Incremental Global Event Extraction

Alex Judea and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH
Schloss-Wolfsbrunnenweg 35
69118 Heidelberg, Germany
(alex.judea|michael.strube)@h-its.org

Abstract

Event extraction is a difficult information extraction task. Li et al. (2014) explore the benefits of modeling event extraction and two related tasks, entity mention and relation extraction, jointly. This joint system achieves state-of-the-art performance in all tasks. However, as a system operating only at the sentence level, it misses valuable information from other parts of the document. In this paper, we present an incremental approach to make the global context of the entire document available to the intra-sentential, state-of-the-art event extractor. We show that our method robustly increases performance on two datasets, namely ACE 2005 and TAC 2015.

1 Introduction

*But the **strikes** prove controversial.*

In many cases, it is not sufficient to look at one sentence when extracting events. In our example, *strikes* has no potential event arguments, and the context is not sufficient to disambiguate it correctly: Is it the trigger of an ATTACK event because it actually means *bomb strikes*? Or is it not a trigger at all because it refers to the industrial action?

*Soon after dawn on this fourth day, confirmation of the ship's first **strike** arrived . . . This is the first of an unknown number of **strikes** we'll conduct during our watch in "operation enduring freedom" . . . But the **strikes** prove controversial.*

Given the other sentences it is easier to infer that *strikes* is indeed the trigger of an ATTACK event. In this paper, we present a system that makes the global context of a document available to a state-of-the-art event extractor. Looking at a broader context also benefits argument detection. For example, within one document entities play coherent roles in different events. Consider the following text:

*Sam Waksal was **sentenced** to seven years and three months in federal prison . . . He's being released on his own cog in a sans before he's to **report** to jail.*

Looking only at individual sentences, it is hard to predict that *report* triggers an ARREST-JAIL event, which in turn makes it hard to predict that *he* is the Person of this event. If the system looks at the entire document and knows that *he* and *Sam Waksal* are coreferent, it can better infer that the Defendant of a SENTENCE event can be the Person of an ARREST-JAIL event, which in turn makes it easier to infer that *he* is this person.

In this paper, we present a method to incorporate the global, document-wide context into the decision process of a system that predicts entity mentions, events, and relations jointly. We use features that are based on the 'one sense per discourse' assumption (Gale et al., 1992), a concept widely used in Word Sense Disambiguation (e.g., Navigli and Lapata (2007)), and on the coherence of roles an entity plays in different events. We show that our method robustly increases performance on two datasets, namely ACE 2005 and TAC 2015.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2279–2289, Osaka, Japan, December 11-17 2016.

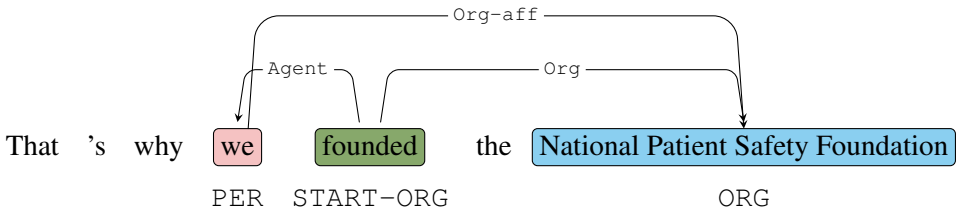

Figure 1: The configuration of a sentence. Depicted are two entities, a trigger, and the semantic relations and event argument relations between them.

The paper is structured as follows. Section 2 describes the task in more detail. Section 3 puts our work in context of other approaches to event extraction. Section 4 describes an intra-sentential, state-of-the-art system introduced by Li et al. (2014). In Section 5 we present *Incremental Global Inference*, a multi-pass procedure that makes the global context of a document accessible to the joint decoding of our intra-sentential event extractor. Section 6 reports evaluation results and Section 7 gives conclusions.

2 Task Description

Event extraction is an information extraction task where mentions of predefined event types are extracted from texts. We follow the task definition of the Automatic Content Extraction (ACE) program of 2005 which defines 33 event types, organized in eight categories. We also evaluate on another dataset, namely on the Event Nugget data of TAC 2015 (Mitamura et al., 2015).

