CANDEL: An Algorithm for Same-Sentence Pronominal Resolution

Cristina Nicolae and **Gabriel Nicolae**
Human Language Technology Research Institute
Department of Computer Science
The University of Texas at Dallas
Richardson, TX 75083-0688
{cristina, gabriel}@hlt.utdallas.edu

Abstract

This paper presents a syntactic path-based learning algorithm (CANDEL from CANDIDATE-ELIMINATION) for the coreference resolution of pronouns that have their antecedents in the same sentence. Syntactic paths are treated as hypotheses to be learned. The hypotheses make up a version space that is delimited by a specific set and a general set, which grow closer to each other as the algorithm runs, in order to be consistent with the training examples encountered. Experiments on the MUC-6 and MUC-7 datasets reveal that this resolution method is a viable alternative to acquiring large amounts of data from the web.

Introduction

Pronoun resolution is a vital part of the more general task of coreference resolution. The aim of coreference resolution is, given a natural language text, to discover the entities mentioned in it and the textual mentions of these entities. A coreference system must correctly identify the sets into which the mentions of the text must be clustered so that each set corresponds to a real-world or imagined entity. As a step in this process, the system has to detect pairs of mentions that are assumed to refer to the same entity with a certain confidence. When one of these mentions is a pronoun, the task is called pronominal resolution. In the following example, the pronoun *its* refers back to the named entity *Dawn Capital* (its 'antecedent'):

Dawn Capital changes *its* name.

Previous work in pronominal resolution ranges from naive syntactic approaches to semantically rich approaches and from supervised to unsupervised learning. (Hobbs 1978) designed two methods for resolving third person pronouns: a naive algorithm based on syntactic parse trees and some selectional constraints, and a semantic approach that assumes knowledge available in the form of predicate calculus axioms. Due to the preconditions that were not available at the time, none of the algorithms was implemented, but running them by hand on a set of examples obtained good results. (Lappin & Leass 1994) designed an algorithm (RAP) for third person pronouns that is based on the syntactic representation of a slot grammar parser, and manually attached scores of salience to possible antecedents. The experimental results placed it higher in performance than an implementation of Hobbs' algorithm. (Kehler *et al.* 2004) showed that predicate-argument statistics offer little predictive power to a system for pronoun resolution trained with morphosyntactic features, and concluded that predicate-argument statistics are not suitable to replace world knowledge that might be necessary in the resolution. (Yang *et al.* 2004) incorporated coreferential information about the candidate antecedents of the mentions, information ignored by other reference resolution systems. The experimental results showed that this information boosted the resolution performance. (Yang, Su, & Tan 2005) used, in addition, statistic-based semantic compatibility information and data mined from the web. The results showed that using the semantic information coupled with the web for the twin-candidate model increased the performance of a system based on the single-candidate model and a corpus. (Cherry & Bergsma 2005) compared an unsupervised expectation maximization learning system with current supervised learning methods and found their performance almost equal, proving thus that unsupervised approaches are viable for pronoun resolution.

The approach presented in this paper is most similar to the one reported by (Bergsma & Lin 2006), because both systems take advantage of syntactic path information in pronominal resolution, but there are marked differences between the two approaches. First, Bergsma and Lin work with dependency paths obtained by the minimalist parser Minipar (Lin 1998), while CANDEL makes use of parse tree paths obtained by Collins' parser (Collins 1999). Additionally, they treat path coreference as just one of the features used by an SVM^{light} classifier (Joachims 1999) for pronominal resolution, whereas our approach is solely based on parse path information, and the learning algorithm is an adaptation of CANDIDATE-ELIMINATION (Mitchell 1997). They calculate the coreference of millions of paths in a bootstrapping framework, while CANDEL is a supervised learning based on a sparse corpus. Bergsma and Lin treat paths as a whole and obtain their likelihood of coreference by counting their appearances; in contrast, our approach 'looks inside' the paths in order to learn patterns of coreference and their likelihoods. Finally, CANDEL does not treat the case of

pronouns with antecedents in another sentence, as opposed to Bergsma and Lin's algorithm.

