Kernel-Based Pronoun Resolution with Structured Syntactic Knowledge

Xiaofeng Yang[†] **Jian Su**[†] **Chew Lim Tan**[‡]

[†]Institute for Infocomm Research
21 Heng Mui Keng Terrace,
Singapore, 119613
{xiaofengy,sujian}@i2r.a-star.edu.sg

[‡] Department of Computer Science
National University of Singapore,
Singapore, 117543
tancl@comp.nus.edu.sg

Abstract

Syntactic knowledge is important for pronoun resolution. Traditionally, the syntactic information for pronoun resolution is represented in terms of features that have to be selected and defined heuristically. In the paper, we propose a kernel-based method that can automatically mine the syntactic information from the parse trees for pronoun resolution. Specifically, we utilize the parse trees directly as a structured feature and apply kernel functions to this feature, as well as other normal features, to learn the resolution classifier. In this way, our approach avoids the efforts of decoding the parse trees into the set of flat syntactic features. The experimental results show that our approach can bring significant performance improvement and is reliably effective for the pronoun resolution task.

1 Introduction

Pronoun resolution is the task of finding the correct antecedent for a given pronominal anaphor in a document. Prior studies have suggested that syntactic knowledge plays an important role in pronoun resolution. For a practical pronoun resolution system, the syntactic knowledge usually comes from the parse trees of the text. The issue that arises is how to effectively incorporate the syntactic information embedded in the parse trees to help resolution. One common solution seen in previous work is to define a set of features that represent particular syntactic knowledge, such as the grammatical role of the antecedent candidates, the governing relations between the candidate and the pronoun, and so on. These features are calculated by mining the parse trees, and then could be used for resolution by using manually designed rules (Lappin and Leass, 1994; Kennedy and Boguraev, 1996; Mitkov, 1998) or machine-learning methods (Aone and Bennett, 1995; Yang et al., 2004; Luo and Zitouni, 2005).

However, such a solution has its limitation. The syntactic features have to be selected and defined manually, usually by linguistic intuition. Unfortunately, what kinds of syntactic information are effective for pronoun resolution still remains an open question in this research community. The heuristically selected feature set may be insufficient to represent all the information necessary for pronoun resolution contained in the parse trees.

In this paper we will explore how to utilize the syntactic parse trees to help learning-based pronoun resolution. Specifically, we directly utilize the parse trees as a structured feature, and then use a kernel-based method to automatically mine the knowledge embedded in the parse trees. The structured syntactic feature, together with other normal features, is incorporated in a trainable model based on Support Vector Machine (SVM) (Vapnik, 1995) to learn the decision classifier for resolution. Indeed, using kernel methods to mine structural knowledge have shown success in some NLP applications like parsing (Collins and Duffy, 2002; Moschitti, 2004) and relation extraction (Zelenko et al., 2003; Zhao and Grishman, 2005). However, to our knowledge, the application of such a technique to the pronoun resolution task still remains unexplored.

Compared with previous work, our approach has several advantages: (1) The approach utilizes the parse trees as a structured feature, which avoids the efforts of decoding the parse trees into a set of syntactic features in a heuristic manner. (2) The approach is able to put together the structured feature and the normal features in a trainable model, which allows different types of information to be considered in combination for both learning and resolution. (3) The approach is applicable for practical pronoun resolution as the syntactic information can be automatically obtained from machine-generated parse trees. And our study shows that the approach works well under the commonly available parsers.

We evaluate our approach on the ACE data set. The experimental results over the different domains indicate that the structured syntactic feature incorporated with kernels can significantly improve the resolution performance (by 5%∼8% in the success rates), and is reliably effective for the pronoun resolution task.

The remainder of the paper is organized as follows. Section 2 gives some related work that utilizes the structured syntactic knowledge to do pronoun resolution. Section 3 introduces the framework for the pronoun resolution, as well as the baseline feature space and the SVM classifier. Section 4 presents in detail the structured feature and the kernel functions to incorporate such a feature in the resolution. Section 5 shows the experimental results and has some discussion. Finally, Section 6 concludes the paper.