In ACE, events are annotated only intra-sentential. Each event type has roles, e.g., `START-ORG`, an event indicating the founding of an organization, has the roles `Agent` and `Org`, whereas `DIE`, an event indicating the death of a person, has the roles `Agent`, `Victim`, and `Instrument`. The roles `Place` and `Time` are shared by all event types.

Roles are filled by zero or more *arguments*, that is, spans of text. The same span of text may be shared by multiple events as arguments, and may fill different roles in each of them. Finally, every event is indicated by a *trigger*.

Besides event annotations, ACE provides annotations of entity mentions and semantic relations. Detecting entity mentions is a task strongly related to event extraction because most arguments are mentions of persons, locations, organizations, etc.[1] Semantic relations and event arguments are also related because they coincide often with the start or end points of arguments.

Consider Figure 1. Depicted is the sentence *That's why we founded the National Patient Safety Foundation*. We can find two entity mentions, namely *we* as a `PER` and *National Patient Safety Foundation* as an `ORG` mention. Furthermore, there is one trigger of a `START-ORG` event, namely *founded*. This event has two arguments, namely *we* filling the role `Agent`, and *National Patient Safety Foundation* filling the role `Org`. Finally, there is an `Org-aff` relation between *we* and *National Patient Safety Foundation*.

While ACE provides annotations for all tasks involved (entity mentions, event triggers, event arguments), the TAC 2015 Event Nugget data provides only annotations for event triggers. The trigger annotation schemes are similar, the two data sets share many event types, and events occur only intra-sentential. However, some event types in TAC are more fine-grained, e.g., `TRANSPORT` in ACE was split into `TRANSPORT-PERSON` and `TRANSPORT-ARTIFACT` in TAC.

3 Related Work

The base system we use is the one described in Li et al. (2014)[2]. To our knowledge it is the only system to predict entity mentions, relations, event triggers, and event arguments jointly. It achieves state-of-the-art performance in all four tasks.

Many approaches to event extraction, including our base system, do not cross sentence boundaries (Grishman et al., 2005; Ahn, 2006; Lu and Roth, 2012; Li et al., 2013; Li et al., 2014). Only a few

[1] Some arguments are mentions of points in time, amounts of money, etc.

[2] This system is a combination of the systems described in Li et al. (2013) and Li and Ji (2014).

approaches go beyond sentences (Liao and Grishman, 2010; Hong et al., 2011) or beyond documents (Ji and Grishman, 2008) in order to exploit richer contexts for the extraction of events.

Recently, three deep learning systems were proposed. Nguyen and Grishman (2015) use a Convolutional Neural Network (CNN) to detect event triggers. They achieve good trigger detection performance, but they do not tackle the full event task. Chen et al. (2015) propose a more complicated version of a CNN which is able to detect multiple arguments in addition to triggers. Their system is a pipeline system (it predicts triggers and arguments separately) and suffers from error propagation. Nguyen et al. (2016) combine the advantages of joint models, explicit feature engineering and neural networks. They propose a joint, bi-directional recurrent network which additionally uses the features in Li et al. (2013). However, both Chen et al. (2015) and Nguyen et al. (2016) rely on gold entity mentions for trigger prediction.

The cross-sentential systems proposed in Liao and Grishman (2010) and Yang and Mitchell (2016) are closest to ours. We will describe them in the following.

Liao and Grishman (2010) propose a pipeline system that performs easy-first global inference. Local classifiers find triggers and arguments based solely on local information. Confident decisions are collected and used to inform global trigger and argument classifiers. In the local, intra-sentential phase the system performs pattern matching to align the entity mention context of a trigger to some known patterns. Similar to our approach, confident decisions are collected during the intra-sentential pass and used to infer harder cases.

