Coreference Systems based on Kernels Methods

Yannick Versley
SFB 441
University of Tübingen
versley@sfs.uni-tuebingen.de

Alessandro Moschitti
DISI
University of Trento
moschitti@disi.unitn.it

Massimo Poesio
DISI
University of Trento
massimo.poesio@unitn.it

Xiaofeng Yang
Data Mining Department
Institute for Infocomm Research
xiaofengy@i2r.a-star.edu.sg

Abstract

Various types of structural information - e.g., about the type of constructions in which binding constraints apply, or about the structure of names - play a central role in coreference resolution, often in combination with lexical information (as in expletive detection). Kernel functions appear to be a promising candidate to capture structure-sensitive similarities and complex feature combinations, but care is required to ensure they are exploited in the best possible fashion. In this paper we propose kernel functions for three subtasks of coreference resolution - binding constraint detection, expletive identification, and aliasing - together with an architecture to integrate them within the standard framework for coreference resolution.

1 Introduction

Information about coreference relations–i.e., which noun phrases are mentions of the same entity–has been shown to be beneficial in a great number of NLP tasks, including information extraction (McCarthy and Lehnert 1995), text planning (Barzilay and Lapata 2005) and summarization (Steinberger et al. 2007). However, the performance of coreference resolvers on unrestricted text is still quite low. One reason for this is that coreference resolution requires a great deal of information, ranging from string matching to syntactic constraints to semantic knowledge to discourse salience information to full common sense reasoning (Sidner 1979; Hobbs 1978, 1979; Grosz et al. 1995; Vieira and Poesio 2000; Mitkov 2002). Much of this information won't be available to robust coreference resolvers until better methods are found to represent and encode common sense knowledge; but part of the problem is also the need for better methods to encode information that is in part structural, in part lexical. Enforcing binding constraints –e.g., ruling out *Peter* as antecedent of *him* in (1a) requires recognizing that the anaphor occurs in a particular type of construction (Chomsky 1981; Lappin and Leass 1994; Yang et al. 2006) whose exact definition however has not yet been agreed upon by linguists (indeed, it may only be definable in a graded sense (Sturt 2003; Yang et al. 2006)), witness examples like (1b). Parallelism effects are a good example of structural information inducing preferences rather than constraints. Recognizing that *It* in examples such as (1c,d) are expletives requires a combination of structural information and lexical information (Lappin and Leass 1994; Evans 2001). But some sort of structure also underlies our interpretation of other types of coreference: e.g., knowledge about the structure of names certainly plays a role in recognizing that *BJ Habibie* is a possible antecedent for *Mr. Habibie*.

(1) a. John thinks that Peter hates *him*.

    b. John hopes that Jane is speaking only to *himself*.

    c. *It*'s lonely here.

    d. *It* had been raining all day.

The need to capture such information suggests a role for kernel methods (Vapnik 1995) in coreference resolution. Kernel functions make it possible to capture the similarity between structures

without explicitly enumerating all the substructures, and have therefore been shown to be a viable approach to feature engineering for natural language processing for any task in which structural information plays a role, e.g. (Collins and Duffy 2002; Zelenko et al. 2003; Giuglea and Moschitti 2006; Zanzotto and Moschitti 2006; Moschitti et al. 2007). Indeed, they have already been used in NLP to encode the type of structural information that plays a role in binding constraints (Yang et al. 2006); however, the methods used in this previous work do not make it possible to exploit the full power of kernel functions. In this work, we extend the use of kernel functions for coreference by designing and testing kernels for three subtasks of the coreference task:

- Binding constraints
- Expletive detection
- Aliasing

and developing distinct classifiers for each of these tasks. We show that our developed kernels produce high accuracy for both distinct classifiers for these subtasks as well as for the complete coreference system.

In the remainder: Section 2, briefly describes the basic kernel functions that we used; Section 3 illustrates our new kernels for expletive, binding and name alias detection along with a coreference context kernel; Section 4 reports the experiments on individual classifiers on expletives, binding and names whereas Section 5 shows the results on the complete coreference task; Finally, Section 6 derives the conclusions.

2 Kernel for Structured Data

We used three kernel functions in this work: the String Kernel (SK) proposed in Shawe-Taylor and Cristianini (2004) to evaluate the number of subsequences between two sequences, the Syntactic Tree Kernel (STK; see Collins and Duffy 2002) which computes the number of syntactic tree fragments and the Partial Tree Kernel (PTK; see Moschitti 2006) which provides a more general representation of trees in terms of tree fragments. We discuss each in turn.

