Zhou GD, Kong F. Learning noun phrase anaphoricity in coreference resolution via label propagation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 26(1): 34–44 Jan. 2011. DOI 10.1007/s11390-011-1109-8

Learning Noun Phrase Anaphoricity in Coreference Resolution via Label Propagation

Guo-Dong Zhou (周国栋), *Senior Member*, *CCF*, *Member*, *ACM*, *IEEE*, and Fang Kong (孔　芳), *Member*, *CCF*

NLP Lab, School of Computer Science and Technology, Soochow University, Suzhou 215006, China

E-mail: {gdzhou, kongfang}@suda.edu.cn

Received December 28, 2009; revised October 28, 2010.

Abstract Knowledge of noun phrase anaphoricity might be profitably exploited in coreference resolution to bypass the resolution of non-anaphoric noun phrases. However, it is surprising to notice that recent attempts to incorporate automatically acquired anaphoricity information into coreference resolution systems have been far from expectation. This paper proposes a global learning method in determining the anaphoricity of noun phrases via a label propagation algorithm to improve learning-based coreference resolution. In order to eliminate the huge computational burden in the label propagation algorithm, we employ the weighted support vectors as the critical instances to represent all the anaphoricity-labeled NP instances in the training texts. In addition, two kinds of kernels, i.e., the feature-based RBF (Radial Basis Function) kernel and the convolution tree kernel with approximate matching, are explored to compute the anaphoricity similarity between two noun phrases. Experiments on the ACE2003 corpus demonstrate the great effectiveness of our method in anaphoricity determination of noun phrases and its application in learning-based coreference resolution.

Keywords coreference resolution, anaphoricity determination, label propagation, RBF kernel, convolution tree kernel

1 Introduction

Coreference resolution, the task of determining which noun phrases (NPs) in a text refer to the same real-world entity, has been long considered an important and difficult problem in natural language processing. Identifying the linguistic constraints on when two NPs can co-refer remains an active research area in the community. One significant constraint on coreference, the anaphoricity constraint, specifies that a non-anaphoric NP cannot be coreferent with any of its preceding NPs in a given text. Therefore, it is useful to skip over these non-anaphoric NPs rather than attempt an unnecessary search for an antecedent for each of them, only ending up with inaccurate outcomes. Although many existing machine learning approaches to coreference resolution have performed reasonably well without explicit anaphoricity determination (e.g., [1-5]), anaphoricity determination has been studied fairly extensively in the literature, given the potential usefulness of NP anaphoricity in coreference resolution. One common approach involves the design of heuristic rules to identify specific types of non-anaphoric NPs, such as pleonastic pronouns (e.g., [6-9]) and existential definite descriptions (e.g., [10]). More recently, the problem has been tackled using statistics-based (e.g., [11-12]) and learning-based (e.g., [13-17]) methods. Although there is empirical evidence (e.g. [14-15]) that coreference resolution might be further improved with proper anaphoricity information, its contribution is still far from expectation.

This paper employs a label propagation (LP) algorithm for global learning of NP anaphoricity in coreference resolution. Given the anaphoricity-labeled NP instances and the anaphoricity-unlabeled NP instances, the LP algorithm first represents the labeled and unlabeled instances as vertices in a connected graph, then propagates the label information from any vertex to nearby vertices through weighted edges and finally infers the labels of anaphoricity-unlabeled instances until a global stable stage is achieved. In this paper, the labeled data include all the NP instances in the training texts with the anaphoricity labeled, and the unlabeled data include all the NP instances in a test text with the anaphoricity unlabeled. We also explore to represent all the anaphoricity-labeled NP instances with some critical instances in order to eliminate the huge computational burden in the LP algorithm. In particular, we adopt the weighted support vectors returned by an SVM training process as the critical instances. The intuition behind is that all the anaphoricity-labeled NP instances may be well represented by some critical

Regular Paper
Supported by the National Natural Science Foundation of China under Grant Nos. 60873150, 90920004 and 61003153.

instances, i.e., the weighted support vectors as done in this paper. One major advantage of LP-based anaphoricity determination is that the anaphoricity of all the NP instances in a test text can be determined together in a global way. Compared with previous methods, the LP algorithm can effectively capture the natural clustering structure in both the labeled and unlabeled data to smooth the labeling function. In addition, two kinds of kernels, i.e., the feature-based RBF (Radial Basis Function) kernel and the convolution tree kernel with approximate matching, are employed to compute the anaphoricity similarity between two NP instances and weigh the edge between them. Experiments on the ACE2003 corpus show that our LP-based anaphoricity determination significantly outperforms the locally-optimized one, which adopts a classifier (e.g., SVM) to determine the anaphoricity of NP instances in a test text individually and significantly improves the performance of learning-based coreference resolution. It also shows that, while feature-based anaphoricity determination contributes much to pronoun resolution, its contribution on definite NP resolution can be ignored. In comparison, it shows that tree kernel-based anaphoricity resolution contributes significantly to the resolution of both pronouns and definite NPs due to the inclusion of various kinds of syntactic structured information.

The rest of this paper is organized as follows. In Section 2, we review related work in anaphoricity determination. Then, the LP algorithm is introduced in Section 3 while Section 4 describes different similarity measurements employed in the algorithm. Section 5 shows the experimental results. Finally, we conclude our work in Section 6.

2 Related Work

Given its potential usefulness in coreference resolution, anaphoricity determination has been studied fairly extensively in the literature and can be classified into three categories: heuristic rule-based (e.g., [6-10]), statistics-based (e.g., [11-12, 18]) and learning-based (e.g., [13-17]).

For the heuristic rule-based approaches, Paice and Husk[6], Lappin and Leass[7], Kennedy and Boguraev[8], Denber[9], and Cherry and Bergsma[18] looked for particular constructions using certain trigger words to identify pleonastic pronouns① while Vieira and Poesio[10] recognized non-anaphoric definite NP instances through the use of syntactic cues and case-sensitive rules and found that nearly 50% of definite NP instances are non-anaphoric.

For the statistics-based approaches, Bean and Riloff[11] developed a statistics-based method for automatically identifying existential definite NP instances which are non-anaphoric. The intuition behind is that many definite NP instances are not anaphoric since their meanings can be understood from general world knowledge. They found that existential NP instances account for 63% of all definite NP instances and 76% of them could be identified by syntactic or lexical means. Using 1600 MUC-4 terrorism news documents as the training data, they achieved 87% in precision and 78% in recall at identifying non-anaphoric definite NPs. Cherry and Bergsma[18] extended the work of Lappin and Leass[7] for large-scale anaphoricity determination by additionally detecting non-anaphoric instances of *it* using Minipar's pleonastic category *Subj*. This is done by both employing Minipar's named entity recognition to identify time expressions, such as "it was midnight...", and providing a number of other linguistic patterns to match common non-anaphoric *it* cases, such as in expressions "darn it" and "don't overdo it". Bergsma *et al.*[12] proposed a distributional method in detecting non-anaphoric pronouns by first extracting the surrounding textual context of the pronoun, then gathering the distribution of words that occurred within that context from a large corpus and finally learning to classify these distributions as representing either anaphoric or non-anaphoric pronoun instances. Experiments on the Science News Corpus of It-Bank② in identifying non-anaphoric pronoun *it* show that their distributional method achieved the performance of 81.4%, 71.0% and 75.8 in precision, recall and F1-measure, respectively, compared with the performance of 93.4%, 21.0% and 34.3 in precision, recall and F1-measure, respectively, using the rule-based approach as described in Lappin and Leass[7], and the performance of 66.4%, 49.7% and 56.9 in precision, recall and F1-measure, respectively, using the rule-based approach as described in Cherry and Bergsma[18].

