A Study of Convolution Tree Kernel with Local Alignment

Lidan Zhang
Department of Computer Science
HKU, Hong Kong
lzhang@cs.hku.hk

Kwok-Ping Chan
Department of Computer Science
HKU, Hong Kong
kpchan@cs.hku.hk

Abstract

This paper discusses a new convolution tree kernel by introducing local alignments. The main idea of the new kernel is to allow some syntactic alternations during each match between subtrees. In this paper, we give an algorithm to calculate the composite kernel. The experiment results show promising improvements on two tasks: semantic role labeling and question classification.

1 Introduction

Recently kernel-based methods have become a state-of-art technique and been widely used in natural language processing applications. In this method, a key problem is how to design a proper kernel function in terms of different data representations. So far, there are two kinds of data representations. One is to encode an object with a flat vector whose element correspond to an extracted feature from the object. However the feature vector is sensitive to the structural variations. The extraction schema is heavily dependent on different problems. On the other hand, kernel function can be directly calculated on the object. The advantages are that the original topological information is to a large extent preserved and the introduction of additional noise may be avoided. Thus structure-based kernels can well model syntactic parse tree in a variety of applications, such as relation extraction(Zelenko et al., 2003), named entity recognition(Culotta and Sorensen, 2004), semantic role labeling(Moschitti et al., 2008) and so on.

To compute the structural kernel function, Haussler (1999) introduced a general type of kernel function, called“ Convolution kernel”. Based on this work, Collins and Duffy (2002) proposed a tree kernel calculation by counting the common subtrees. In other words, two trees are considered if and only if these two trees are exactly same. In real sentences, some structural alternations within a given phrase are permitted without changing its usage. Therefore, Moschitti (2004) proposed partial trees to partially match between subtrees. Kashima and Koyanagi (2002) generalize the tree kernel to labeled order tree kernel with more flexible match. And from the idea of introducing linguistical knowledge, Zhang et al. (2007) proposed a grammar-driven tree kernel, in which two subtrees are same if and only if the corresponding two productions are in the same manually defined set. In addition, the problem of hard matching can be alleviated by processing or mapping the trees. For example, Tai mapping (Kuboyama et al., 2006) generalized the kernel from counting subtrees to counting the function of mapping. Moreover multi-source knowledge can benefit kernel calculation, such as using dependency information to dynamically determine the tree span (Qian et al., 2008).

In this paper, we propose a tree kernel calculation algorithm by allowing variations in productions. The variation is measured with local alignment score between two derivative POS sequences. To reduce the computation complexity, we use the dynamic programming algorithm to compute the score of any alignment. And the top n alignments are considered in the kernel.

Proceedings of the EACL 2009 Workshop on GEMS: GEometical Models of Natural Language Semantics, pages 25–32,
Athens, Greece, 31 March 2009.

Another problem in Collins and Duffy's tree kernel is context-free. It does not consider any semantic information located at the leaf nodes of the parsing trees. To lexicalized tree kernel, Bloehdorn et al. (2007) considered the associated term similarity by virtue of WordNet. Shen et al. (2003) constructed a separate lexical feature containing words on a given path and merged into the kernel in linear combination.

The paper is organized as follows. In section 2, we describe the commonly used tree kernel. In section 3, we propose our method to make use of the local alignment information in kernel calculation. Section 4 presents the results of our experiments for two different applications (Semantic Role Labeling and Question Classification). Finally section 5 provides our conclusions.