Same-Sentence Pronominal Resolution

One way to resolve the coreference of the mentions in a text through machine learning is to cluster them by using the more clear-cut cases (e.g. pronouns, names) as seeds (in the vein of (Cardie & Wagstaff 1999)). The initial clusters are formed by the most confident pairs pronoun-antecedent or name-antecedent. Clustering proceeds by incrementally adding to each cluster new mentions that are the most strongly connected to the ones that are already in the cluster. This addition can be stopped after a confidence threshold is reached, that is when mentions outside a cluster are sufficiently weakly linked to the ones inside.

Pronominal resolution is an important step in the above method. The method relies on correct seeding; otherwise, the errors introduced right at the beginning can multiply and negatively affect performance. The more confident the detection of the pronouns' antecedents is, the less chances the subsequent clustering has for wrong starts.

An analysis of the MUC-6 (MUC-6 1995) training data reveals that out of 292 total pronouns, 187 have an antecedent in the same sentence, 129 have one in the previous sentence, and 81 pronouns have antecedents both in the same and the previous sentences. This sums up to 236 pronouns who have an antecedent in either the same or the previous sentence. This sum constitutes approximately 80% of the total number of pronouns. Therefore, focusing just on the pronouns that resolve in the same and the previous sentences would solve a large majority of the task. This paper presents a novel method for resolving pronouns with same-sentence antecedents, leaving pronouns with candidates in the previous sentence to be resolved by a naive algorithm that links each pronoun to its closest candidate.

Pronouns are resolved through a syntactic pattern-based approach. The starting points for this approach are the syntactic paths between each pronoun and its candidate antecedent. The learning algorithm, the training instances, and the choice of candidate are all built to handle these paths. In particular, the learning algorithm employed (CANDEL) is a variation on the classic machine learning algorithm CANDIDATE-ELIMINATION. This algorithm was chosen for its relatively easy adaptability to a pattern-based approach, compared to popular machine learning methods like decision trees or maximum entropy.

Training instances

A training instance consisting of a feature vector and an outcome is created for each pair pronoun-candidate, where the pronoun and the candidate are in the same sentence. Candidates are all the nouns and pronouns that appear before the pronoun. If the two corefer according to the annotation, the outcome associated is positive; otherwise, the outcome associated is negative.

Learning with CANDEL

The original CANDIDATE-ELIMINATION algorithm computes a set of hypotheses (version space), all of which are consistent with an observed sequence of training examples. The version space is specified through two boundary hypotheses sets, the most specific (S) and the most general (G), and comprises all the hypotheses in the two sets plus the ones between them in generality. During the course of the algorithm, each new training example seen brings the two boundaries closer through generalization and specialization, respectively, such that the version space remains consistent throughout with all the training examples. For a more in-depth understanding of the algorithm, please refer to (Mitchell 1997) page 32.

The CANDEL algorithm is an adaptation of CANDIDATE-ELIMINATION to the case of pattern-based pronoun resolution. Our method differs from the original algorithm in three main points: (1) it is specifically designed for this task rather than a general machine learning technique; (2) it introduces statistical values attached to hypotheses; and (3) it changes the order in which the training examples are presented to the algorithm. All three points make the algorithm suitable for pronoun resolution, which cannot be said of the original one, so a performance comparison between them is not possible.

To elaborate on the three points, first CANDEL instantiates the original hypotheses into sentential parse tree paths, and their attributes into the nodes of these paths, which represent mentions that appear in the same sentence. The operations of generalization and specialization are adapted for parse tree paths. Secondly, in the original algorithm the hypotheses that are inconsistent with a new training example are eliminated. This tactic would be short-lived in our task, where the amount of training data and its inconsistency would reduce the version space to void in just a few steps. The original algorithm had to be changed to introduce a probabilistic approach. Each path (hypothesis) is associated with a confidence value throughout the running of the algorithm. The more examples it is consistent with, the greater its confidence is, while inconsistencies decrease confidence. In this way, we can keep a large version space at any time, and also prevent the elimination of useful hypotheses because of noise in the training data. Finally, all the positive examples are treated first, followed by all the negative examples. The reasoning behind this is presented in the subsection dedicated to describing the algorithm.