Yang and Mitchell (2016) propose a pipeline system with global inference. In contrast to most other approaches, it also predicts entity mentions. Yang and Mitchell (2016) first train two Conditional Random Fields to generate mention and trigger candidates for a document. They only keep the 50 best mention candidates, and the 10 best trigger candidates. Then, they train three classifiers to capture entity mentions, within-event structures, and event-event relations. Finally, they formalize an Integer Linear Program that, given the local classifiers and the (global) event-event classifier, produces a globally optimal solution, instead of refining locally-optimal solutions with global information.

In terms of label inference (assigning types to candidates), our approach lies between Liao and Grishman (2010) and Yang and Mitchell (2016). The former apply global inference after local inference, the latter model both jointly. We let local and global inference inform each other and iteratively refine both.

A major difference between Liao and Grishman (2010) one the one hand, and Yang and Mitchell (2016) and our approach on the other is that the first approach uses gold entity mentions, while the other two use predicted entity mentions. Comparing full inference, we note that Yang and Mitchell (2016) use a pipeline approach: They first tag sentences for potential entity mention and event trigger candidates, and apply label inference afterwards. In contrast, we model both jointly.

4 Intra-Sentential Event Detection

In this section we describe our reimplementation of the system presented in Li et al. (2014), a state-of-the-art event detector. It employs joint decoding of entity mentions, events, and relations in order to make use of a rich feature set, including features which capture interdependencies of different subtasks. It uses a structured perceptron with beam search to explore different segmentations of a sentence and possible connections between segments.

4.1 Terminology

Following Li et al. (2014) we will call a specific segment of a sentence a *node* and a relation connecting two nodes an *arc*. The entire set of nodes and arcs for a sentence will be called *configuration*.

Formally, a node n is a tuple (b, e, n_t) where b and e are begin and end offsets, and n_t is an entity, event, or 'null' type. We encode offsets as token indices. An arc is a tuple (e_1, e_2, a_t) with $e_1 < e_2$, where e_1 and e_2 are the end offsets of the non-overlapping nodes of the connection, and a_t is a semantic relation or event argument type. Because semantic relations are directed and nodes are ordered by offset, we have to mark relation direction by defining an 'inverse' version of each semantic relation type.

4.2 Decoding and Training

Our decoding (producing configurations for a given sentence) is built around the idea of *Semi-Markov Chains* (Sarawagi and Cohen, 2004). Here, in contrast to more traditional token sequence labelers like CRFs, decoding produces segments of arbitrary length, possibly spanning multiple tokens and is therefore able to use features characterizing entire segments, and not only individual tokens. In the following we describe the decoding and training procedures in detail. Algorithm 1 formalizes the procedure.

We start our description with the method GENERATE NODES in Algorithm 1. Given a sentence s with tokens $t_1 \dots t_m$, the procedure generates nodes of different types and lengths ending at each t_i. It starts with t_1 and generates nodes of length one with different types, e.g., $(1, 1, \texttt{PER})$ or $(1, 1, \texttt{START-ORG})$. The procedure moves on to the next token, t_2. Now, it generates segments of length one and length two, e.g., $(2, 2, \texttt{PER})$ or $(1, 2, \texttt{PER})$, and adds them to the previously constructed configurations such that nodes do not overlap and there are no gaps.

Algorithm 1:
BEAM SEARCH$(s = t_1 \dots t_m, y, z, \textit{training})$

```
1  beam b ← ∅
2  for i = 1 ... m do
3      b(i) ← GENERATE NODES(b, i)
4      b(i) ← top_z(b(i))
5      if training ∧ y_i ∉ b(i) then
6          UPDATE(b(i, 1), y_i)
7          exit
8      for j = i − 1 ... 0 do
9          b(i) ← GENERATE ARCS(b, j, i)
10         b(i) ← top_z(b_i)
11         if training ∧ y_i ∉ b(i) then
12             UPDATE(b(i, 1), y_i)
13             exit
14 if training ∧ y ≠ b(m, 1) then
15     UPDATE(b(m, 1), y)
16     exit
17 return b(m, 0)
```

Some configurations may now contain two nodes which means that the procedure can predict arcs (GENERATE ARCS in Algorithm 1). It generates a compatible arc for each node pair, but does not overwrite configurations without arcs. This ensures that not predicting an arc between any two nodes is always a valid hypothesis. We collect type restrictions for arcs from the training data. For example, it is not possible that Organization-Affiliation relations hold between person mentions. Such restrictions keep the search space smaller and make learning easier.