2 Related Work

One of the early work on pronoun resolution relying on parse trees was proposed by Hobbs (1978). Hobbs' algorithm works by searching the parse trees in a heuristic order. Specifically, the algorithm processes one sentence at a time, using a left-to-right breadth-first searching strategy. It first checks the current sentence where the pronoun occurs. The first NP that satisfies constraints, like number and gender agreements, would be selected as the antecedent. If the antecedent is not found in the current sentence, the algorithm would traverse the trees of previous sentences in the text. As the searching processing is completely done on the parse trees, the performance of the algorithm would rely heavily on the accuracy of the parsing results.

Lappin and Leass (1994) reported a pronoun resolution algorithm which uses the syntactic representation output by McCord's Slot Grammar parser. A set of salience measures (e.g. *Subject*, *Object* or *Accusative* emphasis) is derived from the syntactic structure. The candidate with the highest salience score would be selected as the antecedent. In their algorithm, the weights of salience measures have to be assigned manually.

Category: whether the candidate is a definite noun phrase, indefinite noun phrase, pronoun, named-entity or others.

Reflexiveness: whether the pronominal anaphor is a reflexive pronoun.

Type: whether the pronominal anaphor is a male-person pronoun (like *he*), female-person pronoun (like *she*), single gender-neuter pronoun (like *it*), or plural gender-neuter pronoun (like *they*)

Subject: whether the candidate is a subject of a sentence, a subject of a clause, or not.

Object: whether the candidate is an object of a verb, an object of a preposition, or not.

Distance: the sentence distance between the candidate and the pronominal anaphor.

Closeness: whether the candidate is the candidate closest to the pronominal anaphor.

FirstNP: whether the candidate is the first noun phrase in the current sentence.

Parallelism: whether the candidate has an identical collocation pattern with the pronominal anaphor.

Table 1: Feature set for the baseline pronoun resolution system

Luo and Zitouni (2005) proposed a coreference resolution approach which also explores the information from the syntactic parse trees. Different from Lappin and Leass (1994)'s algorithm, they employed a maximum entropy based model to automatically compute the importance (in terms of weights) of the features extracted from the trees. In their work, the selection of their features is mainly inspired by the government and binding theory, aiming to capture the c-command relationships between the pronoun and its antecedent candidate. By contrast, our approach simply utilizes the parse trees as a structured feature, and lets the learning algorithm discover all possible embedded information that is necessary for pronoun resolution.

3 The Resolution Framework

Our pronoun resolution system adopts the common learning-based framework similar to those by Soon et al. (2001) and Ng and Cardie (2002).

In the learning framework, a training or testing instance is formed by a pronoun and one of its antecedent candidate. During training, for each pronominal anaphor encountered, a positive instance is created by paring the anaphor and its closest antecedent. Also a set of negative instances is formed by paring the anaphor with each of the non-coreferential candidates. Based on the training instances, a binary classifier is generated using a particular learning algorithm. During resolution, a pronominal anaphor to be resolved is paired in turn with each preceding antecedent candidate to form a testing instance. This instance is presented to the classifier which then returns a class label with a confidence value indicating the likelihood that the candidate is the antecedent. The candidate with the highest confidence value will be selected as the antecedent of the pronominal anaphor.

3.1 Feature Space

As with many other learning-based approaches, the knowledge for the reference determination is represented as a set of features associated with the training or test instances. In our baseline system, the features adopted include lexical property, morphologic type, distance, salience, parallelism, grammatical role and so on. Listed in Table 1, all these features have been proved effective for pronoun resolution in previous work.

3.2 Support Vector Machine

In theory, any discriminative learning algorithm is applicable to learn the classifier for pronoun resolution. In our study, we use Support Vector Machine (Vapnik, 1995) to allow the use of kernels to incorporate the structured feature.

Suppose the training set *S* consists of labelled vectors $\{(x_i, y_i)\}$, where x_i is the feature vector of a training instance and y_i is its class label. The classifier learned by SVM is

$$f(x) = sgn(\sum_{i=1} y_i a_i x * x_i + b) \qquad (1)$$

where a_i is the learned parameter for a support vector x_i. An instance x is classified as positive (negative) if $f(x) > 0$ ($f(x) < 0$)[1].