2 Convolution Tree Kernel

The main idea of tree kernel is to count the number of common subtrees between two trees T_1 and T_2. In convolutional tree kernel (Collins and Duffy, 2002), a tree(T) is represented as a vector $h(T) = (h_1(T), ..., h_i(T), ..., h_n(T))$, where $h_i(T)$ is the number of occurrences of the i^{th} tree fragment in the tree T. Since the number of subtrees is exponential with the parse tree size, it is infeasible to directly count the common subtrees. To reduce the computation complexity, a recursive kernel calculation algorithm was presented. Given two trees T_1 and T_2,

$$\begin{aligned} K(T_1, T_2) &= < h(T_1), h(T_2) > \\ &= \sum_i h_i(T_1) h_i(T_2) \\ &= \sum_i (\sum_{n_1 \in N_{T_1}} I_i(n_1) \sum_{n_2 \in N_{T_2}} I_i(n_2)) \\ &= \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \triangle(n_1, n_2) \end{aligned} \quad (1)$$

where, N_{T_1} and N_{T_2} are the sets of all nodes in trees T_1 and T_2, respectively. $I_i(n)$ is the indicator function to be 1 if i-th subtree is rooted at node n and 0 otherwise. And $\triangle(n_1, n_2)$ is the number of common subtrees rooted at n_1 and n_2. It can be computed efficiently according to the following rules:

(1) If the productions at n_1 and n_2 are different, $\triangle(n_1, n_2) = 0$

(2) If the productions at n_1 and n_2 are same, and n_1 and n_2 are pre-terminals, then $\triangle(n_1, n_2) = \lambda$

(3) Else, $\triangle(n_1, n_2) = \lambda \prod_j^{nc(n_1)} (1 + \triangle(ch(n_1, j), ch(n_2, j)))$

where $nc(n_1)$ is the number of children of n_1 in the tree. Note that $n_1 = n_2$ because the productions at n_1 and n_2 are same. $ch(n_1, j)$ represents the j^{th} child of node n_1. And $0 < \lambda \leq 1$ is the parameter to downweight the contribution of larger tree fragments to the kernel. It corresponds to $K(T_1, T_2) = \sum_i \lambda^{size_i} h_i(T_1) h_i(T_2)$, where $size_i$ is the number of rules in the i'th fragment. The time complexity of computing this kernel is $O(|N_{T_1}| \cdot |N_{T_2}|)$.

3 Tree Kernel with Local Alignment

3.1 General Framework

As we referred, one of problems in the basic tree kernel is its hard match between two rules. In other words, at each tree level, the two subtrees are required to be perfectly equal. However, in real sentences, some modifiers can be added into a phrase without changing the phrase's function. For example, two sentences are given in Figure 1. Considering "A1" role, the similarities between two subtrees(in circle) are 0 in (Collins and Duffy, 2002), because the productions "NP→DT ADJP NN" and "NP→DT NN" are not identical. From linguistical point of view, the adjective phrase is optional in real sentences, which does not change the corresponding semantic role. Thus the modifier components(like "ADJP" in the above example) should be neglected in similarity comparisons.

To make the hard match flexible, we can align two string sequences derived from the same node. Considering the above example,

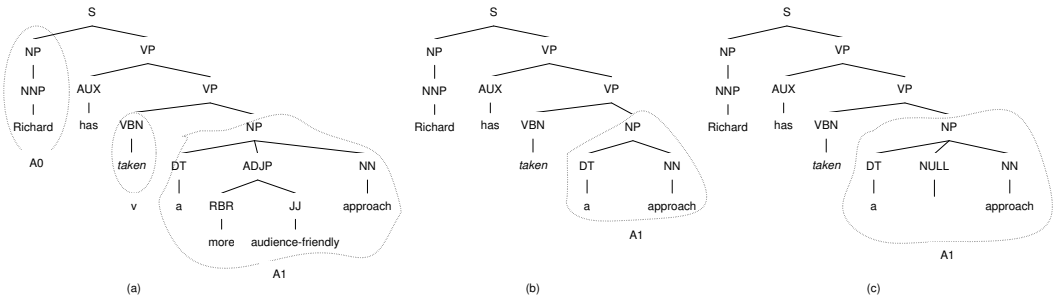

Figure 1: Syntactic parse tree with "A1" semantic role

an alignment might be "DT ADJP NN" vs "DT - NN", by inserting a symbol(-). The symbol(-) corresponds to a "NULL" subtree in the parser tree. And the "NULL" subtree can be regarded as a null character in the sentence, see Figure 1(c).