Among the learning-based methods, Evans[13] applied a machine learning approach to identifying the non-anaphoricity of pronoun *it*. Ng and Cardie[14] employed various domain-independent features in identifying anaphoric NPs and showed how such information can be incorporated into a coreference resolution system. Experiments show that their method improves the performance of coreference resolution by 2.0 and 2.6 to 65.8 and 64.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively, due to much more gain in precision compared with the loss in recall. Ng[15]

①For example, Lappin and Leass[7], and Kennedy and Boguraev[8] looked for modal adjectives (e.g., "necessary") or cognitive verbs (e.g., "It is thought that $\cdots$") in a set of patterned constructions.

②www.cs.ualberta.ca/~bergsma/ItBank/.

examined the representation and optimization issues in computing and using anaphoricity information to improve learning-based coreference resolution. He used an anaphoricity classifier as a filter for coreference resolution. Evaluation on the ACE2003 corpus shows that, compared with a baseline coreference resolution system of no explicit anaphoricity determination, their method improves the performance by 2.8, 2.2 and 4.5 to 54.5, 64.0 and 60.8 in F1-measure (due to the gain in precision) on the NWIRE, NPAPER and BNEWS domains, respectively, via careful determination of an anaphoricity threshold with proper constraint-based representation and global optimization. However, he did not look into the contribution of anaphoricity determination on coreference resolution of different NP types, such as pronoun and definite NPs. Yang *et al.*[16] made use of non-anaphors to create a special class of training instances in the twin-candidate model[4] and thus equipped it with the non-anaphoricity determination capability. Experiments show that the proposed method improves the performance by 2.9 and 1.6 to 67.3 and 67.2 in F1-measure on the MUC-6 and MUC-7 corpora, respectively, due to much more gain in precision compared with the loss in recall. However, surprisingly, their experiments show that eliminating non-anaphors using an anaphoricity determination module in advance harms the performance. Denis and Balbridge[17] employed an integer linear programming (ILP) formulation for coreference resolution which models anaphoricity and coreference as a joint task, such that each local model informs the other for final assignments. Experiments on the ACE2003 corpus show that this joint anaphoricity-coreference ILP formulation outperforms the coreference-only ILP formulation by 0.7∼1.0 in F1-measure. However, their experiments assume true ACE mentions (i.e., all the ACE mentions are already known from the annotated corpus). Therefore, the actual effect of this joint anaphoricity-coreference ILP formulation on fully-automatic coreference resolution is still unclear.

3 Label Propagation

In the LP algorithm[19], the natural clustering structure in data is represented as a connected graph. Given the labeled data and unlabeled data, the LP algorithm first represents labeled and unlabeled instances as vertices in a connected graph, then propagates the label information from any vertex to nearby vertices through weighted edges and finally infers the labels of unlabeled instances until a global stable stage is achieved. Fig.1 presents the label propagation algorithm.

Here, each vertex corresponds to an instance, and

```
Assume:
    $\boldsymbol{Y}$: the $n \times r$ labeling matrix, where $y_{ij}$ represents the
        probability of vertex $x_i$ $(i = 1 \ldots n)$ with label $r_j$
        $(j = 1 \ldots r)$;
    $\boldsymbol{Y}_L$: the top $l$ rows of $\boldsymbol{Y}^0$. $\boldsymbol{Y}_L$ corresponds to the $l$ labeled
        instances;
    $\boldsymbol{Y}_U$: the bottom $u$ rows of $\boldsymbol{Y}^0$. $\boldsymbol{Y}_U$ corresponds to the $u$
        unlabeled instances;
    $\overline{\boldsymbol{T}}$: an $n \times n$ matrix, with $\bar{t}_{ij}$ is the probability jumping
        from vertex $x_i$ to vertex $x_j$;
BEGIN (the algorithm)
  Initialization:
    Set the iteration index $t = 0$;
    Let $\boldsymbol{Y}^0$ be the initial soft labels attached to each vertex;
    Let $\boldsymbol{Y}_L^0$ be consistent with the labeling in the labeled
        data, where $y_{ij}^0$ = the weight of the labeled instance
        if $x_i$ has the label $r_j$;
    Initialize $\boldsymbol{Y}_U^0$;
  REPEAT
    Propagate the labels of any vertex to nearby vertices by
    $\boldsymbol{Y}^{t+1} = \overline{\boldsymbol{T}}\boldsymbol{Y}^t$;
    Clamp the labeled data, that is, replace $\boldsymbol{Y}_L^{t+1}$ with $\boldsymbol{Y}_L^0$;
  UNTIL $\boldsymbol{Y}$ converges (e.g., $\boldsymbol{Y}_L^{t+1}$ converges to $\boldsymbol{Y}_L^0$);
  Assign each unlabeled instance with a label: for $x_i(l \prec$
  $i \leqslant n)$, find its label with $\arg\max_j y_{ij}$;
END (the algorithm)
```

Fig.1. LP algorithm.

the edge between any two instances x_i and x_j is weighted by w_{ij} to measure their similarity. In principle, larger edge weights allow labels to travel through easier. Thus the closer the instances are, the more likely they have similar labels. The algorithm first calculates w_{ij} using a kernel, then transforms it to $t_{ij} = p(j \to i) = w_{ij} \big/ \sum_{k=1}^{n} w_{kj}$, which measures the probability of propagating a label from instance x_j to instance x_i, and finally normalizes t_{ij} row by row using $\bar{t}_{ij} = t_{ij} \big/ \sum_{k=1}^{n} t_{ik}$ to maintain the class probability interpretation of the labeling matrix $\boldsymbol{Y}$.

During the label propagation process, the label distribution of the labeled data is clamped in each loop using their initial weights and acts like forces to push out labels through the unlabeled data. With this push originating from the labeled data, the label boundaries will be pushed faster along edges with larger weights and settle in gaps along those with lower weights. Ideally, we can expect that w_{ij} across different classes should be as small as possible and w_{ij} within the same class as big as possible. In this way, label propagation tends to happen within the same class. This algorithm has been shown to converge to a unique solution[19], which can be obtained without iteration in theory, and the initialization of $\boldsymbol{Y}_U^0$ (the unlabeled data) is not important since $\boldsymbol{Y}_U^0$ does not affect its estimation. However, proper initialization of $\boldsymbol{Y}_U^0$ actually helps the algorithm converge more rapidly in practice. In this paper, each row in $\boldsymbol{Y}_U^0$, i.e., the label distribution for each test instance, is initialized to the weighted similarity of the test instance with the labeled instances.