One advantage of SVM is that we can use kernel methods to map a feature space to a particular high-dimension space, in case that the current problem could not be separated in a linear way. Thus the dot-product $x_1 * x_2$ is replaced by a kernel function (or kernel) between two vectors, that is $K(x_1, x_2)$. For the learning with the normal features listed in Table 1, we can just employ the well-known polynomial or radial basis kernels that can be computed efficiently. In the next section we will discuss how to use kernels to incorporate the more complex structured feature.

4 Incorporating Structured Syntactic Information

4.1 Main Idea

Parse trees that cover the pronoun and the candidate under consideration could provide much syntactic information related to the two expressions. The commonly used syntactic knowledge for pronoun resolution, such as grammatical roles or the governing relations, can be directly described by the tree structures. Other syntactic knowledge that may be helpful for resolution could also be implicitly represented in the trees. Therefore, by comparing the common substructures between two trees we can find out to what degree two trees contain similar syntactic information, which can be done using a convolution tree kernel.

The value returned from the tree kernel reflects the similarity between two instances in syntax. Such syntactic similarity can be further combined with other knowledge to compute the overall similarity between two instances, through a composite kernel. And thus a SVM classifier can be learned and then used for resolution. This is just the main idea of our approach.

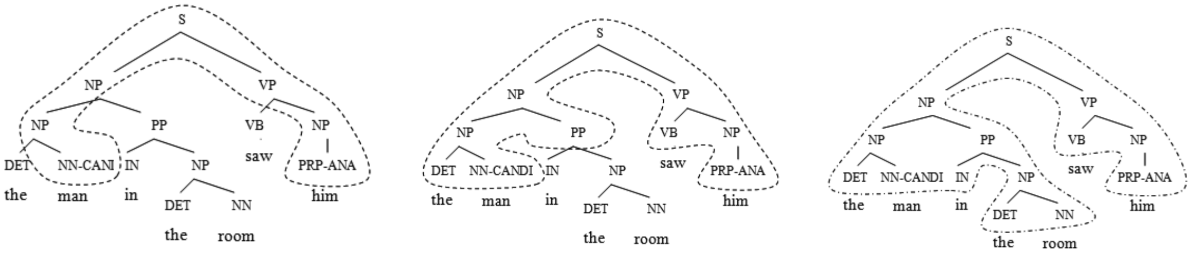
4.2 Structured Syntactic Feature

Normally, parsing is done on the sentence level. However, in many cases a pronoun and an antecedent candidate do not occur in the same sentence. To present their syntactic properties and relations in a single tree structure, we construct a syntax tree for an entire text, by attaching the parse trees of all its sentences to an upper node.

Having obtained the parse tree of a text, we shall consider how to select the appropriate portion of the tree as the structured feature for a given instance. As each instance is related to a pronoun and a candidate, the structured feature at least should be able to cover both of these two expressions. Generally, the more substructure of the tree is included, the more syntactic information would be provided, but at the same time the more noisy information that comes from parsing errors would likely be introduced. In our study, we examine three possible structured features that contain different substructures of the parse tree:

Min-Expansion This feature records the minimal structure covering both the pronoun and

[1] For our task, the result of $f(x)$ is used as the confidence value of the candidate to be the antecedent of the pronoun described by x.



Min-Expansion Simple-Expansion Full-Expansion

Figure 1: structured-features for the instance i{“him”, “the man”}

the candidate in the parse tree. It only includes the nodes occurring in the shortest path connecting the pronoun and the candidate, via the nearest commonly commanding node. For example, considering the sentence “*The man in the room saw him.*”, the structured feature for the instance i{“him”,“the man”} is circled with dash lines as shown in the leftmost picture of Figure 1.

Simple-Expansion *Min-Expansion* could, to some degree, describe the syntactic relationships between the candidate and pronoun. However, the syntactic properties of the candidate or the pronoun might not be captured, because the tree structure surrounding the expression is not taken into consideration. To incorporate such information, feature *Simple-Expansion* not only contains all the nodes in *Min-Expansion*, but also includes the first-level children of these nodes[2]. The middle of Figure 1 shows such a feature for i{“him”, ”the man”}. We can see that the nodes “PP” (for “in the room”) and “VB” (for “saw”) are included in the feature, which provides clues that the candidate is modified by a prepositional phrase and the pronoun is the object of a verb.