2.1 String Kernels (SK)

The string kernels that we consider count the number of substrings shared by two sequences containing gaps, i.e. some of the characters of the original
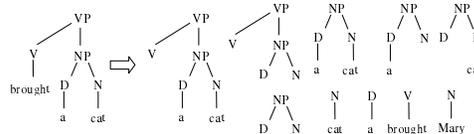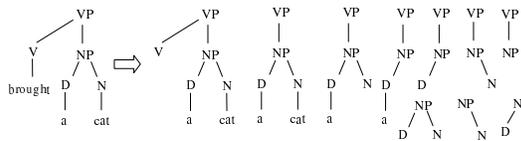

Figure 1: A tree with some of its STFs .

Figure 2: A tree with some of its PTFs.

string are skipped. Gaps penalize the weight associated with the matched substrings. More in detail, (a) longer subsequences receive lower weights. (b) Valid substrings are sequences of the original string with some characters omitted, i.e. gaps. (c) Gaps are accounted by weighting functions and (d) symbols of a string can also be whole words, i.e. the word sequence kernel Cancedda et al. (2003).

2.2 Tree Kernels

The main idea underlying tree kernels is to compute the number of common tree fragments between two trees without explicitly considering the whole fragment space. The type of fragments characterize different kernel functions. We consider syntactic tree fragments (STFs) and partial tree fragments (PTFs)

2.2.1 Syntactic Tree Kernels (STK)

An STF is a connected subset of the nodes and edges of the original tree, with the constraint that any node must have all or none of its children. This is equivalent to stating that the production rules contained in the STF cannot be partial. For example, Figure 1 shows a tree with its PTFs: *[VP [V NP]]* is an STF, *[VP [V]]* or *[VP [NP]]* are not STFs.

2.2.2 Partial Tree Kernel (PTK)

If we relax the production rule constraint over the STFs, we obtain a more general substructure type, i.e. PTF, generated by the application of partial production rules, e.g. Figure 2 shows that *[VP [NP[D]]]* is indeed a valid fragment. Note that PTK can be seen as a STK applied to all possible child sequences of the tree nodes, i.e. a string kernel combined with a STK.

2.3 Kernel Engineering

The Kernels of previous section are basic functions that can be applied to feature vectors, strings and

trees. In order to make them effective for a specific task, e.g. for coreference resolution: (a) we can combine them with additive or multiplicative operators and (b) we can design specific data objects (vectors, sequences and tree structures) for the target tasks.

It is worth noting that a basic kernel applied to an innovative view of a structure yields a new kernel (e.g. Moschitti and Bejan (2004); Moschitti et al. (2006)), as we show below:

Let $K(t_1, t_2) = \phi(t_1) \cdot \phi(t_2)$ be a basic kernel, where t_1 and t_2 are two trees. If we map t_1 and t_2 into two new structures s_1 and s_2 with a mapping $\phi_M(\cdot)$, we obtain: $K(s_1, s_2) = \phi(s_1) \cdot \phi(s_2) = \phi(\phi_M(t_1)) \cdot \phi(\phi_M(t_2)) = \phi'(t_1) \cdot \phi'(t_2) = K'(t_1, t_2)$, which is a noticeably different kernel induced by the mapping $\phi' = \phi \circ \phi_M$.

3 Kernels for Coreference Resolution

In this paper we follow the standard learning approach to coreference developed by Soon et al. (2001) and also used the few variants in Ng and Cardie (2002). In this framework, training and testing instances consist of a pair (anaphor, antecedent). During training, a positive instance is created for each anaphor encountered by pairing the anaphor with its closest antecedent; each of the non-coreferential mentions between anaphor and antecedent is used to produce a negative instance. During resolution, every mention to be resolved is paired 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 nearest candidate with a positive classification will be selected as the antecedent of the possible anaphor. The crucial point is that in this approach, the classifier is trained to identify positive and negative instances of the resolution process. In previous work on using kernel functions for coreference (Yang et al. 2006), structural information in the form of tree features was included in the instances. This approach is appropriate for identifying contexts in which the binding constraints apply, but not, for instance, to recognize expletives. In this work we adopted therefore a more general approach, in which separate classifiers are used to recognize each relevant configuration, and their output is then used as an input to the coreference classifier. In this section we discuss the types of structures and kernel functions we used for three different kinds of classifiers: expletive, binding and alias classifiers. We then present the results of these classifiers, and finally the results with the coreference resolver as a whole.