The LP algorithm may suffer from huge computational burden when the number of labeled instances is large given a certain number of unlabeled instances (e.g., the anaphoricity-unlabeled NP instances in a test text). In order to resolve this problem, we simply keep some critical instances in the label propagation process. In this paper, we employ the weighted support vectors returned by a Support Vector Machine (SVM) training process as the critical instances. The intuition behind is that the labeled instances may be well represented by some critical instances, i.e., the weighted support vectors as done in this paper. In addition, w_{ij} (the edge weight between any two instances x_i and x_j) is multiplied by the weights of x_i and x_j (for an unlabeled instance, 1.0 and for a support vector, its weight returned by SVM).

4 Kernel-Based Similarity

The key issue in the label propagation algorithm is how to compute the similarity w_{ij} between two instances x_i and x_j. This paper examines two similarity measures: the feature-based RBF (Radial Basis Function) kernel and the convolution tree kernel with approximate matching.

4.1 Feature-Based RBF Kernel

In our feature-based RBF kernel to anaphoricity determination, an instance is represented by 17 lexical, syntactic and semantic features, as shown in Table 1, which are specifically designed for distinguishing anaphoric and non-anaphoric NP instances, according to widely used features in literature. Since the local context surrounding an NP instance plays a critical role in discriminating whether an NP instance is anaphoric or not, the features in Table 1 can be classified into two categories:

(a) current NP instance (i.e., the NP instance in anaphoricity consideration) itself, e.g., types and semantic roles of current NP instance;

(b) contextual information, e.g., whether current NP instance is nested in another NP instance, the distance between current NP instance and a clause structure, indicated by coordinating words (e.g., that, this, which).

4.2 Convolution Tree Kernel with Approximate Matching

Given an NP instance in anaphoricity determination, a parse tree represents the local context surrounding

Table 1. Features in Anaphoricity Determination of NP Instances (Note: The semantic role-related features are derived from an in-house state-of-the-art semantic role labeling system.)

Feature Type	Feature	Description
Features related with current NP instance itself	IsPronoun	1 if current NP instance is a pronoun, else 0.
	IsDefiniteNP	1 if current NP instance is a define NP, else 0.
	IsDemonstrativeNP	1 if current NP instance is a demonstrative NP, else 0.
	IsArg0	1 if the semantic role of current NP instance is Arg0/agent, else 0.
	IsArg0MainVerb	1 if current NP instance has the semantic role of Arg0/agent for the main predicate of the sentence, else 0.
	IsArgs	0 if current NP instance has no semantic role, else 1.
	IsSingularNP	1 if current NP instance is a singular noun, else 0.
	IsPersonalPronoun	1 if current NP instance is a male/female personal pronoun, else 0.
Features related with the local context surrounding current NP instance	StringMatch	1 if there is a full string match between current NP instance and one of other phrases in the context, else 0.
	NameAlias	1 if current NP instance and one of other phrases in the context is a name alias or abbreviation of the other, else 0.
	Appositive	1 if current NP instance and one of other phrases in the context are in an appositive structure, else 0.
	NPNested	1 if current NP instance is nested in another NP instance, else 0.
	NPNesting	1 if current NP instance nests another NP instance, else 0.
	WordSenseAgreement	1 if current NP instance and one of other phrases in the context agree in the WordNet sense, else 0.
	IsFirstNPinSentence	1 if current NP instance is the first NP of this sentence, else 0.
	BackwardDistance	The distance between current NP instance and the nearest backward clause, indicated by coordinating words (e.g., that, which).
	ForwardDistance	The distance between the nearest forward clause, indicated by coordinating words (e.g., that, which), and current NP instance.

current NP instance in a structural way and thus contains much information in determining whether current NP instance is anaphoric or not. For example, the commonly used knowledge for anaphoricity determination, such as the grammatical role of current NP instance or whether current NP instance is nested in other NP instances, can be directly captured by a parse tree structure.

4.2.1 *Parse Tree Structure*

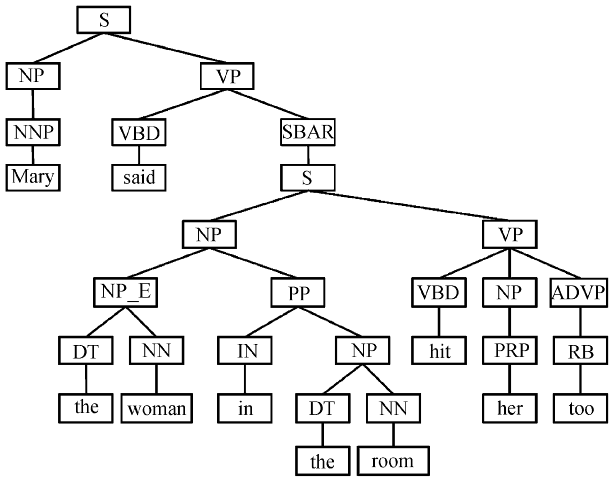
Given a parse tree and an NP instance in consideration, the problem is how to choose a proper parse tree structure to cover syntactic structured information well in the tree kernel computation. Generally, the more the parse tree structure includes, the more syntactic structured information would be provided, at the expense of more noisy/unnecessary information. In this paper, we limit the window size to 5 chunks (either NPs or non-NPs), including previous two chunks, current chunk (i.e., current NP instance) and following two chunks, and prune out the substructures outside the window. Fig.2 shows the full parse tree for the sentence "Mary said the woman in the room hit her too", using the Charniak parser[20], and the chunk sequence derived from the parse tree using the Perl script[③] written by Sabine Buchholz.

Here, we explore four parse tree structures in NP anaphoricity determination: the common tree (CT), the shortest path-enclosed tree (SPT), the minimum tree (MT) and the dynamically extended tree (DET), motivated by Yang *et al.*[21] and Zhou *et al.*[22] The following are the examples of the four parse tree structures, corresponding to the full parse tree and the chunk sequence as shown in Fig.2, with the NP chunk "(NP (DT the) (NN woman))" in anaphoricity determination.

Common Tree (CT). As shown in Fig.3(a), CT is the complete sub-tree rooted by the nearest common ancestor of the first chunk "(NP (NNP Mary))" and the last chunk "(NP (DT the) (NN room))" of the five-chunk window.

Shortest Path-Enclosed Tree (SPT). As shown in Fig.3(b), SPT is the smallest common sub-tree enclosed by the shortest path between the first chunk "(NP (NNP Mary))" and the last chunk "(NP (DT the) (NN room))" of the five-chunk window.

Minimum Tree (MT). As shown in Fig.3(c), MT only keeps the root path from the NP in anaphoricity determination to the root node of SPT.