Convolution kernels, studied in (Haussler, 1999) gave the framework to construct a complex kernel from its simple elements. Suppose $x \in X$ can be decomposed into $x_1, ..., x_m \equiv \vec{x}$. Let R be a relation over $X_1 \times ... \times X_m \times X$ such that $R(\vec{x})$ is true iff $x_1, ..., x_m$ are parts of x. $R^{-1}(x) = \{\vec{x}|R(\vec{x}, x)\}$, which returns all components. For example, x is any string, then $\vec{x}$ can be its characters. The convolution kernel K is defined as:

$$K(x, y) = \sum_{\vec{x} \in R^{-1}(x), \vec{y} \in R^{-1}(y)} \prod_{d=1}^{m} K_d(x_d, y_d) \quad (2)$$

Considering our problem, for example, a derived string sequence x by the rule "$n_1 \rightarrow x$". $R(x_i, x)$ is true iff x_i appears in the right hand of x. Given two POS sequences x and y derived from two nodes n_1 and n_2, respectively, $A(x, y)$ denotes all the possible alignments of the sequence. The general form of the kernel with local alignment is defined as:

$$K'(n_1, n_2) = \sum_{(i,j) \in A(x,y)} K(n_1^i, n_2^j) \quad (3)$$

$$\triangle'(n_1, n_2) = \lambda \sum_{(i,j) \in A(x,y)} AS^{(i,j)} \prod_{d=1}^{nc(n_1,i)} (1 + \triangle'(ch(n_1, i, d), ch(n_2, j, d)))$$

where, (i, j) denotes the i^{th} and j^{th} variation for x and y, $AS^{(i,j)}$ is the score for alignment i and j. And $ch(n_1, i, d)$ selects the d^{th} subtree for the i^{th} aligned schema of node n_1.

It is easily to prove the above kernel is positive semi-definite, since the kernel $K(n_1^i, n_2^j)$ is positive semi-definite. The native computation is impractical because the number of all possible alignments($|A(x, y)|$) is exponential with respect to $|x|$ and $|y|$. In the next section, we will discuss how to calculate $AS^{(i,j)}$ for each alignment.

3.2 Local Alignment Kernel

The local alignment(LA) kernel was usually used in bioinformatics, to compare the similarity between two protein sequences(x and y) by exploring their alignments(Saigo et al., 2004).

$$K_{LA}(x, y) = \sum_{\pi \in A(x,y)} \exp^{\beta s(x,y,\pi)} \quad (4)$$

where $\beta \geq 0$ is a parameter, $A(x, y)$ denotes all possible local alignments between x and y, and $s(x, y, \pi)$ is the local alignment score for a given alignment schema π, which is equal to:

$$s(x, y, \pi) = \sum_{i=1}^{|\pi|} S(x_{\pi_1^i}, y_{\pi_2^i}) - \sum_{j=1}^{|\pi|-1} [g(\pi_1^{i+1} - \pi_1^i) + g(\pi_2^{i+1} - \pi_2^i)] \quad (5)$$

In equation(5), S is a substitution matrix, and g is a gap penalty function. The alignment score is the sum of the substitution score between the correspondence at the aligned position, minus the sum of the gap penalty for the

case that '-' symbol is inserted. In natural language processing, the substitution matrix can be selected as identity matrix and no penalty is accounted.

Obviously, the direct computation of the original K_{LA} is not practical. Saigo (2004) presented a dynamic programming algorithm with time complexity $O(|x|\cdot|y|)$. In this paper, this dynamic algorithm is used to compute the kernel matrix, whose element(i,j) is used as $AS^{(i,j)}$ measurement in equation(3).

