Abstract

Coreference resolution is a key problem in natural language understanding that still escapes reliable solutions. One fundamental difficulty has been that of resolving instances involving pronouns since they often require deep language understanding and use of background knowledge. In this paper we propose an algorithmic solution that involves a new representation for the knowledge required to address hard coreference problems, along with a constrained optimization framework that uses this knowledge in coreference decision making. Our representation, Predicate Schemas, is instantiated with knowledge acquired in an unsupervised way, and is compiled automatically into constraints that impact the coreference decision. We present a general coreference resolution system that significantly improves state-of-the-art performance on hard, Winograd-style, pronoun resolution cases, while still performing at the state-of-the-art level on standard coreference resolution datasets.

1 Introduction

Coreference resolution is one of the most important tasks in Natural Language Processing (NLP). Although there is a plethora of works on this task (Soon et al., 2001a; Ng and Cardie, 2002a; Ng, 2004; Bengtson and Roth, 2008; Pradhan et al., 2012; Kummerfeld and Klein, 2013; Chang et al., 2013), it is still deemed an unsolved problem due to intricate and ambiguous nature of natural language text. Existing methods perform particularly poorly on pronouns, specifically when gender or plurality information cannot help. In this paper, we aim to improve coreference resolution by addressing these hard problems. Consider the following examples:

Ex.1 [A bird]$_{e1}$ perched on the [limb]$_{e2}$ and [it]$_{pro}$ bent.

Ex.2 [Robert]$_{e1}$ was robbed by [Kevin]$_{e2}$, and [he]$_{pro}$ is arrested by police.

In both examples, one cannot resolve the pronouns based on only gender or plurality information. Recently, Rahman and Ng (2012) gathered a dataset containing 1886 sentences of such challenging pronoun resolution problems (referred to later as the Winograd dataset, following Winograd (1972) and Levesque et al. (2011)). As an indication to the difficulty of these instances, we note that a state-of-the-art coreference resolution system (Chang et al., 2013) achieves precision of 53.26% on it. A special purpose classifier (Rahman and Ng, 2012) trained on this data set achieves 73.05%. The key contribution of this paper is a general purpose, state-of-the-art coreference approach which, at the same time, achieves precision of 76.76% on these hard cases.

Addressing these hard coreference problems requires significant amounts of background knowledge, along with an inference paradigm that can make use of it in supporting the coreference decision. Specifically, in Ex.1 one needs to know that “a limb bends” is more likely than “a bird bends”. In Ex.2 one needs to know that the subject of the verb “rob” is more likely to be the object of “arrest” than the object of the verb “rob” is. The knowledge required is, naturally, centered around
the key predicates in the sentence, motivating the central notion proposed in this paper, that of *Predicate Schemas*. In this paper, we develop the notion of *Predicate Schemas*, instantiate them with automatically acquired knowledge, and show how to compile it into constraints that are used to resolve coreference within a general *Integer Linear Programming* (ILP) driven approach to coreference resolution. Specifically, we study two types of *Predicate Schemas* that, as we show, cover a large fraction of the challenging cases. The first specifies one predicate with its subject and object, thus providing information on the subject and object preferences of a given predicate. The second specifies two predicates with a semantically shared argument (either subject or object), thus specifies role preferences of one predicate, among roles of the other. We instantiate these schemas by acquiring statistics in an unsupervised way from multiple resources including the Gigaword corpus, Wikipedia, Web Queries and polarity information.

A lot of recent work has attempted to utilize similar types of resources to improve coreference resolution (Rahman and Ng, 2011a; Ratinov and Roth, 2012; Bansal and Klein, 2012; Rahman and Ng, 2012). The common approach has been to inject knowledge as features. However, these pieces of knowledge provide relatively strong evidence that loses impact in standard training due to sparsity. Instead, we compile our *Predicate Schemas* knowledge automatically, at inference time, into constraints, and make use of an ILP driven framework (Roth and Yih, 2004) to make decisions. Using constraints is also beneficial when the interaction between multiple pronouns is taken into account when making global decisions. Consider the following example:

Ex.3 *Jack*$_{e_1}$ threw the bags of *John*$_{e_2}$ into the water since *he*$_{pro_1}$ mistakenly asked *him*$_{pro_2}$ to carry *his*$_{pro_3}$ bags.

