Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Joint Inference over a Lightly Supervised Information Extraction Pipeline: Towards Event Coreference Resolution for Resource-Scarce Languages

Chen Chen and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{yzcchen,vince}@hlt.utdallas.edu

Abstract

We address two key challenges in end-to-end event coreference resolution research: (1) the *error propagation* problem, where an event coreference resolver has to assume as input the noisy outputs produced by its upstream components in the standard information extraction (IE) pipeline; and (2) the *data annotation* bottleneck, where manually annotating data for all the components in the IE pipeline is prohibitively expensive. This is the case in the vast majority of the world's natural languages, where such annotated resources are not readily available. To address these problems, we propose to perform joint inference over a lightly supervised IE pipeline, where all the models are trained using either active learning or unsupervised learning. Using our approach, only 25% of the training sentences in the Chinese portion of the ACE 2005 corpus need to be annotated with entity and event mentions in order for our event coreference resolver to surpass its fully supervised counterpart in performance.

1 Introduction

Event coreference resolution is the task of determining which event mentions in a text refer to the same real-world event. Compared to entity coreference, event coreference is not only much less studied but arguably more challenging.

Our goal in this paper is to examine two unaddressed, yet significant challenges in end-to-end event coreference research. First, event coreference resolvers suffer from the *error propagation* problem. Recall that for two event mentions to be coreferent, both their *triggers* (i.e., the words realizing the occurrence of the events) and their corresponding *arguments* (e.g., the times, places, and people involved in them) have to be compatible. However, identifying potential arguments (which are typically provided by an entity extraction system), linking arguments to their event mentions (which is typically performed by an event extraction system), and determining the compatibility between two event arguments (which is the job of an entity coreference resolver), are all non-trivial tasks. The errors made by any of these upstream components in the information extraction (IE) pipeline can propagate to the event coreference resolver. For instance, if the upstream event extraction system fails to extract certain event mentions, they will not be accessible to the event coreference resolver, thus harming its recall. Some previous work sidestepped this error propagation problem by evaluating their event coreference resolvers in a rather unrealistic setting (Chen, Ji, and Haralick 2009), where they assume that all but the event coreference system in the pipeline are implemented by an oracle.

Second, event coreference suffers from the *data annotation bottleneck*. To date, state-of-the-art end-to-end event coreference performance has been achieved by training not only the event coreference resolver but also the other systems in the IE pipeline in a *supervised* manner (Chen and Ng 2014). In other words, annotated data needs to be provided to train each of the systems in the pipeline. Unfortunately, such annotated data is available only for a handful of languages (e.g., English, Chinese, and Arabic, as provided by the ACE corpora). This data annotation bottleneck makes it challenging to apply existing event coreference resolution approaches to most of the world's natural languages.

To address the *error propagation* problem, we propose to perform *joint inference* via Integer Linear Programming (ILP) (Roth and Yih 2004) over the outputs of the models trained for the four key tasks in the IE pipeline, namely entity extraction, entity coreference, event extraction, and event coreference. To our knowledge, while there has been previous work on applying ILP to entity coreference (Denis and Baldridge 2007; Finkel and Manning 2008) and entity typing and coreference (Durrett and Klein 2014), this is the first attempt to perform joint inference for four key IE tasks.

To relieve the *data annotation bottleneck*, we reduce the amount of labeled data needed to train the systems in the IE pipeline by employing a combination of *unsupervised learning* and *active learning* (Cohn, Atlas, and Ladner 1994). The key idea is to divide the IE tasks in the pipeline into two groups: domain-independent tasks and domain-dependent tasks. Entity coreference and event coreference can be modeled as domain-independent processes once a set of entity/event mentions is given. Taking advantage of their domain-independent nature, we design generative models for these coreference tasks and train them in an unsupervised manner. On the other hand, the remaining tasks are domain-dependent. For instance, event extraction is by definition domain-dependent in the sense that the types/subtypes

of the events to be extracted and the semantic roles of their arguments need to be specified *a priori*. Given their domain-dependent nature, we train classifiers for these tasks in a supervised manner, but to reduce the amount of manual annotation effort, we train them on informative instances that are intelligently selected via active learning. Importantly, recall that while we employ active learning to select instances for training classifiers for entity and event extraction, our ultimate goal is to use as little data as possible to build state-of-the-art event coreference models. Hence, we propose to incorporate the feedback provided by our unsupervised event coreference model into the active learning process. This will allow us to select more informative entity and event extraction instances to train on, thus enhancing event coreference performance. This results in our *coreference-aware* active learning algorithm.[1]

Using our active-learning, joint-inference approach, only 25% of the training sentences in the Chinese portion of the ACE 2005 corpus need to be annotated with entity and event mentions in order for our event coreference resolver to surpass its fully supervised counterpart in performance. We believe that our work represents an important step towards building event coreference resolvers for languages for which annotated resources for IE tasks are not readily available.

2 Related Work

Almost all existing approaches to event coreference were developed for English. While early work (e.g., Humphreys, Gaizauskas, and Azzam (1997)) in MUC was limited to several scenarios, ACE takes a further step towards processing more fine-grained events. Most ACE event coreference resolvers are supervised (e.g., Ahn (2006), Chen and Ji (2009), McConky et al. (2012), Sangeetha and Arock (2012)).