Full-Expansion This feature focusses on the whole tree structure between the candidate and pronoun. It not only includes all the nodes in *Simple-Expansion*, but also the nodes (beneath the nearest commanding parent) that cover the words between the candidate and the pronoun[3]. Such a feature keeps the most information related to the pronoun and candidate pair. The rightmost picture of Figure 1 shows the structure for feature *Full-Expansion* of i{“him”, ”the man”}. As illustrated, different from in *Simple-Expansion*, the subtree of “PP” (for “in the room”) is fully expanded and all its children nodes are included in *Full-Expansion*.

Note that to distinguish from other words, we explicitly mark up in the structured feature the pronoun and the antecedent candidate under consideration, by appending a string tag “ANA” and “CANDI” in their respective nodes (e.g.,“NN-CANDI” for “man” and “PRP-ANA” for “him” as shown in Figure 1).

4.3 Structural Kernel and Composite Kernel

To calculate the similarity between two structured features, we use the convolution tree kernel that is defined by Collins and Duffy (2002) and Moschitti (2004). Given two trees, the kernel will enumerate all their subtrees and use the number of common subtrees as the measure of the similarity between the trees. As has been proved, the convolution kernel can be efficiently computed in polynomial time.

The above tree kernel only aims for the structured feature. We also need a composite kernel to combine together the structured feature and the normal features described in Section 3.1. In our study we define the composite kernel as follows:

$$K_c(x_1, x_2) = \frac{K_n(x_1, x_2)}{|K_n(x_1, x_2)|} * \frac{K_t(x_1, x_2)}{|K_t(x_1, x2)|} \quad (2)$$

where K_t is the convolution tree kernel defined for the structured feature, and K_n is the kernel applied on the normal features. Both kernels are divided by their respective length[4] for normaliza-

[2] If the pronoun and the candidate are not in the same sentence, we will not include the nodes denoting the sentences before the candidate or after the pronoun.

[3] We will not expand the nodes denoting the sentences other than where the pronoun and the candidate occur.

[4] The length of a kernel K is defined as $|K(x_1, x_2)| = \sqrt{K(x_1, x_1) * K(x_2, x_2)}$

tion. The new composite kernel K_c, defined as the multiplier of normalized K_t and K_n, will return a value close to 1 only if both the structured features and the normal features from the two vectors have high similarity under their respective kernels.

5 Experiments and Discussions

5.1 Experimental Setup

In our study we focussed on the third-person pronominal anaphora resolution. All the experiments were done on the ACE-2 V1.0 corpus (NIST, 2003), which contain two data set, training and devtest, used for training and testing respectively. Each of these sets is further divided into three domains: newswire (NWire), newspaper (NPaper), and broadcast news (BNews).

An input raw text was preprocessed automatically by a pipeline of NLP components, including sentence boundary detection, POS-tagging, Text Chunking and Named-Entity Recognition. The texts were parsed using the maximum-entropy-based Charniak parser (Charniak, 2000), based on which the structured features were computed automatically. For learning, the SVM-Light software (Joachims, 1999) was employed with the convolution tree kernel implemented by Moschitti (2004). All classifiers were trained with default learning parameters.

The performance was evaluated based on the metric *success*, the ratio of the number of correctly resolved[5] anaphor over the number of all anaphors. For each anaphor, the NPs occurring within the current and previous two sentences were taken as the initial antecedent candidates. Those with mismatched number and gender agreements were filtered from the candidate set. Also, pronouns or NEs that disagreed in person with the anaphor were removed in advance. For training, there were 1207, 1440, and 1260 pronouns with non-empty candidate set found pronouns in the three domains respectively, while for testing, the number was 313, 399 and 271. On average, a pronoun anaphor had 6∼9 antecedent candidates ahead. Totally, we got around 10k, 13k and 8k training instances for the three domains.

5.2 Baseline Systems

Table 2 lists the performance of different systems. We first tested Hobbs' algorithm (Hobbs, 1978). Described in Section 2, the algorithm uses heuristic rules to search the parse tree for the antecedent, and will act as a good baseline to compare with the learned-based approach with the structured feature. As shown in the first line of Table 2, Hobbs' algorithm obtains 66%∼72% success rates on the three domains.