3.1 Expletive Kernels

In written text, about a third of the occurrences of the pronoun *it* are not coreferent to a previous mention, but either refer to a general discourse topic (*it's a shame*) or do not refer at all, as in the case of extraposed subjects (*it is thought that ...*) or weather verbs (*it's raining*). It is desirable to minimize the impact that these non-anaphoric pronouns have on the accuracy of a anaphora resolution: Lappin and Leass (1994), for example, use several heuristics to filter out expletive pronouns, including a check for patterns including modal adjectives (it is good/necessary/... that ...), and cognitive verbs (it is thought/believed/... that ...).

Newer approaches to the problem use machine-learning on hand-annotated examples: Evans (2001) compares a shallow approach based on surrounding lemmas, part-of-speech tags, and the presence of certain elements such as modal adjectives and cognitive verbs, trained on 3171 examples from Susanne and the BNC to a reimplementation of a pattern-based approach due to Paice and Husk (1987) and finds that the shallower machine-learning approach compares favorably to it. Boyd et al. (2005) use an approach that combines some of Evans' shallow features with hand-crafted patterns in a memory based learning approach and find that the more informative features are beneficial for the system's performance (88% accuracy against 71% for their reimplementation using Evans' shallow features).

Evans' study also mentions that incorporating the expletive classifier as a filter for a pronoun resolver gives a gain between 2.86% (for manually determined weights) and 1% (for automatically optimized weights).

Tree kernels are a good fit for expletive classification since they can naturally represent the lexical and structural context around a word. Our final classifier uses the combination of an unmodified tree (UT) (where the embedding clause or verb phrase of the pronoun is used as a tree), and a tree that only preserves the most salient structural features (ST).

The reduced representation prunes all nodes that

would not be seen as indicative in a pattern approach, essentially keeping verb argument structure and important lexical items, such as the governing verb and, in the case of copula constructions, the predicate. For example, the phrase

```
(S (NP (PRP It))
   (VP (VBZ has)
       (NP (NP (DT no) (NN bearing))
       (PP (IN on)
           (NP (NP (PRP$ our)
                   (NN work)
                   (NN force))
               (NP (NN today)))))))
   (. .))
```

would be reduced to the ST:

```
(S-I (NP-I (PRP-I It))
     (VP (VBX have)
         (NP))
     (.))
```

or, in a similar fashion,

```
(S (NP (PRP it))
   (VP (VBZ 's)
       (NP (NP (NN time))
           (PP (IN for)
               (NP (PRP$ their)
                   (JJ biannual)
                   (NN powwow))))))
```

would just be represented as the ST:

```
(S-I (NP-I (PRP-I it))
     (VP (BE VBZ)
         (NP-PRD (NN time))))
```

3.2 Binding Kernels

The resolution of pronominal anaphora heavily relies on the syntactic information and relationships between the anaphor and the antecedent candidates, including binding and other constraints, but also context-induced preferences in sub-clauses.

Some researchers (Lappin and Leass 1994; Kennedy and Boguraev 1996) use manually designed rules to take into account the grammatical role of the antecedent candidates as well as the governing relations between the candidate and the pronoun, while others use features determined over the parse tree in a machine-learning approach (Aone and Bennett 1995; Yang et al. 2004; Luo and Zitouni 2005). However, such a solution has limitations, since the syntactic features have to be selected and defined manually, and it is still partly an open question which syntactic properties should be considered in anaphora resolution.

We follow (Yang et al. 2006; Iida et al. 2006) in using a tree kernel to represent structural information using the subtree that covers a pronoun and its antecedent candidate. Given a sentence like "The
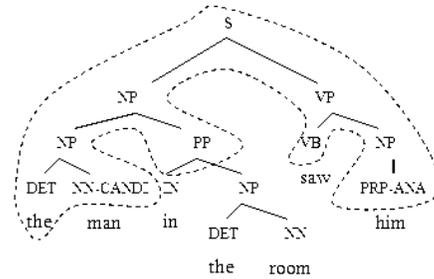

Figure 3: The structure for binding detection for the instance inst("the man", "him") in the sentence 'the man in the room saw him'

man in the room saw him.", we represent the syntactic relation between "*The man*" and "*him*", by the shortest node path connecting the pronoun and the candidate, along with the first-level of the node children in the path.

