# Iterative Document-level Information Extraction via Imitation Learning

**Yunmo Chen**[1] **William Gantt**[2] **Weiwei Gu**[2]
**Tongfei Chen**[3] **Aaron Steven White**[2] **Benjamin Van Durme**[1]
[1] Johns Hopkins University [2] University of Rochester [3] Microsoft Semantic Machines
{yunmo|vandurme}@jhu.edu, {wgantt@ur.|wgu7@ur.|aaron.white@}rochester.edu, tongfei@pm.me

## Abstract

We present a novel iterative extraction model, IterX, for extracting complex relations, or *templates*, i.e., $N$-tuples representing a mapping from named slots to spans of text within a document. Documents may feature zero or more instances of a template of any given type, and the task of *template extraction* entails *identifying* the templates in a document and *extracting* each template's slot values. Our imitation learning approach casts the problem as a Markov decision process (MDP), and relieves the need to use *predefined template orders* to train an extractor. It leads to state-of-the-art results on two established benchmarks – 4-ary relation extraction on SciREX and template extraction on MUC-4 – as well as a strong baseline on the new BETTER Granular task.[1]

## 1 Introduction

A variety of tasks in information extraction (IE) require synthesizing information across multiple sentences, up to the length of an entire document. The centrality of document-level reasoning to IE has been underscored by an intense research focus in recent years on problems such as argument linking (Ebner et al., 2020; Li et al., 2021, *i.a.*), $N$-ary relation extraction (Quirk and Poon, 2017; Yao et al., 2019; Jain et al., 2020, *i.a.*), and — our primary focus — template extraction (Du et al., 2021b; Huang et al., 2021, *i.a.*).

Construed broadly, template extraction is general enough to subsume certain other document-level extraction tasks, including $N$-ary relation extraction. Motivated by this consideration, we propose to treat these problems under a unified framework of *generalized template extraction* (§2).[2] Figure 1 shows 4-ary relations from the SciREX dataset (Jain et al., 2020), presented as simple templates.

[1] Code available at github.com/wanmok/iterx.

[2] We encourage the reader to consult Appendix A for a discussion of some important differences between generalized template extraction and traditional event extraction.

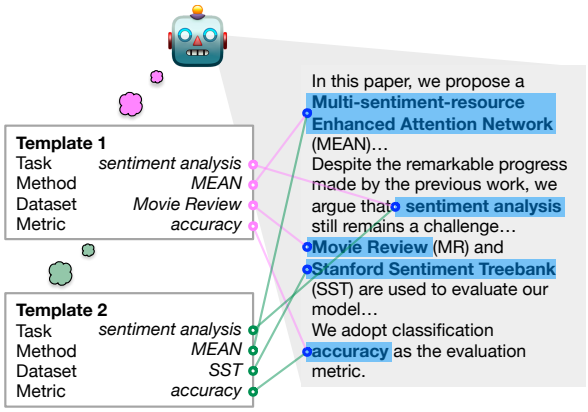


Figure 1: An example of multi-template extraction on a document (an NLP paper; Lei et al. (2018)) from the SciREX dataset. An agent reads the entire paper and *iteratively* generates templates, each consisting of slots for `Task`, `Method`, `Dataset`, and `Metric`.

Since documents typically describe multiple complex events and relations, template extraction systems must be capable of predicting multiple templates per document. Existing approaches such as Du et al. (2021b) and Huang et al. (2021) rely on a linearization strategy to force models to learn to predict templates in a pre-defined order. In general, however, such orderings are arbitrary. Others have instead focused on the simplified problem of role-filler entity extraction (REE), which entails extracting all slot-filling entities but does not involve mapping them to individual templates (Patwardhan and Riloff, 2009; Du et al., 2021a, *i.a.*).

We present a new model for generalized template extraction, IterX, that iteratively extracts multiple templates from a document *without* requiring a pre-defined linearization scheme. We formulate the problem as a Markov decision process (MDP, §2), where an action corresponds to the generation of a single template (§3.2), and states are sets of predicted templates (§3.3). Our system is trained via *imitation learning*, where the agent imitates a dynamic oracle drawn from an expert policy (§3.4). Our contributions can be summarized as follows:



*Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1858–1874
May 2-6, 2023 

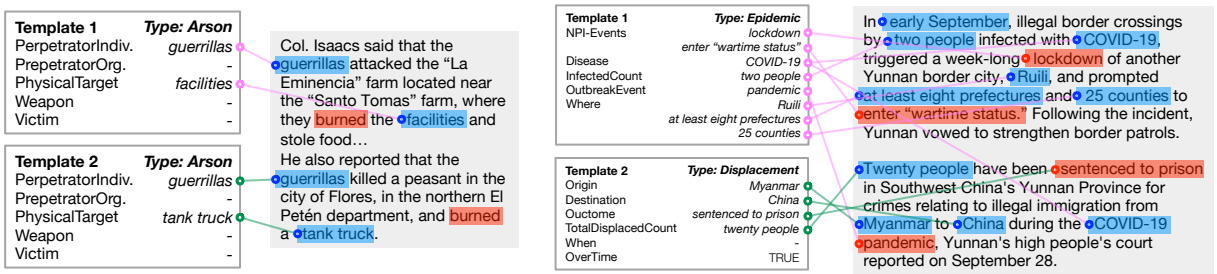


Figure 2: Template examples from MUC-4 (left) and BETTER Granular (right) datasets. Event triggers (e.g. *burned* above) are *not* annotated in MUC-4 and are highlighted here only for clarity.

