Anaphora Resolution in Biomedical Literature: A Hybrid Approach

Jennifer D'Souza and Vincent Ng
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{jld082000,vince}@hlt.utdallas.edu

ABSTRACT

While traditional work on anaphora resolution has focused on resolving anaphors in newspaper and newswire articles, the surge of interest in biomedical natural language processing in recent years has stimulated work on anaphora resolution in biomedical texts. Existing anaphora resolvers, whether applied to the biomedical domain or not, have adopted either a learning-based or a rule-based approach. We hypothesize that both approaches have their unique strengths, and propose in this paper a hybrid approach to anaphora resolution in biomedical texts that aims to combine their strengths. Our hybrid approach achieves an F-score of 60.9 on the BioNLP-2011 coreference dataset, which to our knowledge is the best result reported to date on this dataset.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text Analysis

General Terms

Algorithms, Experimentation

Keywords

coreference resolution, anaphora resolution, bioinformatics

1. INTRODUCTION

Anaphora is a linguistic device commonly used in narratives and dialogs to avoid repetitions of phrases in human communication. By definition, an *anaphor* depends on another phrase, namely its *antecedent*, for its semantic interpretation. Hence, the automatic resolution of anaphors to antecedents, a task known as anaphora resolution, is a core (and challenging) issue in natural language processing (NLP). There are subtle differences between anaphora resolution and another task, coreference resolution, but for our purposes, it is not crucial to distinguish them.[1] Hence, following common practice, we will use the terms *anaphora* and *coreference* interchangeably in this paper.

Anaphora and coreference resolution is an enabling technology for many high-level NLP applications. In fact, coreference resolution was identified as a core task in the Sixth and Seventh Message Understanding Conferences [19, 20] needed to support high-level information-extraction tasks such as slot/template filling. More recently, the BioNLP-2011 shared task organizers have also identified coreference as an important supporting task for event extraction from biomedical texts [24], and have provided researchers with coreference-annotated biomedical texts for training and evaluating coreference resolvers. To illustrate the role played by coreference in biomedical event extraction, consider the following sentence, which is taken from a biomedical text used in the aforementioned shared task:

A mutant of KBF1/*p50* (delta SP), unable to bind to DNA but able to form homo- or heterodimers, has been constructed. This protein *reduces or abolishes* in vitro the DNA binding activity of wild-type proteins of the same family ...

This example describes a *negative regulation* event triggered by the words *reduces or abolishes*. The goal of an event extraction system is to automatically identify the existence of a negative regulation event (by identifying the trigger *reduces or abolishes*) as well as its *arguments*, such as its *cause* (which in this example is the protein *p50*). As we can see, the identification of *p50* as the cause of the event can be facilitated by the resolution of the definite noun phrase (NP) *This protein* to *A mutant of KBF1/p50 (delta SP)*.

It is worth mentioning that the BioNLP-2011 event extraction tasks focus on extracting events related to proteins/genes.[2] Hence, as a task supporting event extraction, the BioNLP-2011 coreference task focuses on *protein coreference*, which involves resolving anaphors that have protein references as their antecedents. Despite the restriction to

[1]Coreference resolution is concerned with clustering noun phrases that refer to the same real-world entity. Hence, two noun phrases that refer to *Barack Obama*, such as *Obama* and *President Obama*, should be grouped together by a coreference resolver. In contrast, anaphora resolution is concerned with identifying an antecedent for an anaphor, and does not involve establishing links between non-anaphors such as *Obama* and *President Obama*. Other differences between anaphora and coreference can be found in [33].

[2]As far as the shared task is concerned, the terms *proteins* and *genes* are synonymous and will be used interchangeably in this paper.

protein coreference, the BioNLP coreference task is very challenging: the best-performing coreference resolver in the shared task, Reconcile, achieves an F-measure of 34.05 [13], and a more recent resolver, which is implemented as part of the EventMine event extraction system [17], achieves an F-measure of 55.9 on the same dataset.

Our goal in this paper is to design a coreference resolver for improving the resolution of anaphors in the BioNLP protein coreference shared task. Unlike existing coreference resolvers, which adopt either a rule-based approach (as in EventMine) or a learning-based approach (as in Reconcile), we hypothesize that both of these approaches have their unique strengths. Consequently, we propose a *hybrid* approach to coreference resolution for combining their strengths. When evaluated on the BioNLP protein coreference dataset, our resolver achieves an F-measure of 60.9, surpassing EventMine's resolver by 5 points in absolute F-measure. To our knowledge, this is the best score reported to date on this dataset.

The rest of the paper is organized as follows. In Section 2, we review related work on coreference resolution. Sections 3 and 4 give an overview of the BioNLP coreference dataset and the architecture of our resolver. Sections 5 and 6 describe the two major components of our resolver, namely mention detection and anaphora resolution, respectively. Finally, we present evaluation results in Section 7 and conclude in Section 8.

2. RELATED WORK

In this section, we give a brief overview of the related work on coreference resolution.

Non-biomedical coreference resolution. Traditional anaphora and coreference resolvers were developed primarily for resolving anaphors in newspaper and newswire articles. There are two research trends that we believe are particularly worth mentioning.

The first is the shift from rule-based approaches to learning-based approaches. Specifically, rule-based anaphora resolvers were popular prior to the mid-1990s, and the design of rules in these resolvers was motivated to a large extent by well-known discourse theories [5, 6]. The advent of the statistical NLP era, as well as the public availability of coreference-annotated corpora produced by shared evaluations such as the Message Understanding Conferences (MUC), the Automatic Content Evaluations (ACE), and more recently, the CoNLL shared tasks, have prompted the development of machine learning approaches to coreference resolution. Various learning-based models of coreference resolution have been developed, ranging from the simple *mention-pair* model [23, 28] to *cluster-based* ranking models [26]. See Ng [22] for a detailed description of these models.

The second trend involves a shift from knowledge-lean approaches to knowledge-rich approaches. While there is general consensus that the difficulty of coreference resolution requires the use of sophisticated knowledge, it is by no means easy to compute such knowledge accurately. As a result, Mitkov [16] advocates the use of simple knowledge sources involving grammatical knowledge and shallow syntactic knowledge for coreference resolution. Recent work has focused on employing semantic knowledge extracted from lexical knowledge bases [7, 25, 27] or automatically acquired from an unannotated corpus [1, 21, 34].