	NWire	NPaper	BNews
Hobbs (1978)	66.1	66.4	72.7
NORM	74.4	77.4	74.2
NORM_MaxEnt	72.8	77.9	75.3
NORM_C5	71.9	75.9	71.6
S_Min	76.4	81.0	76.8
S_Simple	73.2	82.7	82.3
S_Full	73.2	80.5	79.0
NORM+S_Min	77.6	82.5	**82.3**
NORM+S_Simple	79.2	82.7	**82.3**
NORM+S_Full	**81.5**	**83.2**	81.5

Table 2: Results of the syntactic structured features

The second block of Table 2 shows the baseline system (NORM) that uses only the normal features listed in Table 1. Throughout our experiments, we applied the polynomial kernel on the normal features to learn the SVM classifiers. In the table we also compared the SVM-based results with those using other learning algorithms, i.e., Maximum Entropy (Maxent) and C5 decision tree, which are more commonly used in the anaphora resolution task.

As shown in the table, the system with normal features (NORM) obtains 74%∼77% success rates for the three domains. The performance is similar to other published results like those by Keller and Lapata (2003), who adopted a similar feature set and reported around 75% success rates on the ACE data set. The comparison between different learning algorithms indicates that SVM can work as well as or even better than Maxent (NORM_MaxEnt) or C5 (NORM_C5).

5.3 Systems with Structured Features

The last two blocks of Table 2 summarize the results using the three syntactic structured features, i.e, *Min_Expansion* (S_MIN), *Simple_Expansion* (S_SIMPLE) and *Full_Expansion* (S_FULL). Between them, the third block is for the systems using the individual structured feature alone. We can see that all the three structured features per-

[5] An anaphor was deemed correctly resolved if the found antecedent is in the same coreference chain of the anaphor.

	NWire			NPaper			BNews		
Sentence Distance	0	1	2	0	1	2	0	1	2
(Number of Prons)	(192)	(102)	(19)	(237)	(147)	(15)	(175)	(82)	(14)
NORM	80.2	72.5	26.3	81.4	75.5	33.3	80.0	65.9	50.0
S_Simple	79.7	70.6	21.1	**87.3**	**81.0**	26.7	**89.7**	70.7	**57.1**
NORM+S_Simple	**85.4**	**76.5**	**31.6**	**87.3**	79.6	**40.0**	88.6	**74.4**	50.0

Table 3: The resolution results for pronouns with antecedent in different sentences apart

	NWire		NPaper		BNews	
Type	person	neuter	person	neuter	person	neuter
(Number of Prons)	(171)	(142)	(250)	(149)	(153)	(118)
NORM	81.9	65.5	80.0	73.2	74.5	73.7
S_Simple	81.9	62.7	83.2	**81.9**	82.4	**82.2**
NORM+S_Simple	**87.1**	**69.7**	**83.6**	81.2	**86.9**	76.3

Table 4: The resolution results for different types of pronouns

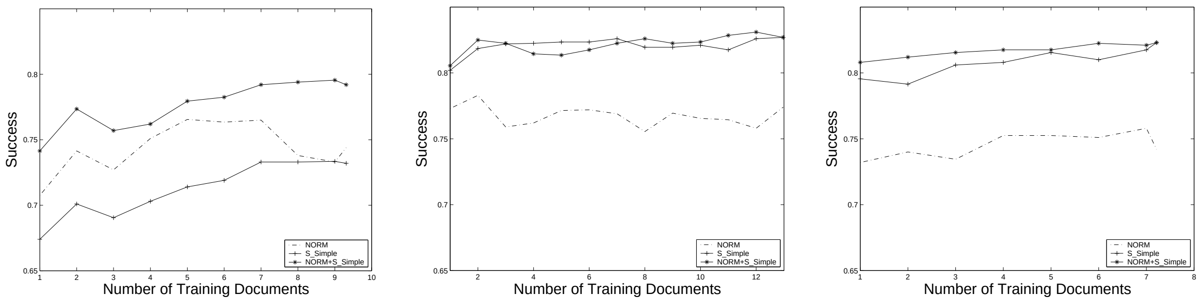
form better than the normal features for NPaper (up to 5.3% *success*) and BNews (up to 8.1% *success*), or equally well (±1 $\sim$ 2% in *success*) for NWire. When used together with the normal features, as shown in the last block, the three structured features all outperform the baselines. Especially, the combinations of NORM+S_SIMPLE and NORM+S_FULL can achieve significantly[6] better results than NORM, with the success rate increasing by (4.8%, 5.3% and 8.1%) and (7.1%, 5.8%, 7.2%) respectively. All these results prove that the structured syntactic feature is effective for pronoun resolution.