Input to Algorithm 1 is a sentence s, a gold configuration y, and the beam size z. The beam b is a ranked list of hypotheses (that is, configurations) ending at any token position in the sentence. If the procedure is not in training mode, or if it did not make any prediction errors, it returns the top hypothesis ending at the last token position, $b(m, 1)$ for a sentence with m tokens.

Ideally, one would enumerate all possible configurations for a sentence and pick the best one. However, such an exhaustive enumeration is infeasible because the number of configurations grows exponentially with the number of tokens. Following Li et al. (2014) we approximate the enumeration by using *beam search*. We first expand configurations with new nodes and keep only the best z configurations. These are then expanded with arcs. During this process we again keep only the best z configurations. After arc generation, we move to the next position.

In training mode the procedure updates feature weights as soon as y_i, a prefix of the gold configuration, is not predictable anymore because it is no longer part of the beam. This is called *early update*. Huang et al. (2012) proved that, in structured perceptrons, standard updates may lead to bad performance with inexact search strategies such as beam search because there may be updates which actually lower the score of the gold solution, thus leading the model in a wrong direction. Early updates prevent this problem at the expense of higher training time. Note that we exit the procedure when early updates happen. There are two positions where early updates occur, namely after node generation and during arc generation (Lines 6 and 12).

4.3 Features

The feature set of our base system is complex because each subtask requires its own rich feature set. We can divide features into two broad categories: Static features, which do not depend on previous decisions, and dynamic features, which depend on previous decisions. Table 1 contains a feature summary struc-

Static	Common	lexical information* (segment tokens, context, dependencies)
		brown clusters, gazetteer entries*
	Mentions	character suffixes of length 3 and 4*
	Triggers	possible FrameNet frames in case of verbs*
	Arguments	trigger type, mention and mention context
		trigger-mention dependency/constituency paths
	Relations	entity types, contexts, extent overlaps
		mention-mention dependency/constituency paths
		syntactico-semantic structures (Chan and Roth, 2011)
Dynamic	Common	(event or entity) type bigrams
		consistencies (same text=same type, coordinations, pronouns)
	Triggers	dependency path between trigger pairs
	Arguments	characterize mentions filling the same role in the same event
		characterize triggers sharing a mention
	Relations	characterize triangles like A-C, B-C
		characterize constructions where the parts tend to have same relation types, e.g., coordinations
	Joint Argument-Relation	relation types and overlapping argument types

Table 1: A description of the base system's feature set. Static features marked with * apply to the entire segment and not to each token in the segment.

tured in this way. All features are concatenated with the node or arc type under consideration. Argument features additionally come without types in order to characterize properties of arguments in general.

In order to simulate the feature set of a more traditional sequence labeler, the base system adds a BILU-tag ("B" for first token of a segment, "I" for an intermediate token, "L" for the last token, and "U" in case of length-1 segments) to each token feature of the node under consideration. If this node is an entity and the last node in the configuration is also an entity, it additionally adds the entity type of the last segment to token features, otherwise it adds a 'null' label.

5 Incremental Global Inference

Our base system is limited in two ways. It operates intra-sententially and is thus limited to the information in a single sentence. In some cases this may be sufficient to predict all entity mentions, events, and relations, in many others it is not. An approach operating on the document level has access to a broader context and may be able to resolve more difficult cases.

Even within a sentence the base system is limited to the left context of the token under consideration. The entire sentence is only available when decoding reaches the last token. At that point many decisions are not reversible anymore because of the approximate search procedure.

In the following we present Incremental Global Inference, a method to cope with both limitations. It makes the global, document-wide context available to the local decoding of our base system.