Biomedical coreference resolution. The lack of a publicly available annotated corpus for biomedical coreference resolution until recently has made it relatively difficult for researchers to develop machine learning approaches and to perform *comparative* evaluations of their resolvers. Consequently, early approaches to biomedical coreference resolution are primarily rule-based [2, 10, 14], and researchers interested in developing machine learning approaches have to annotate their own corpus [4, 30, 32, 36]. Two biomedical coreference corpora were recently made publicly available, one released by Gasperin [4] and the other produced by the BioNLP-2011 shared task [24].

In addition to the distinction between rule-based approaches and machine learning approaches, existing work on biomedical coreference resolution can be classified along two dimensions. First, while most resolvers are evaluated on Medline abstracts [2, 10, 30, 32, 36], some are evaluated on full-text articles [4, 8]. Second, while some work targets the resolution of both pronominal and non-pronominal anaphors [30, 36], some focuses on resolving specific types of anaphors, such as non-pronominal anaphors [4, 8] and demonstratives [32]. However, unlike in non-biomedical coreference resolution, a clear distinction between knowledge-rich approaches and knowledge-lean approaches does not appear to exist in biomedical coreference resolution. Most existing resolvers have made use of features commonly employed by non-biomedical resolvers, such as string-matching and grammatical features, together with a few features specific to the biomedical domain. For example, Torii and Vijay-Shanker [32] have designed *highlighting features* that exploit certain lexical regularities in Medline abstracts.

We conclude this section by mentioning that virtually all existing coreference resolvers have adopted either a rule-based or a learning-based approach, unlike ours, which adopts a hybrid approach that aims to combine the strengths of rule-based and learning-based approaches.

3. DATASET

As mentioned before, we use as our evaluation dataset the BioNLP-2011 coreference dataset. The dataset is composed of 1210 documents, of which 800 are designated by the shared task organizers (and used by us) for training, 150 for development, and 260 for testing. These documents are taken from three sources: the MedCO dataset [30], the Genia event annotation [12], and the Genia Treebank [31]. There are 2309 anaphors in the training set and 473 anaphors in the development set. The percentages of different types of anaphors in these datasets are shown in Table 1. Note that we do not have the statistics for the test set, since coreference annotations on the test set are not made publicly available by the shared task organizers.[3]

4. SYSTEM ARCHITECTURE

We adopt a fairly standard pipeline architecture consisting of two components, mention detection and anaphora resolution. Given a text to be coreference-annotated, the mention detection component first extracts the anaphors and the candidate antecedents. Then, the anaphora resolution com-

[3]To evaluate the performance of our resolver on the test set, we have to submit our system output to a server, which will return the performance scores to us.

Anaphor Type	Training	Development
Relative pronoun	54.3%	56.9%
Personal pronoun	26.6%	26.0%
Definite NP	15.4%	14.0%
D&I pronoun	2.4%	2.1%
Others	1.3%	1.1%

Table 1: Statistics of the datasets.

ponent will select an antecedent for each extracted anaphor from the list of extracted candidate antecedents.

A natural question is: how do we implement these two components? The mention detection component can be implemented using a rule-based approach or a learning-based approach. As we will see, we implement both approaches, and adopt the one that yields the better performance on the development set.

Like the mention detection component, the anaphora resolution component can also be implemented using a rule-based approach or a learning-based approach. As mentioned before, we adopt a *hybrid* approach. More specifically, we hypothesize that different types of anaphors might be better resolved using different approaches. Personal pronouns, for example, are subject to syntactic constraints on coreference such as Binding Constraints, and might be better resolved by using a syntactic tree as a *structured* feature for a learning algorithm. Relative pronouns, on the other hand, can be resolved with a fairly high accuracy using simple heuristics. Definite NPs, as well as demonstrative and indefinite (D&I) pronouns, do not occur sufficiently frequently in the given training set to enable a learning algorithm to collect the kind of statistics needed to acquire accurate resolution rules, so a rule-based method might yield better results for these anaphors than learning-based methods. Importantly, however, we do **not** use these intuitions to determine whether a rule-based method or a learning-based method should be used to resolve a particular type of anaphors: as we will see in Section 6, we use the development set to guide the selection of the method for resolving a particular type of anaphors.

5. MENTION DETECTION COMPONENT

In this section, we will describe the first component of our system, mention detection. We first describe how to implement this component using machine learning (Section 5.1) and heuristic rules (Section 5.2), and then empirically compare these two mention detection methods (Section 5.3).

5.1 Learning-Based Mention Detection

Following Reconcile [13], we train two mention detectors independently, one for extracting candidate antecedents and one for extracting anaphors, on 400 of the 800 training documents.[4] Like the Reconcile team, we recast mention detection as a sequence labeling task. Specifically, the anaphor detector is trained to assign to each token in a development or test document a label that indicates whether it **b**egins an anaphor, is **i**nside an anaphor, or is **o**utside an anaphor. Hence, to learn the anaphor detector, we create one training instance for each token in the 400-document training set and derive its class value (one of **b**, **i**, and **o**) from the annotated data. Each instance represents the token under consideration, and consists of linguistic features including the token itself, its part-of-speech (POS) tag[5], affixes in the range of 1–3, orthographic features, and various combinations of these features as was done in Reconcile. The candidate antecedent detector is trained in a similar fashion using the same set of features, except that it is used to label candidate antecedents rather than anaphors.[6] We employ CRF++[7] to train both detectors.

5.2 Heuristic-Based Mention Detection

Our heuristic-based mention detector employs different methods for extracting anaphors and extracting candidate antecedents. Below we will begin by describing the heuristic anaphor detector.