In order to correctly resolve the pronouns in Ex.3, one needs to have the knowledge that "*he asks him*" indicates that *he* and *him* refer to different entities (because they are subject and object of the same predicate; otherwise, *himself* should be used instead of *him*). This knowledge, which can be easily represented as constraints during inference, then impacts other pronoun decisions in a global decision with respect to all pronouns: *pro$_3$* is likely to be different from *pro$_2$*, and is likely to refer to *e$_2$*. This type of inference can be easily represented as a constraint during inference, but hard to inject as a feature.

We then incorporate all constraints into a general coreference system (Chang et al., 2013) utilizing the mention-pair model (Ng and Cardie, 2002b; Bengtson and Roth, 2008; Stoyanov et al., 2010). A classifier learns a pairwise metric between mentions, and during inference, we follow the framework proposed in Chang et al. (2011) using ILP.

The main contributions of this paper can be summarized as follows:

1. We propose the *Predicate Schemas* representation and study two specific schemas that are important for coreference.
2. We show how, in a given context, *Predicate Schemas* can be automatically compiled into constraints and affect inference.
3. Consequently, we address hard pronoun resolution problems as a standard coreference problem and develop a system\(^1\) which shows significant improvement for hard coreference problems while achieving the same state-of-the-art level of performance on standard coreference problems.

The rest of the paper is organized as follows. We describe our *Predicate Schemas* in Section 2 and explain the inference framework and automatic constraint generation in Section 3. A summary of our knowledge acquisition steps is given in Section 4. We report our experimental results and analysis in Section 5, and review related work in Section 6.

2 *Predicate Schema*

In this section we present multiple kinds of knowledge that are needed in order to improve hard coreference problems. Table 1 provides two example sentences for each type of knowledge. We use $m$ to refer to a mention. A mention can either be an entity $e$ or a pronoun $pro$. $pred_m$ denotes the predicate of $m$ (similarly, $pred_{pro}$ and $pred_e$ for pronouns and entities, respectively). For instance, in sentence 1.1 in Table 1, the predicate of $e_1$ and $e_2$ is...
Paula liked [Ness]

Explanation of examples from Table 1

Also the entity mention it-bends”. Note that the predicate of the pronoun is that “a limb bends” is more reasonable than “a bird as mentioned in Section 1, we need the knowledge that “bee had pollen” is more reasonable than “flower had pollen”. Here, in addition to entity mention and the predicate of the pronoun, we need the argument which shares the predicate with the pronoun. To formally define the type of knowledge needed we denote it with “pred(m, a)” where m and a are a mention and an argument, respectively.

3.2 Paula liked [Ness] more than [Pokey] because [he] was mean to her.

Table 1: Example sentences for each schema category. The annotated entities and pronouns are hard coreference problems.

<table>
<thead>
<tr>
<th>Category</th>
<th>#</th>
<th>Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.1</td>
<td>[The bird] perched on the [limb] and [it] had bent.</td>
</tr>
<tr>
<td></td>
<td>1.2</td>
<td>[The bee] landed on [the flower] because [it] had pollen.</td>
</tr>
<tr>
<td>2</td>
<td>2.1</td>
<td>[Bill] was robbed by [John] so the officer arrested [him].</td>
</tr>
<tr>
<td></td>
<td>2.2</td>
<td>[Jimbo] was afraid of [Bobbert] because [he] gets scared around new people.</td>
</tr>
<tr>
<td>3</td>
<td>3.1</td>
<td>[Lakshman] asked [Vivan] to get him some ice cream because [he] was hot.</td>
</tr>
</tbody>
</table>

Table 2: Predicate Schemas and examples of the logic behind the schema design. Here * indicates that the argument is dropped, and S(.) denotes the scoring function defined in the text.

<table>
<thead>
<tr>
<th>Type</th>
<th>Schema form</th>
<th>Explanation of examples from Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>pred(m, a)</td>
<td>Example 1.2: It is enough to know that:</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a))</td>
<td>S have (m = [the flower], a = [pollen]) &gt;</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a))</td>
<td>S have (m = [the bee], a = [pollen])</td>
</tr>
<tr>
<td>2</td>
<td>pred(m, a, m)</td>
<td>Example 2.2: It is enough to know that:</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a, m))</td>
<td>S (be afraid of (m = *, a = *)) get scared (m = *, a =__), because &gt;</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a, m))</td>
<td>S (be afraid of (a = <em>, m =</em>)) get scared (m = *, a =__, because)</td>
</tr>
</tbody>
</table>

Table 3: Possible variations for scoring function statistics. Here * indicates that the argument is dropped.

