Reading the Manual: Event Extraction as Definition Comprehension

Yunmo Chen[1]   **Tongfei Chen**[1]   **Seth Ebner**[1]
Aaron Steven White[2]   **Benjamin Van Durme**[1]
[1]Johns Hopkins University   [2]University of Rochester
{yunmo,tongfei,seth,vandurme}@jhu.edu
aaron.white@rochester.edu

Abstract

We ask whether text understanding has progressed to where we may extract event information through incremental refinement of *bleached statements* derived from annotation manuals. Such a capability would allow for the trivial construction and extension of an extraction framework by intended end-users through declarations such as, *Some person was born in some location at some time*. We introduce an example of a model that employs such statements, with experiments illustrating we can extract events under closed ontologies and generalize to unseen event types simply by reading new definitions.

1 Introduction

This work is aimed at the disconnect between how human annotators and machines carry out information extraction: humans read annotation manuals consisting of guidelines and illustrative examples then label data, whereas machines label data (by making predictions) based purely on previously seen examples (Figure 1). We explore the feasibility of building a model that has access to information derived from annotation manuals. Specifically we focus on the task of event extraction and convert annotation guidelines describing event types into natural language *bleached statements*. An example bleached statement for the ACE 2005 (Walker et al., 2006) LIFE:BE-BORN event type is:

<u>some person</u> was born in <u>some location</u> at <u>some time</u>
   PERSON   PLACE   TIME

The bleached statement describes a general occurrence of an event of a given type. The event's arguments are initialized with bleached placeholders (e.g. *some person*) to be replaced with extracted spans from the text, eventually resulting in, e.g.:

<u>Barack Obama</u> was born in <u>Hawaii</u> at <u>some time</u>
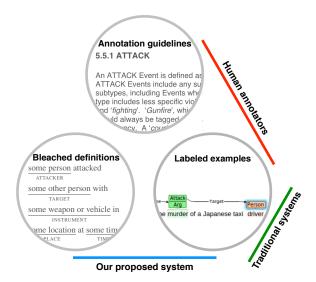   PERSON   PLACE   TIME

Figure 1: Comparison of data sources for human annotators, traditional information extraction systems, and our proposed approach. Human annotators use annotation guidelines and limited illustrative examples, traditional systems use large amounts of labeled examples, and our system uses bleached statements (derived from annotation guidelines) and labeled examples.

We are motivated to consider these statements owing to the rapid progress in sentence-level representation learning and machine reading comprehension: can a contemporary encoder *understand* event statements well enough that their derived representation may be directly employed in extraction?

Bleached statements are straightforward to write and accommodate various levels of expressiveness in both the choice of arguments present and in the lexicalization of the trigger.[1] These features allow for easy adaptation as an ontology changes: simply introduce new or modified statements. In the case where the amount of labeled examples is small or non-existent, a bleached statement serves as a sort

---

[1]For example, for a CONFLICT:ATTACK event, we could use the simple trigger "attacked" or the more descriptive phrase "violently caused physical harm or damage to."
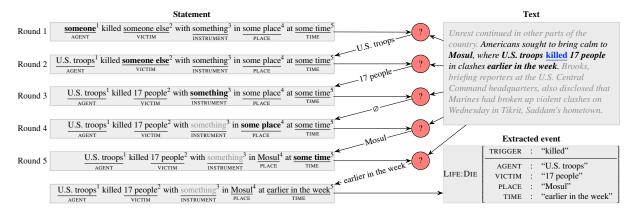
Figure 2: An example of our approach on a sentence from the ACE 2005 dataset for the LIFE:DIE event. The bleached statement is incrementally populated with values from the text (in the order denoted by the superscripts), and not all event arguments are supported by the text. The grayed out text in the paragraph is given for context to the reader, but our model operates on single-sentence contexts.

of canonical example provided ahead of further annotation. We would like a solution where one simply declared the intended information, where any traditional labeled examples were used for disambiguation and fine-tuning, rather than being a critical component in building a model.