Extended parse paths A parse tree path that plays the role of hypothesis is denoted by a sequence of pairs sign-node extracted from the path between the pronoun and the candidate antecedent in the syntactic tree. To compose such a path from the tree, the tree is traversed from right to left, starting at the later occurring word of the pair. In our notation, the first node is preceded by the '#' sign. Afterwards, whenever the path goes upwards on a branch in the tree, a '+' sign is added to the path notation, and whenever the path descends, a '-' sign is added. All nodes encountered are added to the path. An example of a parse tree path is $\#PRP\$ + NP + PP + VP + S - NP$, which has been extracted as the path between antecedent *Dawn Capital* and pronoun *its* from the first parse tree illustrated in Figure 1(a). All the sign-node pairs in this path ($\#PRP\$, +NP, +PP, +VP, +S, -NP$) are attributes of

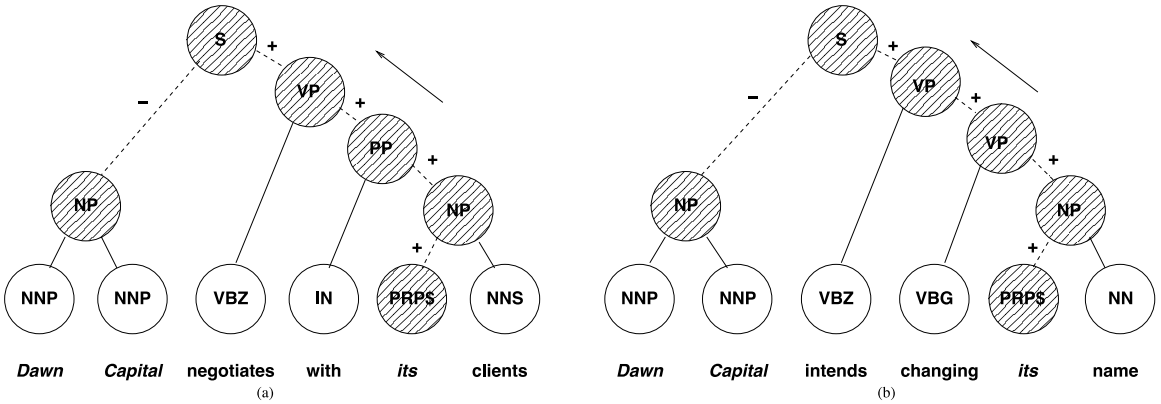

Figure 1: An example of two parse tree paths.

the hypothesis.

The parse paths have been extended in two ways: (a) each node can correspond to multiple mentions (connected by OR); and (b) a path can contain special nodes: '*', which replaces an unspecified set of nodes (which can be empty), and '?', which is a placeholder for *exactly one* unspecified node. These extensions give a pattern quality to the parse paths, which enables the operations performed on them by the learning algorithm.

To understand the rationale for this extension, consider the two sentences in Figure 1:

S_1 *Dawn Capital* negotiates with *its* clients.

S_2 *Dawn Capital* intends changing *its* name.

In both S_1 and S_2, the pronoun *its* refers to the named entity *Dawn Capital*. The parse paths (highlighted in the figure) between the pronoun and its antecedent are:

P_1	=	#PRP$	+NP	+PP	+VP	+S	-NP
P_2	=	#PRP$	+NP	+VP	+VP	+S	-NP

If the system treated these two paths (and the others) just as different strings, on a small corpus it would learn a number of unique paths comparable to the count of the training examples. Thus it would overfit on the training data, and any new variation on the paths already met would be classified as negative. The paths seen as black boxes are not sufficient; we need to search inside them for patterns. If, in the current example, a new (extended) path is generated out of the two:

G = #PRP$ +NP +(PP|VP) +VP +S -NP

this generalization will cover both situations.