There have also been attempts to evaluate event coreference resolvers on other corpora. For example, OntoNotes (Pradhan et al. 2007) was used by Chen et al. (2011); EventCorefBank (ECB) (Bejan and Harabagiu 2010) was used by Lee et al. (2012); ECB+ (Cybulska and Vossen 2014), an extension to ECB, was used by Yang, Cardie, and Frazier (2015); and the Intelligent Community (IC) corpus (Hovy et al.. 2013) was used by Cybulska and Vossen (2013), Goyal et al. (2013) and Araki et al. (2014). However, as noted by Liu et al. (2014), OntoNotes and ECB are only partially annotated with event coreference links. The IC corpus is not publicly available at the time of writing.

Compared to English event coreference, there has been much less work on Chinese event coreference. SinoCoreferencer (Chen and Ng 2014), a publicly-available ACE-style event coreference resolver for Chinese that achieves state-of-the-art results[2], employs a supervised approach where a pairwise classifier is trained to determine whether two event mentions are coreferent.

[1]Note that our coreference-aware active learner is *not* an active learner for coreference resolution (Laws, Heimerl, and Schütze 2012; Miller, Dligach, and Savova 2012). It is an active learner for entity and event extraction whose results are to be used for unsupervised event coreference resolution.

[2]www.hlt.utdallas.edu/~yzcchen/coreference/

(沙米里) 在 [离](家) 时，遭到 (歹徒)[暗杀]。这次 [攻击] 显示了 (叛乱分子) 能力。
(Shameri) was [assassinated] by (scoundrels) when [leaving] (home). The [attack] revealed (the insurgents)' ability.

Figure 1: Example of ACE IE.

3 ACE IE Tasks

Next, we give an overview of the four key IE tasks defined in ACE 2005. The ACE **event coreference** task involves performing coreference on the *event* mentions that belong to one of the 33 ACE event subtypes in a document. The ACE **entity coreference** task involves performing coreference on the *entity* mentions that belong to one of the 7 ACE entity types. The ACE **entity extraction** task involves identifying the entity mentions needed to perform entity coreference. Finally, the ACE **event extraction** task involves identifying the event mentions needed to perform event coreference. An event mention is composed of a *trigger* (i.e., the word realizing the event's occurrence) and a set of *arguments* (i.e., the event's participants). Each event trigger has a *subtype*, while each event argument has a *role*. Hence, identifying an event mention involves (1) **event trigger identification and subtype determination** (i.e., identifying and assigning a subtype to its trigger); and (2) **argument identification and role determination** (i.e., identifying and assigning a role to each of its arguments).

To better understand the ACE 2005 IE tasks, consider the sentence in Figure 1. An **entity extraction** system should determine that the four parenthesized phrases are entity mentions. An **entity coreference** resolver should determine that the two entity mentions (scoundrels) and (the insurgents) are coreferent. An **event extraction** system should identify three event mentions, whose trigger words are bracketed. Taking the last event mention as an example, a trigger identification and subtype determination system should determine that the trigger word is (attack) with subtype *Die*. After that, an argument identification and role determination system should identify (the insurgents) as the argument of this event mention with role *Agent*. Finally, an **event coreference** resolver should determine that the two event mentions triggered by (assassinated) and (attack) are coreferent.

4 Models for the IE Tasks

Recall from the introduction that to minimize reliance on annotated data, we design generative models for entity and event coreference resolution, training them in an unsupervised manner. On the other hand, we train classifiers for entity and event extraction in a supervised manner, reducing the annotation effort via active learning.

Coreference Models

We train three coreference models, one for pronoun resolution, one for name and nominal resolution, and one for event coreference resolution. The first two models assume as input a set of entity mentions and output an antecedent

for each anaphoric entity mention. The event coreference model essentially has the same goal, except that it operates on event mentions. All three models employ the same underlying generative process that we previously used for unsupervised event coreference resolution (Chen and Ng 2015), differing only w.r.t. the features used.

Our pronoun resolver employs nine features to represent the pronoun to be resolved, p, and one of its candidate antecedents, c, as described below:

Lexical (1): whether p and c are lexically identical.

Grammatical (5): whether p and c have the same grammatical role; whether they agree in number/gender/person/animacy.

Semantic (1): whether c is semantically compatible with p's governing verb.

Distance (2): the sentence distance between p and c; the number of intervening entity mentions.

Our name/nominal resolution model employs 12 features to represent the name/nominal to be resolved, n, and one of its candidate antecedents, c, as described below:

Lexical (5): whether n and c are lexically identical; whether they have the same head word; whether they have different modifiers; whether one mention is an abbreviation of the other; whether n contains all the words appearing in c.

Syntactic (3): whether n and c are in a copular construction; whether they have an appositive relationship; whether they are in an i-within-i construct.

Semantic (2): whether n and c have the same semantic type according to a Chinese lexical database[3]; whether they have the same entity subtype according to the entity extraction model described below.

Distance (2): the sentence distance between n and c; the number of intervening entity mentions.

Finally, our event coreference model employs six features to represent the event mention to be resolved, e, and one of its candidate antecedents, c, as described below:

Trigger-based (2): whether e and c's triggers have the same basic verb and compatible verb structures[4]; whether they are both nominal and are incompatible w.r.t. number.

Argument-based (3): whether e and c possess two arguments that have the same semantic role but different semantic classes; whether they possess two arguments that have the same semantic role but are not coreferent; whether they possess two named entity arguments that both have VALUE as their NE type but are lexically different.

Distance (1): the number of intervening event mentions.

Entity and Event Extraction Models

We train three classifiers, one for entity extraction and two for event extraction, using SVM^{multiclass} (Joachims, Finley, and Yu 2009).