As an example of such a solution, we propose a model which incrementally populates bleached statements by querying partially filled statements against text. This strategy is similar to the tasks of machine reading comprehension (MRC) and question answering (QA), in which an answer span is predicted in response to a question about a document. Conceptually this may also be considered a form of incremental recognizing textual entailment (RTE) (Dagan et al., 2006) where we iteratively refine a hypothesis that is supported by the document context. Experimental results demonstrate that zero- and few-shot event extraction are feasible with this approach. While our intent here is exploratory, we manage to achieve state-of-the-art performance on trigger identification and trigger classification on the ACE 2005 dataset. The contributions of this work are: **(i)** A novel approach to event extraction that takes into account annotation guidelines through bleached statements; **(ii)** A multiple-span selection model that demonstrates the feasibility of the approach for event extraction as well as for zero- and few-shot settings.

2 Background

Event extraction is traditionally viewed as three subtasks: (1) event trigger detection, where triggers of events (words that most clearly express the occurrences of events) are detected; (2) entity men-

tion detection, where all potential arguments (entity mentions) to events are detected; and (3) argument role prediction, where relations between detected arguments and trigger words are recognized with respect to each event type's defined set of roles.

Much prior work adopts a pipelined approach to these 3 subtasks or focuses on a subset of the subtasks based on gold entity mention spans. These include feature-based approaches (Ji and Grishman, 2008; Liao and Grishman, 2010; McClosky et al., 2011; Huang and Riloff, 2012; Li et al., 2013, *inter alia*) and neural approaches (Nguyen and Grishman, 2015; Chen et al., 2015, 2017; Nguyen and Grishman, 2018; Sha et al., 2018, *inter alia*).

Because pipelined approaches suffer from error propagation in which the error from earlier subtasks (e.g. entity mention detection) is inherited by later subtasks, joint modeling of the 3 subtasks has been attempted. Yang and Mitchell (2016) attempts to jointly model the three components with hand-crafted features, but still need to detect entity mentions and event triggers separately. Nguyen and Nguyen (2019) jointly models the three tasks using neural networks with shared underlying representations. The models proposed in these two works are the baselines used in this paper.

Huang et al. (2018) approach zero-shot event extraction by stipulating a graph structure for each event type and finding the event type graph structure whose learned representation most closely matches the learned representation of the parsed AMR (Banarescu et al., 2013) structure of a text. In contrast, our approach forgoes explicit graph-structured semantic representations such as AMR.

Researchers have introduced large question an-

swering (QA) / machine reading comprehension (MRC) datasets in a cloze style (Hermann et al., 2015; Onishi et al., 2016), where a query sentence contains a placeholder and the model fills the blank. Our work can be viewed as an extension to such work, where multiple placeholders are extracted.

Li et al. (2019) casts relation extraction as multi-turn QA with natural language questions, where in each turn one argument of the relation is found.. The method requires writing a question for each entity type and each relation type. In (Levy et al., 2017), sets of crowdsourced paraphrastic questions are written for each relation type in the ontology. In contrast, for each event type we use a single declarative bleached statement derived from the annotation guidelines. Soares et al. (2019) proposes a model for relation extraction by filling in two blanks given a contextual relation statement.

These three methods focus on binary relation extraction, and do not readily generalize to *n*-ary events or relations. Our approach naturally supports variable arity events and relations.

3 Problem Formulation

A *bleached statement* consists of: the statement tokens $S = (s_1, s_2, \cdots, s_n)$; a placeholder dictionary $R = \{(r_k : I_k)\}_{k=1,\cdots,K}$, where r_k is the predefined *role* of that argument (e.g. AGENT, PATIENT); and an index set $I_k \subseteq \{1, \cdots, n\}$, a set containing indices of tokens in the statement S (i.e. if $I = \{i_1, \cdots, i_l\}$, then $(s_{i_1}, \cdots, s_{i_l})$ is a placeholder).[2] An example bleached statement in the ACE 2005 dataset for the event type LIFE:DIE (also used in our Figure 2 for illustration purposes) is:

$$\underset{\text{AGENT}}{\underline{\text{someone}}} \text{ killed } \underset{\text{VICTIM}}{\underline{\text{someone else}}} \text{ with } \underset{\text{INSTRUMENT}}{\underline{\text{something}}}$$

$$\text{in } \underset{\text{PLACE}}{\underline{\text{some place}}} \text{ at } \underset{\text{TIME}}{\underline{\text{some time}}}$$

This statement is accompanied by the following placeholder dictionary, in which each role is mapped to an index set that highlights the placeholder in the bleached statement[3]:

$$R = \begin{bmatrix} \text{AGENT} & : & \{1\} \\ \text{VICTIM} & : & \{3, 4\} \\ \text{INSTRUMENT} & : & \{6\} \\ \text{PLACE} & : & \{8, 9\} \\ \text{TIME} & : & \{11, 12\} \end{bmatrix}$$

---

[2] Our bleached statements are inspired in part by linguistic resource creation efforts by White and Rawlins (2018).