<table>
<thead>
<tr>
<th>Type 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S(pred(m, a))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, *))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, *))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S(pred(m, a))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, a))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, *))</td>
</tr>
<tr>
<td></td>
<td>S(pred(m, *))</td>
</tr>
</tbody>
</table>

is pred = “perch on”. cn refers to the discourse connective (cn=“and” in sentence 1.1). a denotes an argument of pred other than m. For example, in sentence 1.1, assuming that m = e, the corresponding argument is a = e.

We represent the knowledge needed with two types of Predicate Schemas (as depicted in Table 2). To solve the assignment of [it] pro in sentence 1.1, as mentioned in Section 1, we need the knowledge that “a limb bends” is more reasonable than “a bird bends”. Note that the predicate of the pronoun is playing a key role here. Also the entity mention itself is essential. Similarly, for sentence 1.2, to resolve [it] pro, we need the knowledge that “bee had pollen” is more reasonable than “flower had pollen”. Here, in addition to entity mention and the predicate (of the pronoun), we need the argument which is being used in the case of Sentence 1.2.

For sentence 2.2, we need to have the knowledge that the subject of the verb phrase “be afraid of” is more likely than the object of the verb phrase “be afraid of” to be the subject of the verb phrase “get scared”. The structure here is more complicated than that of Type 1 schema. To make it clearer, we analyze sentence 2.1. In this sentence, the object of “be robbed by” is more likely than the subject.

Note that the order of m and a relative to the predicate is a critical issue. To keep things general in the schemas definition, we do not show the ordering; however, when using scores in practice the order between a mention and an argument is a critical issue.
of the verb phrase “be robbed by” to be the object of “the officer arrest”. We can see in both examples (and for the Type 2 schema in general), that both predicates (the entity predicate and the pronoun predicate) play a crucial role. Consequently, we design the Type 2 schema to capture the interaction between the entity predicate and the pronoun predicate. In addition to the predicates, we may need mention-argument information. Also, we stress the importance of the discourse connective between entity mention and pronoun; if in either sentence 2.1 or 2.2, we change the discourse connective to “although’, the coreference resolution will completely change. Overall, we can represent the knowledge as \( \text{pred}_m(u, a) \mid \text{pred}_m(m, \bar{a}), \text{cn} \). Just like for Type 1 schema, we can represent Type 2 schema with a score function for different variations of arguments (lower half of Table 3). In Table 2, we exhibit this for sentence 2.2.

Type 3 contains the set of instances which cannot be solved using schemas of Type 1 or 2. Two such examples are included in Table 1. In sentence 3.1 and 3.2, the context containing the necessary information goes beyond our triple representation and therefore this instance cannot be resolved with either of the two schema types. It is important to note that the notion of Predicate Schemas is more general than the Type 1 and Type 2 schemas introduced here. Designing more informative and structured schemas will be essential to resolving additional types of hard coreference instances.

3 Constrained ILP Inference

\begin{equation}
\text{arg max}_{u,v} \sum_{u \in \mathcal{M}, v \in \mathcal{M}} f_{u,v} y_{u,v}
\end{equation}

subject to \( y_{u,v} \in \{0, 1\}, \forall u,v \in \mathcal{M} \)

Constraints from Predicate Schemas Knowledge

Constraints between pronouns.

Here, \( u, v \) are mentions and \( y_{u,v} \) is the decision variable to indicate whether or not mention \( u \) and mention \( v \) are coreferents. As the first constraint shows, \( y_{u,v} \) is a binary variable. \( y_{u,v} \) equals 1 if \( u, v \) are coreferents and 0 otherwise. The second constraint indicates that we only choose at most one antecedent to be coreferent with each mention \( v \). \( (u < v \) represents that \( u \) appears before \( v \), thus \( u \) is an antecedent of \( v \).) In this work, we add constraints from Predicate Schemas Knowledge and between pronouns.