(a)

(NP (NNP Mary)) (VP (VBD said)) (***NP_E*** (***DT the***) (***NN woman***)) (PP (IN in)) (NP (DT the) (NN room)) (VP (VBD hit)) (NP (PRP her)) (ADVP (RB too))

(b)

Fig.2. Full parse tree for the sentence "Mary said the woman in the room hit her too", using the Charniak parser, and the corresponding chunk sequence derived from it. Here, the label "E" indicates the NP instance in consideration. (a) Full parse tree. (b) Chunk sequence.

Dynamically Extended Tree (DET). The intuitions behind DET are that the information related with antecedent candidates (all the antecedent candidates compatible[④] with current NP instance in anaphoricity consideration), predicates[⑤] and right siblings plays a critical role in coreference resolution. Given an MT, this is done as follows.

1) Attaching all the compatible antecedent candidates and their corresponding paths. As shown in Fig.3(d), "Mary" is attached while "the room" is not since the former is compatible with the NP instance "the woman" and the latter is not compatible with the NP instance "the woman". In this way, possible coreference between current NP instance and the compatible antecedent candidates can be included in the parse tree structure. In some sense, this is a natural extension of the twin-candidate learning method proposed in Yang *et al.*[4,23], which explicitly models the competition between two antecedent candidates.

2) For each node in MT, attaching the path from the node to the leaf node of the corresponding predicate if it is predicate-headed. The intuition behind is that such predicate-related information is useful in identifying certain kinds of expressions with non-anaphoric NP

[③]http://ilk.kub.nl/~sabine/chunklink/.

[④]With matched number, person and gender agreements.

[⑤]For simplicity, only verbal predicates are considered in this paper. However, this can be extended to nominal predicates with automatic identification of nominal predicates.

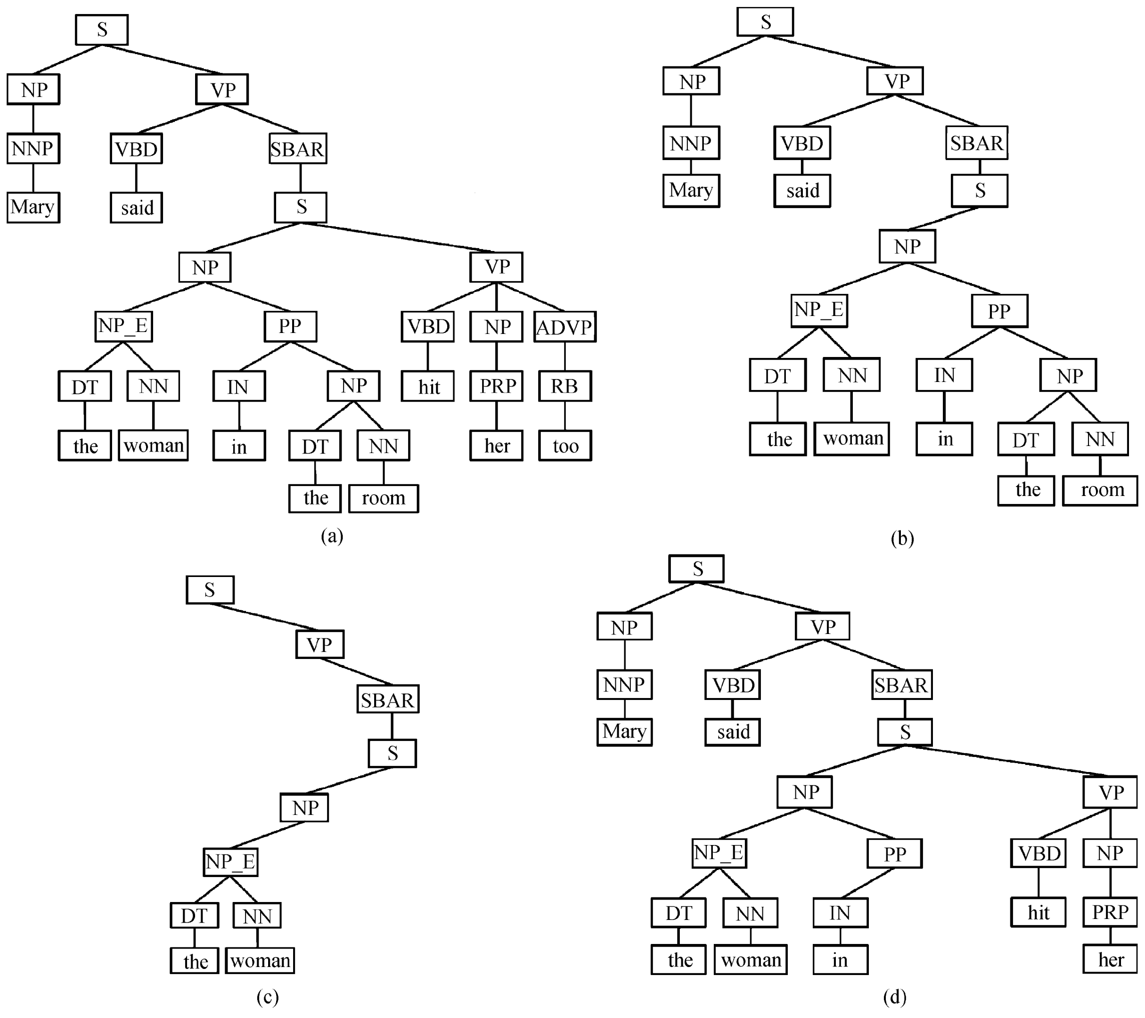

S NP VP NNP VBD SBAR Mary said S NP VP NP_E PP VBD NP ADVP DT NN IN NP hit PRP RB the woman in DT NN her too the room
(a)
S NP VP NNP VBD SBAR Mary said S NP NP_E PP DT NN IN NP the woman in DT NN the room
(b)
S VP SBAR S NP NP_E DT NN the woman
(c)
S NP VP NNP VBD SBAR Mary said S NP VP NP_E PP VBD NP DT NN IN hit PRP the woman in her
(d)

Fig.3. Examples of parse tree structures. (a) CT: Common tree. (b) SPT: Shortest path-enclosed tree. (c) MT: Minimum tree. (d) DET: Dynamically extended tree.

instances, e.g., the non-anaphoric *it* in "darn it". As shown in Fig.3(d), "said" and "hit" are attached.

3) Attaching the path to the head word of the first right sibling if the parent of current NP instance is an NP instance and current NP instance has one or more right siblings. Normally, the NP instance in anaphoricity consideration, NP_E, in the production of "NP→NP_E+PP" introduces a new entity and thus non-anaphoric.

4) Pruning those nodes (except possessive nodes) with the single in-arc and the single out-arc and with the syntactic phrase type same as its child node.