[3] The model itself does not see the role names. They are used only for human readability and evaluation.

The event extraction task as defined in the ACE 2005 dataset also requires finding an *event trigger*—a span in the text that most clearly expresses the event's occurrence. In this example, the trigger is the word "*killed*". For a consistent implementation, we consider the trigger to be a special argument of the event, with role name TRIGGER.

Formally, the task is: given a bleached statement S, its placeholder dictionary R, and text tokens T, return a dictionary $\hat{R}$ that contains the event trigger and the the extracted arguments. Such a result is shown in the bottom right of Figure 2. Note that the INSTRUMENT role is not filled in the example because the model does not find a span to fill it.

4 Approach

Given a bleached statement with multiple placeholders, we do not fill the placeholders in parallel—instead, we fill them incrementally in an enforced order.[4] In each step, the model attempts to fill a single focused placeholder, which is replaced by the extracted span(s) thereby creating a refined statement (see Figure 2). In this work we fill the placeholders in the statement *from left to right* and leave other orders as future work.

Formally, in each round, our model returns multiple arguments (see "Multiple Argument Selector" section below) for each placeholder:

$$A \leftarrow \text{GETARGS}(S, I, T) ,$$

where S is the (partially refined) statement, I is the index set that covers the focused placeholder (which corresponds to a role), and T is the text to extract from. The returned argument set A contains a number of text spans (potentially zero) in T that replace the placeholder in S picked out by I.

If A is the empty set, then the model did not find an appropriate text span to replace the placeholder. If the answer set A is not empty, we replace the placeholder with the extracted span. Note that in some cases, there can be more than one argument that fits a role. Consider the following bleached statement (for the ACE 2005 event LIFE:MARRY), focused on the first placeholder "*some people*":

$$\underset{\text{PERSON}}{\underline{\text{some people}}}^1 \text{ married in } \underset{\text{PLACE}}{\underline{\text{some location}}}^2 \text{ at } \underset{\text{TIME}}{\underline{\text{some time}}}^3$$

We expect multiple arguments for the same role PERSON in this event. If our model returns

---

[4] The enforced order is denoted in our examples by indices on the placeholders and the values that fill them.

Algorithm 1 Argument extraction

Input: statement S, placeholder dictionary R, text T
Output: extracted argument structure E
  function EXTRACTARGS(S, R, T)
    $i \leftarrow 1$ ▷ i-th round
    $S^{(1)} \leftarrow S$ ▷ the initial statement
    $E \leftarrow \varnothing$ ▷ extracted event
    for $(r, I) \in R$ **do**
      $A \leftarrow$ GETARGS($S^{(i)}, I, T$)
      if $A \neq \varnothing$ **then**
        $S^{(i+1)} \leftarrow$ replace the I tokens in $S^{(i)}$
with A ▷ refine the statement
          $E \leftarrow E \cup (r : A)$
      else $S^{(i+1)} \leftarrow S^{(i)}$ ▷ skip to the next role
      end if
      $i \leftarrow i + 1$
    end for
  return E
  end function

Algorithm 2 Event extraction

Input: ontology $\mathcal{O}$, text T
Output: extracted event structures $\mathcal{E}$
  function EXTRACTEVENTS($\mathcal{O}, T$)
    $\mathcal{E} \leftarrow \varnothing$
    for $(\tau, S, R) \in \mathcal{O}$ **do**
      $triggers \leftarrow$ TRIGGERID(S, R, T)
      for $t \in triggers$ **do**
        $S' \leftarrow$ ANCHORTRIGGER(S, t)
        $E \leftarrow$ EXTRACTARGS(S', R, T)
        $\hat{R} \leftarrow E \cup$ (TRIGGER : t)
        $\mathcal{E} \leftarrow \mathcal{E} \cup \hat{R}$
      end for
    end for
  return $\mathcal{E}$ ▷ all events extracted from T
  end function