- We show that generalized template extraction can be treated as a Markov decision process, and that imitation learning can be effectively used to train a model to learn this process without making explicit assumptions about template orderings.
- We demonstrate state-of-the-art results with ITERX on two established benchmarks for complex relation extraction: 4-ary relation extraction on SCIREX and template extraction on MUC-4.
- We introduce strong baselines for the recently introduced English BETTER Granular template extraction task.

## 2 Problem Formulation

We propose to treat both classic template extraction and $N$-ary relation extraction under a unified framework of *generalized template extraction*. Given a document $D = (w_1, \cdots, w_N)$ where each $w_i$ is a token, we assume that some system (or model component) generates a set of candidate *mention spans* $\mathcal{X} = \{x_1, \cdots, x_M\}$, where each $x_i = D[l_i : r_i] \in \mathcal{X}$ is contiguous with left and right span boundary indices $l_i$ and $r_i$.

We define a *template ontology* as a set of template types $\mathcal{T}$, where each type $t \in \mathcal{T}$ is associated with a set of slot types $S_t$. A *template instance* is defined as a pair $(t, \{(s_k : X_k), \cdots\})$ where $t \in \mathcal{T}$ is a template type, $s_k \in S_t$ is a *slot type* associated with $t$, and $X_k \subseteq \mathcal{X}$ is a subset of all mention spans extracted from the document that fills slot type $s_k$ ($X_k = \emptyset$ indicates that slot $s_k$ has no filler).[3] Taking Figure 2 (left) as an example, Template 1 has type $t$ = `Arson` and slots {`PerpetratorIndiv` : {"guerrillas"}, `PhysicalTarget` : {"facilities"}}.

We reduce the problem of extracting a *single* template to the problem of assigning a slot type to each extracted span $x_i \in \mathcal{X}$, where some spans may be assigned a special *null* type ($\varepsilon$), indicating that they fill no slot in the current template. Given this formulation, we can equivalently specify a template instance as $(t, a)$ where $a$ is an *assignment* of spans to slot types: $\{x_i : s_i\}_{s_i \in S_t}$. We denote the union of all slot types across all template types, along with the empty slot type $\varepsilon$, as $\mathcal{S} = \{\varepsilon\} \cup \bigcup_{t \in \mathcal{T}} S_t$.

With these definitions in hand, the problem of *generalized template extraction* can be stated succinctly: Given a template ontology $\mathcal{T}$, a document $D$, and a set of candidate mentions $\mathcal{X}$ extracted from $D$, generate a set of template instances $\{(t_1, a_1), \cdots, (t_K, a_K)\}$, where $t_i \in \mathcal{T}$.

**As an MDP** We treat generalized template extraction as a Markov decision process (MDP), where each step of the process generates one whole template instance. For simplicity, we consider the problem of extracting templates of a specific type $t \in \mathcal{T}$; extracting *all* templates then simply requires iterating over $\mathcal{T}$, where $|\mathcal{T}|$ is typically very small. This MDP $(2^{\mathcal{A}}, \mathcal{A}, E, R)$ comprises the following:[4]

- $2^{\mathcal{A}}$: the set of states. In our case, this is the set of all template generation histories. Each state $A \subseteq \mathcal{A}$ is a set of generated templates;
- $\mathcal{A}$: the set of *actions* or *assignments*: an action is the generation of a single template (an assignment of slot types to spans);
- $E$: the environment that dictates state transitions. Here, each transition simply adds a generated template to the set of all templates generated for the document: $E(A, a) = A \cup \{a\}$;
- $R(A, a)$: the reward from action $a$ under the current state $A$.

These components are detailed in the following section. Figure 3 shows ITERX in action: the MDP

[3] In this work, we use *template* as an abbreviation for *template instance*, relying on the *type* vs. *instance* usage only when necessary for clarity.

[4] Our notation is consistent with prior NLP work that uses MDPs, e.g., Levenshtein Transformers (Gu et al., 2019).