4.2.2 *Convolution Tree Kernel with Approximate Matching*

Given two parse trees, we now study how to measure the similarity between them, using a convolution kernel. A convolution kernel[24] aims to capture structural information in terms of substructures. As a specialized convolution kernel, the convolution tree kernel[25] $K_{\rm CTK}(T_1, T_2)$ ("CTK" for convolution tree kernel) counts the number of common sub-trees as the syntactic structure similarity between two parse trees T_1 and T_2:

$$K_{\rm CTK}(T_1, T_2) = \sum_{n_1 \in N_1, n_2 \in N_2} \Delta(n_1, n_2) \qquad (1)$$

where N_j is the set of nodes in tree T_j, and $\varDelta(n_1, n_2)$ evaluates the common sub-trees rooted at n_1 and n_2[⑥] and is computed recursively as follows.

1) If the context-free productions (i.e., context-free grammar rules) at n_1 and n_2 do not match, then $\varDelta(n_1, n_2) = 0$; otherwise go to 2).

[⑥]That is, each node n encodes the identity of a sub-tree rooted at n and, if there are two nodes in the tree with the same label, the summation will go over both of them.

2) If both n_1 and n_2 are POS tags, then $\Delta(n_1, n_2) = \lambda$; otherwise (i.e., if both n_1 and n_2 are constituent tags, such as NP and S) go to 3).

3) Calculate $\Delta(n_1, n_2)$ recursively as:

$$\Delta(n_1, n_2) = \lambda \prod_{k=1}^{\#ch(n_1)} (1 + \Delta(ch(n_1, k), ch(n_2, k))) \quad (2)$$

where $\#ch(n)$ is the number of children of node n, $ch(n, k)$ is the k-th child of node n and λ $(0 < \lambda < 1)$ is the decay factor in order to make the kernel value less variable with respect to different sub-tree sizes[25]. Note that, in the above standard CTK, n_1 and n_2 in (2) should have the same number of children, i.e., $\#ch(n_1) = \#ch(n_2)$.

One major problem with the above standard CTK is that it only allows exact matching (i.e., full matching) of sub-trees and thus may fail to effectively capture the commonality between similar sub-trees. In order to resolve this problem, this paper explores a new CTK which allows approximate matching. This is done by modifying (2) in computing $\Delta(n_1^i[1], n_1^i[2])$ to allow insertion/deletion/substitution of tree nodes in the sub-trees:

$$\Delta(n_1^i[1], n_1^i[2]) = \lambda \cdot \lambda_1^{\#InsDels} \cdot \lambda_2^{\#Subs} \cdot \prod_{k=1}^{\#ch(x_1^i[1])} (1 + \Delta(ch(x_1^i[1], k), ch(x_1^i[2], k))) \quad (2')$$

where

1) $\#InsDels$ is the number of insertions/deletions of optional tree nodes and $\#Subs$ is the number of substitutions between similar tree nodes while λ_1 and λ_2 $(1 < \lambda_1, \lambda_2 < 1)$ are their weights.

2) $x_1^i[j]$ is the best matched child sequence between $n_1^i[1]$ and $n_1^i[2]$ of length $\#ch(x_1^i[j])$ with approximate matching. Here, $ch(x_1^i[j], k)$ is the k-th child of $x_1^i[j]$.

Obviously, the number of insertions/deletions/substitutions can be easily determined by computing the least edit distance between two productions by maximizing $\lambda_1^{\#InsDels} \cdot \lambda_2^{\#Subs}$ using dynamic programming. This is different from the complicated dynamic programming algorithm as proposed in Zhang *et al*[26], which is motivated by the partial matching algorithm as proposed in Moschitti[27]. In particular, all the children of a node are optional tree nodes except its head child since head children normally contain critical information and should be similar for two productions to be matched (either exactly or approximately). In addition, substitutions only apply between similar tree nodes, e.g., the part-of-speeches (POSs) for singular nouns and plural nouns. This means insertions and deletions only apply to non-head children while substitutions only happen in the same group of similarly functioned tree nodes. For approximate matching, we only consider five groups of similar tree nodes: 1) adjective group: ADJP, JJ, JJR, JJS; 2) adverb group: ADVP, RB, RBR, RBS; 3) noun group: NP, NN, NNS, NNP, NNPS, NAC, NX; 4) active verb group: VP, VB, VBD, VBG, VBP, VBZ, VBN (in the active voice only); 5) passive verb group: VPP, VBN (in the passive voice only). Since voice information is very indicative in distinguishing subjects and objects, VBNs and VPs in the passive voice are determined heuristically and differentiated from other verbs.

5 Experimentation

We have systematically evaluated the label propagation algorithm on global learning of NP anaphoricity determination on the ACE2003 corpus, and its application in coreference resolution.

5.1 Experimental Setting

The ACE2003 corpus contains three domains: newswire (NWIRE), newspaper (NPAPER), and broadcast news (BNEWS). For each domain, there exist two datasets, training and devtest, which are used for training and testing respectively.

As a baseline coreference resolution system, a raw test text is first preprocessed automatically by a pipeline of NLP components, including sentence boundary detection, POS tagging, named entity recognition and phrase chunking, and then a training or test instance is formed by an anaphor and one of its antecedent candidates, similar to Soon *et al.*'s method[1]. Among them, named entity recognition, part-of-speech tagging and noun phrase chunking apply the same Hidden Markov Model (HMM)-based engine with error-driven learning capability[28-29]. During training, for each anaphor encountered, a positive instance is created by pairing the anaphor and its closest antecedent while a set of negative instances is formed by pairing the anaphor with each of the non-coreferential candidates. Based on the training instances, a binary classifier is generated using a particular learning algorithm. In this paper, we use SVMLight developed by Joachims[30] with the default setting. During resolution, an anaphor is first paired in turn with each preceding antecedent candidate to form a test instance, which is presented to a classifier. The classifier then returns a confidence value indicating the likelihood that the candidate is the antecedent. Finally, the candidate with the highest confidence value is selected as the antecedent. In particular, the NP instances with mismatched number, person and gender agreements are filtered out. As a result, an anaphor has ~ 7 antecedent candidates. In addition, the test corpus is resolved in document-level, i.e., one document by one document.

For anaphoricity determination, we report the performance in Acc$^+$ and Acc$^-$, which measure the accuracies of identifying anaphoric NP instances and non-anaphoric NP instances, respectively. Obviously, higher Acc$^+$ means that more anaphoric NP instances would be identified correctly, while higher Acc$^-$ means that more non-anaphoric NP instances would be filtered out. For coreference resolution, we report the performance in terms of recall, precision, and F1-measure using the commonly-used model theoretic MUC scoring program[31]. For separate scoring of different NP types, a recognized reference is considered correct if the recognized antecedent is in the coreferential chain of the anaphor. To see whether an improvement is significant, we conduct significance testing using paired t-test. In this paper, "$\ggg$", "$\gg$" and "$>$" denote p-values of an improvement smaller than 0.01, in-between (0.01, 0.05] and bigger than 0.05, which means significantly better, moderately better and slightly better, respectively.

