Document-Level Event Argument Extraction by Conditional Generation: Appendix

Sha Li and **Heng Ji** and **Jiawei Han**
University of Illinois at Urbana-Champaign, IL, USA
{shal2, hengji, hanj}@illinois.edu

1 Trigger Extraction Model Details

1.1 Tagging Scheme

We use the IO tagging scheme, where I stands for "inside a span" and O stands for "outside any span". This simplified tagging scheme was selected to reduce parameters without much loss of modeling power since (1) triggers are often single words and I tags in the BIO (B stands for "beginning of a span") scheme are infrequent and (2) we rarely see two consecutive event triggers of the same type.

1.2 Class Vectors

For each of the event types, we provided 3 keywords as initial seeds. If the event type can be triggered by nominals, we additionally add keywords for the nominal form. Our chosen keywords will be provided along with the ontology file as supplementary materials.

For each event type, we search for its corresponding keywords' occurrence in the Gigaword corpus. To filter out ambiguous usages of the keywords, we apply BERT-large as a masked language model and predict words that can replace the current mention of the keyword. If another keyword for this event type appears among the top 50 candidates, we accept this example. The vector representation for this example is the average of the wordpiece tokens that consist the keyword.

The class vector is an average over all the examples for the event type.

2 Solving for M

The following section is a simplified version of the derivation from TapNet (Yoon et al., 2019).

In order to correctly classify c_k, we would like to maximize the dot product with ϕ_k and minimize the dot product with $\phi_{l\neq k}$ in the subspace defined by M. A possible solution would be to find the projection matrix M so that:

$$M(c_k) = \lambda M(\phi_k - \frac{1}{m-1}\sum_{l\neq k}\phi_l) \quad \text{s.t.} \|\phi_i\| = 1, \phi_i^T\phi_{j\neq i} = 0. \tag{1}$$

This implies that

$$M(c_k)^T M(\phi_k) = \lambda\|M\|^2 \quad M(c_k)^T M(\phi_l) = -\lambda\frac{1}{m-1}\|M\|^2 \tag{2}$$

which is a reasonably good separation between the classes.

Let $\hat{\phi}_k = \phi_k - \sum_{l\neq k}\phi_l$, then we can rearrange the previous equation as:

$$M^T(c_k - \lambda\hat{\phi}_k) = 0 \tag{3}$$

Note that this holds for every k. If we define $D \in \mathbf{R}^{d\times n}$ as the matrix with $c_k - \lambda\hat{\phi}_k$ as its kth column, we have $M^T D = \vec{0}$, implying that the columns in M are in the null space of D^T. This null space of D^T can be obtained by QR decomposition.

$$D^T = QR = [Q_1, Q_2]\begin{bmatrix}R_1\\0\end{bmatrix} \tag{4}$$

Although the rank of D is unknown, it will not be larger than n (and with high probability close to n), and thus we can take m columns starting from the $n+1$ column of Q for $M \in \mathbf{R}^{d\times m}$.

In order to account for the new types, we apply some leniency at training time and learn $n' > n$ reference vectors instead of only n vectors for the n classes that appear in the training set. Then when we are asked to identify new types during inference, we update M based on the new class vectors c'_n.

The complete algorithm is listed in Algorithm 1.

2.1 Pseudo Labeling

In the pseudo-labeling process, we compute the token-wise cosine similarity between class vectors and averaged sentence-piece embeddings from

Algorithm 1: Event trigger extraction.

Training;
Input: Label names of n event types. Training examples $\{x, y\}$.
Compute class vectors $c_1, \cdots, c_n$;
Initialize $\phi_1, \cdots, \phi_{n'}$;
for *training episode* **do**
  Compute D;
  Compute M by decomposing D;
  for *training batch* **do**
    Optimize $\theta = \{\Phi, W, \theta_f\}$ w.r.t Equation 8.
  end
end
Testing;
Input: Label names of n' event types.Test examples $\{x\}$.
Result: Set of $\{e, p\}$ event type, position pairs.
Compute class vectors for new classes $c_{n+1}, \cdots, c_{n'}$;
Compute D;
Compute M by decomposing D;
for *test example* **do**
  Predict y for x sequence.
end

BERT-Large. The event type token labels are accepted if the similarity is higher than 0.65 and the O label is assigned if none of the similarity scores are higher than 0.4. For cases in between, we assign an X label which means ignoring the token for loss computation.

3 Dataset Collection and Annotation Details

We removed documents that have less than 100 tokens, and off-topic documents such as excerpts from history books. In the annotation process, annotators can also flag documents as duplicates, or irrelevant. All documents are in English.

When using the KAIROS event ontology, out of the 67 defined event types, we use 51 types that were found in our dataset and merge some rarely seen sub-subevent types. In particular, event sub-subtypes under Contact.Prevarication, Contact.RequestCommand, Contact.ThreatenCoerce were merged. Movement.Transportation.GrantAllowPassage, Transaction.AidBetweenGovernments.Unspecified, Personnel.ChangePosition types were omitted.