Our anaphor detector assumes as input four lists: a list of personal pronouns, a list of relative pronouns, a list of D&I pronouns, and a list of definite NPs. The first three lists are manually created based on our commonsense knowledge of which words are pronouns, relative pronouns, and demonstratives.[8] On the other hand, the list of definite NPs is simply composed of all the definite NPs that appear in the training set. Given these four lists, the anaphor detector employs a two-step approach for extracting anaphors from a development/test document as follows. In the first step, the detector posits a word/phrase in the given document as a candidate anaphor if it appears in one of the four lists. Then, in the second step, a set of simple heuristics are applied to prune spurious candidate anaphors, in an attempt to improve the precision of the detector. For example, occurrences of "that" which serve as complementizers (e.g., "found that", "suggests that"), occurrences of demonstrative or indefinite pronouns which are part of a demonstrative/indefinite NP (e.g., "this transcription factor", "both enzymes"), and pleonastic pronouns (e.g., "It is found that", "It was possible that") are identified using simple patterns and are subsequently removed from the list of candidate anaphors.

On the other hand, our heuristic-based candidate antecedent detector operates simply by taking all base NPs from a syntactic parse that appear before an anaphor as the list of candidate antecedents for the anaphor.

5.3 Results

To get an idea of whether the learning-based or the rule-based mention detector is better, we conduct experiments on the development set to evaluate their performance. Specifically, Table 2 shows the number of gold anaphors in the development set, as well as the number of gold anaphors that are identified by the two mention detectors. Table 3 shows the same statistics for the candidate antecedents. As

[4]It will become obvious in the next section why we do not use all of the 800 training documents for training the mention detectors.

[5]All the POS tags used in our experiments are obtained using the McClosky-Charniak parser [15].

[6]Note that the candidate antecedents and the anaphors that we use in the training process include all and only those that appear in the .a2 files.

[7]Available from http://crfpp.sourceforge.net.

[8]For personal pronouns, we only include *it*, *its*, *itself*, *they*, and *their*, since the rest of them are rarely used as anaphors in a biomedical text.

Anaphor type	Gold	Learning	Heuristic
Relative pronoun	269	257	262
Personal pronoun	123	106	120
D&I pronoun	59	21	32
Definite NP	10	5	10

Table 2: Comparison of the number of gold anaphors recovered by the two mention detection methods.

Gold	Learning	Heuristic
449	186	313

Table 3: Comparison of the number of gold candidate antecedents recovered by the two methods.

we can see, the heuristic detector surpasses the learning-based detector in extracting candidate antecedents and all types of anaphors.

In addition, to determine how effective the pruning step employed by our heuristic detector is, we show in Table 4 the number of true positives (TP) and false positives (FP) extracted by the two mention detectors for each type of anaphors. While the number of TPs decrease slightly for relative pronouns after pruning, overall pruning helps improve the precision of the heuristic mention detector.

An important point deserves mention. While our results indicate that the heuristic mention detector outperforms the learning-based mention detector, it is still possible that the learning-based detector, when used in combination with the anaphora resolution component (see the next section), will produce better *coreference* results than the heuristic mention detector. Hence, in the remaining sections, we will use both mention detectors in combination with the anaphora resolution component to produce coreference results.

6. ANAPHORA RESOLUTION COMPONENT

Now that we have a list of anaphors and a list of candidate antecedents for each development/test document, our second component, the anaphora resolution component, will attempt to find an antecedent for each anaphor.

As mentioned before, we hypothesize that different resolution methods may work well for different types of anaphors. Hence, in this section, we describe six resolution methods that we employ to resolve each of the four types of anaphors in our dataset (namely, relative pronouns, personal pronouns, D&I pronouns, and definite NPs), and determine which of the six methods works best for each type of anaphors on the development set. The first five methods (Sections 6.1–6.5) are learning-based methods, and the last one is a rule-based method.

6.1 Reconcile Features

To determine whether the features commonly used for coreference resolution in newspaper/newswire articles are effective for biomedical coreference resolution, we employ in our first resolution method the feature set used by Reconcile [29], a state-of-the-art supervised resolver developed for the MUC and ACE coreference corpora. This feature set is composed of more than 66 commonly used string-matching,

Anaphor type	Before Pruning TP/FP	After Pruning TP/FP
Relative pronoun	269/313	262/22
Personal pronoun	123/235	120/5
D&I pronoun	32/19	32/13
Definite NP	10/12	10/2

Table 4: Effect of heuristic pruning.

grammatical, semantic, and positional features defined between an anaphor and a candidate antecedent.

Before we describe how these features can be used to train a coreference model, one point regarding the anaphors and the candidate antecedents used to generate instances for training the model deserves mention. As noted before, the anaphors and the candidate antecedents are obtained via either the learning-based mention detector or the heuristic-based mention detector. While all the anaphors and candidate antecedents are automatically extracted when the heuristic mention detector is used, the situation for the learning-based mention detector is different. Recall that our learning-based mention detector was trained on 400 of the 800 available training documents. When generating instances for training the coreference model from these 400 documents, we use the gold (i.e., correct) candidate antecedents and gold anaphors. For the remaining 400 documents, we generate training instances by using the candidate antecedents and anaphors extracted by the CRF models. The reason for using automatically extracted candidate antecedents and anaphors to generate training instances for the coreference learner is simple: it creates an environment for the learner that more closely resembles the condition during testing, where only automatically extracted candidate antecedents and anaphors are available.

Next, we describe how a coreference model can be trained using these anaphors, candidate antecedents, and the Reconcile features. Unlike Reconcile, which trains a classifier to determine whether an anaphor m_k and a candidate antecedent m_j are coreferent, we train a *ranker*, as ranking has been shown to outperform classifiers for coreference resolution [3, 9, 37]. Specifically, the ranker aims to impose a ranking on the candidate antecedents for each anaphor in a test document, so that the correct antecedent is assigned the highest rank. Hence, each training instance for training the ranker is an ordered pair ($\mathbf{x}_{m_i,m_k}$, $\mathbf{x}_{m_j,m_k}$), where $\mathbf{x}_{m_i,m_k}$ is a feature vector generated between an anaphor m_k and a correct antecedent m_i, and $\mathbf{x}_{m_j,m_k}$ is a feature vector generated between m_k and an incorrect candidate antecedent m_j. The goal of the ranker-learning algorithm, then, is to acquire a ranker that minimizes the number of violations of pairwise rankings provided in the training set. We train this ranker using Joachims' [11] SVM^{light} package on all 800 training documents.