5.2 Experimental Results

Table 2 shows the performance of LP-based anaphoricity determination using the feature-based RBF kernel. It shows that our method achieves the accuracies of 74.8%/84.4%, 76.2%/81.3% and 71.8%/81.7% on identifying anaphoric/ non-anaphoric NP instances in the NWIRE, NPAPER and BNEWS domains, respectively, when all the anaphoricity-labeled NP instances are used in the LP algorithm. This suggests that our approach can effectively filter out about 82% of non-anaphoric NP instances. However, it can only keep about 74% of anaphoric NP instances. Table 2 also shows the performance on different NP types. Considering the effectiveness of anaphoricity determination on indefinite NP instances (due to that most of anaphoric indefinite NP instances are in an appositive structure and thus can be easily captured by the IsAppositive feature) and that most of errors in anaphoricity determination on proper nouns are caused by the named entity recognition module in the preprocessing, it indicates the difficulty of anaphoricity determination on pronoun and definite NP instances. As a comparison, Table 2 also shows the performance of locally-optimized anaphoricity determination using a classifier (SVM with the feature-based RBF kernel, as adopted in this paper), which determines the anaphoricity of NP instances in a test text individually. Our significance testing shows that the LP-based method systematically outperforms ($\ggg$) the SVM-based method. This suggests the effectiveness of the LP algorithm in global modeling of the natural clustering structure in anaphoricity determination. Finally, Table 2 shows the performance when only the weighted support vectors are employed in the LP algorithm. Here, SVMLight (with the default setting) is employed to extract the weighted support vectors as the critical instances to represent all the anaphoricity-labeled NP instances. Our evaluation shows that the weighted support vectors occupy 18$\sim$25% of all the anaphoricity-labeled NP instances and the computational burden can be largely reduced by more than 90%. Table 2 also shows that the LP algorithm with the weighted support vectors even outperforms (slightly but consistently) the one with all the anaphoricity-labeled NP instances. This indicates that the weighted support vectors can well represent all the anaphoricity-labeled NP instances and label propagation may be better done via the weighted support vectors, which determine the decision boundary between anaphoric and non-anaphoric NP instances.

Table 3 shows the performance of LP-based anaphoricity determination using the standard convolution tree kernel on different parse tree structures. It shows that while MT performed worst due to its simple structure, DET outperforms MT ($\ggg$), SPT ($\ggg$) and CT ($\ggg$) on all the three domains due to fine inclusion of necessary structural information, although inclusion of more information in both CT and SPT also improves the performance. As shown in Table 4, all the three kinds of structural information related with antecedent candidates, predicates and right siblings in DET contribute significantly ($\ggg$). For LP with the weighted support vectors, our evaluation shows that the weighted

Table 2. The Performance of NP Anaphoricity Determination Using the Feature-Based RBF Kernel (Figures outside the parentheses: LP with all the labeled training instances; figures inside the parentheses: LP with the weighted support vectors.)

Anaphor Type	NWIRE		NPAPER		BNEWS	
	Acc$^+$ (%)	Acc$^-$ (%)	Acc$^+$ (%)	Acc$^-$ (%)	Acc$^+$ (%)	Acc$^-$ (%)
Pronoun	88.7 (89.5)	56.2 (56.4)	90.2 (90.7)	58.6 (58.9)	87.4 (87.7)	57.8 (58.1)
ProperNoun	72.5 (73.1)	85.2 (85.4)	74.6 (74.9)	80.5 (80.7)	70.6 (71.1)	78.8 (79.0)
DefiniteNP	66.6 (66.9)	83.1 (83.4)	72.1 (72.4)	77.5 (77.8)	65.3 (65.7)	81.5 (81.8)
InDefiniteNP	95.4 (95.8)	93.7 (93.9)	90.5 (91.0)	95.8 (96.2)	87.2 (87.4)	97.3 (97.4)
Overall	74.8 (75.3)	84.4 (84.6)	76.2 (76.6)	81.3 (81.6)	71.8 (72.2)	81.7 (81.9)
Overall (SVM)	*71.3*	*80.2*	*73.5*	*79.1*	*68.4*	*78.6*

Table 3. Performance of NP Anaphoricity Determination Using the Standard Convolution Tree Kernel on Different Parse Tree Structures (Figures outside the parentheses: LP with all the labeled training instances; figures inside the parentheses: LP with the weighted support vectors.)

Parse Tree Structure	Scheme	NWIRE (%)	NPAPER (%)	BNEWS (%)
CT	Acc^+	72.6 (72.9)	74.3 (74.8)	74.2 (74.6)
	Acc^-	82.1 (82.7)	80.2 (80.9)	72.3 (72.5)
SPT	Acc^+	72.4 (72.6)	74.1 (74.5)	73.8 (74.1)
	Acc^-	80.8 (81.3)	79.5 (80.1)	72.5 (72.8)
MT	Acc^+	71.4 (71.8)	70.5 (70.8)	66.9 (67.1)
	Acc^-	77.2 (77.6)	75.3 (75.6)	78.2 (78.4)
DET	Acc^+	79.2 (79.6)	81.2 (81.5)	76.5 (76.8)
	Acc^-	87.8 (88.1)	84.5 (84.7)	85.3 (85.4)
DET (SVM)	*Acc^+*	*76.5*	*78.9*	*74.3*
	Acc^-	*82.3*	*81.6*	*83.2*

Table 4. Contribution of Structural Information in DET Using the Standard Convolution Tree Kernel (Figures outside the parentheses: LP with all the labeled training instances; figures inside the parentheses: LP with the weighted support vectors.)

Performance Change		NWIRE (%)	NPAPER (%)	BNEWS (%)
-Antecedent candidates	Acc^+	−4.0 (−4.2)	−3.8 (−4.3)	−4.3 (−4.9)
	Acc^-	−5.2 (−5.5)	−5.3 (−5.6)	−4.5 (−4.7)
-Predicate	Acc^+	−5.2 (−5.6)	−4.8 (−5.4)	−5.6 (−5.9)
	Acc^-	−4.3 (−4.6)	−3.5 (−4.0)	−4.9 (−5.4)
-First right sibling	Acc^+	−3.6 (−4.0)	−4.1 (−4.6)	−3.1 (−3.8)
	Acc^-	−4.8 (−5.2)	−5.2 (−5.5)	−4.4 (−4.8)

support vectors only occupy 29∼36% of all the anaphoricity-labeled NP instances and the computational burden can be largely reduced by more than 80% using the standard convolution tree kernel in various parse tree structures. Table 3 also shows that the LP algorithm with the weighted support vectors outperforms (slightly but consistently) the one with all the anaphoricity-labeled NP instances using the standard convolution tree kernel in all considered parse tree structures. Finally, Table 3 verifies that LP-based anaphoricity determination outperforms ($\ggg$) the SVM-based one, using the standard convolution tree kernel.