We further compare the performance of the three different structured features. As shown in Table 2, when used together with the normal features, *Full_Expansion* gives the highest success rates in NWire and NPaper, but nevertheless the lowest in BNews. This should be because feature *Full-Expansion* captures a larger portion of the parse trees, and thus can provide more syntactic information than *Min_Expansion* or *Simple_Expansion*. However, if the texts are less-formally structured as those in BNews, *Full-Expansion* would inevitably involve more noises and thus adversely affect the resolution performance. By contrast, feature *Simple_Expansion* would achieve balance between the information and the noises to be introduced: from Table 2 we can find that compared with the other two features, *Simple_Expansion* is capable of producing average results for all the three domains. And for this reason, our subsequent reports will focus on *Simple_Expansion*, unless otherwise specified.

As described, to compute the structured feature, parse trees for different sentences are connected to form a large tree for the text. It would be interesting to find how the structured feature works for pronouns whose antecedents reside in different sentences. For this purpose we tested the success rates for the pronouns with the closest antecedent occurring in the same sentence, one-sentence apart, and two-sentence apart. Table 3 compares the learning systems with/without the structured feature present. From the table, for all the systems, the success rates drop with the increase of the distances between the pronoun and the antecedent. However, in most cases, adding the structured feature would bring consistent improvement against the baselines regardless of the number of sentence distance. This observation suggests that the structured syntactic information is helpful for both intra-sentential and inter-sentential pronoun resolution.

We were also concerned about how the structured feature works for different types of pronouns. Table 4 lists the resolution results for two types of pronouns: person pronouns (i.e., “he”, “she”) and neuter-gender pronouns (i.e., “it” and “they”). As shown, with the structured feature incorporated, the system NORM+S_Simple can significantly boost the performance of the baseline (NORM), for both personal pronoun and neuter-gender pronoun resolution.

[6] $p < 0.05$ by a 2-tailed *t* test.



NWire NPaper BNews

Figure 2: Learning curves of systems with different features

5.4 Learning Curves

Figure 2 plots the learning curves for the systems with three feature sets, i.e, normal features (NORM), structured feature alone (S_Simple), and combined features (NORM+S_Simple). We trained each system with different number of instances from 1k, 2k, 3k, . . . , till the full size. Each point in the figures was the average over two trails with instances selected forwards and backwards respectively. From the figures we can find that (1) Used in combination (NORM+S_Simple), the structured feature shows superiority over NORM, achieving results consistently better than the normal features (NORM) do in all the three domains. (2) With training instances above 3k, the structured feature, used either in isolation (S_Simple) or in combination (NORM+S_Simple), leads to steady increase in the success rates and exhibit smoother learning curves than the normal features (NORM). These observations further prove the reliability of the structured feature in pronoun resolution.

5.5 Feature Analysis

In our experiment we were also interested to compare the structured feature with the normal flat features extracted from the parse tree, like feature *Subject* and *Object*. For this purpose we took out these two grammatical features from the normal feature set, and then trained the systems again. As shown in Table 5, the two grammatical-role features are important for the pronoun resolution: removing these features results in up to 5.7% (NWire) decrease in *success*. However, when the structured feature is included, the loss in *success* reduces to 1.9% and 1.1% for NWire and BNews, and a slight improvement can even be achieved for NPaper. This indicates that the structured feature can effectively provide the syntactic information important for pronoun resolution.