Operations on paths Given two extended paths P_1 and P_2, three main operations can be performed on them: matching, generalization, and specialization. For the sake of simplicity in expression, let us call a path that can contain '*' nodes a starred path, and one that cannot– a non-starred path. Note that a non-starred path can still contain '?' nodes. Furthermore, lowercase single letters will be used instead of part of speech tags in the examples, but these letters stand for part of speech tags like 'NP'.

Matching

This operation verifies if starred path P_1 matches non-starred path P_2 in the spirit of regular expressions. Path P_1 is converted into a regular expression pattern, while path P_2 is converted into a text that will be compared against the pattern. For an example, the following are two paths that match.

P_1	=	#a | b | c	+*		+a | d	-b	-*
P_2	=	#a	+b	+?	+d	-b	

Generalization

For starred path P_1 and non-starred path P_2 that do not match, this operation obtains the minimal generalization G of P_1 such that G matches P_2. Preconditions for this operation are: (a) the paths have the same number of non-'*' nodes; and (b) the paths have the same tree structure. Nodes from path P_1 are coupled with nodes at the same position in path P_2. The generalization performs unions between the two sets of mentions inside the corresponding nodes. The resulting path will match P_2 by the definition of this operation. As an example, consider the following two paths:

P_1	=	#a | b	+a | c	+*	-d
P_2	=	#a	+d		-e

The two paths do not match, and they satisfy the preconditions for generalization. The generalized path is:

G = #a | b +a | c | d +* -d | e

Specialization

For starred path P_1 and non-starred path P_2 that match, this operation obtains the minimal specializations of P_1 such that none of them matches P_2. Since they match, the two hypotheses can be padded with null nodes or '?' nodes in order to align. After that, the specialization is done in two different ways according to the type of the node in P_1 that is being specialized:

i. For a node P_{1i} that was not obtained through expanding a '*', P_1 is specialized by two new paths that differ from it only by node i:
   a. S_1 with $sign(S_{1i}) = sign(P_{1i})$ and $tags(S_{1i}) = tags(P_{1i}) - tags(P_{2i})$;
   b. (only if $sign(S_{1i})$ is unspecified) S_2 with $sign(S_{2i}) = opposite of\ sign(P_{2i})$ and $tags(S_{2i}) = tags(P_{1i})$.

ii. For a sequence of n nodes that were obtained through expanding a '*', P_1 is specialized by replacing the sequence with two sets of subpaths:

1. subpaths that have a length different from n, no matter what nodes they contain:
   a. $length < n$: hypotheses made up only of '?' nodes, with lengths between 1 and $n-1$;
   b. $length > n$: a hypothesis with $n+1$ '?' nodes and a '*' at the end;
2. subpaths that can have length n that differ from the corresponding sequence in P_2 by one node (as described in (i.)).

A simple example is in order. Consider the following two paths that match:

P_1	=	#a | b	_?	_*		-c | d
P_2	=	#a	+b	+d	+e	-c

They are padded as follows:

P_1	=	#a | b	_?	_?	_?	-c | d
P_2	=	#a	+b	+d	+e	-c

The specialization hypotheses, in which $\wedge a$ means anything but a, and '_' stands for any sign, will be:

S_1	=	#b	_?	_*	-c | d		
S_2	=	#a | b	+$\wedge$b	_*	-c | d		
S_3	=	#a | b	-b	_*	-c | d		
S_4	=	#a | b	_?	_*	-d		
S_5	=	#a | b	_?	_?	-c | d		
S_6	=	#a | b	_?	_?	_?	_?	_*	-c | d
S_7	=	#a | b	_?	+$\wedge$d	_*	-c | d	
S_8	=	#a | b	_?	-d	_*	-c | d	
S_9	=	#a | b	_?	_?	+$\wedge$e	_*	-c | d
S_{10}	=	#a | b	_?	_?	-e	_*	-c | d
S_{11}	=	#a | b	_?	_*	+$\wedge$e	-c | d	
S_{12}	=	#a | b	_?	_*	-e	-c | d	
S_{13}	=	#a | b	_?	_*	+$\wedge$d	_?	-c | d
S_{14}	=	#a | b	_?	_*	-d	_?	-c | d

The set generated in (i) contains $S_1 - S_4$; the one in (ii.1.a) contains S_5; the one in (ii.1.b) contains S_6; and $S_7 - S_{14}$ are generated in (ii.2). None of these hypotheses match P_2. Note that in the end the sequence of + and - must have only one inflection.