$A = \{$"Kim", "Pat"$\}$, i.e. a set containing multiple extracted arguments, we replace the placeholder with all arguments, concatenated with the "*and*" token, and shift the focus to the next placeholder, creating the refined statement:

Kim and Pat[1] married in **some location**[2] at some time[3]
  PERSON   PLACE   TIME

If our model returns nothing, i.e. $A = \varnothing$, we simply skip the placeholder and move the focus to the next placeholder. For example, if the model finds no argument for the PLACE role, the refined statement of the next iteration would be

Kim and Pat[1] married in some location[2] at **some time**[3]
  PERSON   PLACE   TIME

We run this iterative process until all roles of an event are visited. An advantage of this method is that during the incremental refinement process, the statement always remains a natural language sentence. The incremental process for extracting event arguments is formalized in Algorithm 1, given the initial bleached statement S, the role dictionary R, and the text T to extract from.

Annotation manuals of interest usually define multiple event types. For each event type τ described in the manual, we require a bleached statement S and a role dictionary R. These together form our ontology $\mathcal{O} = \{(\tau_k, S_k, R_k)\}$. To perform full event extraction (Algorithm 2), we first run a trigger detection model for *all event types* (see

"Trigger Identification" section below) specified in the ontology. For those event types whose trigger is found, we proceed with argument extraction (Algorithm 1).

4.1 Model

Architecture for MRC In light of recent advancements in NLP from large-scale pre-training, we use BERT (Devlin et al., 2019) as our sequence encoder. We first review the answer selector architecture for machine reading comprehension (MRC) used in BERT, then extend it for our approach.

Under the formulation of MRC, each training data point is of the form (S, T) where S is a natural language question with tokens $S = (s_1, \cdots, s_n)$ and T is the text to extract answers from, with tokens $T = (t_1, \cdots, t_m)$. The model returns a span in T or predicts that the question is not answerable, in which case an empty span is returned.

To perform MRC, Devlin et al. (2019) proposed the following architecture. First the question S and the text T are concatenated with special delimiters and passed through the BERT contextualizer:

$$\text{BERT} \left([\text{CLS}, s_1, \cdots, s_n, \text{SEP}, t_1, \cdots, t_m, \text{SEP}] \right),$$

where CLS is a special sentinel token whose embedding encompasses the whole string, and SEP is a sentence separator. We denote the output encoding of each question token s_i ($1 \le i \le n$) as $\mathbf{s}_i \in \mathbb{R}^d$, and the encoding of each text token t_j ($1 \le j \le m$) as $\mathbf{t}_j \in \mathbb{R}^d$. Additionally, two vectors, $\mathbf{b}_{\text{left}}$ and $\mathbf{b}_{\text{right}}$, for the left and right boundaries of the answer span are learned. The probability of

each token t_j ($1 \le j \le m$) being the left or right boundary of the answer span is computed as

$$P_{\text{left}}(t_j) \propto \exp(\mathbf{b}_{\text{left}} \cdot \mathbf{t}_j); \; P_{\text{right}}(t_j) \propto \exp(\mathbf{b}_{\text{right}} \cdot \mathbf{t}_j)$$

The two vectors $\mathbf{b}_{\text{left}}$ and $\mathbf{b}_{\text{right}}$ act as attention *query vectors* to the text, resulting in a soft pointer over the text tokens.

Multiple Argument Selector Our scenario is fundamentally different from MRC in two ways: (1) Our query is not formulated as a natural language question; instead, it is a cloze-style problem with a natural language statement and a highlighted blank to fill; (2) For some cases, there can be more than one answer for a given blank. Previous MRC models support extracting only at most one answer.

To accommodate these requirements, we propose a new architecture for this scenario that describes the GETARGS function in Algorithm 1. Given a bleached statement $S = (s_1, \cdots, s_n)$ with a highlighted placeholder span with indices $I = \{i_1, \cdots, i_l\} \subseteq \{1, \cdots, n\}$, instead of two attention query vectors $\mathbf{b}_{\text{left}}$ and $\mathbf{b}_{\text{right}}$ to get the left and right boundary for the answer span, we consider the problem of answer span selection as a *tagging* problem, first proposed in Yao et al. (2013), where answer spans are tagged using a linear chain CRF (Lafferty et al., 2001). By considering answer span selection as tagging, our model selects potentially multiple spans for a query.