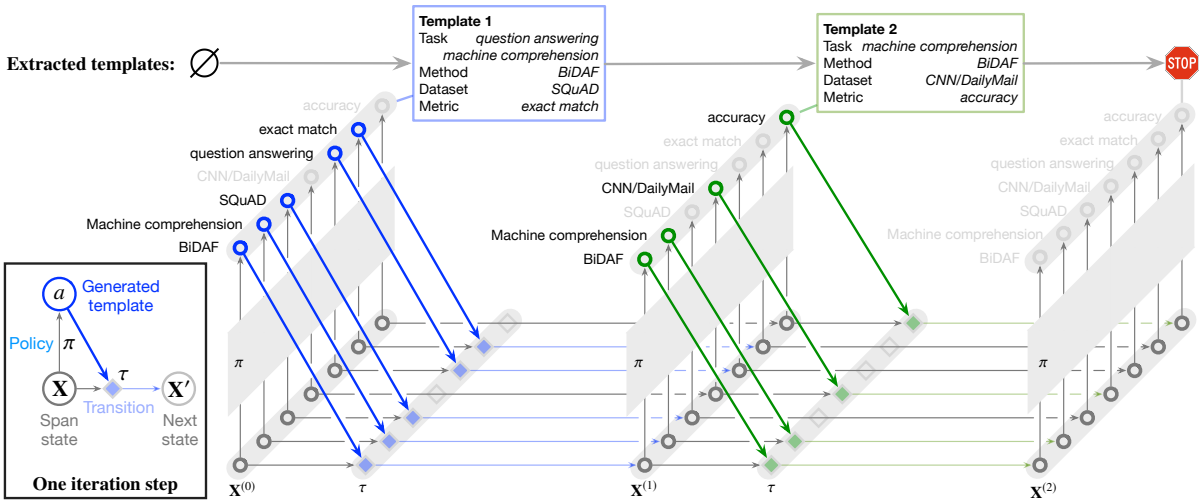


Figure 3: The basic iteration step of ITERX (left box), and an **unrolled** version on SCIREX 4-ary relation extraction (extraction of templates in the form {`Task`, `Method`, `Dataset`, `Metric`}) executed on the NLP paper *Bidirectional Attention Flow for Machine Comprehension* (Seo et al., 2017). Span embeddings ($\mathbf{X}^{(0)}$) are passed as input to the first step, where the model extracts the template {`Task`: *Machine comprehension, Question answering*; `Method`: *BiDAF*; `Dataset`: *SQuAD*; `Metric`: *Exact match*}. This information is propagated via our *memory mechanism* to the second step, and informs prediction of the next template: {`Task`: *Machine comprehension*; `Method`: *BiDAF*; `Dataset`: *CNN/DailyMail*; `Metric`: *Accuracy*}. The third step assigns the null slot type $\varepsilon$ to all spans, indicating that the model is unable to find any further templates, thus stopping the generation process.

produces two templates sequentially, terminating on a null assignment to all input spans in $\mathcal{X}$.

## 3 Model

Our ITERX model is a parameterized agent that makes decisions under the MDP above: conditioned on an input document $D$, spans $\mathcal{X}$ extracted from $D$, and a specific template type $t$, ITERX generates a single template of type $t$ at each step. The model consists of two parameterized components:

- **Policy $\pi$**: A policy $\pi(a \mid t, \mathbf{X})$ that generates a distribution over potential assignments of spans to slots in the current template of type $t$;
- **State transition model $\tau$**: An autoregressive state encoder that maps a state (i.e., a set of predicted templates) $A$ to a continuous representation $\mathbf{X}$ via a state transition model $\tau$, where $\mathbf{X}^{(k+1)} = \tau(\mathbf{X}^{(k)}, a_k)$. Here the state representation $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_M)$ comprises of a vector $\mathbf{x}_i \in \mathbb{R}^d$ for each span $x_i$.

ITERX generates a sequence of templates: It starts with initial state $\mathbf{X}^{(0)}$ (see Figure 3 for a running example) comprising only span representations derived from the encoder (as no templates have been predicted) and ends when no new template is generated (§3.5). ITERX is trained via imitation learning, aiming to imitate an expert policy $\pi^*$ derived from reference templates.

### 3.1 Span Extraction and Representation

ITERX takes *spans* as input and thus relies on a span proposal component to obtain candidate spans.[5] For all experiments, we use the neural CRF-based span finder employed for FrameNet parsing in Xia et al. (2021) and for the BETTER Abstract task in Yarmohammadi et al. (2021).[6] CRF-based span finders have been empirically shown to excel at IE tasks (Gu et al., 2022).

The input document is first embedded using a pretrained Transformer encoder (Devlin et al., 2019; Raffel et al., 2020) that is fine-tuned during model training.[7] Each span $x = D[l : r]$ extracted by the span extractor is encoded as a vector $\mathbf{x}_{\text{enc}}$, which is obtained by first concatenating three vectors of dimension $d$: the embeddings of the first and the last tokens in the span, and a weighted sum

[5] Although many systems since Lee et al. (2017) that require spans as input have trained span proposal modules end-to-end, we found this to be unnecessary to obtain strong results and leave this extension for future work.