The **entity extraction model** extracts and determines the type of each entity mention in a document. To train this classifier, we create one training instance for each noun n in the training data. If n heads an entity mention whose entity type belongs to one of the seven ACE entity types, then the class label is its entity type. Otherwise, we set the class label to *None*. Each instance is represented using 17 features, as described below:

Lexical (6): n's string; n's characters; characters in a window of five surrounding n.

Wordlist-based (10): We employ 10 Chinese wordlists to create 10 binary features, each of which encodes whether n appears in a wordlist. The 10 wordlists are (a) Chinese surnames; (b) famous geo-political entity (GPE) and location names (three wordlists); (c) Chinese location suffixes; (d) Chinese GPE suffixes; (e) famous international organization names; (f) famous company names; (g) famous person names; and (h) a list of pronouns.

Semantic (1): n's semantic category according to a Chinese lexical database.[3]

The **trigger identification and subtyping classifier** takes as input a candidate trigger word (i.e., a word that survives Li et al.'s (2012) filtering rules) and outputs its event subtype (if it is a true trigger) or *None* (if it is not a trigger). To train this classifier, we create one training instance for each word w in each training document. If w does not correspond to a trigger, the class label of the corresponding instance is *None*. Otherwise, the class label is the subtype of the trigger. Each instance is represented by 26 features, as described below:

Lexical (7): w's string; w's characters; w's part-of-speech (POS) tag; word and POS bigrams formed from w and its preceding or following word.

Syntactic (6): the depth of w's node in its syntactic parse tree; the path from w's node to the root of the tree; the phrase type of w's grandparent node and its phrase structure; the path from w's node to its governing IP node; whether a zero pronoun (ZP) precedes w.

Semantic (7): whether w exists in the list of predicates extracted from the Chinese PropBank (Xue and Palmer 2005); w's entry number in a Chinese lexical database[3]; whether w is a predicate according to a semantic role labeler[5]; if yes, the entity type and subtype of its Arg0 and Arg1.

Nearest entity (6): the entity type of the entity syntactically/textually closest to w; the entity types of the left and right entities syntactically/textually closest to w.

The **argument identification and role determination classifier** takes as input a *candidate argument link* a_{en}, linking (1) a candidate event mention e (i.e., an event mention whose trigger is identified by the aforementioned trigger identification and subtyping classifier), and (2) a candidate argument n of e. It outputs the role of n w.r.t. e (if n is a true argument of e) or *None* (if n is not an argument of e). To train this classifier, we create training instances by pairing each true event mention e (i.e., event mention consisting of a true trigger) with each of e's candidate arguments. If the candidate argument n is indeed a true argument of e, the class label of the training instance is the argument's role. Otherwise, its class label is *None*. Each instance is represented by 19 features, as described below:

[3]The dictionary is available from the Harbin Institute of Technology NLP Group's website.

[4]See Chen and Ng (2015) for details on the definitions of basic verbs and verb structure compatibility.

[5]https://code.google.com/p/mate-tools/

Basic (6): the POS of e's trigger; e's event subtype; n's entity type; n's head word; e's event subtype + n's head word; e's event subtype + n's entity subtype.
Neighbors (6): the word to the left/right of n; the word to the left/right of n + the word's POS; the word to the left/right of e's trigger + the word's POS.
Syntactic (4): the phrase structure obtained by expanding the grandparent of e's trigger in its syntactic parse tree; the shortest path from n's head word to e's trigger; the length of this shortest path.
Positional (1): whether e appears before or after n.
Dependency-based (1): the dependency path from n's head word to e's trigger.
ZP-based (2): whether a ZP z precedes e's trigger; if so, whether n is coreferent with z.

5 Coreference-Aware Active Learning

Recall from the above that annotated data is needed to train the three entity and event extraction models. To reduce annotation effort, we employ active learning to intelligently select data for manual annotation.

Our active learning algorithm selects the most informative *sentences* to annotate. Since the annotations are only used to train the entity and event extraction models, when a sentence s is selected for manual annotation, only the event mentions and the entity mentions in s will be annotated.[6] In particular, we will *not* annotate s with any entity coreference and event coreference information, as we opted to use our unsupervised generative models to produce entity and event coreference chains. As mentioned before, since our ultimate goal is to perform event coreference, our active learning algorithm seeks to select sentences whose annotated entity and event mentions will likely improve event coreference performance. To this end, we propose a *coreference-aware* active learning algorithm, which exploits the information provided by the event coreference model in the learning process.

Our active learning algorithm is shown in Figure 2. In each iteration, it (1) retrains the entity and event extraction models on the sentences annotated so far; (2) retrains the unsupervised entity and event coreference models using the entity and event mentions extracted by the entity and event extraction models; (3) applies all these models to the unlabeled documents; and (4) selects the sentences for manual annotation based on the outputs of the event coreference model and the event trigger classifier.

Before describing our sentence selection algorithm, recall that our active learning algorithm takes as one of its inputs a set of seed sentences $\mathcal{L}$ that are annotated with entity and event mentions. The simplest way to create $\mathcal{L}$ is to select these seeds randomly. However, we hypothesize that event mentions whose triggers appear more frequently tend to have a higher impact on improving event coreference, and as a result, we prefer selecting as seeds those sentences containing these frequently occurring triggers. Specifically, we

[6]When annotating an event mention, we identify its trigger and its arguments, and annotate its event type/subtype and the role of each of its arguments. When annotating an entity mention, we annotate its entity type.