We enforce the constraint that all extracted spans come from the same sentence in the text, but in general this constraint need not be enforced. Additionally, our model operates on single-sentence contexts, so information available in other sentences is not considered.

We use the BIO tagging scheme (Ramshaw and Marcus, 1995), where each token in the text is tagged with B (beginning), I (inside), or O (outside). In a linear-chain CRF, the probability of an output tag sequence $y_1, \cdots, y_j$ (for each j, $y_j \in \{\text{B}, \text{I}, \text{O}\}$) given the text $T = (t_1, \cdots, t_j)$ is

$$P(y_1, \cdots, y_j | t_1, \cdots, t_j) \propto \prod_{j=1}^{m} \psi(y_{j-1}, y_j, j), \quad (1)$$

where we define the potential function $\psi(y_{j-1}, y_j, j)$ as the output of a neural function described below. Our model is trained to maximize P.

We first compute an attentive representation for a placeholder with respect to each text token t_j, using the attention mechanism proposed by Luong et al. (2015), since the placeholder is of variable length but we desire a fixed-size vector representation:

$$a_{ij} = \frac{\exp\left(\mathbf{s}_i \cdot \mathbf{t}_j\right)}{\sum_{i' \in I} \exp\left(\mathbf{s}_{i'} \cdot \mathbf{t}_j\right)} \quad (2)$$

$$\tilde{\mathbf{s}}_j = \sum_{i \in I} a_{ij} \mathbf{s}_i \quad (3)$$

Then the attentive placeholder representation $\tilde{\mathbf{s}}_j$, together with its corresponding text token representation $\mathbf{t}_j$, are joined using various matching methods proposed in Mou et al. (2016):[5]

$$\mathbf{x}_j = \left[\tilde{\mathbf{s}}_j \; ; \; \mathbf{t}_j \; ; \; |\tilde{\mathbf{s}}_j - \mathbf{t}_j| \; ; \; \tilde{\mathbf{s}}_j \odot \mathbf{t}_j\right] \quad (4)$$

yielding the joined feature vector $\mathbf{x}_j \in \mathbb{R}^{4d}$.

Finally the joined feature vector $\mathbf{x}_j$ is passed through a multi-layer feed-forward neural network to get the final potential function for each token and each predicted tag type $y_j \in \{\text{B}, \text{I}, \text{O}\}$:

$$\psi(y_{j-1}, y_j, j) = \text{FFNN}_{y_j}(\mathbf{x}_j) \quad (5)$$

In our experiments, we pass $\mathbf{x}_j$ through 4 layers, with output dimensions $2d, d, d$, and 1, respectively, and tanh as the nonlinearity function between layers.

Trigger Identification Triggers of events can be thought as a special argument, which usually is the main verb (or a nominalized verb) that expresses the occurrence of an event. We reuse the argument selection model for trigger identification: the highlighted token set for the trigger is all tokens in the statement that are not part of any standard argument:

$$I_{\text{trigger}} = \{1, \cdots, n\} \setminus \bigcup_{(r, I) \in R} I \quad (6)$$

For example, the highlighted token set for the trigger of the statement in Figure 2 consists of the tokens underlined below:

> someone <u>killed</u> someone else <u>with</u> something <u>in</u> some place <u>at</u> some time

4.2 Training Data Generation

We generate data examples in the form of (S, I, T) triples to train the argument extractor, where S is a bleached statement, I is the index set of the focused

---

[5] $|\cdot|$ is elementwise absolute value, $\odot$ is elementwise product, and $[\,;\,]$ is vector concatenation.

placeholder, and T is the text. Algorithm 1 generates a sequence of bleached statements, where each successive statement is a refinement of its predecessor. During training, instead of replacing placeholders with their predicted arguments $A \leftarrow \text{GETARGS}(S, I, T)$, we replace them with the gold argument(s) from the event extraction dataset.

Negative Sampling For trigger identification, we augment each example with negative samples from the set of event types not found in the example's text. For each event, $\alpha\%$ of the non-occurring event types are taken as negative samples. We tune $\alpha \in \{10, 20, 30, 40, 50\}$.