Table 5 shows the performance of LP-based anaphoricity determination using the convolution tree kernel with approximate matching on different parse tree structures. Similar to the standard convolution tree kernel, it shows that the LP algorithm with the weighted support vectors outperforms (slightly but consistently) the one with all the anaphoricity-labeled NP instances in all considered parse tree structures, using the convolution tree kernel with approximate matching. Compared with the standard convolution tree kernel as shown in Table 3, the convolution tree kernel with approximate matching significantly ($\ggg$) outperforms the standard convolution tree kernel in all considered parse tree structures. This suggests that approximate matching in terms of tree node insertion/deletion/substitution is very useful in capturing the similarity between two sub-trees. However, such improvements are at the cost of about 32∼41% more running time, largely due to dynamic programming in computing the edit distance between two productions to achieve approximate matching. In this paper, λ_1 (for insertion and deletion) and λ_2 (for substitution) in $(2')$ are fine-tuned to 0.6 and 0.4 respectively.

Table 5. Performance of NP Anaphoricity Determination Using the Convolution Tree Kernel with Approximate Matching on Different Parse Tree Structures (Figures outside the parentheses: LP with all the labeled training instances; figures inside the parentheses: LP with the weighted support vectors.)

Parse Tree Structure	Scheme	NWIRE (%)	NPAPER (%)	BNEWS (%)
CT	Acc^+	73.8 (74.2)	75.7 (76.2)	75.8 (76.1)
	Acc^-	83.6 (84.3)	81.5 (82.3)	73.7 (74.1)
SPT	Acc^+	73.5 (73.8)	75.2 (75.7)	75.1 (75.5)
	Acc^-	82.3 (82.6)	80.9 (81.6)	73.9 (74.2)
MT	Acc^+	72.3 (72.8)	71.6 (71.7)	67.8 (67.9)
	Acc^-	78.3 (78.7)	76.3 (76.8)	79.1 (79.3)
DET	Acc^+	80.7 (81.2)	82.6 (82.9)	77.9 (78.3)
	Acc^-	89.2 (89.6)	85.9 (86.3)	86.9 (87.4)
DET (SVM)	*Acc^+*	*77.8*	*80.4*	*75.9*
	Acc^-	*83.7*	*83.1*	*85.8*

In addition, Table 6 shows the detailed performance of LP-based anaphoricity determination on different anaphor types using the convolution tree kernel with approximate matching on DET. Compared with the feature-based RBF kernel as shown in Table 2, it shows that the convolution tree kernel significantly outperforms ($\ggg$) the feature-based RBF kernel in all the three domains, with much contribution due to performance improvement on both pronouns and definite NPs, although the tree kernel performs worse than the feature-based RBF kernel on proper nouns and indefinite NPs due to the effectiveness of anaphoricity determination on proper nouns and indefinite NPs using the simple IsNameAlias and IsAppositive features respectively.

Finally, Table 7 shows the effectiveness of LP-based anaphoricity determination on coreference resolution by including it as a preprocessing step to a baseline coreference resolution system without explicit anaphoricity

Table 6. Performance of NP Anaphoricity Determination Using the Convolution Tree Kernel with Approximate Matching on DET (Figures outside the parentheses: LP with all the labeled training instances; figures inside the parentheses: LP with the weighted support vectors.)

NWIRE		NPAPER		BNEWS	
Acc$^+$ (%)	Acc$^-$ (%)	Acc$^+$ (%)	Acc$^-$ (%)	Acc$^+$ (%)	Acc$^-$ (%)
91.4 (91.8)	77.3 (77.9)	91.2 (91.4)	80.7 (81.2)	89.8 (90.3)	81.0 (81.4)
71.6 (71.8)	83.8 (83.9)	73.1 (73.4)	78.4 (78.7)	68.6 (68.8)	77.5 (77.6)
76.4 (77.0)	90.5 (90.6)	79.4 (80.1)	87.2 (87.4)	77.1 (77.6)	89.8 (89.9)
91.3 (92.5)	91.2 (91.3)	87.1 (89.4)	93.1 (93.4)	85.8 (87.2)	93.8 (94.4)
80.5 (80.9)	89.2 (89.4)	82.5 (82.8)	86.0 (86.3)	77.9 (78.3)	86.9 (87.1)

Table 7. Employment of Anaphoricity Determination in Coreference Resolution

		NWIRE			NPAPER			BNEWS		
System	NP Types	R (%)	P (%)	F	R (%)	P (%)	F	R (%)	P (%)	F
BaseLine (No anaphoricity)	Pronoun	66.5	61.6	64.0	70.1	64.2	67.0	61.7	63.2	62.4
	DefiniteNP	26.9	80.3	40.2	34.5	62.4	44.4	30.5	71.4	42.9
	Overall	53.1	67.4	59.4	57.7	67.0	62.1	48.0	65.9	55.5
+Anaphoricity determination: RBF kernel and LP with the weighted support vectors	Pronoun	64.3	68.5	66.3	67.7	72.9	70.2	60.1	75.1	66.8
	DefiniteNP	26.9	80.7	40.4	34.7	62.5	44.6	30.8	71.9	43.1
	Overall	51.0	75.8	61.0	55.1	75.6	63.7	46.3	77.6	58.0
+Anaphoricity determination: CTK with approximate matching and LP with the weighted support vectors	Pronoun	64.9	71.9	68.2	69.2	75.7	72.3	62.5	77.6	68.5
	DefiniteNP	30.4	83.5	44.6	38.6	66.3	48.8	34.0	73.9	46.6
	Overall	54.1	79.1	62.8	57.3	77.4	65.7	48.9	81.1	61.0

determination, which employs the same set of features, as adopted in the single-candidate model of Yang *et al.*[4], using an SVM-based classifier and the feature-based RBF kernel. Here, the LP algorithm with the weighted support vectors is employed while for kernel methods, the convolution tree kernel with approximate matching is done on DET. Besides the overall performance, Table 7 also shows the performance for two major NP types, pronoun and definite NP. It shows that anaphoricity determination with the feature-based RBF kernel much improves ($\ggg$) the performance of coreference resolution with most of the contribution due to pronoun resolution while its contribution on definite NPs can be ignored. It indicates the usefulness of anaphoricity determination in filtering out non-anaphoric pronouns and the difficulty in identifying anaphoric definite NPs, using the feature-based RBF kernel. It also shows that tree kernel-based anaphoricity determination can not only improve ($\ggg$) the performance on pronoun resolution but also improve ($\ggg$) the performance on definite NP resolution due to the much better performance of tree kernel-based anaphoricity determination on definite NPs. This suggests the necessity of exploring structural information in identifying anaphoric definite NPs.

6 Conclusion

This paper systematically studies a global learning method of identifying the anaphoricity of noun phrases via a label propagation algorithm and the application of an explicit anaphoricity determination module in improving learning-based coreference resolution. In particular, two kinds of kernels, i.e., the feature-based RBF kernel and the convolution tree kernel with approximate matching, are employed to compute the anaphoricity similarity between two NPs. Besides, the weighted support vectors are employed to represent all the anaphoricity-labeled instances to eliminate the huge computational burden in the LP algorithm. Evaluation on the ACE2003 corpus indicates that LP-based anaphoricity determination using both the kernels much improves the performance of coreference resolution. It also shows the usefulness of various structural information (related with antecedent candidates, predicates and right siblings) in tree kernel-based anaphoricity determination and in coreference resolution of both pronouns and definite NPs.