Algorithm 2:
INCREMENTAL GLOBAL INFERENCE (d)

```
1  decision map r = ∅
2  for i = 1 ... k do
3      for Sentence s ∈ document do
4          r ∪ SEARCH(s, y, z, training, r)
           // no updates
5  for Sentence s ∈ document do
6      r_s = SEARCH(s, y, z, training, r)
7      if error(r_s) then
           // updates
8          UPDATE(x*, y*)
9      r ∪ r_s
10 return r
```

5.1 Procedure

Incremental Global Inference is a multi-pass approach. We iterate several times through the document. The decisions in each iteration are input to the next iteration. We first perform standard decoding as described in Section 4 for all sentences in a document, and feed back the decisions in this step for a second pass. In the second pass we extract global features, that is, features measuring the similarity of a new node or arc to decisions made in the last iteration. We again keep the decisions for the entire document and continue with a third decoding pass, etc. Algorithm 2 formalizes the procedure.

Feature type	Applies to	Condition	Description
Local	n is trigger, m is entity	dependency graph matching known	graph connecting event type and lemmas of n, and entity type of m
			indicator
	n is argument, m is trigger	dependency graph matching known	graph connecting role of n, event type, and lemmas of m
			indicator
Global	n and m are triggers	always	The event types of n and m
		lemmas equal	The event types of n and m
		coreferent arguments	The event types of n and m
	n and m are arguments	coreferent arguments	the roles and event types of n and m

Table 2: New local and global features for a node n. Global features are for node pairs (n, m), where m may come from the entire document.

Input to Algorithm 2 is a document d, output is a structure r holding the configuration for each sentence in the document. The algorithm can be divided in two parts, namely *collection* and *final decoding*. We will now describe the two parts.

In the collection phase (Lines 2 - 4), the procedure iterates k times through the document. In each iteration, and for each sentence, it performs SEARCH, a slightly different version of Algorithm 1, the only difference being that feature updates are omitted. We perform updates in *final decoding*. The decoding decisions are recorded per sentence in the decision map r. The decision map r is updated such that the decisions for a sentence in one iteration are overwritten with the decisions for the same sentence in the next iteration. r is input for the next iteration.[3] With each iteration, and with each update of r, SEARCH has a more reliable view on mentions, triggers, and arguments in the entire document, helping it to find new nodes and arcs which would be hard to get otherwise.

After collection comes final decoding (Lines 5 - 8). Here, we again perform SEARCH. Now, we also perform weight updates as soon as they are necessary. The final output of the algorithm is the final sate of r.

5.2 New Features

In this subsection we describe the new features we use. First, we will describe new local features. Then, we will describe the global features needed for our Incremental Global Inference. Table 2 summarizes the features we describe in the following. The first column reports feature types (local or global), the second column reports class requirements (triggers or arguments), the third column reports conditions under which features fire, and the last column gives short descriptions.

Local Features

The first local feature we introduce to event extraction is based on so-called *hidden units* (Das et al., 2014). The hidden units of an event type are the exclusive triggers it has in the training data. Hidden units define a semantic space for their event types. Measuring semantic relations of the node under consideration with this space helps to find triggers never seen during training. For example, let the candidate trigger be *meeting* and the candidate event type be `MEET`. We have several hidden units of `MEET` sharing semantic relations with the predominant sense of *meeting*: *convention* and *summit* as WordNet hypernyms (Fellbaum, 1998), and *meet* as a morphological variant. We can draw features from the hidden units themselves and from the semantic relations involved.

The second local feature we introduce to event extraction is based on dependency graphs between triggers and arguments. We collect all these graphs in the training data and index them by the involved argument and entity types. For example, we collect all graphs between `MEET` events and their `Place` arguments. One such graph is the following: `MEET` $\xrightarrow{\text{nsubjpass}}$ set $\xleftarrow{\text{nmod:in}}$ `LOC/Place` , coming from sentences like *The meeting was set in Beijing*, or *The conference was set in New York*. The graph encodes that a

[3] Initially, the decision map is empty.

MEET event trigger is the passive subject of *set*, and a LOC entity/a Place argument is connected via the nominal modifier *in* to the same word.