4.3 Recasting MRC Data for Pre-training

SQuAD (Rajpurkar et al., 2016) is a reading comprehension dataset consisting of questions on a set of Wikipedia articles, where the answer to each question is a span of text extracted from the corresponding reading passage. Its version 2.0 (Rajpurkar et al., 2018) contains additional data that poses unanswerable questions to reading comprehension systems. To do well, a system should learn to abstain from answering when no answer is supported by the text.

We employ recast versions of the training and development splits of SQuAD 2.0 as pre-training data for our event extraction system. We cast each SQuAD natural language question to a format similar to our bleached statements, where the *wh-* question phrases of the questions are tagged as the placeholders to be filled. For example, given the following SQuAD question,

> <u>What form of oxygen</u> is composed of 3 oxygen atoms?

the extracted *wh-* phrase is "*What form of oxygen*", which is chosen as the single placeholder in this statement. This is answered as

> <u>ozone</u> is composed of 3 oxygen atoms?
> _{ANSWER}

This methodology is linguistically motivated, as both questions (as in SQuAD) and bleached statements (this work) reduce to logical forms with the same predicate. The denotations can be written (using a generic operator Q) as

$Qx. [\![\text{form of oxygen}]\!](x)$

$\wedge [\![\text{composed of three oxygen atoms}]\!](x)$

where Q is λ for the question and is $\exists$ for the bleached statement. Hence the *wh-* phrase is semantically similar to an existentially quantified phrase (e.g. *some form of oxygen*, where *some* introduces existential quantification), despite their pragmatic difference in illocutionary force (inquiring vs. stating). Additionally, *wh-* phrases presuppose the existence of their answer referent; to use a *wh-*phrase when no referent exists would be infelicitous. Hence *wh-* question phrases serve the same function as the existentially quantified placeholder phrases in our bleached statements, and so the recast SQuAD questions are appropriate data for pre-training.

We extract *wh-* phrases through syntactic analysis of the questions. We define the *wh-* phrase of a question to be the maximum span in its constituency parse that bears any of the question tags in the Penn Treebank (Marcus et al., 1993) parsing annotation guideline.[6]

We employ the neural span-based constituency parser (Stern et al., 2017) in the AllenNLP (Gardner et al., 2018) toolkit to parse the SQuAD questions for extracting the *wh-* phrases.

5 Experiments and Discussions

5.1 Event Extraction on ACE 2005

We evaluate our approach on the ACE 2005 dataset and use the same data splits as previous work, in which 40 newswire documents are used as the test set, another 30 documents of different genres are selected as the development set, and the remaining 529 documents constitute the training set (Li et al., 2013; Yang and Mitchell, 2016; Nguyen and Nguyen, 2019). Following previous work, we use four evaluation metrics: (1) *Trigger Identification*: a trigger is correctly identified if its span offsets exactly match a reference trigger; (2) *Trigger Classification*: a trigger is correctly classified if its span offsets and event subtype exactly match a reference trigger; (3) *Argument Identification*: an argument is correctly identified if its span offsets and corresponding event subtype exactly match a reference argument; and (4) *Argument Classification*: an argument is correctly classified if its span offsets, corresponding event subtype, and argument role exactly match a reference argument. The overall

---

[6]These include the following tags (followed by examples): WHADJP (how many), WHADVP (why), WHNP (which book), texttttWHPP (by whose authority), WDT (which), WP (who), WP$ (whose), WRB (where).

Model	Trigger						Argument					
	Identification			Classification			Identification			Classification		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
JOINTFEATURE	**77.6**	65.4	71.0	**75.1**	63.3	68.7	**73.7**	38.5	50.6	**70.6**	36.9	48.4
JOINT3EE	70.5	74.5	72.5	68.0	71.8	69.8	59.9	**59.8**	**59.9**	52.1	**52.1**	**52.1**
Ours w/ partial data	64.5	62.3	63.4	60.9	58.7	59.8	43.1	42.8	43.0	35.2	34.9	35.0
Ours w/o pre-training	50.0	**85.7**	63.2	48.1	**82.4**	60.7	29.3	55.0	38.2	24.7	46.4	32.2
Ours w/ full data	68.9	77.3	**72.9**	66.7	74.7	**70.5**	44.9	41.2	43.0	44.3	40.7	42.4

Table 1: P(recision), R(ecall), and F_1 obtained by models on the ACE 2005 dataset. Best results are **bolded**. Using the full training set improves F_1 performance over using the partial training set on all metrics except argument identification (equal). Pre-training on the recast SQuAD 2.0 dataset improves F_1 performance over no pre-training on all metrics.

performance is evaluated using *precision* (P), *recall* (R), and *F-measure* (F_1) for each metric.