Input:
- $\mathcal{D}$, the set of all documents in the training data
- $\mathcal{L}$, the set of all labeled sentences in the docs in $\mathcal{D}$
- $\mathcal{U}$, the set of all unlabled sentences in the docs in $\mathcal{D}$

Output:
- $\mathcal{L}$, the set of all labeled sentences in the docs in $\mathcal{D}$

Repeat:
1. Train entity extraction model SVM_{entity} on $\mathcal{L}$
2. Apply SVM_{entity} to extract entity mentions from $\mathcal{U}$
3. For each document D_i, train the unsupervised entity coreference resolution models on the extracted entity mentions and apply them to create entity coreference chains
4. Train event trigger classifier $SVM_{trigger}$ and event argument classifier SVM_{arg} on $\mathcal{L}$
5. Apply $SVM_{trigger}$ and SVM_{arg} to extract event triggers and event arguments from $\mathcal{U}$ respectively
6. For each document D_i, train unsupervised event coreference model $Coref_{event}$ on the extracted event mentions and apply it to create event coreference chains
7. Select the sentences $\mathcal{A}$ from $\mathcal{U}$ based on the outputs of $Coref_{event}$ and $SVM_{trigger}$
8. Annotate sentences in $\mathcal{A}$ manually
9. $\mathcal{L} = \mathcal{L} \cup \mathcal{A}, \mathcal{U} = \mathcal{U} - \mathcal{A}$

Until: a stopping criterion is reached

Figure 2: Our active learning algorithm.

create $\mathcal{L}$, which is initially empty, by incrementally populating it with annotated sentences, as follows. Observing that the part-of-speech (POS) tags of most trigger words are NN, VV or P, we first collect the list of words with these POS tags and then sort them in decreasing order of frequency of occurrence in the unlabeled set $\mathcal{U}$. We process each word w in the sorted list as follows. If w never appears in $\mathcal{L}$, we randomly select and annotate two unlabeled sentences containing w and add them to $\mathcal{L}$; if w only appears once in $\mathcal{L}$, we randomly select and annotate one unlabeled sentence containing w and add it to $\mathcal{L}$; otherwise, we do nothing. We repeat the above steps until $\mathcal{L}$ is populated with 200 sentences.

Figure 3 shows our sentence selection algorithm, which proceeds as follows. $\mathcal{A}$, which is initially empty, will be incrementally populated with the selected sentences. First, it divides the set of candidate event triggers (i.e., event triggers annotated by the trigger classifier) into two subsets: $\mathcal{T}_1$ contains those that are involved in event coreference chains, whereas $\mathcal{T}_2$ contains those that are singletons (i.e., not involved in any event coreference chains). It considers selecting sentences containing triggers in $\mathcal{T}_1$ before considering selecting any of the sentences containing triggers in $\mathcal{T}_2$. The reason is that annotating the triggers in $\mathcal{T}_1$ would have a larger impact on event coreference performance: by annotating the event mentions in $\mathcal{T}_1$, we can remove the already-established event coreference links between those candidate event mentions that turn out *not* to be true event mentions, thus potentially improving event coreference precision. Compared to the triggers in $\mathcal{T}_1$, those in $\mathcal{T}_2$ have a lesser impact on event coreference performance: in virtually all commonly-used coreference scoring programs, all sin-

Input:
- $\mathcal{L}$, the set of all labeled sentences in the docs in $\mathcal{D}$
- $\mathcal{U}$, the set of all unlabled sentences in the docs in $\mathcal{D}$
- $\mathcal{C}$, the set of all event coreference clusters in the docs in $\mathcal{D}$
- $\mathcal{T} = \{t_{ijk}\}$, the set of all candidate event triggers in $\mathcal{U}$ (t_{ijk} is the kth candidate trigger in the jth sentence of the ith training document)
-n, batch size (number of sentences to be selected)

Output:
- $\mathcal{A}$, the set of selected sentences

Initialize:
- $\mathcal{A} = \{\}$

Steps:
1. Divide $\mathcal{T}$ into two disjoint sets $\mathcal{T}_1$ and $\mathcal{T}_2$ based on $\mathcal{C}$, where $\mathcal{T}_1$ contains those candidate event triggers of non-singleton event mentions, and $\mathcal{T}_2$ contains those candidate event triggers of singleton event mentions
2. Sort $\mathcal{T}_1$ and $\mathcal{T}_2$ in ascending order of the confidence value returned by $SVM_{trigger}$
3. For every candidate trigger t_{ijk} in $\mathcal{T}_1$,
   unless (a) the size of $\mathcal{A}$ equals n; (b) the lexical string of t_{ijk} has appeared in the sentences in $\mathcal{L}$ more than twice; or (c) the sentence s_{ij} containing t_{ijk} has already been selected,
      add the sentence containing t_{ijk} to $\mathcal{A}$
4. For every candidate trigger t_{ijk} in $\mathcal{T}_2$,
   unless (a) the size of $\mathcal{A}$ equals n; (b) the lexical string of t_{ijk} appears in the sentences in $\mathcal{L}$ more than twice; or (c) the sentence containing t_{ijk} has been selected,
      add the sentence containing t_{ijk} to $\mathcal{A}$

Figure 3: Sentence selection algorithm.

gleton event mentions are removed before the scorers are applied. In fact, the sentences containing triggers in $\mathcal{T}_2$ are considered only if the batch size (i.e., the number of sentences to be selected) has not been reached after all the sentences containing triggers in $\mathcal{T}_1$ are considered. It is this preference that makes our sentence selection algorithm coreference-aware.