The Predicate Schemas knowledge provides a vector of score values \( S(u, v) \) for mention pairs \( \{(u, v) | (u \in \mathcal{M}, v \in \mathcal{P}\} \), which concatenates all the schemas involving \( u \) and \( v \). Entries in the score vector are designed so that the larger the value is, the more likely \( u \) and \( v \) are to be coreferents. We have two ways to use the score values: 1) Augmenting the feature vector \( \phi(u, v) \) with these scores. 2) Casting the scores as constraints for the coreference resolution ILP in one of the following forms:

\begin{equation}
\begin{cases}
\text{if } s_i(u, v) \geq \alpha_i s_i(w, v) \Rightarrow y_{u,v} \geq y_{w,v}, \\
\text{if } s_i(u, v) \geq s_i(w, v) + \beta_i \Rightarrow y_{u,v} \geq y_{w,v},
\end{cases}
\end{equation}

where \( s_i(.) \) is the \( i \)-th dimension of the score vector \( S(.) \) corresponding to the \( i \)-th schema represented for a given mention pair. \( \alpha_i \) and \( \beta_i \) are threshold values which we tune on a development set.\(^3\) If an inequality holds for all relevant schemas (that is, all the dimensions of the score vector), we add an inequality between the corresponding indicator variables inside the ILP.\(^4\) As we increase the value of a

\(^3\) For the \( i \)-th dimension of the score vector, we choose \( \alpha_i \) or \( \beta_i \) as the threshold.

\(^4\) If the constraints dictated by any two dimensions of \( S \) are contradictory, we ignore both of them.
threshold, the constraints in (1) become more conservative, thus it leads to fewer but more reliable constraints added into the ILP. We tune the threshold values such that their corresponding scores attain high enough accuracy, either in the multiplicative form or the additive form.\footnote{The choice is made based on the performance on the development set.} Note that, given a pair of mentions and context, we automatically instantiate a collection of relevant schemas, and then generate and evaluate a set of corresponding constraints. To the best of our knowledge, this is the first work to use such automatic constraint generation and tuning method for coreference resolution with ILP inference. In Section 4, we describe how we acquire the score vectors $S(u,v)$ for the Predicate Schemas in an unsupervised fashion.

We now briefly explain the pre-processing step required in order to extract the score vector $S(u,v)$ from a pair of mentions. Define a triple structure $t_m \triangleq \text{pred}_m(m,a,m)$ for any $m \in M$. The subscript $m$ for $\text{pred}$ and $a$, emphasizes that they are extracted as a function of the mention $m$. The extraction of triples is done by utilizing the dependency parse tree from the Easy-first dependency parser (Goldberg and Elhadad, 2010). We start with a mention $m$, and extract its related predicate and the other argument based on the dependency parse tree and part-of-speech information. To handle multiword predicates and arguments, we use a set of hand-designed rules. We then get the score vector $S(u,v)$ by concatenating all scores of the Predicate Schemas given two triples $t_u$, $t_v$. Thus, we can expand the score representation for each type of Predicate Schemas given in Table 2: 1) For Type 1 schema, $S(u,v) \equiv S(\text{pred}_v(m = u, a = a_v))$ \footnote{In $\text{pred}_v(m = u, a = a_v)$ the argument and the predicate are extracted relative to $v$ but the mention $m$ is set to be $u$.} 2) For Type 2 schema, $S(u,v) \equiv S(\text{pred}_a(m = v, a = a_v)|\text{pred}_v(m = v, a = a_v),cn)$.

In additional to schema-driven constraints, we also apply constraints between pairs of pronouns within a fixed distance\footnote{We set the distance to be 3 sentences.}. For two pronouns that are semantically different (e.g. he vs. it), they must refer to different antecedents. For two non-possessive pronouns that are related to the same predicate (e.g. he saw him), they must refer to different antecedents.\footnote{Three cases are considered: he-him, she-her, they-them}

\section{Knowledge Acquisition}

One key point that remains to be explained is how to acquire the knowledge scores $S(u,v)$. In this section, we propose multiple ways to acquire these scores. In the current implementation, we make use of four resources. Each of them generates its own score vector. Therefore, the overall score vector is the concatenation of the score vector from each resource: $S(u,v) = [S_{\text{giga}}(u,v) S_{\text{wiki}}(u,v) S_{\text{web}}(u,v) S_{\text{pol}}(u,v)]$.