3.3 Local Alignment Tree Kernel

Now we embed the above local alignment score into the general tree kernel computation. Equation(3) can be re-written into following:

$$\triangle'(n_1,n_2)=\lambda\sum_{\pi\in A(x,y)}(\exp^{\beta s(x,y,\pi)}\times \prod_{k=1}^{nc(n_1,i)}(1+\triangle'(ch(n_1,i,k),ch(n_2,j,k)))) \quad (6)$$

To further reduce the computation complexity, a threshold (ξ) is used to filter out alignments with low scores. This can help to avoid over-generated subtrees and only select the significant alignments. In other words, by using the threshold (ξ), we can select the salient subtree variations for kernels. The final kernel calculation is shown below:

$$\triangle'(n_1,n_2)=\lambda\sum_{\substack{\pi\in A(x,y)\\ s(x,y,\pi)>\xi}}(\varepsilon^{\beta s(x,y,\pi)}\times \prod_{k=1}^{nc(n_1,i)}(1+\triangle'(ch(n_1,i,k),ch(n_2,j,k)))) \quad (7)$$

After filtering, the kernel is still positive semi-definite. This can be easily proved using the theorem in (Shin and Kuboyama, 2008), since this subset selection is transitive. More specifically, if $s(x,y,\pi)>\xi \bigwedge s(y,z,\pi')>\xi$, then $s(x,z,\pi+\pi')>\xi$.

The algorithm to compute the local alignment tree kernel is given in algorithm 1. For any two nodes pair(x_i and y_j), the local alignment score $M(x_i,y_j)$ is assigned. In the kernel matrix calculation, the worst case occurs when the tree is balanced and most of the alignments are selected.

Algorithm 1 algorithm for local alignment tree kernel

Require: 2 nodes n_1,n_2 in parse trees;The productions are $n_1\rightarrow x_1,...,x_m$ and $n_2\rightarrow y_1,...,y_n$

```
return △'(n1, n2)
if n1 and n2 are not same then
    △'(n1, n2) = 0
else
    if both n1 and n2 are pre-terminals then
        △'(n1, n2) = 1
    else
        calculate kernel matrix by equation( 4)
        for each possible alignment do
            calculate △'(n1, n2) by equation(7)
        end for
    end if
end if
```

4 Experiments

4.1 Semantic Role Labeling

4.1.1 Experiment Setup

We use the CoNLL-2005 SRL shared task data(Carreras and Marquez, 2005) as our experimental data. It is from the Wall Street Journal part of the Penn Treebank, together with predicate-arguments information from the PropBank. According to the shared task, sections 02-21 are used for training, section 24 for development and section 23 as well as some data from Brown corpus are left for test. The data sets are described in Table 1.

		Sentences	Arguments
Training		39,832	239,858
Dev		1,346	8,346
Test	WSJ	1,346	8,346
	Brown	450	2,350

Table 1: Data sets statistics

Considering the two steps in semantic role labeling, i.e. semantic role identification and recognition. We assume identification has been done correctly, and only consider the semantic role classification. In our experiment, we focus on the semantic classes include 6 core (A0-A5), 12 adjunct(AM-) and 8 reference(R-) arguments.

In our implementation, SVM-Light-TK[1] (Moschitti, 2004) is modified. For SVM multi-classifier, the ONE-vs-ALL (OVA) strategy is selected. In all, we prepare the data for each semantic role (r) as following:

(1) Given a sentence and its correct full syntactic parse tree;

(2) Let P be the predicate. Its potential arguments A are extracted according to (Xue and Palmer, 2004)

(3) For each pair $< p, a > \in P \times A$: if a covers exactly the words of semantic role of p, put minimal subtree $< p, a >$ into positive example set (T_r^+); else put it in the negative examples (T_r^-)

In our experiments, we set $\beta = 0.5$.

4.1.2 Experimental Results

The classification performance is evaluated with respect to accuracy, precision(p), recall(r) and $F_1 = 2pr/(p + r)$.