Figure 3 graphically shows such tree highlighted with dash lines. More in detail we operate the following tree transformation:
(a) 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, i.e. "NN-CANDI" for "man" and "PRP-ANA" for "him".
(b) To reduce the data sparseness, the leaf nodes representing the words are not incorporated in the feature, except that the word is the word node of the "DET" type (this is to indicate the lexical properties of an expression, e.g., whether it is a definite, indefinite or bare NP).
(c) If the pronoun and the candidate are not in the same sentence, we do not include the nodes denoting the sentences (i.e., "S" nodes) before the candidate or after the pronoun.

The above tree structures will be jointly used with the basic STK which extracts tree fragments able to characterize the following information: (a) the candidate is post-modified by a preposition phrase, (the node "PP" for "in the room" is included), (b) the candidate is a definite noun phrase (the article word "the" is included), (c) the candidate is in a subject position (NP-S-VP structure), (d) the anaphor is an object of a verb (the node "VB" for "saw" is included) and (e) the candidate is c-commanding the anaphor (the parent of the NP node for "the main in the room" is dominating the anaphor ("him"), which are important for reference determination in the pronoun resolution.

3.3 Encoding Context via Word Sequence Kernel

The previous structures aim at describing the interaction between one referential and one referent; if such interaction is observed on another mention pair, an automatic algorithm can establish if they corefer or not. This kind of information is the most useful to characterize the target problem, however, the context in which such interaction takes place is also very important. Indeed, natural language proposes many exceptions to linguistic rules and these can only be detect by looking at the context. To be able to represent context words or phrases, we use context word windows around the mentions and the subsequence kernel function (see section 2.1) to extract many features from it.

For example, in the context of "*and so **Bill Gates** says that*", a string kernel would extract features including: *Bill_Gates_says_that*, *says_that*, *Gates*, *Gates_says_that*, *Bill_says_that*, *so_Gates_says_that*, *and_so_that* and so on.

Name	Alias
BJ Habibie	Mr. Habibie
Federal Express	Fedex
Ju Rong Zhi	Ju

Table 1: Examples of coreferent named entities (aliases) taken from the MUC 6 corpus.

3.4 Kernels for Alias Resolution

Most methods currently employed by coreference resolution (CR) systems for identifying coreferent named entities, i.e. aliases, are fairly simplistic in nature, relying on simple surface features such as the edit distance between two strings representing names. We investigate the potential of using the structure contained within names. This can be very useful to solve complex cases like those shown in Table 1, taken from the MUC 6 corpus (Chinchor and Sundheim 2003). For this purpose, we add syntactic information to the feature set by tagging the parts of a name (e.g. *first name*, *last name*, etc.) as illustrated in Figure 4.

To automatically extract such structure we used the High Accuracy Parsing of Name Internal Structure (HAPNIS) script[1]. HAPNIS takes a name as input and returns a tagged name like what is shown in Figure 4. It uses a series of heuristics in making its classifications based on information such as the
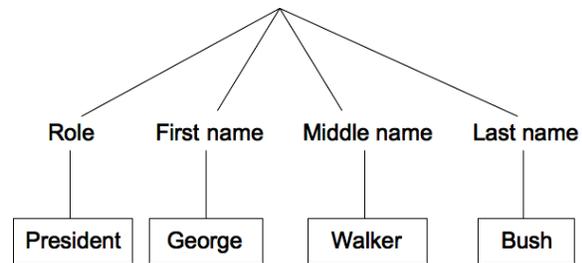

Figure 4: A proper name labeled with syntactic information.

serial positions of tokens in a name, the total number of tokens, the presence of meaningful punctuation such as periods and dashes, as well as a library of common first names which can be arbitrarily extended to any size. The tag set consists of the following: *surname*, *forename*, *middle*, *link*, *role*, and *suffix*[2].

Once the structure for a name has been derived, we can apply tree kernels to represent it in the learning algorithms thus avoiding the manual feature design. Such structures are not based on any particular grammar, therefore, any tree subpart may be relevant. In this case the most suitable kernel is PTK, which extracts any tree subpart. It is worth to note that the name tree structure can be improved by inserting a separate node for each name character and exploiting the string matching approximation carried out by PTK. For example, *Microsoft Inc.* will have a large match with *Microsoft Incorporated* whereas the standard string matching would be null.