\subsection{Gigaword Co-occurrence}

We extract triples $t_m \triangleq \text{pred}_m(m, a_m)$ (explained in Section 3) from Gigaword data (4,111,240 documents). We start by extracting noun phrases using the Illinois-Chunker (Punyakanok and Roth, 2001). For each noun phrase, we extract its head noun and then extract the associated predicate and argument to form a triple.

We gather the statistics for both schema types after applying lemmatization on the predicates and arguments. Using the extracted triples, we get a score vector from each schema type: $S_{\text{giga}} = (S_{\text{giga}}^{(1)} S_{\text{giga}}^{(2)})$.

To extract scores for Type 1 Predicate Schemas, we create occurrence counts for each schema instance. After all scores are gathered, our goal is to query $S_{\text{giga}}^{(1)}(u,v) \equiv S(\text{pred}_v(m = u, a = a_v))$ from our knowledge base. The returned score is the $\log(.)$ of the number of occurrences.

For Type 2 Predicate Schemas, we gather the statistics of triple co-occurrence. We count the co-occurrence of neighboring triples that share at least one linked argument. We consider two triples to be neighbors if they are within a distance of three sentences. We use two heuristic rules to decide whether a pair of arguments between two neighboring triples are coreferents or not: 1) If the head noun of two arguments can match, we consider them coreferents. 2) If one argument in the first triple is a person name and there is a compatible pronoun (based on its gender and plurality information) in the second triple, they are also labeled as coreferents. We also extract the discourse connectives between triples (because, ...
Given the co-occurrence information, we get a score vector \( S_{\text{wiki}}(u, v) \) corresponding to Type 1 Predicate Schemas, and hence \( S(u, v)_{\text{wiki}} \equiv S(\text{pred}_v(m = u, a = a_v)). \)

### 4.3 Web Search Query Count

Our third source of score vectors is web queries that we implement using Google queries. We extract a score vector \( S_{\text{web}}(u, v) \equiv S(\text{pred}_v(m = u, a = a_v)) \) (Type 1 Predicate Schemas) by querying for 1) “\( u a_v \)” 2) “\( u \text{ pred}_v \)” 3) “\( u \text{ pred}_v, a_v \)” 4) “\( a_v, u \)”.11 For each variation of nouns (plural and singular) and verbs (different tenses) we create a different query and average the counts over all queries. Concatenating the counts (each is a separate dimension) would give us the score vector \( S_{\text{web}}(u, v) \).

### 4.4 Polarity of Context

Another rich source of information is the polarity of context, which has been previously used for Winograd schema problems (Rahman and Ng, 2012). Here we use a slightly modified version. The polarity scores are used for Type 1 Predicate Schemas and therefore we want to get \( S_{\text{pol}}(u, v) \equiv S(\text{pred}_v(m = u, a = a_v)) \). We first extract polarity values for \( \text{Po(pred}_u) \) and \( \text{Po(pred}_v) \) by repeating the following procedures for each of them:

- We extract initial polarity information given the predicate (using the data provided by Wilson et al. (2005)).
- If the role of the mention is object, we negate its polarity.
- If there is a polarity-reversing discourse connective (such as “but”) preceding the predicate, we reverse the polarity.
- If there is a negative comparative adverb (such as “less”, “lower”) we reverse the polarity.

---

9We use the \( \log(.) \) of the counts here.

10Conditional probability of “limb” immediately following the given verb “bend”.

11We query this only when \( a_v \) is an adjective and \( \text{pred}_v \) is a to-be verb.
Given the polarity values $Po(pred_u)$ and $Po(pred_v)$, we construct the score vector $S_{pol}(u, v)$ following Table 4.

5 Experiments

In this section, we evaluate our system for both hard coreference problems and general coreference problems, and provide detailed analysis on the impact of our proposed Predicate Schemas. Since we treat resolving hard pronouns as part of the general coreference problems, we extend the Winograd dataset with a more complete annotation to get a new dataset. We evaluate our system on both datasets, and show significant improvement over the baseline system and over the results reported in Rahman and Ng (2012). Moreover, we show that, at the same time, our system achieves the state-of-the-art performance on standard coreference datasets.