In addition, the algorithm prefers selecting sentences containing the *low-confidence* triggers in $\mathcal{T}_1$ according to the event trigger classifier (i.e., candidate triggers that are closest to the SVM hyperplane), unless (1) the low-confidence trigger has been annotated at least twice in $\mathcal{L}$ or (2) the sentence containing the low-confidence trigger has previously been selected. The algorithm employs the same preference when selecting sentences from $\mathcal{T}_2$.

6 Applying Joint Inference

In this section, we explain how we employ joint inference to address the error propagation problem.

Notation

Let D be a test document. Let E be the set consisting of the $|E|$ candidate event mentions in D, $e_1, ...e_{|E|}$, which are sorted by the order in which they appear in D. We use $EP(D)$ to denote the set of $\binom{|E|}{2}$ candidate event mention pairs, i.e., $EP(D) = \{(e_i, e_j)|1 \leq i < j \leq |E|\}$. Similarly, N is defined as the set consisting of the $|N|$ entity mentions in D, $n_1, ...n_{|N|}$, which are sorted by the order in which they appear in D. We use $NP(D)$ to denote the set of $\binom{|N|}{2}$ entity mention pairs, i.e., $NP(D) = \{(n_i, n_j)|1 \leq i < j \leq |N|\}$. Also, we use a_{ij} to denote the candidate argument link connecting candidate event mention e_i and entity mention n_j, $Role(a_{ij})$ to denote the role played by n_j in e_i, and A to denote the set of all candidate argument links in D.

We define T as the 34-element set consisting of the 33 ACE event subtypes and *None*, and t as a subtype in T. Similarly, we define R as the 36-element set consisting of the 35 ACE argument roles and *None*, and r as a role in R. Also, we define S as the 8-element set consisting of the seven ACE entity types and *None*, and s as a type in S.

Also, we define five terms. $P_{ec}(e_i, e_j)$ is the event coreference classifier's confidence that two candidate event mentions e_i and e_j are coreferent. $P_{nc}(n_i, n_j)$ is the entity coreference classifier's confidence that two candidate entity mentions n_i and n_j are coreferent. $P_{et}(e, t)$ is the trigger identification and subtyping classifier's confidence that the candidate event mention e has subtype t. $P_{ns}(n, s)$ is the entity extraction model's confidence that the candidate entity mention n has entity type s. $P_{ar}(a, r)$ is the argument identification and role determination classifier's confidence that the candidate argument link a involves role r. Since the last three classifiers are trained using SVM^{multiclass}, we normalize their confidence values to $[0, 1]$.[7]

Finally, we define binary indicator variables whose values are to be determined by an ILP solver. Specifically, the variable $x(e_i, e_j)$ takes on the value 1 if and only if the solver posits e_i and e_j as coreferent. Similarly, the variable $u(n_i, n_j)$ has the value 1 if and only if the solver posits n_i and n_j as coreferent. The variable $y(e, t)$ has the value 1 if and only if the solver determines that e has subtype t. The variable $v(n, s)$ has the value 1 if and only if the solver determines that n has type s. The variable $z(a, r)$ has the value 1 if and only if the solver decides that a has role r.

ILP Formulation

For each test document D, we create one ILP program. Its objective function is a linear combination of the five confidence terms from the five classifiers in Section 4:

$$\begin{aligned}\max \quad & \sum_{(e_i,e_j)\in EP(D)} P_{ec}(e_i, e_j)x(e_i, e_j) + \\ & \sum_{(n_i,n_j)\in NP(D)} P_{nc}(n_i, n_j)u(n_i, n_j) + \\ & \sum_{e\in E}\sum_{t\in T} P_{et}(e, t)y(e, t) + \sum_{n\in N}\sum_{s\in S} P_{ns}(n, s)v(n, s) + \\ & \sum_{a\in A}\sum_{r\in R} P_{ar}(a, r)z(a, r)\end{aligned} \tag{1}$$

subject to:

$$x(e_i, e_j) \in \{0, 1\}, \forall (e_i, e_j) \in EP(D) \tag{2}$$

$$u(n_i, n_j) \in \{0, 1\}, \forall (n_i, n_j) \in NP(D) \tag{3}$$

$$y(e, t) \in \{0, 1\}, \forall e \in E, \forall t \in T \tag{4}$$

[7] For each test instance, we first apply a sigmoid function to the confidence value assigned by the classifier to each class and then rescale the resulting values so that they sum to 1.

$$v(n,s) \in \{0,1\}, \forall n \in N, \forall s \in S \quad (5)$$

$$z(a,r) \in \{0,1\}, \forall a \in A, \forall r \in R \quad (6)$$

$$\sum_{t \in T} y(e,t) = 1, \forall e \in E \quad (7)$$

$$\sum_{s \in S} v(n,s) = 1, \forall n \in N; \sum_{r \in R} z(a,r) = 1, \forall a \in A \quad (8)$$

Equations (2)−(8) are integrity constraints. Equations (2), (3), (4), (5) and (6) ensure that $x(e_i, e_j)$, $u(n_i, n_j)$, $y(e,t)$, $v(n,s)$ and $z(a,r)$ are binary values. Equations (7) and (8) ensure that each candidate event mention e has exactly one subtype, each candidate entity mention n has exactly one type, and each candidate argument link a involves exactly one role.

Next, we define constraints that enforce the consistency among the outputs of the five classifiers.

Constraint 1. Intuitively, two event mentions e_i and e_j are not coreferent if they possess two non-coreferent arguments that have the same role. To implement this constraint, we rely on the following definition. We say that two true argument links a_{ik} and a_{jl} are *incompatible* if $Role(a_{ik}) = Role(a_{jl}) \neq$ *None* and their underlying entity mentions n_k and n_l are not coreferent.