The CANDEL **algorithm** The initial step of the algorithm is to insert a '*' hypothesis into G (the general boundary set) and a null hypothesis into S (the specific boundary set). Then, for all positive examples S changes through generalizations; after that, for all negative examples G changes through specializations. S and G change to keep the version space as consistent as possible with all training examples. For each training example considered, all hypotheses consistent with it increase their confidence, while some of the inconsistent ones decrease it and others are generalized or specialized– hypotheses in S are generalized to include positive examples and hypotheses in G are specialized to exclude negative examples. In the case of a positive example, a consistent hypothesis is one that matches it, while in the case of a negative example a consistent hypothesis is one that does not match it.

The order in which the training examples are fed to the algorithm is motivated by the specifics of the task. In the original algorithm, the order didn't matter; in our case, all the positive examples are presented first, followed by all the negative examples. This change was introduced because a few positive examples at the beginning, coupled with negative examples, can reduce G very fast, in a Greedy style. At each step, the hypotheses in G that are inconsistent with the current training example decrease their confidence. If the value becomes negative, they are eliminated. Because the confidence values start out low, a single positive example after some previous negative example modified G can delete at one step many hypotheses in G that might be consistent with a later positive example. By presenting all the positive examples first, this does not happen, since during this first half G will always contain a '*' hypothesis that will not be deleted.

The algorithm works in the following steps:

CANDEL

```
* Initialize G to the set of maximally
  general hypotheses in H.
* Initialize S to the set of maximally
  specific hypotheses in H.
* For each positive training example d, do
  1. Increase the confidence of any
     hypotheses in S or G consistent with
     d.
  2. Decrease the confidence of any
     hypotheses in G inconsistent with
     d.  Remove the ones with negative
     confidence.
  3. For each hypothesis s in S that is not
     consistent with d do
     a. Remove s from S.
     b. Add to S all minimal generalizations
        h of s such that h is consistent with
        d and some member of G is more general
        than h.  Associate initial confidence
        with h.
  4. Remove from S any hypothesis that is
     more general than another hypothesis in
     S.
* For each negative training example d, do
  1. Increase the confidence of any
     hypotheses in S or G consistent with
     d.
  2. Decrease the confidence of any
     hypotheses in S inconsistent with
     d.  Remove the ones with negative
     confidence.
  3. For each hypothesis g in G that is not
     consistent with d do
     a. Remove g from G.
     b. Add to G all minimal specializations
        h of g such that h is consistent with d
        and some member of S is more specific
        than h.  Associate initial confidence
        with h.
  4. Remove from G any hypothesis that is
     less general than another hypothesis in
     G.
```

Pronoun type	*Patterns*
personal	$\#P + S + SBAR + VP - N$ $\#P + S + VP + hVP + hS - NP - P$ $\#P + S \mid VP + hS \mid hVP - hS \mid hVP - NP \mid S - N$ $\#O\_ \wedge hS\_ \wedge SBAR + \wedge(S \mid SBAR) + \wedge S - O\_ \wedge (N \mid NP \mid S)\_*$
possessive	$\#P + hNP + NP + PP + N$ $\#P + NP \mid hNP + NP \mid VP + VP \mid hVP + S \mid hS \mid hVP - N \mid P$ $\#P + NP + N$ $\#O\_O\_O - \wedge(N \mid VP)\_ \wedge N\_O$
reflexive	$\#P + hVP + hS - N$ $\#P + PP + VP + VP + hVP + S - P$ $\#P + hS - N$ $\#* + \wedge(NP \mid S \mid VP)\_O$

Table 1: Patterns from final version spaces. P and N are used at the ends to stand for pronoun and noun, respectively. hS means an S node with the role of head for its parent node.