4 Experiments with Coreference Subtask Classifiers

In these experiments we test the kernels devised for expletive (see Section 3.1), binding (see Section 3.2) and alias detection (see Section 3.4), to study the level of accuracy reachable by our kernel-based classifiers. The baseline framework is constituted by SVMs along with a polynomial kernel over the Soon et al.'s features.

4.1 Experiments on Expletive Classification

We used the BBN Pronoun corpus[3] as a source of examples, with the training set consisting of sections 00-19, yielding more than 5800 instances of

---

[1] The script is freely available at http://www.cs.utah.edu/ hal/HAPNIS/.

[2] Daumé reports a 99.1% accuracy rate on his test data set. We therefore concluded that it was sufficient for our purposes.

[3] Ralph Weischedel and Ada Brunstein (2005): BBN Pronoun Coreference and Entity Type Corpus, LDC2005T33

it, with the testing set consisting of sections 20 and 21, using the corresponding parses from the Penn Treebank for the parse trees. Additionally, we report on the performance of the classifier learnt on only the first 1000 instances to verify that our approach also works for small datasets. The results in Table 2 show that full tree (UT) achieves good results whereas the salient tree (ST) leads to a better ability to generalize, and the combination approach outperforms both individual trees.

	BBN large			BBN small		
	Prec	Recl	Acc	Prec	Recl	Acc
UT	83.87	61.54	84.35	78.76	52.66	80.85
ST	78.08	67.46	83.98	77.61	61.54	82.50
UT+ST	81.12	68.64	85.27	80.74	64.50	84.16

Table 2: Results for kernel-based expletive detection (using STK)

Note that the accuracy we get by training on 1000 examples (84% accuracy; see the *small* column in Table 2) is better than Boyd's replication of Evans (76% accuracy) or their decision tree classifier (81% accuracy) even though Boyd et al.'s dataset is three times bigger. On the other hand, Boyd et al's full system, which uses substantial hand-crafted knowledge, gets a still better result (88% accuracy), which is also higher than the accuracy of our classifier even when trained on the full 5800 instances.

	MUC-6		
	Prec	Recl	F
Soon et al.	51.25	55.51	53.29
STK	71.93	55.41	62.59

Table 3: Binding classifier: coreference classification on same-sentence pronouns

4.2 Experiments with the Binding Classifier

To assess the effect of the binding classifier on same-sentence pronoun links, we extracted 1398 mention pairs from the MUC-6 training data where both mentions were in the same sentence and at least one item of the pair included a pronoun, using the first 1000 for training and the remaining 398 examples for testing. The results (see Table 3) show that the syntactic tree kernel (STK) considerably improves the precision of classification of the Soon et al.'s features.

4.3 Experiments on Alias Classification

For our preliminary experiments, we extracted only pairs in the MUC 6 testing set in which both mentions were proper names, as determined by the coreference resolver's named entity recognizer. This set of proper names contained about 37,000 pairs of proper names of which about 600 were positive instances. About 5,500 pairs were randomly selected as test instances and the rest were used for training.

In the first experiment, we trained a decision tree classifier to detect if two names are aliases. For this task, we used either the string kernel score over the sequence of characters or the edit distance. The results in Table 4 show that the string kernel score performs better by 21.6 percentage points in F-measure.

In the second experiments we used SVMs trained with the string kernel over the name-character sequences and with PTK, which takes into account the structure of names. The results in Table 5 show that the structure improves alias detection by almost 5 absolute percent points. This suggests that an effective coreference system should embed PTK and name structures in the coreference pair representation.

	Recall	Precision	F-measure
String kernel	49.5%	60.8%	54.6%
Edit distance	23.9%	53.1%	33.0%

Table 4: Decision-tree based classification of name aliases using string kernels and edit distance.

	Recall	Precision	F-measure
String kernel	58.4%	67.5%	62.6%
PTK	64.8%	70.0%	67.3%

Table 5: SVM-based classification of name aliases using string kernels and tree-based feature.

5 Experiments on Coreference Systems

In this section we evaluate the contribution in the whole coreference task of the expletive classifier and the binding kernel. The predictions of the former are used as a feature of our basic coreference system whereas the latter is used directly in the coreference classifier by adding it to the polynomial kernel of the basic system.