There is a caveat, however. Since the anaphors and the candidate antecedents are automatically extracted, it is possible that (1) the anaphor m_k is erroneous (i.e., m_k is in fact *not* anaphoric), or (2) m_k is truly anaphoric, but its correct antecedent was not extracted by the detector. Note that when generating training instances for m_k that belongs to one of these cases, none of the extracted candidate antecedents is the correct antecedent. To address this problem, we posit that each anaphor has a *null* candidate antecedent. Specifically, if m_k belongs to one of these two cases, then we generate training instances of the form (*null*, $\mathbf{x}_{m_j,m_k}$), where m_j is a (wrong) candidate antecedent for m_k, so that the learner can learn that the *null* candidate antecedent should be ranked higher than all other candidates. Otherwise, we generate training instances of the form ($\mathbf{x}_{m_i,m_k}$, *null*), where m_i is a correct antecedent for m_k, so that the learner can learn that *null* is not the correct antecedent.

After training, the ranker can be applied to the test instances, which are created in the same way as the training instances. An anaphor is resolved to the highest-ranked candidate antecedent.

6.2 Sentence-Based Flat Parse Features

Our second resolution method is identical to the first method, except that the Reconcile features are replaced with features that encode the *paths* in a parse tree, where a path from node j and node k is an ordered sequence of nodes that need to be traversed in order to reach k from j.[9]

It may not be immediately clear whether we are indeed gaining anything by replacing the Reconcile features with path-based features, since some of the Reconcile features are already encoding information extracted from parse trees. To see the reason, recall that many parse-based features employed by existing coreference resolvers, including Reconcile, are computed by *heuristically* extracting information from parse trees. For instance, to compute the syntactic salience of an NP, the typical way is to extract information such as how far the NP is from the root of the parse tree in which it appears and whether it is embedded within a prepositional phrase. Note that both pieces of information can be captured in a simpler way using a *path-based* feature that encodes the path from the NP to the root of the tree. Hence, the main advantage of employing path-based features is *simplicity*: they obviate the need to design heuristics for extracting information from a parse tree, and therefore are especially useful in cases where it may not be easy to design such heuristics.

Given the potential advantages of employing paths as features, we employ in our second resolution method a feature set composed solely of six *path-based* features capturing the context of an anaphor m_k and/or one of its candidate antecedents m_j, as described below.

Feature 1: The path from the parent of the node corresponding to the first word of m_j to the root of the tree.

Feature 2: The path from the parent of the node corresponding to the last word of m_j to the root of the tree.[10]

Feature 3: The path from the parent of m_k to the root of the tree.[11]

Feature 4: Let P_j and P_k be the paths from Feature 1 and Feature 3, respectively, and CA_{jk} be the first node appearing in both P_j and P_k (i.e., CA_{jk} is the lowest common ancestor of m_j and m_k in the parse tree). Moreover, let A_j be the node immediately preceding CA_{jk} in P_j and A_k be the node immediately preceding CA_{jk} in P_k. Feature 4 encodes the sequence of nodes in the same level as A_j and A_k that lie between A_j and A_k.

Feature 5: Using the notation from Feature 4, Feature 5 encodes just one node, CA_{jk}.

Feature 6: The path from the parent of the node corresponding to the first word of m_j to the node corresponding to the parent of m_k.

As mentioned before, these six features aim to capture the context of an anaphor and/or one of its candidate antecedents. Specifically, Features 1 and 2 encode the paths from a candidate antecedent to the root of the tree, which, among other things, indirectly capture syntactic salience, as discussed before. Feature 3 is essentially the same as the first two features except that it operates on an anaphor. The remaining three features are *relational* features, capturing the relationship between an anaphor m_k and a candidate antecedent m_j. Specifically, Features 4 and 6 encode the lexical context and the syntactic context in which an anaphor and a candidate antecedent occur respectively, whereas Feature 5 encodes their lowest common ancestor. Encoding their lowest common ancestor could be useful for various reasons. For instance, if this ancestor is an NP but m_k and m_j are not in an appositive construction, then they are not likely to be coreferent. As another example, if this ancestor is a VP, m_k and m_j may correspond to different arguments of the VP and are therefore less likely to be coreferent.

To enable the reader to better understand how these six features are computed, we show in Figure 1 the values of these six features computed for the anaphor *which* and the candidate antecedent *these regulatory activities*. As we can see, Features 1 and 2 both encode the path NP-NP-PP-NP-PP-VP-S; Feature 3 encodes the path WHNP-WHPP-WHNP-SBAR-NP-PP-NP-PP-VP-S; Feature 4 is a sequence of length one consisting of the "," node; Feature 5 is composed of the "NP" node; and Feature 6 encodes the path NP-NP-SBAR-WHNP-WHPP-WHNP.

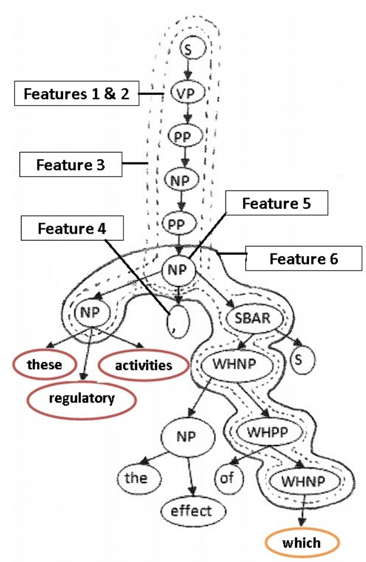

Figure 1: Example of path-based features.

6.3 Document-Based Flat Parse Features

[9]In our experiments, the parse trees are obtained using the McClosky-Charniak parser [15].

[10]At first glance, Feature 1 seems identical to Feature 2. However, their values are different for those candidate antecedents in our dataset that are very long and are spanned by more than one parent.

[11]An anaphor in this dataset is always spanned by the same parent, so it does not matter whether we use the first or the last word to compute the parent.