Version space For the purpose of using regular expression matching, the members of the version space of hypotheses obtained at the end of CANDEL must be spelled out. In order to do this, a queue structure is used. For each pair (s, g) with s in S, g in G, and g more general than S, the queue (which initially contains s) is updated with all minimal generalizations of its first element, which is moved to the version space set. At the end, the queue is empty and the version set contains all hypotheses in the version space bordered by S and G.

```
Version space set

For all hypotheses g in G do

  For all hypotheses s in S less general
  than g do

  1. Initialize queue to contain s
  2. While queue is not empty do
   a. Pop s' from head of queue and insert
    it into the version space set
   b. Introduce all minimal
    generalizations of s' at the end of
    queue.
```

Choosing a candidate For each pronoun in the testing data, the program generates testing examples from the paths between it and all its candidate antecedents that appear in the same sentence. For each of these paths, the program searches for all the patterns in the learned version space that match it. Every pattern has associated the number of positive and negative training examples it matches. (Note that the patterns from G will not match any negative example at the end of CANDEL, but the patterns from S might.) The score of the candidate is the total number of positive examples minus the total number of negative examples, over the patterns that match it. Constraints apply: agreement in gender and number is required, and only certain entity types can corefer with certain pronouns (e.g. PERSONS and ORGANIZATIONS with possessive pronouns). The entity types are automatically detected at preprocessing by a separate module. The valid candidate with the highest score is designated the antecedent of the pronoun.

Experimental Results

It might be interesting to see some examples of the patterns in the version space that emerged from the execution of CANDEL. Some of them, grouped according to pronoun type (personal, possessive, and reflexive), are presented in Table 1. Even if without the whole parse tree they are not exactly intuitive, we can observe certain preferences about the parts of speech, depending on the type of pronoun. Personal pronouns appear to resolve mostly with antecedents in other clauses, while possessive pronouns resolve in the same clause, in patterns that contain mostly noun phrases or pronominal phrases; and reflexive pronouns have short, simple patterns that keep them close to the noun they modify.

Experimenting was performed on the MUC-6 (MUC-6 1995) and MUC-7 (MUC-7 1997) datasets. The two corpora are annotated with coreference information and easily accommodate pronominal resolution by considering just the pronouns detected in the texts. MUC-6 contains 30 "dryrun" training files and 30 "formal" testing files, while MUC-7 contains 30 dryrun and 20 formal files. We trained our classifier on the dryrun texts and tested it on the formal texts.

Three types of pronouns were considered, taken separately and all together: personal, possessive, and reflexive. The scoring was calculated as precision P, recall R, and F-factor $F = \frac{2*P*R}{P+R}$. The results are summarized in Table 2. From the numbers of instances presented in the table it can be observed that the two corpora are sparse for this task. Indeed, the worst performance was recorded on the reflexive pronouns (F = 40% and F = 0% on the two corpora, respectively) that are 12 in total. The best performance was obtained in both cases on personal pronouns, which are also the greatest in number, followed by the possessives. The overall score, including all types of pronouns, was F = 78.6% and F = 75.7% respectively. The precision is always much higher than the recall, denoting that, even if not all antecedents were detected, the ones that were detected were correct in high proportions.

In order to compare our system against a challenging baseline, we implemented Hobbs' algorithm for pronominal resolution (Hobbs 1978). Some adaptation was required. First, Hobbs' algorithm only handles third person

Pronoun type	MUC-6				MUC-7			
	P	*R*	*F*	*instances*	*P*	*R*	*F*	*instances*
personal	93.6	73.0	**82.0**	278	93.1	74.1	**82.6**	201
possessive	85.1	70.8	**77.3**	137	67.9	49.3	**57.1**	73
reflexive	50.0	33.3	40.0	9	0.00	0.00	0.00	3
all types	88.7	70.5	**78.6**	424	87.2	66.9	**75.7**	277