	Accuracy(%)
(Collins and Duffy, 2002)	84.35
(Moschitti, 2004)	86.72
(Zhang et al., 2007)	87.96
Our Kernel	88.48

Table 2: Performance comparison between different kernel performance on WSJ data

[1]http://dit.unitn.it/ moschitt/Tree-Kernel.htm

	P(%)	R(%)	$F_{\beta=1}$
Development	81.03	68.91	74.48
WSJ Test	84.97	79.45	82.11
Brown Test	76.95	70.94	73.51
WSJ+Brown	82.98	75.40	79.01

WSJ	P(%)	R(%)	F
A0	81.28	83.90	82.56
A1	84.22	66.39	74.25
A2	77.27	62.36	69.02
A3	93.33	21.21	34.57
A4	82.61	51.35	63.33
A5	100.00	40.00	57.41
AM-ADV	74.21	56.21	63.92
AM-CAU	75.00	46.09	57.09
AM-DIR	57.14	16.00	25.00
AM-DIS	77.78	70.00	73.68
AM-EXT	75.00	53.10	62.18
AM-LOC	89.66	74.83	81.57
AM-MNR	84.62	48.20	61.41
AM-MOD	96.64	92.00	94.26
AM-NEG	99.30	95.30	97.26
AM-PNC	48.20	28.31	35.67
AM-PRD	50.00	30.00	37.50
AM-TMP	87.87	73.43	80.00
R-A0	81.08	67.80	73.85
R-A1	77.50	49.60	60.49
R-A2	58.00	42.67	49.17
R-AM-CAU	100.00	25.00	40.00
R-AM-EXT	100.00	100.00	100.00
R-AM-LOC	100.00	55.00	70.97
R-AM-MNR	50.00	25.00	33.33
R-AM-TMP	85.71	52.94	65.46

Table 3: top: overall performance result on data sets ; bottom: detail result on WSJ data

Table 2 compares the performance of our method and other three famous kernels on WSJ test data. We implemented these three methods with the same settings described in the papers. It shows that our kernel achieves the best performance with 88.48% accuracy. The advantages of our approach are: 1). the alignments allow soft syntactic structure match; 2). threshold can avoid over-generation and selected salient alignments.

Table 3 gives our performance on data sets and the detail result on WSJ test data.

Similarity	Definition
Wu and Palmer	$sim_{WUP}(c_1, c_2) = \frac{2dep(lso(c_1,c_2))}{d(c_1,lso(c_1,c_2))+d(c_2,lso(c_1,c_2))+2dep(lso(c_1,c_2))}$
Resnik	$sim_{RES}(c_1, c_2) = -\log P(lso(c_1, c_2))$
Lin	$sim_{LIN}(c_1, c_2) = \frac{2\log P(lso(c_1,c_2))}{\log P(c_1)+\log P(c_2)}$

Table 4: popular semantic similarity measurements

4.2 Question Classification

4.2.1 Semantic-enriched Tree Kernel

Another problem in the tree kernel (Collins and Duffy, 2002) is the lack of semantic information, since the match stops at the pre-terminals. All the lexical information is encoded at the leaf nodes of parsing trees. However, the semantic knowledge is important in some text applications, like Question Classification. To introduce semantic similarities between words into our kernel, we use the framework in Bloehdorn et al. (2007) and rewrite the rule (2) in the iterative tree kernel calculation(in section 2).

> (2) If the productions at n_1 and n_2 are same, and n_1 and n_2 are pre-terminals, then $\triangle(n_1, n_2) = \lambda\alpha k_w(w_1, w_2)$

where w_1 and w_2 are two words derived from pre-terminals n_1 and n_2, respectively, and the parameter α is to control the contribution of the leaves. Note that each preterminal has one child or equally covers one word. So $k_w(w_1, w_2)$ actually calculate the similarity between two words w_1 and w_2.

In general, there are two ways to measure the semantic similarities. One is to derive from semantic networks such as WordNet (Mavroeidis et al., 2005; Bloehdorn et al., 2006). The other way is to use statistical methods of distributional or co-occurrence (Ó Séaghdha and Copestake, 2008) behavior of the words.