Our third resolution method is identical to the second resolution method, except that it aims to address one of its weaknesses. Specifically, if an anaphor m_k and its candidate antecedent m_j are not in the same sentence, Features 4, 5, and 6 cannot be computed, since m_k and m_j no longer have a common ancestor. To address this problem, we create for each document a *super-root* node, and add an edge between the super-root and the root of each of the parse trees for the sentences in the document. This construction ensures that Features 4, 5, and 6 can always be computed, since there is always a common ancestor for the nodes corresponding to any pair of mentions.

6.4 Sentence-Based Structured Parse Feature

While path-based features may obviate the need to design heuristics for effectively extracting information from syntactic parse trees, we are faced with another non-trivial problem: which paths should be used as features? In fact, one may question whether the six features introduced in the preceding two resolution methods can adequately capture the context of an anaphor and a candidate antecedent. Fortunately, advanced machine learning algorithms such as SVMs have enabled a parse tree to be used directly as a *structured* feature (i.e., a feature whose value is a linear or hierarchical structure, as opposed to a *flat* feature, which has a discrete or real value), owing to their ability to employ *kernels* to efficiently compute the similarity between two potentially complex structures.[12] In other words, by employing trees as features, we no longer need to design heuristics to extract information from parse trees or determine which paths to use as features.

Note, however, that while we want to use a parse tree directly as a feature, we do *not* want to use the *entire* parse tree as a feature. Specifically, while using the entire parse tree enables a richer representation of the syntactic context than using a *partial* parse tree, the increased complexity of the tree also makes it more difficult for the SVM learner to make generalizations.

To strike a better balance between having a rich representation of the context and improving the learner's ability to generalize, we extract a subtree from a parse tree and use it as the value of the structured feature of an instance. Specifically, given anaphor m_k, candidate antecedent m_j, and the associated syntactic parse tree T, we follow Yang et al. [35], retaining as our subtree the portion of T that covers (1) all the nodes lying on the shortest path from m_k and m_j (see Feature 6 in Section 6.2 for details on how this shortest path is computed), and (2) all the immediate children of these nodes that are not the leaves of T.

This subtree is known as a *simple expansion tree* [35]. To better understand how a simple expansion tree is computed, we show in Figure 2 the simple expansion tree (the subtree being circled) for the anaphor *which* and the candidate antecedent *these regulatory activities*.

We train a classifier on the 800 training documents for determining whether an anaphor m_k and a candidate antecedent m_j are coreferent, using a learning algorithm that can exploit tree-structured features, SVM$^{light-TK}$ [18], and a feature set composed of one feature, the simple expansion tree.[13] We follow Soon et al. [28] to create training instances: we create (1) a positive instance for each anaphor m_k and its closest antecedent m_j; and (2) a negative instance for m_k paired with each of the intervening candidate antecedents, $m_{j+1}, m_{j+2}, \ldots, m_{k-1}$. As in the first resolution method, we create an additional training instance between m_k and the *null* candidate antecedent. If m_k is a spurious anaphor or the correct antecedent for m_k was not extracted by the mention detector, this additional training instance will be labeled as positive; otherwise, it is labeled as negative.[14]

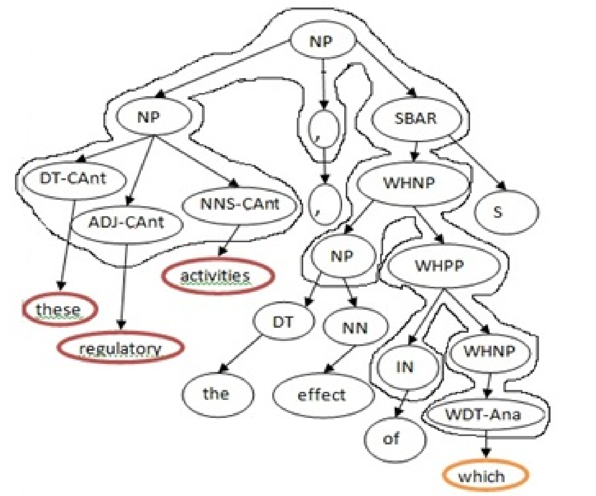

Figure 2: Example of a simple expansion tree.

After training, the classifier is used to identify an antecedent for an anaphor m_k in a test text. The test instances are generated in the same way as training, and m_k is resolved to the candidate antecedent that is classified as having the most positive classification confidence with m_k.

6.5 Document-Based Structured Parse Feature

As in the second resolution method, the simple expansion tree in the fourth resolution method is not computable if an anaphor m_k and its candidate antecedent m_j are not in the same sentence. To address this problem, we adopt the same solution as in the third resolution method, creating for each document a *super-root* node, and adding an edge between the super-root and the root of each of the parse trees for the sentences in the document. Other than this modification, our fifth resolution method is identical to the fourth one.

6.6 Rule-Based Method

Our final resolution method is rule-based. Since differ-

[12] One may wonder why our path-based features, which have linear structure, are not structured features. The reason is that that we are exploiting a path as a *value* rather than as a sequence. We could have employed the path-based features as structured features had we defined and applied a kernel function that operates on sequences to them.

[13] A classifier is trained in this case because SVM$^{light-TK}$ does not provide the option of training a ranker.

[14] As in the first resolution method, when the learning-based mention detector is used, gold anaphors and candidate antecedents are used to generate training instances for the 400 documents on which the detector was trained; and for the remaining 400 training documents, anaphors and candidate antecedents that are automatically extracted by the mention detector are used to generate training instances.

ent types of anaphors have different linguistic properties, we hypothesize different strategies are needed for resolving different types of anaphors. Consequently, we develop one ordered list of rules for resolving each type of anaphors.

For a given type of anaphors, the rules should be applied in the order in which they are listed. Specifically, if exactly one candidate antecedent satisfies the conditions specified in a rule, it is selected as the antecedent for the anaphor under consideration. However, if multiple candidate antecedents satisfy the conditions in a rule, the highest-ranked candidate antecedent is chosen to be the antecedent. As we will see, the way the candidate antecedents are ranked is dependent on the anaphor type.