Table 2: The experimental results for CANDEL's pronominal resolution.

personal pronouns (*he*, *she*, *it*, and *they*), therefore, for fairness' sake, both systems were tested only on this type of pronouns. Furthermore, Hobbs' syntactic parse tree notations were not 1-to-1 mappable to the notations of Collins' parser used by CANDEL, so some rules from Hobbs' algorithm had to be slightly modified according to the new tree structure. Hobbs' algorithm has a part that searches for an antecedent in the previous sentence, which was ignored in the testing to match CANDEL's resolution. The world knowledge selectional constraints in Hobbs' algorithm were replaced by the more accessible gender-number agreement and entity type constraints that were used by CANDEL. Finally, Hobbs tested his algorithm on perfect parse trees, while this testing was done using the parse trees detected by Collins' parser with less than perfect accuracy.

Algorithm	MUC-6			MUC-7		
	P	*R*	*F*	*P*	*R*	*F*
CANDEL	95.9	70.0	**80.9**	95.5	69.1	**80.2**
Hobbs	70.2	62.5	66.1	58.9	51.2	54.8

Table 3: Comparison with Hobbs' algorithm on third person personal pronouns only.

As Table 3 illustrates, CANDEL significantly outperforms the adapted Hobbs algorithm on both datasets. The difference in performance stems mainly from the way Hobbs searches for antecedents– his algorithm will almost always pick an antecedent for a pronoun because it also looks in the parse tree to the right of the pronoun, something CANDEL does not do. Thus, the amount of incorrectly found antecedents is larger in Hobbs' algorithm, and that contributes to the smaller scores.

Conclusions

We have presented CANDEL, an algorithm that learns patterns out of paths in syntactic parse trees, and uses them to identify the correct antecedents for pronouns that resolve in the same sentence. Syntactic paths are treated as hypotheses with sequences of nodes as their attributes, and the algorithm maintains a version space of hypotheses consistent with the training examples seen. The version space is delimited by a specific set and a general set that draw closer with every iteration, through generalization and specialization, respectively. Experiments on the MUC-6 and MUC-7 datasets show that the method is adequate in the case of sparsity of data, and thus offers an alternative to acquiring large amounts of data.

References

Bergsma, S., and Lin, D. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, 33–40. Sydney, Australia: Association for Computational Linguistics.

Cardie, C., and Wagstaff, K. 1999. Noun phrase coreference as clustering. In *Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, Association for Computational Linguistics*, 82–89.

Cherry, C., and Bergsma, S. 2005. An expectation maximization approach to pronoun resolution. In *Ninth Conference on Natural Language Learning*, 88–95.

Collins, M. 1999. *Head-Driven Statistical Models for Natural Language Parsing. PhD Dissertation, University of Pennsylvania.*

Hobbs, J. R. 1978. Resolving pronoun references. In Grosz, B.; Sparck-Jones, K.; and Webber, B., eds., *Readings in Natural Language Processing*. Morgan Kaufmann. 339–352.

Joachims, T. 1999. Making large-scale svm learning practical. In Scholkopf, B., and Burges, C., eds., *Advances in Kernel Methods*. MIT-Press.

Kehler, A.; Appelt, D.; Taylor, L.; and Simma, A. 2004. The (non)utility of predicate-argument frequencies for pronoun interpretation. In *Proceedings of the Human Language Technology Conference/North American chapter of the Association for Computational Linguistics, 2004*.

Lappin, S., and Leass, H. J. 1994. An algorithm for pronominal anaphora resolution. volume 20, 535–561.

Lin, D. 1998. Dependency-based evaluation of minipar. In *Workshop on the Evaluation of Parsing Systems, First International Conference on Language Resources and Evaluation*.

Mitchell, T. E. 1997. *Machine Learning*. McGraw-Hill.

MUC-6. 1995. Coreference task definition.

MUC-7. 1997. Coreference task definition.

Yang, X.; Su, J.; Zhou, G.; and Tan, C. L. 2004. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*.

Yang, X.; Su, J.; and Tan, C. 2005. Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics.*