Note that the base system also uses dependencies to characterize syntactic connections of triggers and arguments. We extend this feature type in two important ways. First, we normalize dependency graphs by entity types, event types, or roles. Second, we can utilize dependency graphs during trigger prediction. The base system is always limited to the left context. When it predicts a trigger, it misses arguments to the right of it. Our Incremental Global Inference makes the entire sequence of entity mentions/event arguments available, regardless of the actual decoding position. This increases the chance to find the MEET trigger in our example, which may in turn influence other decisions in the next decoding pass.

eventmeet trigger. However, Incremental Global Inference makes the entire mention sequence available, regardless of the actual decoding position, increasing the chance to find the MEET trigger in our example, which may in turn influence other argument or entity mention decisions in the next decoding pass.

Global Features

We now describe the global features we use for our Incremental Global Inference. The global feature function takes two arguments (n, m), where n is the argument or trigger under consideration, and m is a trigger or argument coming from the entire document.

We have three types of global features. The first applies to triggers and catches all event types present in the document. As Liao and Grishman (2010) show, certain event types often co-occur, e.g. ATTACK and DIE often co-occur in news documents.

Another feature type fires if two triggers have equal event types and equal lemmas[4]. This is based on the observation that one word tends to trigger the same event type within one document (Ji and Grishman, 2008). This is strongly related to the 'one sense per discourse' assumption (Gale et al., 1992).

The third feature type applies to triggers and arguments. It catches event types of events with coreferent arguments, and the roles entities play in events. In ACE, an entity usually plays the same role for events with the same type (which is again realated to the 'one sense per discourse' assumption) and plays coherent roles for events with different types (Liao and Grishman, 2010). For example, if an entity is the Target of an ATTACK event, it rarely becomes the Attacker, at least not within one document. However, the same entity may be the Victim of a DIE event, but never the Agent. In order to increase recall of our coreference resolution system, we regard all non-pronoun string matches as coreferent, as well as partial string matches if the involved entities are persons.

6 Evaluation

In the following, we describe the data sets and the evaluation metrics we used. We report evaluation numbers and discuss them.

6.1 Data and Evaluation Metrics

We devise evaluation on two data sets, namely on ACE 2005 and on the Event Nugget data of TAC 2015. For the TAC evaluation, we use the official training and test sets. We manually split the training set into 132 documents for training and 26 for development. The test set consists of 202 documents. We removed the lemmas 'it' and 'say' from decoding because they proved to be highly ambiguous (Reimers and Gurevych, 2015).

For the ACE evaluation, we adopt the evaluation setting of Li et al. (2014). We train on English documents and remove the two smallest and most informal parts of the data, 'conversational telephone speech' and 'Usenet newsgroups'. From the remaining 511 documents, 351 are used for training, 80 for development, and 80 for testing. For direct comparison we use the same data split as Li et al. (2014). We set the beam size to 8. For Incremental Global Inference we perform 3 iterations. After each training epoch, we measure performance on the development set (in terms of trigger and argument F_1) and use the model with the best overall performance for evaluation on the test set.

[4]In case of multi-word triggers, we take the lemma of each token.

	Triggers						Arguments					
	Identification			Classification			Identification			Classification		
System	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Baseline	**67.2**	62.2	64.6	**64.7**	59.9	62.2	70.7	36.6	48.2	57.3	29.7	39.1
IGI	67.0	**65.9**	**66.5**	64.2	**63.1**	**63.7**	**76.1**	**40.8**	**53.1**	**59.9**	**32.1**	**41.8**

(a) Trigger and argument performance. 'Identification' reports performance without trigger type/argument role assignment.

	ATTACK(206)			TRANSPORT(98)			DIE(49)			MEET(42)		
System	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Baseline	**72.1**	71.4	**71.7**	44.0	49.0	46.4	**64.6**	85.7	73.7	66.7	76.2	71.1
IGI	70.1	**71.8**	71.0	**48.1**	**53.1**	**50.5**	63.8	**89.8**	**74.6**	**70.2**	**78.6**	**74.2**

(b) Trigger classification performance for the four most frequent event types.