Note that the rules below are only applicable to candidate antecedents that are either in the same sentence as the anaphor or in one of the two preceding sentences. A natural question, then, is: how were these rules designed, and how were they ordered? The rules are designed and ordered in part based on our commonsense knowledge, and in part based on our inspection of the training data. Hence, even though this rule-based method does not require an explicit training process, it is a "data-driven" rule-based method.

Resolving definite NPs
To resolve definite NP m_k, there are two cases to consider, depending on whether m_k is singular or plural.

If m_k is a *plural* NP, we apply the following rules. Specifically, we first apply them to the candidate antecedents in the same sentence as m_k. If no antecedent is found, we apply them to the candidates in the preceding sentence. If it is still not possible to find an antecedent, we apply them to the candidates in the second preceding sentence before positing m_k as non-anaphoric. Within each sentence, we employ a simple tie-breaking strategy in case more than one candidate satisfies the conditions of a rule: candidates that are closer to m_k are preferred to those that are farther away.

Rule 1: If the head noun of m_k is "gene" or "protein", resolve m_k to candidate m_j if (1) the head noun of m_j is "family" and (2) m_j contains at least one protein name[15].
Rule 2: Resolve m_k to candidate m_j if they have the same head noun.
Rule 3: Resolve m_k to candidate m_j if (1) m_j contains the coordinating conjunction "and", and (2) m_j contains a protein name if the head noun of m_k is "gene" or "protein".

If m_k is a *singular* NP, we apply the following rules, breaking ties simply by preferring candidates that are closer to m_k. Note that in this case, the sentence in which a candidate appears does not play any role in determining its rank.

Rule 1: Resolve m_k to candidate m_j if they have the same head noun.
Rule 2a: Resolve m_k to candidate m_j if the head noun of m_k is "gene" or "protein" and m_j contains a protein name.
Rule 2b (the Pattern rule): Resolve m_k to candidate m_j if one of the words of m_j (1) begins with a lowercase character and contains an uppercase character, a digit, or a special character (e.g., c-Myb, mAb 19C7); or (2) begins with a digit and contains alphabets (e.g., 20-methyl-23-eneanalogues); or (3) begins with an uppercase character and contains a digit (e.g., P450IA1 Elf-1).

Note that Rules 2a and 2b have the same precedence. In other words, if one candidate satisfies 2a and another satisfies 2b, then the higher-ranked candidate is selected as the antecedent.

Resolving personal pronouns
The following rules are used to resolve personal pronoun m_k. In cases where more than one candidate antecedent satisfies the conditions of a rule, we employ a simple tie-breaking strategy: candidate antecedents that are visited earlier when performing a right-to-left, depth-first traversal of the corresponding parse tree have a higher precedence than those that are visited later.

Rule 1: Resolve m_k to candidate m_j if (1) the two agree in number and are in the same sentence; and (2) m_j contains a protein name or one of its words satisfies the three conditions in the aforementioned Pattern rule.
Rule 2: Resolve m_k to candidate m_j if the two agree in number and are in the same sentence.
Rule 3: Resolve m_k to candidate m_j if m_j contains a protein name or one of its words satisfies the three conditions in the aforementioned Pattern rule.
Rule 4: Resolve m_k to candidate m_j if the two are in the same sentence.
Rule 5: Resolve m_k to candidate m_j if the two agree in number.

Resolving D&I pronouns
To resolve D&I pronoun m_k, we first apply the rules below to the candidate antecedents in the same sentence as m_k. If no antecedent is found, we apply them to the candidates in the preceding sentence. If it is still not possible to find an antecedent, we apply them to the candidates in the second preceding sentence before positing m_k as non-anaphoric. Note, however, that Rules 1 and 2 are only applicable to candidates that are in the same sentence as m_k.

Rule 1: Resolve m_k to candidate m_j such that (1) m_j is in the same sentence as m_k and (2) both of them are the subject of the same governing verb.
Rule 2: If m_k is part of a coordinated NP immediately preceded by the coordinating conjunction "or", then resolve m_k to the phrase immediately preceding "or" (motivating example: in the NP *enzyme1, enzyme2, or both*, *both* should be resolved to *enzyme1, enzyme2*).
Rule 3: Resolve m_k to the closest candidate m_j that agrees in number with m_k.

Resolving relative pronouns
Only one rule is used to resolve relative pronoun m_k: Resolve m_k to the closest candidate.

7. EVALUATION

In this section, we evaluate the effectiveness of our resolver using the BioNLP-2011 coreference dataset.

7.1 Experimental Setup

Recall that our resolver comprises two components, the mention detection component and the anaphora resolution component. The mention detection component employs (1) two methods for extracting anaphors, namely a CRF-based method and a heuristic-based method; and (2) two methods for extracting candidate antecedents, namely a CRF-based method and a heuristic-based method. After mention detection, we employ six resolution methods to resolve each of the four types of anaphors. Hence, for each type of anaphors,

[15]Note that for each document, the organizers provided a list of protein names that appeared in the document.

	CRF anaphors						Heuristic anaphors					
	CRF candidates			Heuristic candidates			CRF candidates			Heuristic candidates		
Resolution Method	R	P	F	R	P	F	R	P	F	R	P	F
Ranking-based Reconcile	21.3	60.6	31.5	13.4	47.4	20.8	21.3	62.3	31.7	14.9	53.6	23.3
Sentence-based flat	19.8	83.3	32.0	28.2	83.8	**42.2**	18.8	84.4	30.8	25.2	91.1	39.5
Document-based flat	19.3	83.0	31.3	28.2	78.0	41.4	19.3	84.8	31.5	24.3	90.7	38.3
Sentence-based structured	21.3	75.4	33.2	22.8	79.3	35.4	20.8	77.8	32.8	22.3	78.9	34.7
Document-based structured	21.3	69.4	32.6	22.3	77.6	34.6	20.8	72.4	32.3	22.3	81.8	35.0
Rule-based	—	—	—	27.2	75.3	40.0	—	—	—	27.7	77.8	40.8