[6] The code of the span finder can be found at `github.com/hiaoxui/span-finder`. We refer the reader to these papers for further details.

[7] Different encoders are used in §4 for fair comparison with prior work. For documents exceeding maximum length $N_{\text{max}} = 1024$, tokens are encoded in chunks of size $N_{\text{max}}$.

of the embeddings of all tokens within the span, using a learned global query (Lee et al., 2017). This $3d$-dimensional vector is then compressed to size $d$ using a two-layer feedforward network. Lastly, to incorporate positional information, we add sinusoidal positional embeddings based on the token-level offset of $l$ within the document to yield $\mathbf{x}_{\text{enc}}$.

### 3.2 Policy: Generating a Single Template

A policy $\pi$ generates a single template given span states $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ and template type $t \in \mathcal{T}$, conditioned on the document and all of its candidate mention spans.

Since an action $a$ represents a set of *slot type assignments* for all candidate mentions, the policy $\pi(a \mid t, \mathbf{X})$ can be factorized as

$$\pi(a \mid t, \mathbf{X}) = \prod_{(x:s) \in a} P(s|t, \mathbf{x}) \; . \tag{1}$$

Thus we only need to model the slot type distribution for each candidate span. Here, we employ two models described below.

**Independent Modeling** We train a classifier that outputs a slot type (or $\varepsilon$) given both the template type embedding $\mathbf{t}$ and the slot type embedding $\mathbf{s}$, inspired by a standard practice in binary relation extraction (Ebner et al., 2020; Lin et al., 2020, *i.a.*). It computes the probability with a two-layer feedforward network (FFN), with slots not associated with the template type (i.e., $s \notin S_t \cup \{\varepsilon\}$) assigned 0 probability:

$$P_{\text{ind}}(s|t, \mathbf{x}) \propto \mathbf{1}_{s \in S_t \cup \{\varepsilon\}} \cdot \exp(\mathbf{s}^{\mathrm{T}} \cdot \text{FFN}([\mathbf{t}; \mathbf{x}])) \tag{2}$$

where $[\cdot \; ; \; \cdot]$ denotes vector concatenation.

**Joint Modeling** Following Chen et al. (2020), we create a model that *jointly* considers all candidate spans given the template type. We begin by prepending $\mathbf{t}$ to the sequence of span states $\mathbf{X}$ to yield the sequence $(\mathbf{t}, \mathbf{x}_1, \cdots, \mathbf{x}_n)$. This sequence is fed to a different Transformer encoder, which naturally models interactions both between spans and between a span and the template type via self-attention (Vaswani et al., 2017):

$$(\hat{\mathbf{t}}, \hat{\mathbf{x}}_1, \cdots, \hat{\mathbf{x}}_M) = \text{Transformer}(\mathbf{t}, \mathbf{x}_1, \cdots, \mathbf{x}_M) \tag{3}$$

We emphasize that the inputs to the Transformer are embeddings of *spans* (see §3.1) and not tokens, following Chen et al. (2019, 2020).[8] For each $\mathbf{x}_i$, we pass the representation $\hat{\mathbf{x}}_i$ output by the Transformer to a linear layer with output size $|\mathcal{S}|$, the total number of slot types. A softmax activation is then applied over all slot types $s$ that are valid for $t$ (i.e., $s \in S_t \cup \{\varepsilon\}$), with invalid types masked out, yielding the following distribution:

$$P_{\text{joint}}(s|t, \mathbf{x}) \propto \mathbf{1}_{s \in S_t \cup \{\varepsilon\}} \cdot \exp(\mathbf{s}^{\mathrm{T}} \hat{\mathbf{x}}) \tag{4}$$

### 3.3 State Transition Model

A *state transition model* models the environment $E(A, a)$. Recall that a state transition just consists in the generation of a single template, where the current state $A$ is the set of all templates that have been generated up to the current step.

Here, we propose a neural model that produces a representation of $A$. Specifically, we model $A$ as a sequence of vectors $\mathbf{X}_{\text{mem}}(A) \in \mathbb{R}^{M \times d}$ — one $d$-dimensional state vector for each of the $M$ candidate spans $x \in \mathcal{X}$. Each state vector $\mathbf{x}_{\text{mem}} \in \mathbb{R}^d$ acts as a *span memory*, tracking the *use* of that span across generated templates. We model state transitions using a single gated recurrent unit (GRU; Cho et al., 2014). Given the current template assignment $(x : s) \in a$ of a slot type $s$ to a span $x$, the state transition for $A' = A \cup \{a\}$ is given as follows:

$$\mathbf{x}'_{\text{mem}} = \begin{cases} \text{GRU}(\mathbf{x}_{\text{mem}}, [\mathbf{s} \; ; \; \hat{\mathbf{t}}]) & \text{if } s \neq \varepsilon; \\ \mathbf{x}_{\text{mem}} & \text{if } s = \varepsilon. \end{cases} \tag{5}$$

where $\hat{\mathbf{t}}$ is a template embedding given by $\hat{\mathbf{t}} = \mathbf{t}$ when using the independent policy model given in Equation 2 and given by Equation 3 when using the joint model. Intuitively, $\hat{\mathbf{t}}$ is a summarized vector of the current template, akin to the role of the [CLS] token employed in BERT (Devlin et al., 2019). Here, we use a concatenation of the slot type embedding $\mathbf{s}$ and the template vector $\hat{\mathbf{t}}$ as the input to the state transition GRU to track the use of the span.