To our knowledge, this is the first systematic and successful exploration of both feature-based and tree kernel methods in anaphoricity determination and the application of an explicit anaphoricity determination module in learning coreference resolution.

References

[1] Soon W M, Ng H T, Lim D. A machine learning approach to coreference resolution of noun phrase. *Computational Linguistics*, 2001, 27(4): 521-544.

[2] Ng V, Cardie C. Improving machine learning approaches to coreference resolution. In *Proc. ACL 2002*, Philadelphia, USA, Jul. 6-12, 2002, pp.104-111.

[3] Strube M, Muller C. A machine learning approach to pronoun resolution in spoken dialogue. In *Proc. ACL 2003*, Sapporo, Japan, Jul. 7-12, 2003, pp.68-175.

[4] Yang X F, Zhou G D, Su J, Chew C L. Coreference resolution using competition learning approach. In *Proc. ACL 2003*, Sapporo, Japan, Jul. 7-12, 2003, pp.177-184.

[5] Yang X F, Su J, Lang J, Tan C L, Liu T, Li S. An entity-mention model for coreference resolution with inductive logic programming. In *Proc. ACL 2008*, Columbus, USA, Jun. 15-20, 2008, pp.843-851.

[6] Paice C D, Husk G D. Towards the automatic recognition of anaphoric features in English text: The impersonal pronoun *it*. *Computer Speech and Language*, 1987, 2(2): 109-132.

[7] Lappin S, Leass H J. An algorithm for pronominal anaphora resolution. *Computational Linguistics*, 1994, 20(4): 535-561.

[8] Kennedy C, Boguraev B. Anaphora for everyone: Pronominal anaphora resolution without a parser. In *Proc. COLING 1996*, Copenhagen, Denmark, Aug. 5-9, 1996, pp.113-118.

[9] Denber M. Automatic resolution of anaphora in English. Technical Report, Eastman Kodak Co. 1998.

[10] Vieira R, Poesio M. An empirically based system for processing definite descriptions. *Computational Linguistics*, 2000, 27(4): 539-592.

[11] Bean D, Riloff E. Corpus-based identification of non-anaphoric noun phrases. In *Proc. ACL 1999*, University of Maryland, College Park, USA, Jun. 20-26, 1999, pp.373-380.

[12] Bergsma S, Lin D K, Goebel R. Distributional identification of non-referential pronouns. In *Proc. ACL 2008*, Columbus, USA, Jun. 15-20, 2008, pp.10-18.

[13] Evans R. Applying machine learning toward an automatic classification of *it*. *Literary and Linguistic Computing*, 2001, 16(1): 45-57.

[14] Ng V, Cardie C. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proc. COLING 2002*, Taipei, China, Aug. 24-Sept. 1, 2002, pp.730-736.

[15] Ng V. Learning noun phrase anaphoricity to improve conference resolution: Issues in representation and optimization. In *Proc. ACL 2004*, Barcelona, Spain, Jul. 21-26, 2004, pp.151-158.

[16] Yang X F, Su J, Tan C L. A twin-candidate model of coreference resolution with non-anaphor identification capability. In *Proc. IJCNLP 2005*, Jeju Island, Korea, Oct. 11-13, 2005, pp.719-730.

[17] Denis P, Baldridge J. Joint determination of anaphoricity and coreference using integer programming. In *Proc. NAACL-HLT 2007*, Rochester, USA, Apr. 22-27, 2007, pp.236-243.

[18] Cherry C, Bergsma S. An expectation maximization approach to pronoun resolution. In *Proc. CoNLL 2005*, Ann Arbor, USA, Jun. 29-30, 2005, pp.88-95.

[19] Zhu X, Ghahramani Z. Learning from labeled and unlabeled data with label propagation. CMU CALD Technical Report, CMU-CALD-02-107, 2002.

[20] Charniak E. Immediate-head parsing for language models. In *Proc. ACL 2001*, Toulouse, France, Jul. 9-11, 2001, pp.129-137.

[21] Yang X F, Su J, Tan C L. Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 2006*, Sydney, Australia, Jul. 17-21, 2006, pp.41-48.

[22] Zhou G D, Kong F, Zhu Q M. Context-sensitive convolution tree kernel for pronoun resolution. In *Proc. IJCNLP 2008*, Hyderabad, India, Jan. 7-12, 2008, pp.25-31.

[23] Yang X F, Su J, Tan C L. A twin-candidate model for learning-based anaphora resolution. *Computational Linguistics*, 2008, 34(3): 327-356.

[24] Haussler D. Convolution kernels on discrete structures. Technical Report UCS-CRL-99-10, University of California, Santa Cruz, USA, 1999.

[25] Collins M, Duffy N. Convolution kernels for natural language. In *Proc. NIPS 2001*, Vancouver, Canada, Dec. 3-8, 2001, pp.625-632.

[26] Zhang M, Che W X, Aw A T, Tan C L, Zhou G D, Liu T, Li S. A grammar-driven convolution tree kernel for semantic role classification. In *Proc. ACL 2007*, Prague, Czech Republic, Jun. 23-30, 2007, pp.200-207.

[27] Moschitti A. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proc. ECML 2006*, Berlin, Germany, Sept. 18-22, 2006, pp.318-329.

[28] Zhou G D, Su J. Error-driven HMM-based chunk tagger with context-dependent lexicon. In *Proc. EMNLP-VLC 2000*, Hong Kong, China, Oct. 7-8, 2000, pp.71-79.

[29] Zhou G D, Su J. Named entity recognition using an HMM-based chunk tagger. In *Proc. ACL 2002*, Philadelphia, USA, Jul. 6-12, 2002, pp.473-480.

[30] Joachims T. Text categorization with support vector machine: Learning with many relevant features. In *Proc. ECML 1998*, Chemnitz, Germany, Apr. 21-23, 1998, pp.137-142.

[31] Vilain M, Burger J, Aberdeen J, Connolly D, Hirschman L. A model theoretic coreference scoring scheme. In *Proc. MUC-6*, Columbia, USA, Nov. 1-3, 1995, pp.45-52.

Guo-Dong Zhou received the Ph.D. degree from the National University of Singapore in 1999. He joined the Institute for Infocomm Research, Singapore, in 1999, and had been an associate scientist, scientist and associate lead scientist at the institute until August 2006. Currently, he is a professor at the School of Computer Science and Technology, Soochow University, Suzhou, China. His research interests include natural language processing, information extraction and machine learning. He is a senior member of CCF and has been the member of ACM and IEEE since 1999.

Fang Kong received her Ph.D. degree from Soochow University, Suzhou, China, in 2009. Currently, she is an associate professor at the university. Her research interests include natural language processing and information extraction. She is a member of CCF.