	NWire	NPaper	BNews
NORM	74.4	77.4	74.2
NORM - subj/obj	68.7	76.2	72.7
NORM + S_Simple	79.2	82.7	82.3
NORM + S_Simple - subj/obj	77.3	83.0	81.2
NORM + Luo05	75.7	77.9	74.9

Table 5: Comparison of the structured feature and the flat features extracted from parse trees

Feature	Parser	NWire	NPaper	BNews
S_Simple	Charniak00	73.2	82.7	82.3
	Collins99	75.1	83.2	80.4
NORM+ S_Simple	Charniak00	79.2	82.7	82.3
	Collins99	80.8	81.5	82.3

Table 6: Results using different parsers

We also tested the flat syntactic feature set proposed in Luo and Zitouni (2005)'s work. As described in Section 2, the feature set is inspired the binding theory, including those features like whether the candidate is c_commanding the pronoun, and the counts of "NP", "VP", "S" nodes in the commanding path. The last line of Table 5 shows the results by adding these features into the normal feature set. In line with the reports in (Luo and Zitouni, 2005) we do observe the performance improvement against the baseline (NORM) for all the domains. However, the increase in the success rates (up to 1.3%) is not so large as by adding the structured feature (NORM+S_Simple) instead.

5.6 Comparison with Different Parsers

As mentioned, the above reported results were based on Charniak (2000)'s parser. It would be interesting to examine the influence of different parsers on the resolution performance. For this purpose, we also tried the parser by Collins (1999)

(Mode II)[7], and the results are shown in Table 6. We can see that Charniak (2000)'s parser leads to higher success rates for NPaper and BNews, while Collins (1999)'s achieves better results for NWire. However, the performance difference by the two parsers is not significant (less than 2% *success*) for the three domains, no matter whether the structured feature is used alone or in combination.

6 Conclusion

The purpose of this paper is to explore how to make use of the structured syntactic knowledge to do pronoun resolution. Traditionally, syntactic information from parse trees is represented as a set of flat features. However, the features are usually selected and defined by heuristics and may not necessarily capture all the syntactic information provided by the parse trees. In the paper, we propose a kernel-based method to incorporate the information from parse trees. Specifically, we directly utilize the syntactic parse tree as a structured feature, and then apply kernels to such a feature, together with other normal features, to learn the decision classifier and do the resolution. Our experimental results on ACE data set show that the system with the structured feature included can achieve significant increase in the success rate by around 5%∼8%, for all the different domains. The deeper analysis on various factors like training size, feature set or parsers further proves that the structured feature incorporated with our kernel-based method is reliably effective for the pronoun resolution task.

References

C. Aone and S. W. Bennett. 1995. Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proceedings of the 33rd Annual Meeting of the Association for Compuational Linguistics*, pages 122–129.

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of North American chapter of the Association for Computational Linguistics annual meeting*, pages 132–139.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: kernels over discrete structures and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 263–270.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

J. Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:339–352.

T. Joachims. 1999. Making large-scale svm learning practical. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press.

F. Keller and M. Lapata. 2003. Using the web to obtain freqencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

C. Kennedy and B. Boguraev. 1996. Anaphora for everyone: pronominal anaphra resolution without a parser. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 113–118, Copenhagen, Denmark.

S. Lappin and H. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 20(4):525–561.

X. Luo and I. Zitouni. 2005. Milti-lingual coreference resolution with syntactic features. In *Proceedings of Human Language Techonology conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 660–667.

R. Mitkov. 1998. Robust pronoun resolution with limited knowledge. In *Proceedings of the 17th Int. Conference on Computational Linguistics*, pages 869–875.

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

V. Ng and C. Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Philadelphia.

W. Soon, H. Ng, and D. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

V. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer.

X. Yang, J. Su, G. Zhou, and C. Tan. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of 42th Annual Meeting of the Association for Computational Linguistics*, pages 127–134, Barcelona.

D. Zelenko, C. Aone, and A. Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(6):1083 – 1106.

S. Zhao and R. Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL05)*, pages 419–426.

[7] As in their pulic reports on Section 23 of WSJ TreeBank, Charniak (2000)'s parser achieves 89.6% recall and 89.5% precision with 0.88 crossing brackets (words $\leq$ 100), against Collins (1999)'s 88.1% recall and 88.3% precision with 1.06 crossing brackets.