The input representation of a span $x$ at each step is simply $\mathbf{x} = \mathbf{x}_{\text{enc}} + \mathbf{x}_{\text{mem}}$ — the sum of the original span embeddings $\mathbf{x}_{\text{enc}}$ described in §3.1 and the current memory vector $\mathbf{x}_{\text{mem}}$.

### 3.4 Policy Learning

We use *direct policy learning* (DPL), a type of imitation learning, to train our model. DPL entails training an agent to imitate the behavior of an *interactive demonstrator* as given by optimal actions $a^*$ drawn from some expert policy $\pi^*$, a proposal

[8] Positional embeddings are not needed in this Transformer since sinusoidal embeddings are already added to the span representations.

distribution over actions. This expert policy is computed dynamically based on the current state of the agent, as we describe below. For this reason, the interactive demonstrator is sometimes referred to as a *dynamic oracle* (Goldberg and Nivre, 2012).

*The log-likelihood of the oracle action under the* ITERX *policy model is the reward.* This ensures that the learning problem can be optimized directly using gradient descent, where the objective is given by the expected reward:

$$\mathbb{E}_{\substack{a^* \sim \pi^* \\ A \sim d_{\tilde{\pi}}}} \left[ \sum_{k=0}^{\infty} \gamma^k \log \pi(a^* | t, \mathbf{X}(A)) \right] \quad (6)$$

Here, $\gamma$ is a discount factor, $\tilde{\pi}$ is the *mixed* policy, and states are repeatedly sampled from their induced state distribution $d_{\tilde{\pi}}$. The mixed policy $\tilde{\pi}$ is a mixture of the expert policy and the agent's policy (Ross et al., 2011). Sampling from $\tilde{\pi}$ can thus be described as first sampling some $u \in \{0, 1\}$, then sampling from the agent's parameterized policy $\pi$ if $u = 1$, or sampling an action from the dynamic oracle $\pi^*$ if $u = 0$:

$$\begin{aligned} u &\sim \text{Bernoulli}(\alpha); \\ \hat{a} &\sim u\pi + (1-u)\pi^*. \end{aligned} \quad (7)$$

Here $\alpha$, the *agent roll-out rate*, or the *agent policy mixing rate*, is a hyperparameter that controls the probability of the agent following its own policy vs. the dynamic oracle.

This process resembles *scheduled sampling* (Bengio et al., 2015), a technique commonly employed in training models for sequence generation tasks like machine translation: when updating decoder hidden states, either the gold token $y^*$ or the predicted token $\hat{y}$ may be used, and the decision is made via a random draw. Here, the difference is that we are generating templates at each step instead of tokens.

**Expert Policy** We construct an expert policy based on the agent's policy. At training time, given the set of gold templates $A^*$ and the current state $A$ (all templates predicted thus far), the set $\bar{A} = A^* \setminus A$ contains *all gold templates not yet predicted*. Our expert policy is formulated as

$$\pi^*(a \mid t, \mathbf{X}) \propto \begin{cases} e^{\log \pi(a|t,\mathbf{X})/\beta} & \text{if } a \in \bar{A} \\ 0 & \text{if } a \notin \bar{A} \end{cases} \quad (8)$$

where $\beta$ is a temperature parameter. Intuitively, our expert policy seeks to "please" the agent: a (viable) action's probability under the expert policy is proportional to the probability under the agent's policy. Temperature $\beta$ controls concentration: $\beta \to 0^+$ reduces it to a point distribution over a single action and $\beta \to \infty$ results in equal probability assigned to all remaining gold templates.

### 3.5 Inference

Although many search algorithms for sequence prediction can be employed (e.g. beam search, A*), we find greedy decoding to be effective, and leave further exploration for future work. Setting the initial state $A^{(0)} = \varnothing$, we take actions (i.e., generate templates) by greedy decoding $\hat{a} = \arg\max_a \pi(a \mid t, \mathbf{X})$ for every step. Decoding stops when all spans are assigned the null slot type $\varepsilon$ in $\hat{a}$.