(a) Resolution results for relative pronouns

	CRF anaphors						Heuristic anaphors					
	CRF candidates			Heuristic candidates			CRF candidates			Heuristic candidates		
Resolution Method	R	P	F	R	P	F	R	P	F	R	P	F
Ranking-based Reconcile	3.5	24.1	6.1	19.3	63.9	29.7	5.0	40.0	8.8	19.8	59.7	29.7
Sentence-based flat	3.5	53.8	6.5	21.8	74.6	33.7	3.5	63.6	6.6	21.3	76.8	33.3
Document-based flat	3.0	54.5	5.6	19.8	80.0	31.7	3.5	63.6	6.6	19.8	81.6	31.9
Sentence-based structured	3.5	53.8	6.5	24.3	73.1	36.4	5.0	66.7	9.2	26.3	77.9	**39.3**
Document-based structured	3.5	26.9	6.1	21.8	75.9	33.8	5.0	34.5	8.7	23.8	76.2	36.2
Rule-based	—	—	—	13.9	75.7	23.4	—	—	—	16.3	71.7	26.6

(b) Resolution results for personal pronouns

	CRF anaphors						Heuristic anaphors					
	CRF candidates			Heuristic candidates			CRF candidates			Heuristic candidates		
Resolution Method	R	P	F	R	P	F	R	P	F	R	P	F
Ranking-based Reconcile	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN
Sentence-based flat	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	2.0	12.9	**3.4**
Document-based flat	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	0.0	NaN
Sentence-based structured	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	0.0	NaN
Document-based structured	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN
Rule-based	—	—	—	0.0	NaN	NaN	—	—	—	1.0	100	2.0

(c) Resolution results for demonstrative and indefinite pronouns

	CRF anaphors						Heuristic anaphors					
	CRF candidates			Heuristic candidates			CRF candidates			Heuristic candidates		
Resolution Method	R	P	F	R	P	F	R	P	F	R	P	F
Ranking-based Reconcile	0.0	NaN	NaN	0.5	100	1.0	0.5	11.1	0.9	1.0	50.0	1.9
Sentence-based flat	0.0	NaN	NaN	0.5	7.1	0.9	0.0	NaN	NaN	2.5	14.7	4.2
Document-based flat	0.0	NaN	NaN	1.0	12.5	1.8	0.0	NaN	NaN	0.0	0.0	NaN
Sentence-based structured	0.0	NaN	NaN	0.0	0.0	NaN	0.0	NaN	NaN	0.0	NaN	NaN
Document-based structured	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN	0.0	NaN	NaN
Rule-based	—	—	—	5.0	38.5	8.8	—	—	—	6.9	58.3	**12.4**

(d) Resolution results for definite NPs

Table 5: Development set results for the four types of anaphors. The strongest results are boldfaced.

we have $2 \times 2 \times 6 = 24$ combinations of anaphor extraction method, candidate antecedent extraction method, and resolution method. For each type of anaphors, we determine the combination that yields the best F-measure on the development set. The development set results for the 24 combinations of each type of anaphors, expressed in terms of recall (R), precision (P), and F-measure (F)[16] are shown in Table 5. Note that (1) "NaN" is shown when the denominator involved in computing the corresponding score is zero, and this occurs when none of the anaphors belonging to that particular type was resolved; (2) no rule-based results are available for CRF-based candidate antecedents, since we did not conduct experiments with this particular combination (so only 22 combinations are available); and (3) the recall values indicate the percentages of *all* anaphors that are correctly resolved, so the 21.3% recall shown in row 1 of Table 5(a), for instance, means that 21.3% of all the anaphors (as opposed to just the anaphoric relative pronouns) in the development set are correctly resolved.

[16] This is the F-measure score computed using the *protein* coreference mode, which is the primary evaluation mode for the shared task.

7.2 Results

As we can see from Table 5, the best F-measure scores for different types of anaphors are achieved via different combinations. For example, the best F-measure score for relative pronoun resolution is achieved by training a ranker using sentence-based flat parse features on instances created from CRF-extracted anaphors and heuristically extracted candidate antecedents, whereas the best F-measure score for definite NP resolution is achieved by applying our handcrafted rules to the heuristically extracted anaphors and candidate antecedents. These results substantiate our hypothesis that different methods are needed to resolve different types of anaphors and that a hybrid approach exploiting the strengths of different methods may be desirable.

We employ the best combination learned for each anaphor type from the development set to resolve the anaphors in the test documents. Table 6 shows both the development set results and the test set results of our resolver. For comparison purposes, we also show in the same table the results of Reconcile, the best-performing resolver in the BioNLP-2011 shared task, and EventMine, whose resolver produces better results than Reconcile.[17] As we can see, our resolver outperforms EventMine's resolver by 5 points in F-measure, achieving the best results reported to date on this dataset.

7.3 Error Analysis

While our resolver outperforms state-of-the-art resolvers, there is a lot of room for improvement. To help direct future research on this task, we examine the output produced by our best-performing resolver on the development set, and analyze the major recall and precision problems associated with resolving each type of anaphors. Since D&I pronouns occur infrequently in our dataset, we will leave them out in our analysis.

For relative pronoun resolution, there is no major precision problem: as can be seen from Table 5(a), our resolver achieves fairly high precision (83.8%). This is perhaps not surprising: relative pronouns are comparatively easier to resolve than other types of anaphors, since they typically are in the same sentence as and are in close proximity to their antecedents. On the other hand, recall is limited primarily by the failure of the mention detector to extract the correct antecedents.

For definite NP resolution, precision and recall are limited by the precision and the recall of the anaphor detection method respectively: since our heuristic anaphor detector extracts all and only those definite NPs that appear in the training set, many extracted definite NPs are not anaphoric and many anaphoric definite NPs are not extracted.

Finally, for personal pronoun resolution, recall is limited primarily by the fact that the selected method performs only intra-sentential pronoun resolution. Precision problems can be attributed to two reasons. First, since only intra-sentential candidate antecedents are considered, an incorrect antecedent will be selected for an anaphor whose correct antecedent appears in a preceding sentence. Second, there are many cases where the resolution method incorrectly selects the candidate closest to the given anaphor as the antecedent despite the fact that the correct antecedent appears in the same sentence as the anaphor.