WordNet[2] can be regarded as direct graphs semantically linking concepts by means of relations. Table 4 gives some similarity measures between two arbitrary concepts c_1 and c_2. For our application, the word-to-word similarity can be obtained by maximizing the corresponding concept-based similarity scores. In our implementation, we use WordNet::Similarity package[3](Patwardhan et al., 2003) and the noun hierarchy of WordNet.

In Table 4, dep is the length of path from a node to its global root, $lso(c_1, c_2)$ represents the lowest super-ordinate of c_1 and c_2. The detail definitions can be found in (Budanitsky and Hirst, 2006) .

As an alternative, Latent Semantic Analysis(LSA) is a technique. It calculates the words similarities by means of occurrence of terms in documents. Given a term-by-document matrix X, its singular value decomposition is: $X = U\Sigma V^T$, where Σ is a diagonal matrix with singular values in decreasing arrangement. The column of U are singular vectors corresponding to the individual singular value. Then the latent semantic similarity kernel of terms t_i and t_j is:

$$sim_{LSA} =< U_k^i (U_k^j)^T > \qquad (8)$$

where $U_k = I_k U$ is to project U onto its first k dimensions. I_k is the identity matrix whose first k diagonal elements are 1 and all the other elements are 0. And U_k^i is the i-th row of the matrix U_k. From equation (8), the LSA-based similarity between two terms is the inner product of the two projected vectors. The details of LSA can be found in (Cristianini et al., 2002; Choi et al., 2001).

4.2.2 Experiment Results

In this set of experiment, we evaluate different types of kernels for Question Classification(QC) task. The duty of QC is to categorize questions into different classes. In

[2]http://wordnet.princeton.edu/

[3]http://search.cpan.org/dist/WordNet-Similarity

Accuracy(%)		1000	2000	3000	4000	5500
BOW		77.1	83.3	87.2	87.3	89.2
TK		80.2	86.2	87.4	88.6	91.2
LATK		80.4	86.5	87.5	88.8	91.6
$\alpha = 1$	WUP	81.3	87.3	88.0	89.8	92.5
	RES	81.0	87.1	87.9	89.5	92.2
	LIN	81.1	87.0	88.0	89.3	92.4
	LSA($k = 50$)	80.8	86.9	87.8	89.3	91.7

Table 5: Classification accuracy of different kernels on different data sets

this paper we use the same dataset as introduced in(Li and Roth, 2002). The dataset is divided[4] into 5500 questions for training and 500 questions from TREC 20 for testing. The total training samples are randomly divided into 5 subsets with sizes 1,000, 2,000, 3,000, 4,000 and 5,500 respectively. All the questions are labeled into 6 coarse grained categories and 50 fine grained categories: Abbreviations (abbreviation and expansion), Entity (animal, body, color, creation, currency, medical, event, food, instrument, language, letter, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word), Description (definition, description, manner, reason), Human (description, group, individual, title), Location (city, country, mountain, state) and Numeric (code, count, date, distance, money, order, percent, period, speed, temperature, size, weight).

In this paper, we compare the linear kernel based on bag-of-word (BOW), the original tree kernel (TK), the local alignment tree kernel (section 3, LATK) and its correspondences with LSA similarity and a set of semantic-enriched LATK with different similarity metrics.

To obtain the parse tree, we use Charniak parser[5] for every question. Like the previous experiment, SVM-Light-TK software and the OVA strategy are implemented. In all experiments, we use the default parameter in SVM(e.g. margin parameter) and set $\alpha = 1$. In LSA model, we set $k = 50$. Finally, we use multi-classification accuracy to evaluate the performance.

Table 5 gives the results of the experiments. We can see that the local alignment tree kernel increase the multi-classification accuracy of the basic tree kernel by about 0.4%. The introduction of semantic information further improves accuracy. Among WordNet-based metrics, "Wu and Palmer" metric achieves the best result, i.e. 92.5%. As a whole, the WordNet-based similarities perform better than LSA-based measurement.