## 4 Experiments

We evaluate ITERX on three datasets: SCIREX (Jain et al., 2020), MUC-4 (Grishman and Sundheim, 1996), and BETTER Phase II English Granular.[9] SCIREX is a challenge dataset for 4-ary relation extraction[10] on full academic articles related to machine learning. MUC-4 and Granular are both traditional template extraction tasks, though they differ in important respects, which we discuss in Appendix C. For summary statistics, see Table 1.

| | SciREX | | | MUC-4 | | | Granular | | |
|---|---|---|---|---|---|---|---|---|---|
| | Train | Dev | Test | Train | Dev | Test | Train | Dev | Test |
| # documents | 306 | 66 | 66 | 1,300 | 200 | 200 | 302 | 34 | 32 |
| # templates | 1,627 | 251 | 271 | 1,114 | 191 | 209 | 610 | 57 | 47 |
| # temp. types | 1 | | | 6 | | | 6 | | |
| # slot types | 4 | | | 5 | | | 92 + 4† | | |

Table 1: Summary statistics of the datasets. † indicates slot types that take non-span values as fillers.

### 4.1 Baselines

**GTT (Du et al., 2021b)** To our knowledge, this is the only prior work to have attempted full template extraction in recent years, and it is thus our primary baseline for comparison on MUC-4.[11] GTT first prepends the document text with the valid template types, then passes the result to a BERT

[9] https://ir.nist.gov/better.

[10] SCIREX also contains a binary relation extraction task, but the binary relation is a subrelation of the 4-ary relation, and thus is subsumed by the more difficult task.

[11] We were unfortunately unable to obtain reasonable performance with GTT on SCIREX, and so do not compare ITERX and GTT on this task.

encoder. A Transformer decoder (whose parameters are shared with the encoder) then generates a linearized sequence of template instances.

**TEMPGEN (Huang et al., 2021)** This is the current state-of-the-art system for REE (the simplified slot-filling entity extraction task) on MUC-4. On SCIREX, TEMPGEN may output multiple relation instances, but only one canonical mention as the filler for each role in the relation. On MUC-4, TEMPGEN outputs a single aggregate template per document, but allows multiple spans to fill a template slot. We make minimal modifications to the TEMPGEN source code to support multi-filler, multi-template prediction on both datasets, allowing for direct comparison to ITERX and GTT on full template extraction.

### 4.2 Metrics

The standard metric for template extraction and REE on MUC-4 is CEAF-REE, proposed in Du et al. (2021a) and then used in Du et al. (2021b) and Huang et al. (2021).[12] CEAF-REE is based on the CEAF metric (Luo, 2005) for coreference resolution, that computes an alignment between gold and predicted entities that maximizes a measure of similarity $\phi$ between aligned entities (e.g. $\text{CEAF}_{\phi_4}$ in coreference resolution). This alignment is subject to the constraint that each reference entity is aligned to at most one predicted entity.

The CEAF-REE *implementation* (henceforth, $\text{CEAF-REE}_{\text{impl}}$) employed in Du et al. (2021a,b) and Huang et al. (2021) unfortunately departs from the stated metric *definition* ($\text{CEAF-REE}_{\text{def}}$) in two ways: (1) it eliminates the constraint on entity alignments and (2) it treats the template *type* as an additional slot when reporting cross-slot averages. For maximally transparent comparisons to prior work, we report scores under both $\text{CEAF-REE}_{\text{def}}$ and $\text{CEAF-REE}_{\text{impl}}$, obtaining state-of-the-art results on MUC-4 with each.

However, we argue that neither $\text{CEAF-REE}_{\text{def}}$ nor $\text{CEAF-REE}_{\text{impl}}$ is consistent with historical evaluation of template extraction systems. $\text{CEAF-REE}_{\text{def}}$ errs in enforcing the entity alignment constraint: doing so effectively requires systems to perform coreference resolution, which is too strict and runs contrary to the original MUC-4 evaluation. By contrast, $\text{CEAF-REE}_{\text{impl}}$ also errs in treating the template type as just another slot: this elides the important distinction between the *kind* of event being described and the participants in that event (§6).

In the interest of clarity, we define a modified version of the CEAF-REE metric that avoids both pitfalls: it relaxes the entity alignment constraint and it does not include template type in cross-slot averages. We call this version CEAF-R**M**E, where "M" stands for *mention* and emphasizes the focus on mention-level rather than entity-level ("E") scoring. Intuitively, relaxing this constraint amounts to placing the burden of coreference resolution on the metric: if the scorer aligns two predicted mentions to the same reference entity, the mentions are implicitly deemed coreferent. See Figure 4 for a comparison among these variants.[13]

For SCIREX, we report $\text{CEAF-REE}_{\text{def}}$ and CEAF-RME. For BETTER Granular, we use its official metrics, described in Appendix D.

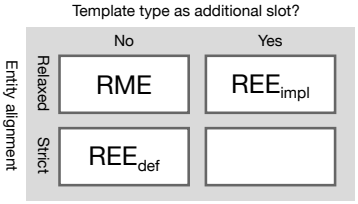


Figure 4: A comparison of the metrics discussed.