Before the event annotation stage, we run a SOTA entity detection model OneIE (Lin et al., 2020) to highlight entity spans. Although this model is not perfect, it can help annotators find candidates for event arguments and reduce annotation time.

The task for the event annotation stage is to identify event trigger and argument spans and label them with the correct event type (argument role). Annotators can also add missing entities or correct the automatic produced entity spans. A two-pass procedure is applied to control the quality of annotation: after annotator A finishes, we randomly assign the annotated document to another more senior annotation B for correction.

After stage 1 finishes, we clean up the annotation by aligning the spans back to word boundaries and then run a joint entity and event coreference system. In stage 2, the annotators are presented with entity (event) clusters and asked to correct them.

Parameter	Value
Base Model	BART-large
Learning rate	[1e-5, 3e-5]
Scheduler	Linear (without warmup)
Batch size	2*8
Max sequence length	512
Training epochs	[3,6]
Beam size	4

Table 1: Hyperparameters for argument extraction

4 Implementation Details

We use the BART-large model (Lewis et al., 2020) for our argument extraction model. Hyperparameters are presented in Table 1. For the zero-shot transfer settings, we trained with a smaller learning rate (1e-5) and more epochs (6).

For the trigger extraction task, we used the BERT-large-cased (Devlin et al., 2019) model. The list of hyperparameters as shown in Table 2.

The BERT-CRF model is similar to (Shi and Lin, 2019). To indicate the trigger, we append the trigger to the input sentence: `[CLS] sentence [SEP] trigger [SEP].`

In order to adapt the BERT-QA model for our event ontology, we use the Template 2 (argument based question template) for argument extraction with trigger information: `[wh_word] is the [role name] in [trigger]?`

5 Additional Experiments on ACE

In Tables 4 and 5 we show the complete trigger extraction and argument extraction results on ACE. Entries with an asterisk (*) indicate that these are reported numbers and may be prone to slight differences in dataset splitting and pre-processing.

Parameter	Value
Base Model	BERT-large-cased
Learning rate	3e-5
Weight decay	1e-5
Scheduler	Linear (without warmup)
Batch size	8
Max sequence length	200 (ACE), 400 (WIKIEVENTS)
Training epochs	10
Projection dim	200
Regularization α	0.5

Table 2: Hyperparameters for trigger extraction

Parameter	Value
Base Model	BERT-large-cased
Learning rate	3e-5
Weight decay	1e-5
CRF learning rate	1e-4
Dropout	0.4
Scheduler	Linear (without warmup)
Batch size	8
Max sequence length	200 (ACE), 400 (WIKIEVENTS)
Training epochs	10

Table 3: Hyperparameters for BERT-CRF baseline.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling.

Sung Whan Yoon, Jun Seo, and Jaekyun Moon. 2019. Tapnet: Neural network augmented with task-adaptive projection for few-shot learning. In *ICML*.

# Seen Event Types	Model	TI Precision	TI Recall	TI F1	TC Precision	TC Recall	TC F1
0	Prototype	1.64	56.03	3.19	1.14	39.01	2.22
	TAPKEY	51.76	59.1	55.19	48.86	55.79	52.10
10 most frequent	Prototype	67.03	72.1	69.48	63.74	68.56	66.06
	BERT-QA	66.25	50.12	57.06	62.5	47.28	53.83
	TAPKEY	67.56	77.78	72.31	64.68	74.47	69.23
1 per general type	Prototype	70.53	66.19	68.29	68.01	63.83	65.85
	BERT-QA	64.91	17.49	27.56	59.64	16.07	25.32
	TAPKEY	66.73	78.72	72.23	63.33	74.7	68.55
All	DYGIE++*	-	-	-	-	-	69.7
	OneIE*	-	-	78.2	-	-	74.7
	BERT-CRF			73.73			69.59
	Prototype	68.1	78.72	73.03	64.83	74.94	69.52
	BERT-QA	68.91	77.54	72.97	65.13	73.29	68.97
	TAPKEY	72.69	76.12	74.36	69.53	72.81	71.13

Table 4: Trigger extraction results on ACE05. Results from DYGIE++ and OneIE are from their papers.

Triggers	Model	AI Precision	AI Recall	AI F1	AC Precision	AC Recall	AC F1
Predicted	DYGIE++*	-	-	55.4	-	-	52.5
	OneIE*	-	-	59.2	-	-	56.8
	BERT-QA*	58.02	50.69	54.11	56.87	49.83	53.12
	BART-Gen	57.57	53.05	55.22	55.99	51.60	53.71
Gold	BERT-QA	69.16	62.65	65.74	66.51	60.47	63.34
	BART-Gen	71.13	68.75	69.92	67.82	65.55	66.67

Table 5: Argument extraction results on ACE05. * indicate reported results.