To implement this as linear constraints, we first define an auxiliary binary variable $c(a_{ik}, a_{jl}, r)$, which takes on the value 0 if and only if a_{ik} and a_{jl} are incompatible and $Role(a_{ik}) = Role(a_{jl}) = r \neq$ *None*[8], as shown below:

$$\begin{aligned} &3 * c(a_{ik}, a_{jl}, r) + z(a_{ik}, r) + z(a_{jl}, r) - u(n_k, n_l) \geq 2 \\ &\forall r \neq None, (a_{ik}, a_{jl}) \in V \end{aligned} \quad (9)$$

$$\begin{aligned} &c(a_{ik}, a_{jl}, r) + z(a_{ik}, r) + z(a_{jl}, r) - u(n_k, n_l) \leq 2, \\ &\forall r \neq None, (a_{ik}, a_{jl}) \in V \end{aligned} \quad (10)$$

where $V = \{(a_{ik}, a_{jl}) | A \times A, i < j\}$.

Now, we can implement Constraint 1 as follows:

$$\begin{aligned} &c(a_{ik}, a_{jl}, r) \geq x(e_i, e_j), \forall a_{ik} \in Arg(e_i), \forall a_{jl} \in Arg(e_j), \\ &\forall r \neq None, \forall (e_i, e_j) \in EP(D) \end{aligned} \quad (11)$$

Constraint 2. If candidate event mention e_i is coreferent with another event mention, then its event subtype cannot be *None*. This constraint enforces the consistency between the output of the event coreference classifier and that of the trigger identification and subtyping classifier. This constraint can be implemented as follows:

$$\begin{aligned} &x(e_i, e_j) + y(e_i, None) \leq 1, \forall (e_i, e_j) \in EP(D) \\ &x(e_k, e_i) + y(e_i, None) \leq 1, \forall (e_k, e_i) \in EP(D) \end{aligned} \quad (12)$$

Similarly, if candidate entity mention n_i is coreferent with another entity mention, then its entity type cannot be *None*. This constraint can be implemented as follows:

$$\begin{aligned} &u(n_i, n_j) + v(n_i, None) \leq 1, \forall (n_i, n_j) \in NP(D) \\ &u(n_k, n_i) + v(n_i, None) \leq 1, \forall (n_k, n_i) \in NP(D) \end{aligned} \quad (13)$$

Constraint 3. If two candidate event mentions e_i and e_j are coreferent, then they must have the same event subtype. This constraint also enforces the consistency between the output of the event coreference classifier and that of the trigger identification and subtyping classifier. This can be implemented as the following linear constraints:

$$\begin{aligned} &x(e_i, e_j) + y(e_i, t) - y(e_j, t) \leq 1, \forall (e_i, e_j) \in EP(D), \forall t \in T \\ &x(e_i, e_j) + y(e_j, t) - y(e_i, t) \leq 1, \forall (e_i, e_j) \in EP(D), \forall t \in T \end{aligned} \quad (14)$$

Similarly, if two candidate entity mentions n_i and n_j are coreferent, then they must have the same entity type. It can be implemented as the following linear constraints:

$$\begin{aligned} &u(n_i, n_j) + v(n_i, s) - v(n_j, s) \leq 1, \\ &\forall (n_i, n_j) \in NP(D), \forall s \in S \\ &u(n_i, n_j) + v(n_j, s) - v(n_i, s) \leq 1, \\ &\forall (n_i, n_j) \in NP(D), \forall s \in S \end{aligned} \quad (15)$$

Constraint 4. If two candidate event mentions e_i and e_j in a given document have the same trigger word, then they must have the same event subtype. This is the well-known "one sense per discourse" constraint, which enforces the consistency between different candidate event mentions with the same trigger word, and can be implemented as follows:

$$y(e_i, t) - y(e_j, t) = 0, \forall t \in T, \forall (e_i, e_j) \in Q \quad (16)$$

where Q is the set of candidate event mention pairs that have the same trigger word. (i.e., $Q = \{(e_i, e_j) | (e_i, e_j) \in EP(D)$ and $Trigger(e_i) = Trigger(e_j)\}$).

Constraint 5. For each candidate event mention or entity mention, we should select for it *at most one* antecedent. Assuming the best-first antecedent selection strategy (Ng and Cardie 2002), this constraint can be implemented as follows:

$$\begin{aligned} &\sum_{i<j} x(e_i, e_j) \leq 1, \forall 1 \leq j \leq |E| \\ &\sum_{i<j} u(n_i, n_j) \leq 1, \forall 1 \leq j \leq |N| \end{aligned} \quad (17)$$

Equipped with the above constraints, an ILP solver can determine the values of $x(e_i, e_j)$, $u(n_i, n_j)$, $y(e,t)$, $v(n,s)$ and $z(a,r)$, from which we then infer our final results of all of our tasks. We solve each ILP program using $lpsolve$.[9]

7 Evaluation

Experimental Setup

Dataset. For evaluation, we employ the 633 Chinese documents of the ACE 2005 training corpus. Statistics on the dataset are shown in Table 1. We obtain our results via 5-fold cross-validation. In each fold experiment, we reserve one fold for testing, and use the sentences in the remaining four folds as unlabeled data for training the (unsupervised) coreference models and for active learning.

Documents	633	Sentences	9,967
Event Mentions	3,333	Event Coref. Chains	2,521
Entity Mentions	34,321	Entity Coref. Chains	15,414

Table 1: Statistics on the dataset.

[8] Without loss of generality, we assume $i < j$ and $k < l$.