Our basic system is based on the standard learning approach to coreference developed by Soon et al. (2001). It uses the features from Soon et al's work, including lexical properties, morphologic type, distance, salience, parallelism, grammatical role and so on. The main difference with

Soon et al. (2001) is the use of SVMs along with a polynomial kernel.

	MUC-6		
	Prec	Recl	F
plain	65.2	66.9	66.0
plain+expletive	66.1	66.9	66.5
upper limit	70.0	66.9	68.4

Table 6: Expletive classification: influence on pronoun resolution

5.1 Influence of Expletive classification

To see how useful a classifier for expletives can be, we conducted experiments using the expletive classifier learned on the BBN pronoun corpus on the MUC-6 corpus. Preliminary experiments indicated that perfect detection of expletives (i.e. using gold standard annotation) could raise the precision of pronoun resolution from 65.2% to 70.0%, yielding a 2.4% improvement in the F-score for pronoun resolution alone, or 0.6% improvement in the overall coreference F-score (see Table 6).

For a more realistic assessment, we used the classifier learned on the BBN pronoun corpus examples as an additional feature to gauge the improvement that could be achieved using it. While the gain in precision is small even in comparison to the achievable error reduction, we need to keep in mind that our baseline is in fact a well-tuned system.

	MUC-6			ACE02-BNews		
	R	P	F	R	P	F
PK	64.3	63.1	63.7	58.9	68.1	63.1
PK+TK	65.2	80.1	71.9	65.6	69.7	67.6

Table 7: Results of the pronoun resolution

5.2 Binding and Context Kernels

In these experiments, we compared our coreference system based on Polynomial Kernel (PK) against its combinations with Syntactic Tree Kernels (STK) over the binding structures (Sec. 3.2) and Word Sequence Kernel (WSK) on context windows (Sec. 3.3). We experimented with both the only pronoun and the complete coreference resolution tasks on the standard MUC-6 and ACE03-BNews data sets.

On the validation set, the best kernel combination between PK and STK was $STK(T1, T2) \cdot PK(\vec{x}_1, \vec{x}_2) + PK(\vec{x}_1, \vec{x}_2)$. Then an improvement arises when simply summing WSK.

Table 7 lists the results for the pronoun resolution. We used PK on the Soon et al.'s features as the baseline. On MUC-6, the system achieves a recall of 64.3% and precision 63.1% and an overall F-measure of 63.7%. On ACE02-BNews, the recall is lower 58.9% but the precision is higher, i.e. 68.1%, for a resulting F-measure of 63.1%. In contrast, adding the binding kernel (PK+STK) leads to a significant improvement in 17% precision for MUC-6 with a small gain (1%) in recall, whereas on the ACE data set, it also helps to increase the recall by 7%. Overall, we can see an increase in F-measure of around 8% for MUC and 4.5% for ACE02-BNews. These results suggest that the structured feature is very effective for pronoun resolution.

	MUC-6			ACE02-BNews		
	R	P	F	R	P	F
PK	61.5	67.2	64.2	54.8	66.1	59.9
PK+STK	63.4	67.5	65.4	56.6	66.0	60.9
PK+STK+WSK	64.4	67.8	66.0	57.1	65.4	61.0

Table 8: Results of the coreference resolution

Table 8 lists the results on the coreference resolution. We note that adding the structured feature to the polynomial kernel, i.e. using the model PK+STK, improves the recall of 1.9% for MUC-6 and 1.8% for ACE-02-BNews and keeps invariant the precision. Compared to pronoun resolution, the improvement of the overall F-measure is smaller (about 1%). This occurs since the resolution of non-pronouns case does not require a massive use of syntactic knowledge as in the pronoun resolution problem. WSK further improves the system's F1 suggesting that adding structured features of different types helps in solving the coreference task.

6 Conclusions

We presented four examples of using kernel-based methods to take advantage of a structured representation for learning problems that arise in coreference systems, presenting high-accuracy classifiers for expletive detection, binding constraints and same-sentence pronoun resolution, and name alias matching. We have shown the accuracy of the individual classifiers for the above tasks and the impact of expletives and binding classifiers/kernels in the complete coreference resolution system. The improvement over the individual and complete tasks suggests that kernel methods

are a promising research direction to achieve state-of-the-art coreference resolution systems.

