \).

\[
\text{SC}_w(n_1, n_2) = \begin{cases} \text{C}_w(n_1, n_2) & \text{if word}(n_1) = \text{word}(n_2) \text{ or child}(n_1) = \text{empty or child}(n_2) = \text{empty,} \\ 0 & \text{if \text{POS}(p_1) \neq \text{POS}(p_2) \text{ or children}(p_1) = \text{empty or children}(p_2) = \text{empty,} } \end{cases}
\]

\[
\text{Kernel approaches}
\]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pword</td>
<td>Predicate word(s)</td>
</tr>
<tr>
<td>Ppos</td>
<td>Pos of the predicate word(s)</td>
</tr>
<tr>
<td>L_srole</td>
<td>Syntactic relation of immediate left child and predicate</td>
</tr>
<tr>
<td>R_srole</td>
<td>Syntactic relation of immediate right child and predicate</td>
</tr>
<tr>
<td>L_head</td>
<td>Word of predicate’s immediate left child</td>
</tr>
<tr>
<td>R_head</td>
<td>Word of predicate’s immediate right child</td>
</tr>
<tr>
<td>SAME_NE</td>
<td>Are the NEs the same?</td>
</tr>
<tr>
<td>SAME_ROLE</td>
<td>Are L_srole and R_srole the same?</td>
</tr>
<tr>
<td>LINK_TYPE</td>
<td>No Link, Link_onePred, Link_Preds</td>
</tr>
</tbody>
</table>

**Table 1. Features for the predicate kernel**

...
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 classification. 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. not in the key file. The formula for the F-score is

\[
\text{PRE}(\text{precision}) = \frac{\text{COR}}{(\text{COR} + \text{SPU})}
\]

\[
\text{REC}(\text{recall}) = \frac{\text{COR}}{(\text{COR} + \text{MIS})}
\]

\[
F-\text{score} = \frac{2 \times (\text{PRE} \times \text{REC})}{(\text{PRE} + \text{REC})}
\]  

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.

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 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. not in the key file. The formula for the F-score is

\[
PRE(\text{precision}) = \frac{\text{COR}}{(\text{COR} + \text{SPU})}
\]

\[
REC(\text{recall}) = \frac{\text{COR}}{(\text{COR} + \text{MIS})}
\]

\[
F-\text{score} = \frac{2 \times (\text{PRE} \times \text{REC})}{(\text{PRE} + \text{REC})}
\]  

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.

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 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. not in the key file. The formula for the F-score is

\[
PRE(\text{precision}) = \frac{\text{COR}}{(\text{COR} + \text{SPU})}
\]

\[
REC(\text{recall}) = \frac{\text{COR}}{(\text{COR} + \text{MIS})}
\]

\[
F-\text{score} = \frac{2 \times (\text{PRE} \times \text{REC})}{(\text{PRE} + \text{REC})}
\]  

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

Table 2. Extraction performance on the LLL data

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Action</th>
<th>Bind</th>
<th>Regulon</th>
<th>No interaction</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate</td>
<td>COR = 28</td>
<td>COR = 8</td>
<td>COR = 3</td>
<td>COR = 0</td>
<td>COR = 39</td>
</tr>
<tr>
<td></td>
<td>MIS = 8/36</td>
<td>MIS = 6/14</td>
<td>MIS = 1/4</td>
<td>MIS = 0/0</td>
<td>MIS = 15/54</td>
</tr>
<tr>
<td>Walk</td>
<td>SPU = 1/29</td>
<td>SPU = 0/3</td>
<td>SPU = 9/9</td>
<td>SPU = 16/55</td>
<td>F-M = 71.5</td>
</tr>
<tr>
<td></td>
<td>MIS = 7/36</td>
<td>MIS = 1/4</td>
<td>MIS = 0/0</td>
<td>MIS = 9/54</td>
<td>REC = 83.3</td>
</tr>
<tr>
<td>Dependency</td>
<td>SPU = 3/32</td>
<td>SPU = 7/20</td>
<td>SPU = 0/3</td>
<td>SPU = 7/7</td>
<td>SPU = 17/62</td>
</tr>
<tr>
<td></td>
<td>COR = 20</td>
<td>COR = 11</td>
<td>COR = 3</td>
<td>COR = 0</td>
<td>COR = 34</td>
</tr>
<tr>
<td></td>
<td>MIS = 16/36</td>
<td>MIS = 3/14</td>
<td>MIS = 1/4</td>
<td>MIS = 0/0</td>
<td>MIS = 20/54</td>
</tr>
<tr>
<td>Hybrid</td>
<td>SPU = 2/22</td>
<td>SPU = 7/18</td>
<td>SPU = 0/3</td>
<td>SPU = 15/15</td>
<td>SPU = 24/58</td>
</tr>
<tr>
<td></td>
<td>COR = 25</td>
<td>COR = 12</td>
<td>COR = 3</td>
<td>COR = 0</td>
<td>COR = 40</td>
</tr>
<tr>
<td></td>
<td>MIS = 8/36</td>
<td>MIS = 2/14</td>
<td>MIS = 1/4</td>
<td>MIS = 0/0</td>
<td>MIS = 14/54</td>
</tr>
<tr>
<td></td>
<td>SPU = 1/29</td>
<td>SPU = 8/20</td>
<td>SPU = 0/3</td>
<td>SPU = 10/10</td>
<td>SPU = 21/61</td>
</tr>
</tbody>
</table>

\(^{3}\)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 morphosyntactic 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.

### 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


---

\(^6\)http://www.it.utu.fi/BioInfer/
IEEE International Conference on Tools with Artificial Intelligence, Sacramento, USA, pp. 467–471.
Hao,Y. et al. (2005) Discovering patterns to extract protein-protein interactions from the literature: Part II. Bioinformatics, 21, 3294–3300.