5.1 Experimental Setup

Datasets: Since we aim to solve hard coreference problems, we choose to test our system on the Winograd dataset\(^{12}\) (Rahman and Ng, 2012). It is a challenging pronoun resolution dataset which consists of sentence pairs based on Winograd schemas. The original annotation only specifies one pronoun and two entities in each sentence, and it is considered as a binary decision for each pronoun. As our target is to model and solve them as general coreference problems, we expand the annotation to include all pronouns and their linked entities as mentions (We call this new re-annotated dataset WinoCoref\(^{13}\)). Ex.3 in Section 1 is from the Winograd dataset. It originally only specifies he as the pronoun in question, and we added him and his as additional target pronouns. We also use two standard coreference resolution datasets ACE(2004) (NIST, 2004) and OntoNotes-5.0 (Pradhan et al., 2011) for evaluation. Statistics of the datasets are provided in Table 5.

Baseline Systems: We use the state-of-the-art Illinois coreference system as our baseline system (Chang et al., 2013). It includes two different versions. One employs Best-Left-Link (BLL) inference method (Ng and Cardie, 2002b), and we name it Illinois\(^{14}\); while the other uses ILP with constraints for inference, and we name it IlliCons. Both systems use Best-Link Mention-Pair (BLMP) model for training.

Developed Systems: We present three variations of the Predicate Schemas based system developed here. We inject Predicate Schemas knowledge as mention-pair features and retrain the system (KnowFeat). We use the original coreference model and Predicate Schemas knowledge as constraints during inference (KnowCons). We also have a combined system (KnowComb), which uses the schema knowledge to add features for learning as well as constraints for inference. A summary of all systems is provided in Table 6.

---

\(^{12}\)Available at http://www.hlt.utdallas.edu/~vince/data/emnlp12/

\(^{13}\)Available at http://cogcomp.cs.illinois.edu/page/data/

\(^{14}\)In implementation, we use the L\(^3\)M model proposed in Chang et al. (2013), which is slightly different. It can be seen as an extension of BLL inference method.
Table 7: Performance results on Winograd and WinoCoref datasets. All our three systems are trained on WinoCoref, and we evaluate the predictions on both datasets. Our systems improve over the baselines by over than 20% on Winograd and over 15% on WinoCoref.

<table>
<thead>
<tr>
<th>System</th>
<th>MUC</th>
<th>BCUB</th>
<th>CEAF</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IlliCons</td>
<td>78.17</td>
<td>81.64</td>
<td>78.45</td>
<td>79.42</td>
</tr>
<tr>
<td>KnowComb</td>
<td>77.51</td>
<td>81.97</td>
<td>77.44</td>
<td>78.97</td>
</tr>
<tr>
<td>OntoNotes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IlliCons</td>
<td>84.10</td>
<td>78.30</td>
<td>68.74</td>
<td>77.05</td>
</tr>
<tr>
<td>KnowComb</td>
<td>84.33</td>
<td>78.02</td>
<td>67.95</td>
<td>76.76</td>
</tr>
</tbody>
</table>

Table 8: Performance results on ACE and OntoNotes datasets. Our system gets the same level of performance compared to a state-of-art general coreference system.

5.2 Results for Hard Coreference Problems

Performance results on Winograd and WinoCoref datasets are shown in Table 7. The best performing system is KnowComb. It improves by over 20% over a state-of-art general coreference system on Winograd and also outperforms Rahman and Ng (2012) by a margin of 3.3%. On the WinoCoref dataset, it improves by 15%. These results show significant performance improvement by using Predicate Schemas knowledge on hard coreference problems. Note that the system developed in Rahman and Ng (2012) cannot be used on the WinoCoref dataset. The results also show that it is better to compile knowledge into constraints when the knowledge quality is high than add them as features.

5.3 Results for Standard Coreference Problems

Performance results on standard ACE and OntoNotes datasets are shown in Table 8. Our KnowComb system achieves the same level of performance as does the state-of-art general coreference system we base it on. As hard coreference problems are rare in standard coreference datasets, we do not have significant performance improvement. However, these results show that our additional Predicate Schemas do not harm the predictions for regular mentions.