### 4.3 Results

**SCIREX** For TEMPGEN, we report models trained with $\text{BART}_{\text{base}}$ and $\text{BART}_{\text{large}}$, where only $\text{BART}_{\text{base}}$ was used in Huang et al. (2021). While BART is an encoder-decoder architecture, ITERX uses only the encoder part, and thus requires about half the pretrained parameters that TEMPGEN does.[14] Even with far fewer parameters, ITERX outperforms the $\text{BART}_{\text{large}}$ baseline by a wide margin. Moreover, our best performing model under $\text{T5}^{\text{enc}}_{\text{large}}$ (Raffel et al., 2020) achieves roughly 2× the performance of TEMPGEN[15] (see Table 2).

**MUC-4** Under the most comparable setting, ITERX outperforms GTT under all metrics by 1–2%, both using $\text{BERT}_{\text{base}}$ (Table 2). With $\text{T5}^{\text{enc}}_{\text{large}}$, ITERX obtains even better performance, with most

[12] The standard metrics for template extraction may be unfamiliar to IE researchers more accustomed to sentence-level *event extraction*. Accordingly, we thoroughly motivate and describe all template extraction metrics we use in this work in Appendix D. What follows is a more abridged discussion.

[13] See github.com/wanmok/iterx for implementations.

[14] We add the superscript "enc" to pretrained models to denote the use of the encoder only (decoder discarded): $\text{T5}^{\text{enc}}_{\text{large}}$.

[15] Replacing BART with T5 in TEMPGEN would have mandated destructive modifications to the pretrained architecture, and we therefore do not report results under this setting.

| Model (Encoder) | SciREX | | | | | | MUC-4 | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CEAF-REE$_{def}$ | | | CEAF-RME | | | CEAF-REE$_{def}$ | | | CEAF-REE$_{impl}$ | | | CEAF-RME | | |
| | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ | P | R | $F_1$ |
| TempGen (BART$_{base}$) | 8.7 | 5.0 | 6.4 | 8.6 | 8.2 | 8.4 | 54.2 | 15.8 | 24.5 | 55.7 | 40.0 | 46.4 | 58.3 | 31.0 | 40.5 |
| TempGen (BART$_{large}$) | 19.9 | 5.0 | 8.0 | 8.9 | 22.3 | 12.7 | **55.8** | 18.9 | 28.3 | **63.7** | 37.4 | 47.2 | **61.3** | 32.9 | 42.8 |
| GTT (BERT$_{base}$) | - | - | - | - | - | - | 54.7 | 23.0 | 32.3 | 61.7 | 42.4 | 50.2 | 55.0 | 36.8 | 44.1 |
| IterX (BERT$_{base}$) | 16.2 | 7.6 | 10.4 | 16.2 | 17.4 | 16.8 | 41.3 | **27.9** | 33.3 | 52.3 | **51.1** | 51.7 | 47.2 | **45.0** | 46.1 |
| IterX (BART$^{enc}_{base}$) | 15.0 | **15.0** | 15.0 | 14.3 | 35.4 | 20.3 | 39.2 | 24.8 | 30.4 | 49.8 | 45.7 | 47.6 | 44.8 | 40.1 | 42.3 |
| IterX (T5$^{enc}_{large}$) | **26.4** | 12.4 | **16.9** | **25.0** | **40.6** | **31.0** | 53.5 | 26.2 | **35.2** | 60.9 | 46.9 | **53.0** | 55.8 | 42.4 | **48.2** |

Table 2: Results on SciREX and MUC-4.

gains coming from increased precision.[16] Furthermore, we note a consistent gap of > 5% $F_1$ between CEAF-RME and CEAF-REE$_{impl}$, which we suspect is due to CEAF-REE$_{impl}$'s inclusion of scores for template type into the aggregated slot $F_1$: as template type scores are higher across models than slot type scores, they are liable to inflate the aggregate score.

**BETTER Granular** We report scores on the English-only Phase II BETTER Granular task using the official BETTER scoring metric in Table 3. Given the complexity of the Granular task, the accompanying difficulty of developing models to perform it, and the lack of existing work on Granular, we report scores only for IterX under T5$^{enc}_{large}$. We intend these to serve as a solid baseline against which future work may be measured.

| Template | | | Slot | | | Combined |
|---|---|---|---|---|---|---|
| P | R | $F_1$ | P | R | $F_1$ | |
| 89.7 | 74.5 | 81.4 | 41.0 | 33.5 | 36.9 | 30.0 |

Table 3: Results on the BETTER Granular dataset. Combined score is Template F1 × Slot F1.

## 5 Analysis

We next conduct ablations to examine how specific aspects of IterX's design affect learning. Here, we focus on SciREX as a case study, as it has the highest average templates per document of the three datasets, allowing us to best investigate the behavior of IterX over long action sequences.