We use BERT for sequence encoding.[7] For pre-training on the recast SQuAD 2.0 dataset, we follow the previously mentioned pre-processing strategy. We pre-train on the training set of SQuAD 2.0 and perform early stopping using the development partition. Examples that do not have exactly 1 *wh*-phrase are discarded.[8] The maximum sequence length is 512 word pieces, the maximum query length is 128 word pieces, the learning rate is 3×10^{-5} with an Adam optimizer, the maximum gradient norm for gradient clipping is set to 1.0, and the number of training epochs is 3.

After pre-training on SQuAD 2.0, we fine-tune the model on ACE 2005. While keeping other hyperparameters unchanged, we set the learning rate to 1×10^{-5} and the number of training epochs to 8. During fine-tuning, we employ negative sampling and set the negative sampling rate to 30%.

In addition to fine-tuning on the full training set of ACE 2005, we consider a single-genre "partial" training setting in which the model is trained only on the 58 documents that appear in the newswire portion of the full training set.

Experimental Results & Discussion   We train our model using full and partial training data and compare with two joint event extraction model baselines. The JOINTFEATURE model (Yang and Mitchell, 2016) is a feature-based model that ex-

ploits document-level information; the JOINT3EE model (Nguyen and Nguyen, 2019) is a neural model that achieves state-of-the-art performance on ACE 2005. These two models represent the state-of-the-art performance for feature-based and neural models, respectively.

Table 1 reports the performance of the systems on the four evaluation metrics. Training on the full training set improves F_1 performance over training on the partial training set, giving the largest improvement on trigger identification and classification. Additionally, pre-training on the recast SQuAD 2.0 dataset provides large F_1 improvements (4.8%–10.2% absolute F_1 increase) on all four evaluation metrics. Our model also tends to have higher recall than precision, especially on trigger identification and classification, and suffers from low precision compared to prior work. Our model achieves state-of-the-art performance on trigger identification and trigger classification.

Because our model does not explicitly incorporate entity mention detection, we hypothesize that our MRC-inspired approach predicts answer spans that are semantically correct but do not exactly match the gold answers, hurting performance on argument-related subtasks. We compare predicted arguments with gold references and find the following sources of errors:

- *Relative clauses*: Our model predicts *Mosul* whereas the gold answer is *Mosul, where U.S. troops killed 17 people in clashes earlier in the week*.

- *Counts*: The gold annotation is *300 billion yen* but our model predicts *300 billion*.

- *Durations*: The gold annotation is *lasted two*

---

[7]We use the BERT-BASE-CASED model, which has 12 layers, 768-dimensional hidden embeddings, 12 attention heads, and 110 million parameters.

[8]Our effective training set contains 128,649 examples after 1,670 were discarded, and the effective development set contains 11,772 examples after 100 were discarded.
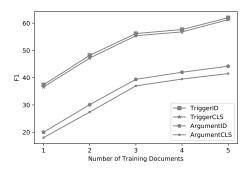
Figure 3: Few-shot experimental results on FrameNet.

Frame	Trigger		Argument	
	Id.	Cls.	Id.	Cls.
ARRIVING	9.1	9.1	6.1	6.1
ATTACK	66.4	66.4	42.1	42.1
GETTING	17.6	17.6	14.3	13.8
INTENTIONALLY_CREATE	12.5	12.5	7.1	7.1
KILLING	22.4	22.4	13.1	7.0
MANUFACTURING	37.4	37.4	27.6	26.7
SCRUTINY	16.6	16.6	11.7	11.0
STATEMENT	0.6	0.6	0.7	0.5
SUPPLY	3.1	3.1	1.7	1.5
USING	10.4	10.4	10.0	8.8
Macro-averaged F_1	19.6	19.6	13.4	12.46

Table 2: Zero-shot results on FrameNet frames. Trigger classification performance is equivalent to trigger identification performance because we evaluate on only one frame in each zero-shot learning experiment.

hours but our model predicts *two hours*.