5.4 Detailed Analysis

To study the coverage of our Predicate Schemas knowledge, we label the instances in Winograd (which also applies to WinoCoref) with the type of Predicate Schemas knowledge required. The distribution of the instances is shown in Table 9. Our proposed Predicate Schemas cover 73% of the instances.

<table>
<thead>
<tr>
<th>Category</th>
<th>Cat1</th>
<th>Cat2</th>
<th>Cat3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>317</td>
<td>1060</td>
<td>509</td>
</tr>
<tr>
<td>Portion</td>
<td>16.8%</td>
<td>56.2%</td>
<td>27.0%</td>
</tr>
</tbody>
</table>

Table 9: Distribution of instances in Winograd dataset of each category. Cat1/Cat2 is the subset of instances that require Type 1/Type 2 schema knowledge, respectively. All other instances are put into Cat3. Cat1 and Cat2 instances can be covered by our proposed Predicate Schemas.
<table>
<thead>
<tr>
<th>Schema</th>
<th>AntePre(Test)</th>
<th>AntePre(Train)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>76.67</td>
<td>86.79</td>
</tr>
<tr>
<td>Type 2</td>
<td>79.55</td>
<td>88.86</td>
</tr>
<tr>
<td>Type 1 (Cat1)</td>
<td>90.26</td>
<td>93.64</td>
</tr>
<tr>
<td>Type 2 (Cat2)</td>
<td>83.38</td>
<td>92.49</td>
</tr>
</tbody>
</table>

Table 10: Ablation Study of Knowledge Schemas on WinoCoref. The first line specifies the performance for KnowComb with only Type 1 schema knowledge tested on all data while the third line specifies the performance using the same model but tested on Cat1 data. The second line specifies the performance results for KnowComb system with only Type 2 schema knowledge on all data while the fourth line specifies the performance using the same model but tested on Cat2 data.

The results use the best performing KnowComb system. They show that both Type 1 and Type 2 schema knowledge have higher precision on Category 1 and Category 2 data instances, respectively, compared to that on full data. Type 1 and Type 2 knowledge have similar performance on full data, but the results show that it is harder to solve instances in category 2 than those in category 1. Also, the performance drop between Cat1/Cat2 and full data indicates that there is a need to design more complicated knowledge schemas and to refine the knowledge acquisition for further performance improvement.

### 6 Related Work

**Winograd Schema:** Winograd (1972) showed that small changes in context could completely change coreference decisions. Levesque et al. (2011) proposed to assemble a set of sentences which comply with Winograd’s schema. Specifically, there are pairs of sentences which are identical except for minor differences which lead to different references of the same pronoun in both sentences. These references can be easily solved by humans, but are hard, he claimed, for computer programs.

**Anaphora Resolution:** There has been a lot of work on anaphora resolution in the past two decades. Many of the early rule-based systems like Hobbs (1978) and Lappin and Leass (1994) gained considerable popularity. The early designs were easy to understand and the rules were designed manually. With the development of machine learning based models (Connolly et al., 1994; Soon et al., 2001b; Ng and Cardie, 2002a), attention shifted to solving standard coreference resolution problems. However, many hard coreference problems involve pronouns. As Winograd’s schema shows, there is still a need for further investigation in this subarea.

**World Knowledge Acquisition:** Many tasks in NLP (such as Textual Entailment, Question Answering, etc.) require World Knowledge. Although there are many existing works on acquiring them (Schwartz and Gomez, 2009; Balasubramanian et al., 2013; Tandon et al., 2014), there is still no consensus on how to represent, gather and utilize high quality World Knowledge. When it comes to coreference resolution, there are a handful of works which either use web query information or apply alignment to an external knowledge base (Rahman and Ng, 2011b; Kobdani et al., 2011; Ratinov and Roth, 2012; Bansal and Klein, 2012; Zheng et al., 2013). With the introduction of Predicate Schema, our goal is to bring these different approaches together and provide a coherent view.

### Acknowledgments

The authors would like to thank Kai-Wei Chang, Alice Lai, Eric Horn and Stephen Mayhew for comments that helped to improve this work. This work is partly supported by NSF grant #SMA 12-09359 and by DARPA under agreement number FA8750-13-2-0008. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

### References


NIST. 2004. The ACE evaluation plan.


multilingual unrestricted coreference in OntoNotes. In Proceedings of the Annual Conference on Computational Natural Language Learning (CoNLL).