	Triggers			Arguments		
System	P	R	F_1	P	R	F_1
Li et al. (2014)	67.9	62.8	65.3	64.7	35.3	45.6
Li et al. (2014) rerun	66.4	60.2	63.1	61.3	30.2	40.4
Baseline	64.7	59.9	62.2	57.3	29.7	39.1

(c) Trigger and argument classification performance. This is a comparison of results published by Li et al. (2014), a version retrained by us ('rerun'), and our reimplementation of the system.

Table 3: Micro-averaged precision, recall, and F_1 for the baseline and IGI on the ACE 2005 test set, and for our baseline compared to published results in Li et al. (2014) and our run using their code. Numbers in bold are the better numbers for the respective measure. Numbers in parentheses are frequencies in the test set.

	Triggers					
	Identification			Classification		
System	P	R	F_1	P	R	F_1
Baseline	**87.3**	47.0	61.1	**73.3**	39.5	51.3
IGI	77.1	**54.7**	**64.0**	64.1	**45.5**	**53.3**

(a) Trigger performance. 'Identification' reports performance without event type assignment.

	ATTACK(591)			CONTACT(587)			TRANSF.MNY(554)			BROADCAST(510)		
System	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Baseline	**69.0**	38.4	49.3	**43.5**	**11.4**	**18.1**	**71.9**	29.1	41.4	**44.0**	20.8	28.2
IGI	67.3	**43.5**	**52.8**	33.5	11.1	16.6	55.4	**34.3**	**42.4**	42.5	**30.4**	**35.4**

(b) Trigger classification performance for the four most frequent event types.

Table 4: Micro-averaged precision, recall and F_1 for the baseline and IGI on the TAC 2015 test set. In TAC 2015 only event trigger annotations are available. Numbers in bold are the better numbers for the respective measure. Numbers in parentheses are frequencies in the test set.

We follow standard evaluation criteria for triggers and events (Ji and Grishman, 2008). A trigger is correct, if its span and event type match a reference trigger. An argument is correct, if its span, event type, and role match a reference argument.

Our evaluations report *identification* and *classification* of event triggers and arguments. An event trigger or argument is correctly identified if its offsets match a reference trigger or argument, and correctly classified, if there is an additional match in event type or role. In all evaluations, we report micro-averaged precision (P), recall (R), and F_1. We devise two evaluation lines: We evaluate trigger and argument identification and classification on the one hand, and trigger classification performance for the four most frequent event types in ACE 2005 and TAC 2015 on the other hand.

6.2 Experiments and Results

Table 3 reports evaluation results for triggers and arguments on ACE 2005. Comparing the baseline with IGI in Table 3a we can see that IGI gives a considerably higher recall and F_1 for triggers, and a higher precision, recall, and F_1 for arguments, both, in terms of identification and classification. For identification, IGI increases trigger performance by 1.9 F_1 points and argument performance by 4.9 F_1 points. For classification, IGI improves 1.5 F_1 points for triggers, and 2.7 F_1 points for arguments.

The same trends can be observed for TAC 2015 (Table 4a). TAC 2015 offers trigger annotations only. For IGI, trigger identification improves by 2.9 F_1 points, and classification improves by 2 F_1 points compared to the baseline.

We now look at the classification performance of the four most frequent event types for both, ACE 2005 (Table 3b) and TAC 2015 (Table 4b). On both data sets, IGI improves F_1 for three of the four most frequent event types. For ACE 2005, `Transport` and `Meet` have higher precision and recall compared to baseline performance. `Attack` loses precision with IGI, resulting in a similar F_1 as the baseline. For TAC 2015, `Contact` loses precision and recall with IGI, resulting in a lower F_1. In all other cases, IGI considerably increases recall, at the expense of precision. For ACE 2005, the event type with most F_1 improvement using IGI is `Transport` (+4.1 points), a very difficult event type. For TAC 2015, the event type with most F_1 improvement using IGI is `Broadcast`, again a very difficult event type.