5.2 Few-shot Learning on FrameNet

In order to evaluate our approach under a lower-resource setting than the partial training setting, we consider few- and zero-shot learning on annotated documents from FrameNet.[9] 7 documents are excluded from FrameNet's 107 annotated documents because they do not have frame annotations. The arguments in our bleached statements consist of only the frame's core *frame elements*. We use the same four evaluation metrics as used for ACE 2005.

We split the remaining 100 documents into 5 training documents and 95 test documents and experiment with training on between 1 and 5 documents.[10] We pick the top-10 most frequent frames that represent events and write bleached statements based on their frame definitions.

Following the same pre-training setup, we then train the model using the same hyperparameters as the model fine-tuned on ACE 2005.

Experimental Results & Discussion The results in Figure 3 show that F_1 performance on all evaluation metrics increases as more documents are added to the training set. The marginal utility of adding training documents almost monotonically decreases as the training set increases, so that performance from training on 3 documents roughly matches performance from training on 5 documents.

5.3 Zero-shot Learning on FrameNet

We additionally investigate the model's ability to generalize to unseen event types using the same

dataset as the few-shot setting. We employ a leave-one-out strategy to the frames in Table 2, training on 9 frames and testing on the other 1.

Experimental Results & Discussion The results in Table 2 reveal a large variation in performance on the frames. The best performance is achieved on the ATTACK frame, but frames such as STATEMENT achieve poor performance.

We report the *macro-averaged* F_1 over all frames to reveal overall performance instead of *micro-average*, since we care how the approach generalizes to different frames. Overall, the macro-averaged F_1 shows that the model can feasibly extract information about events of unseen types, but performance varies greatly across frames.

Possible reasons why the STATEMENT frame has low performance include: (1) the event type being too general, (2) the bleached statement being poorly constructed, (3) the span for the "message" role being long and difficult to tag exactly correctly using the `BIO` scheme.

6 Conclusion & Future Work

We present an approach to event extraction that uses bleached statements to give a model access to information contained in annotation manuals. Our model incrementally refines the statements with values extracted from text. We also demonstrate the feasibility of making predictions on event types seen rarely or not at all. Future work can apply our approach to *n*-ary relation extraction.

---

[9]We use the Full Text Annotation portion of FrameNet.

[10]Because we do not perform a hyperparameter sweep in this setting, we do not use a development set.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In *Proc. ACL*, pages 409–419.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. ACL*, pages 167–176.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL*, pages 4171–4186.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. NeurIPS*, pages 1693–1701.

Lifu Huang, Heng Ji, Kyunghyun Cho, Ido Dagan, Sebastian Riedel, and Clare Voss. 2018. Zero-shot transfer learning for event extraction. In *Proc. ACL*, pages 2160–2170.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proc. AAAI*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL-08: HLT*, pages 254–262.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*, pages 282–289.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proc. CoNLL*, pages 333–342.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL*, pages 73–82.

Xiaoya Li, Fan Yin, Zijun Sun, Xiayu Li, Arianna Yuan, Duo Chai, Mingxin Zhou, and Jiwei Li. 2019. Entity-relation extraction as multi-turn question answering. In *Proc. ACL*, pages 1340–1350.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proc. ACL*, pages 789–797.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*, pages 1412–1421.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proc. ACL*, pages 1626–1635.

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proc. ACL*, pages 130–136.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proc. ACL*, pages 365–371.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proc. AAAI*, pages 5900–5907.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proc. AAAI*, pages 6851–6858.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David A. McAllester. 2016. Who did what: A large-scale person-centered cloze dataset. In *Proc. EMNLP*, pages 2230–2235.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proc. ACL*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proc. EMNLP*, pages 2383–2392.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*.

Lei Sha, Feng Qian, Baobao Chang, and Zhifang Sui. 2018. Jointly extracting event triggers and arguments by dependency-bridge RNN and tensor-based argument interaction. In *Proc. AAAI*, pages 5916–5923.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proc. ACL*, pages 2895–2905.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A minimal span-based neural constituency parser. In *Proc. ACL*, pages 818–827.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus (LDC2006T06). *Philadelphia: Linguistic Data Consortium*.

Aaron Steven White and Kyle Rawlins. 2018. The role of veridicality and factivity in clause selection. In *Proceedings of the 48th Meeting of the North East Linguistic Society*.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proc. NAACL*, pages 289–299.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proc. NAACL*, pages 858–867.