[9] http://sourceforge.net/projects/lpsolve/

System	MUC			B^3			$CEAF_e$			BLANC			CoNLL
	R	P	F	R	P	F	R	P	F	R	P	F	F
SinoCoreferencer	42.7	38.3	40.4	41.5	34.7	37.8	39.9	39.2	39.5	28.1	23.7	25.7	39.2
Ours (Before ILP)	41.3	37.3	39.2	40.3	34.7	37.3	40.1	39.2	39.6	25.4	21.3	23.2	38.7
Ours (After ILP)	44.3	39.0	41.5	43.2	35.9	39.2	42.1	40.9	41.5	27.8	22.5	24.9	40.7

Table 2: Event coreference results.

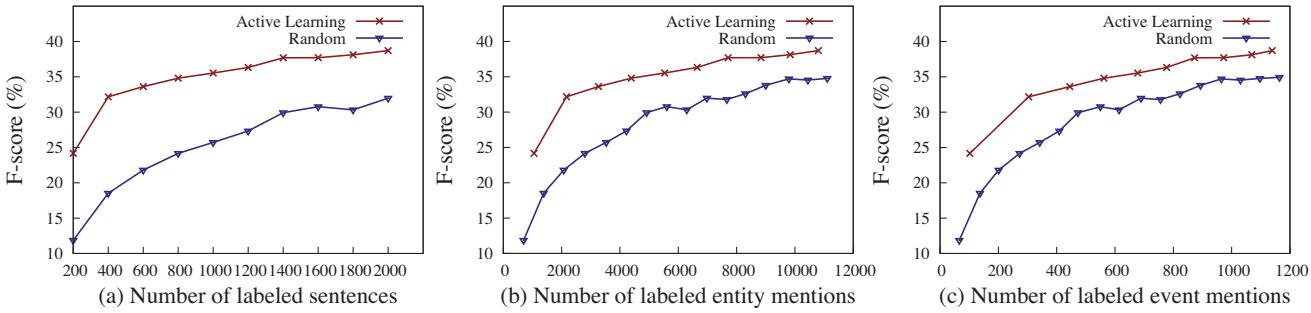

Figure 4: Event coreference performance expressed as the learning curves for active learning and random selection.

Evaluation measures. The event coreference tasks are evaluated using the commonly-used coreference evaluation measures, namely MUC, B^3, $CEAF_e$, and BLANC, all of which report results in terms of recall (R), precision (P), and F-score (F). In addition, we report the CoNLL score (Pradhan et al. 2011), which is the unweighted average of the MUC, B^3, and $CEAF_e$ F-scores. All the coreference results are obtained using the latest version (v8) of the CoNLL scorer (Pradhan et al. 2014). Following the official evaluation methodology employed in the CoNLL-2011 and 2012 shared tasks, we remove all singleton clusters from the outputs before scoring.

Evaluation setting. We run our active learner as follows. Starting from the initial set of the labeled data containing 200 sentences annotated with event mentions and entity mentions, in each iteration we augment the labeled data with 100 sentences annotated with event mentions and entity mentions.[10] Our active learner stops when the performance of our unsupervised event coreference model on the test set approaches that of a state-of-the-art fully supervised resolver, SinoCoreferencer, which is trained on all of the available training documents. After active learning, we perform ILP-based inference on the outputs of the resulting models.

Results and Discussion

Results of the active learner. We employ a random selection baseline, which randomly selects 100 sentences to annotate in each iteration. The learning curves of our active learner and the random selection baseline are shown in Figure 4. Note that these curves show *event coreference* performance despite the fact that the selected sentences were annotated with event mentions and entity mentions. As we can see in Figure 4(a), our active learner achieves significantly better event coreference performance than the random selection baseline throughout.[11] After annotating 2,000 sentences (25% of the training sentences), our active learner achieves a CoNLL score of 38.7%, which is statistically indistinguishable from SinoCoreferencer's performance, while the random baseline only achieves a CoNLL score of 32.0%.

There is a caveat, however. The number of sentences annotated is not necessarily the same as the number of event/entity mentions annotated. In other words, the better performance achieved by our active learner could be attributed to the possibly larger number of event/entity mentions annotated in the selected sentences. To ensure a fair comparison, we also compare the two selection methods with respect to the number of event mentions (Figure 4(c)) and the number of entity mentions (Figure 4(b)) annotated so far. As we can see, our active learner still significantly outperforms the random selection method throughout in both figures.

Results of ILP-based inference. To address the error propagation problem, we apply ILP to the outputs of the models produced at the end of the active learning process. Results of event coreference after applying ILP are shown in row 3 of Table 2. For comparison, we show the results of SinoCoreferencer and our resolver (before applying ILP) in rows 1 and 2, respectively. As we can see, before applying ILP, our resolver is statistically indistinguishable from SinoCoreferencer w.r.t. the CoNLL score. Comparing rows 2 and 3, we can see that the use of ILP significantly improves the F-score of our resolver w.r.t. all scoring measures, achieving a CoNLL score of 40.7%. More importantly, with ILP our resolver significantly outperforms SinoCoreferencer. Note that ILP also improves other IE tasks: the results of entity extraction, entity coreference, trig-

[10] In preliminary experiments, we employed different batch sizes (10, 20, 50, 100, 150, and 200 sentences), but found that none of them yielded significantly better results than the others.