5 Conclusion

In this paper, we propose a tree kernel calculation by allowing local alignments. More flexible productions are considered in line with modifiers in real sentences. Considering text related applications, words similarities have been merged into the presented tree kernel. These similarities can be derived from different WordNet-based metrics or document statistics. Finally experiments are carried on two different applications (Semantic Role Labeling and Question Classification).

For further work, we plan to study exploiting semantic knowledge in the kernel. A promising direction is to study the different effects of these semantic similarities. We are interested in some distributional similarities (Lee, 1999) given certain context. Also the effectivenss of the semantic-enriched tree kernel in SRL is another problem.

References

Stephan Bloehdorn, Roberto Basili, Marco Cammisa, and Alessandro Moschitti. 2006. Semantic kernels for text classification based on topological measures

[4] http://l2r.cs.uiuc.edu/ cogcomp/Data/QA/QC/

[5] ftp://ftp.cs.brown.edu/pub/nlparser/

of feature similarity. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 808–812, Washington, DC, USA. IEEE Computer Society.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32(1):13–47.

X. Carreras and L. Marquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *CoNLL '05: Proceedings of the 9th Conference on Computational Natural Language Learning*.

Freddy Y. Y. Choi, Peter Wiemer-hastings, and Johanna Moore. 2001. Latent semantic analysis for text segmentation. In *In Proceedings of EMNLP*, pages 109–117.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL*, pages 263–270.

Nello Cristianini, John Shawe-Taylor, and Huma Lodhi. 2002. Latent semantic kernels. *J. Intell. Inf. Syst.*, 18(2-3):127–152.

Aron Culotta and Jeffrey Sorensen. 2004. Dependency tree kernels for relation extraction. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 423–429, Morristown, NJ, USA. Association for Computational Linguistics.

David Haussler. 1999. Convolution kernels on discrete structures. Technical report.

Tetsuji Kuboyama, Kilho Shin, and Hisashi Kashima. 2006. Flexible tree kernels based on counting the number of tree mappings. In *ECML/PKDD Workshop on Mining and Learning with Graphs*.

Lillian Lee. 1999. Measures of distributional similarity. In *37th Annual Meeting of the Association for Computational Linguistics*, pages 25–32.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Dimitrios Mavroeidis, George Tsatsaronis, Michalis Vazirgiannis, Martin Theobald, and Gerhard Weikum. 2005. Word sense disambiguation for exploiting hierarchical thesauri in text classification. In Alípio Jorge, Luís Torgo, Pavel Brazdil, Rui Camacho, and Gama Joao, editors, *Knowledge discovery in databases: PKDD 2005 : 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, volume 3721 of *Lecture Notes in Computer Science*, pages 181–192, Porto, Portugal. Springer.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Comput. Linguist.*, 34(2):193–224.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 335–342, Morristown, NJ, USA. Association for Computational Linguistics.

Diarmuid Ó Séaghdha and Ann Copestake. 2008. Semantic classification with distributional kernels. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 649–656, Manchester, UK, August. Coling 2008 Organizing Committee.

Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. 2003. Using measures of semantic relatedness for word sense disambiguation. In *In Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics (CICLING-03)*, pages 241–257.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 697–704, Manchester, UK, August. Coling 2008 Organizing Committee.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. 2004. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689.

Kilho Shin and Tetsuji Kuboyama. 2008. A generalization of haussler's convolution kernel: mapping kernel. In *ICML*, pages 944–951.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 88–94, Barcelona, Spain, July. Association for Computational Linguistics.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106.

Min Zhang, Wanxiang Che, Aiti Aw, Chew Lim Tan, Guodong Zhou, Ting Liu, and Sheng Li. 2007. A grammar-driven convolution tree kernel for semantic role classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 200–207, Prague, Czech Republic, June. Association for Computational Linguistics.