Future work is devoted on making the use of kernels for coreference more efficient since the size of the ACE-2 corpora prevented us to directly use the combination of all kernels that we designed. In this paper, we have also studied a solution which relates to factoring out decisions into separate classifiers and using the decisions as binary features. However, this solution shows some loss in terms of accuracy. We are currently investigating methods that allow us to combine the accuracy and flexibility of the integrated approach with the speed of the separate classifier approach.

References

Aone, C. and Bennett, S. W. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. In *Proc. ACL 1995*, pages 122–129.

Barzilay, R. and Lapata, M. (2005). Modelling local coherence: An entity-based approach. In *Proc. of ACL*, Ann Arbor, MI.

Boyd, A., Gegg-Harrison, W., and Byron, D. (2005). Identifying non-referential it: a machine learning approach incorporating linguistically motivated features. In *Proc. ACL WS on Feature Engineering for Machine Learning in Natural Language Processing*.

Cancedda, N., Gaussier, E., Goutte, C., and Renders, J. M. (2003). Word sequence kernels. *JMLR*, 3:1059–1082.

Chinchor, N. and Sundheim, B. (2003). Muc 6 corpus. *Message Understanding Conference (MUC) 6*.

Chomsky, N. (1981). *Lectures on government and binding*. Foris, Dordrecht, The Netherlands.

Collins, M. and Duffy, N. (2002). New ranking algorithms for parsing and tagging: kernels over discrete structures and the voted perceptron. In *Proc. ACL 2002*, pages 263–270.

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

Giuglea, A.-M. and Moschitti, A. (2006). Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of Coling-ACL*, Sydney, Australia.

Grosz, B., Joshi, A., and Weinstein, S. (1995). Centering: a framework for modeling the local coherence of discourse. *CL*, 21(2):203–225.

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

Hobbs, J. (1979). Resolving pronoun references. *Coherence and Coreference*, 3(1):67–90.

Iida, R., Inui, K., and Matsumoto, Y. (2006). Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proc. Coling/ACL 2006*, pages 625–632.

Kennedy, C. and Boguraev, B. (1996). Anaphora for everyone: pronominal anaphora resolution without a parser. In *Proc. Coling 1996*.

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

Luo, X. and Zitouni, I. (2005). Multi-lingual coreference resolution with syntactic features. In *Proc. HLT/EMNLP 05*.

McCarthy, J. and Lehnert, W. (1995). Using decision trees for coreference resolution. In *Proc. IJCAI 1995*.

Mitkov, R. (2002). *Anaphora resolution*. Longman.

Moschitti, A. (2006). Efficient convolution kernels for dependency and constituent syntactic trees. *Proc. ECML 2006*.

Moschitti, A. and Bejan, C. A. (2004). A semantic kernel for predicate argument classification. In *CoNLL-2004*, USA.

Moschitti, A., Pighin, D., and Basili, R. (2006). Semantic Role Labeling via Tree Kernel Joint Inference. In *Proceedings of CoNLL-X*.

Moschitti, A., Quarteroni, S., Basili, R., and Manandhar, S. (2007). Exploiting syntactic and shallow semantic kernels for question answer classification. In *Proceedings ACL*, Prague, Czech Republic.

Ng, V. and Cardie, C. (2002). Improving machine learning approaches to coreference resolution. In *Proc. ACL 2002*.

Paice, C. D. and Husk, G. D. (1987). Towards an automatic recognition of anaphoric features in english text: The impersonal pronoun 'it'. *Computer Speech and Language*, 2:109–132.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Sidner, C. (1979). Toward a computational theory of definite anaphora comprehension in english. Technical report AI-TR-537, MIT, Cambridge, MA.

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

Steinberger, J., Poesio, M., Kabadjov, M., and Jezek, K. (2007). Two uses of anaphora resolution in summarization. *Information Processing and Management*, 43:1663–1680. Special issue on Summarization.

Sturt, P. (2003). The time-course of the application of binding constraints in reference resolution. *Journal of Memory and Language*.

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

Vieira, R. and Poesio, M. (2000). An empirically based system for processing definite descriptions. *CL*, 27(4):539–592.

Yang, X., Su, J., and Tan, C. (2006). Kernel-based pronoun resolution with structured syntactic knowledge. In *Proc. COLING-ACL 06*.

Yang, X., Su, J., Zhou, G., and Tan, C. (2004). Improving pronoun resolution by incorporating coreferential information of candidates. In *Proc. ACL 2004*.

Zanzotto, F. M. and Moschitti, A. (2006). Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of Coling-ACL*, Sydney, Australia.

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