8. CONCLUSION

We presented a system for resolving anaphors in the BioNLP-2011 coreference dataset. Unlike existing resolvers, which adopt either a rule-based approach or a learning-based approach, our system adopts a hybrid approach, where different types of anaphors are resolved using different combinations of anaphor extraction method, candidate antecedent extraction method, and resolution method. Our resolver achieved an F-measure of 60.9 on held-out test data, surpassing the best known result by 5 points in F-measure.

[17] The results for Reconcile and EventMine are taken directly from the corresponding papers.

	Development Set			Test Set		
System	R	P	F	R	P	F
Reconcile	26.7	74.0	39.3	22.2	73.3	34.1
EventMine	53.5	69.8	60.5	50.4	62.7	55.9
Our system	59.9	77.1	**67.4**	55.6	67.2	**60.9**

Table 6: Resolution results of three resolvers.

9. ACKNOWLEDGMENTS

We thank the four reviewers for their invaluable comments on an earlier draft of the paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142.

10. REFERENCES

[1] D. Bean and E. Riloff. Unsupervised learning of contextual role knowledge for coreference resolution. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 297–304, 2004.

[2] J. Castaño, J. Zhang, and J. Pustejovsky. Anaphora resolution in biomedical literature. In *Proceedings of the 2002 International Symposium on Reference Resolution*, 2002.

[3] P. Denis and J. Baldridge. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660–669, 2008.

[4] C. Gasperin and T. Briscoe. Statistical anaphora resolution in biomedical texts. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 257–264, 2008.

[5] B. J. Grosz, A. K. Joshi, and S. Weinstein. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226, 1995.

[6] B. J. Grosz and C. L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986.

[7] S. Harabagiu, R. Bunescu, and S. Maiorano. Text and knowledge mining for coreference resolution. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 55–62.

[8] C. Huang, Y. Wang, Y. Zhang, Y. Jin, and Z. Yu. Coreference resolution in biomedical full-text articles with domain dependent features. In *Proceedings of the 2nd International Conference on Computer Technology and Development*, 2010.

[9] R. Iida, K. Inui, H. Takamura, and Y. Matsumoto. Incorporating contextual cues in trainable models for coreference resolution. In *Proceedings of the EACL*

Workshop on The Computational Treatment of Anaphora, 2003.

[10] J. jae Kim and J. C. Park. BioAR: Anaphora resolution for relating pronoun names to proteome database entries. In *Proceedings of the ACL Workshop on Reference Resolution and its Applications*, pages 79–86, 2004.

[11] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[12] J.-D. Kim, T. Ohta, and J. Tsujii. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(1):10, 2008.

[13] Y. Kim, E. Riloff, and N. Gilbert. The taming of Reconcile as a biomedical coreference resolver. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 89–93, 2011.

[14] Y.-H. Lin and T. Liang. Pronominal and sortal anaphora resolution for biomedical literature. In *Proceedings of the 16th Conference on Computational Linguistics and Speech Processing*, 2004.

[15] D. McClosky, E. Charniak, and M. Johnson. Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 337–344, 2006.

[16] R. Mitkov. Robust pronoun resolution with limited knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 869–875, 1998.

[17] M. Miwa, P. Thompson, and S. Ananiadou. Boosting automatic event extraction from the literature using domain adaptation and coreference resolution. *Bioinformatics (Advance Access)*, 2012.

[18] A. Moschitti. Making tree kernels practical for natural language processing. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 113–120, 2006.

[19] MUC-6. *Proceedings of the Sixth Message Understanding Conference*. Morgan Kaufmann, San Francisco, CA, 1995.

[20] MUC-7. *Proceedings of the Seventh Message Understanding Conference*. Morgan Kaufmann, San Francisco, CA, 1998.

[21] V. Ng. Semantic class induction and coreference resolution. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 536–543, 2007.

[22] V. Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, 2010.

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

[24] N. Nguyen, J.-D. Kim, and J. Tsujii. Overview of the protein coreference task in BioNLP shared task 2011. In *Proceedings of the BioNLP Shared Task 2011 Workshop*, pages 74–82, 2011.

[25] S. P. Ponzetto and M. Strube. Exploiting semantic role labeling, WordNet and Wikipedia for coreference resolution. In *Proceedings of the Human Language Technology Conference and Conference of the North American Chapter of the Association for Computational Linguistics*, pages 192–199, 2006.

[26] A. Rahman and V. Ng. Supervised models for coreference resolution. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 968–977, 2009.

[27] A. Rahman and V. Ng. Coreference resolution with world knowledge. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 814–824, 2011.

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

[29] V. Stoyanov, C. Cardie, N. Gilbert, E. Riloff, D. Buttler, and D. Hysom. RECONCILE: A coreference resolution research platform. In *Proceedings of the ACL 2010 Conference Short Papers*, 2010.

[30] J. Su, X. Yang, H. Hong, Y. Tateisi, and J. Tsujii. Coreference resolution in biomedical texts: A machine learning approach. In *Ontologies and Text Mining for Life Sciences: Current Status and Future Perspectives*, 2008.

[31] Y. Tateisi, A. Yakushiji, T. Ohta, and J. Tsujii. Syntax annotation for the Genia corpus. In *Proceedings of the Second Interational Joint Conference on Natural Language Processing*, pages 222–227, 2005.

[32] M. Torii and K. Vijay-Shanker. Anaphora resolution of demonstrative noun phrases in medline abstracts. In *Proceedings of 2005 Pacific-Asia Conference on Computational Linguistics*, pages 332–339, 2005.

[33] K. van Deemter and R. Kibble. On coreferring: Coreference in MUC and related annotation schemes. *Computational Linguistics*, 26(4):629–637, 2000.

[34] X. Yang and J. Su. Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 528–535, 2007.

[35] X. Yang, J. Su, and C. L. Tan. Kernel based pronoun resolution with structured syntactic knowledge. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 41–48, 2006.

[36] X. Yang, J. Su, G. Zhou, and C. L. Tan. An NP-cluster based approach to coreference resolution. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 226–232, 2004.

[37] X. Yang, G. Zhou, J. Su, and C. L. Tan. Coreference resolution using competitive learning approach. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 176–183, 2003.