Table 3c is as a comparison of our reimplementation of Li et al. (2014) and the numbers they report. The first column ('Li et al. (2014)') reports published results from the respective paper. Column 'Li et al. (2014) rerun' reports performance of the source code we received from the authors. 'Baseline' is our baseline, a reimplementation of their system. We first compare published results with our rerun of the reference system (Lines 1 and 2). We can see that the published evaluation numbers and the numbers for our rerun of the reference system (using source code we received from the authors) do not match. We can only speculate about the reasons. The biggest difference we can spot is that the source code we received does not use the output of a frame-semantic parser for FrameNet features. It uses all possible frames for a given word instead, if the number of possible frames does not exceed 2. Li et al. (2014) report a gain of 0.8 F_1 points for triggers, and a gain of 2.2 F_1 points for arguments when using a frame-semantic parser. Since the source code we received already uses frames, we expect to gain only a fraction of this improvements using a full-fledged frame-semantic parser. We believe the feature set has changed slightly until we received the source code. It is not easy to reproduce published results (Fokkens et al., 2013). Comparing our reimplementation with our rerun of the reference system (Lines 2 and 3) we can see that the respective performances are close.

7 Conclusions

We presented a method to make the global, document-wide context available to a state-of-the-art, intra-sentential event extractor. The method is an incremental approach: With every iteration it collects new triggers and arguments from the entire document and makes them available to the intra-sentential base system for another decoding pass. Our method leads to considerable increases in recall and F_1 for triggers across different data sets and different event types, and to considerable increases in all measures for arguments.

Lemma matches and coreferent arguments as nuclei for global decoding are only a start. Future research has to investigate other similarity measures based on, e.g., WordNet and word vectors, and to explore the benefits of discourse structure for event extraction.

Acknowledgments

This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a HITS PhD scholarship. We thank the anonymous reviewers for their helpful comments.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events,* Sydney, Australia, 23 July 2006, pages 1–8.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 551–560.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 167–176.

Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, Mass.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 1691–1701.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the DARPA Speech and Natural Language Workshop,* New York, N.Y., 23–26 February 1992, pages 233–237.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. Technical report, Department of Computer Science, New York University, New York, N.Y.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Portland, Oreg., 19–24 June 2011, pages 1127–1136.

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Montréal, Québec, Canada, 3–8 June 2012, pages 142–151.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies,* Columbus, Ohio, 15–20 June 2008, pages 254–262.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Baltimore, Md., 22–27 June 2014, pages 402–412.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Sofia, Bulgaria, 4–9 August 2013, pages 73–82.

Qi Li, Heng Ji, Yu Heng, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing,* Doha, Qatar, 25–29 October 2014, pages 1846–1851.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics,* Uppsala, Sweden, 11–16 July 2010, pages 789–797.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Jeju Island, Korea, 8–14 July 2012, pages 835–844.

Teruko Mitamura, Liu Zhengzhong, and Eduard Hovy. 2015. Overview of TAC KBP 2015 Event Nugget Track. In *Proceedings of the Eighth Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 16–17 November 2015.

Roberto Navigli and Mirella Lapata. 2007. Graph connectivity measures for unsupervised word sense disambiguation. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence,* Hyderabad, India, 6–12 January 2007, pages 1683–1688.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Beijing, China, 26–31 July 2015, pages 365–371.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* San Diego, Cal., 12–17 June 2016, pages 300–309.

Nils Reimers and Iryna Gurevych. 2015. Event Nugget Detection, Classification and Coreference Resolution using Deep Neural Networks and Gradient Boosted Decision Trees. In *Proceedings of the Eighth Text Analysis Conference,* National Institute of Standards and Technology, Gaithersburg, Maryland, USA, 16–17 November 2015.

Sunita Sarawagi and William W. Cohen. 2004. Semi-Markov conditional random fields for information extraction. In *Proceedings of Advances in Neural Information Processing Systems 17.* Vancouver, B.C., Canada, 13–18 December 2004, pages 1185–1192.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 289–299. Association for Computational Linguistics.