[11] All significance tests are paired t-tests, with $p < 0.05$.

ger identification and subtyping, and argument identification and role determination improve by 0.6% (F), 0.5% (CoNLL F), 1.3% (F), and 1.1% (F), respectively. These results suggest the effectiveness of our active-learning, joint-inference approach.[12]

8 Conclusion

We presented an active-learning, joint-inference approach to address two major challenges inherent in event coreference resolution: error propagation and the data annotation bottleneck. When used in tandem, active learning and joint inference enable us to surpass the performance of a state-of-the-art fully supervised event coreference resolver when only 25% of the sentences in the Chinese portion of the ACE 2005 corpus are annotated with entity and event mentions. We believe that our approach has the potential to significantly reduce the annotation effort needed to develop event coreference models for languages for which annotated resources for IE tasks are not readily available.

Acknowledgments

We thank the three anonymous reviewers for their comments. This work was supported in part by NSF Grant IIS-1219142.

References

Ahn, D. 2006. The stages of event extraction. *COLING/ACL Workshop on Annotating and Reasoning about Time and Events*.

Araki, J.; Liu, Z.; Hovy, E.; and Mitamura, T. 2014. Detecting subevent structure for event coreference resolution. *LREC*.

Bejan, C., and Harabagiu, S. 2010. Unsupervised event coreference resolution with rich linguistic features. *ACL*.

Chen, Z., and Ji, H. 2009. Graph-based event coreference resolution. *TextGraphs-4*.

Chen, C., and Ng, V. 2014. SinoCoreferencer: An end-to-end Chinese event coreference resolver. *LREC*.

Chen, C., and Ng, V. 2015. Chinese event coreference resolution: An unsupervised probabilistic model rivaling supervised resolvers. *NAACL HLT*.

Chen, B.; Su, J.; Pan, S. J.; and Tan, C. L. 2011. A unified event coreference resolution by integrating multiple resolvers. *IJCNLP*.

Chen, Z.; Ji, H.; and Haralick, R. 2009. A pairwise event coreference model, feature impact and evaluation for event coreference resolution. *RANLP Workshop on Events in Emerging Text Types*.

Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *Machine Learning* 15(2):201–221.

Cybulska, A., and Vossen, P. 2013. Semantic relations between events and their time, locations and participants for event coreference resolution. *RANLP*.

Cybulska, A., and Vossen, P. 2014. Guidelines for ECB+ annotation of events and their coreference. Technical report, NWR-2014-1, VU University Amsterdam.

[12]To test the utility of ILP independently of active learning, we retrain all the models on *all* of the available training data and apply the same set of ILP constraints to their outputs on the test set. Results show that ILP improves event coreference performance by 5.6% absolute CoNLL score.

Denis, P.; and Baldridge, J. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. *NAACL HLT*.

Durrett, G., and Klein, D. 2014. A joint model for entity analysis: Coreference, typing, and linking. *Transactions of the ACL* 2:477–490.

Finkel, J. R., and Manning, C. D. 2008. Enforcing transitivity in coreference resolution. *ACL*.

Goyal, K.; Jauhar, S. K.; Li, H.; Sachan, M.; Srivastava, S.; and Hovy, E. 2013. A structured distributional semantic model for event co-reference. *ACL*.

Hovy, E.; Mitamura, T.; Verdejo, F.; Araki, J.; and Philpot, A. 2013. Events are not simple: Identity, non-identity, and quasi-identity. *NAACL HLT Workshop on Events*.

Humphreys, K.; Gaizauskas, R.; and Azzam, S. 1997. Event coreference for information extraction. *ACL/EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*.

Joachims, T.; Finley, T.; and Yu, C.-N. J. 2009. Cutting-plane training of structural SVMs. *Machine Learning* 77(1):27–59.

Laws, F.; Heimerl, F.; and Schütze, H. 2012. Active learning for coreference resolution. *NAACL HLT*.

Lee, H.; Recasens, M.; Chang, A.; Surdeanu, M.; and Jurafsky, D. 2012. Joint entity and event coreference resolution across documents. *EMNLP-CoNLL*.

Li, P.; Zhou, G.; Zhu, Q.; and Hou, L. 2012. Employing compositional semantics and discourse consistency in Chinese event extraction. *EMNLP-CoNLL*.

Liu, Z.; Araki, J.; Hovy, E.; and Mitamura, T. 2014. Supervised within-document event coreference using information propagation. *LREC*.

McConky, K.; Nagi, R.; Sudit, M.; and Hughes, W. 2012. Improving event co-reference by context extraction and dynamic feature weighting. *CogSIMA*.

Miller, T. A.; Dligach, D.; and Savova, G. K. 2012. Active learning for coreference resolution. *BioNLP*.

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

Pradhan, S.; Ramshaw, L.; Weischedel, R.; MacBride, J.; and Micciulla, L. 2007. Unrestricted coreference: Identifying entities and events in OntoNotes. *ICSC*.

Pradhan, S.; Ramshaw, L.; Marcus, M.; Palmer, M.; Weischedel, R.; and Xue, N. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. *CoNLL: Shared Task*.

Pradhan, S.; Luo, X.; Recasens, M.; Hovy, E.; Ng, V.; and Strube, M. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. *ACL*.

Roth, D., and Yih, W.-T. 2004. A linear programming formulation for global inference in natural language tasks. *CoNLL*.

Sangeetha, S., and Arock, M. 2012. Event coreference resolution using mincut based graph clustering. *International Journal of Computing and Information Sciences.*

Xue, N., and Palmer, M. 2005. Automatic semantic role labeling for Chinese verbs. *IJCAI*.

Yang, B.; Cardie, C.; and Frazier, P. 2015. A hierarchical distance-dependent Bayesian model for event coreference resolution. *Transactions of the ACL* 3:517–528.