Recall that the dynamic oracle specifies an expert policy $\pi^*$ (Equation 8) from which expert actions $a^*$ are drawn. One design decision concerns the *agent roll-out rate*, $\alpha$, which controls how often we draw from the expert policy vs. the agent policy when making updates. Another decision concerns how *entropic* this policy distribution should be, controlled by the temperature $\beta$. Both decisions reflect a trade-off between exploration and exploitation in the space of action sequences.

**Agent Roll-out Rate $\alpha$** We show how model performance changes as we increase the agent roll-out rate $\alpha \in [0, 1]$ in Figure 5, where $\alpha = 0$ to always following the expert policy, and $\alpha = 1$ corresponds to always updating based on the agent's own policy. The model performs poorly under low $\alpha$, but improves quickly as $\alpha$ increases, reaching a plateau past $\alpha \geq 0.5$. The results are intuitive, as relying more on the expert (lower $\alpha$) for learning would result in a fixed and deterministic set of states that may hinder the agent from visiting new states, which are often encountered at test time. With higher $\alpha$, the agent's behavior is more consistent between train and test time.

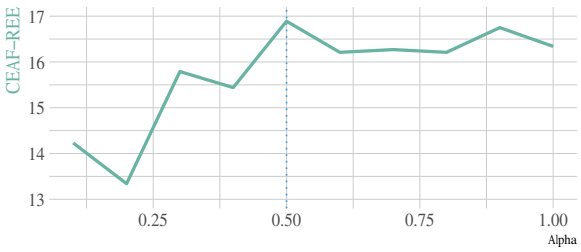


Figure 5: Performance changes on CEAF-REE$_{def}$ with respect to $\alpha$, for which higher means higher probability of rolling out agent's policy for state update.

**Temperature $\beta$** We compare the following four settings for sampling from $\pi^*$, keeping $\alpha = 0$ to control for effects of policy mixing:

- Fixed: Select the next template in the document based on the order as is given in the dataset. In this case, $\beta$ does not come into play. This setting corresponds to the standard practice of using

[16] For GTT, we directly use the output files included in the codebase of Du et al. (2021b). However, we were unsuccessful in adapting their codebase ourselves to make use of T5.

fixed template linearizations (Du et al., 2021b; Huang et al., 2021).

- $\beta \rightarrow 0^+$ (ARGMAX): Select the template that maximizes the likelihood with the system-predicted distributions over slots.
- $\beta = 1$ (XENT): Sample a template according to the distribution defined by the cross entropy between references and predictions.
- $\beta \rightarrow \infty$ (UNIFORM): Sample a template uniformly from the correct template set.

Test set performance for each setting is shown in Table 4. The results for CEAF-RME show a trade-off in precision and recall corresponding to the exploitation-exploration trade-off induced by $\beta$, with the higher $\beta$ (more exploration) of XENT and UNIFORM, yielding higher recall. The trend for CEAF-REE is similar, though less pronounced.

| Approach | CEAF-REE$_{\text{def}}$ | | | CEAF-RME | | |
|---|---|---|---|---|---|---|
| | P | R | $F_1$ | P | R | $F_1$ |
| FIXED | 28.7 | 7.3 | 11.7 | 29.9 | 29.9 | 29.9 |
| ARGMAX | 26.4 | 6.9 | 11.0 | 29.4 | 30.2 | 29.8 |
| XENT | 28.7 | 10.7 | 15.6 | 27.8 | 32.3 | 29.9 |
| UNIFORM | 29.1 | 11.3 | 16.3 | 27.6 | 32.0 | 29.7 |

Table 4: Results with different choices of temperature.

Interestingly, while CEAF-RME $F_1$ scores are consistent across settings, CEAF-REE $F_1$ scores are higher under higher temperature settings. To the extent that the more entropic settings conduce to higher template and mention recall, we would expect these settings to yield more partial-credit template alignments than non-random settings, which tend to focus on correct prediction of fewer templates — thus potentially missing templates entirely and receiving no partial credit.

## 6 Related Work

**Template Extraction** The term *template extraction* was originally proposed in the Message Understanding Conferences (MUC; Sundheim, 1991, *i.a.*) to describe the task of extracting templates from articles. Researchers later focused more heavily on sentence-level IE, especially after the release of the ACE 2005 dataset (Walker et al., 2006). But following renewed interest in document-level IE, researchers (Du et al., 2021b; Huang et al., 2021; Gantt et al., 2022, *i.a.*) have begun to revisit MUC and to develop new template extraction datasets (notably, BETTER Granular).

Traditionally, template extraction comprises two sub-tasks: *template identification*, in which a system identifies and types all templates in a document, and *slot filling* or *role-filler entity extraction* (REE), in which the slots associated with each template are filled with extracted entities. Much recent work in this domain has turned away from the full task, focusing only on REE, which is tantamount to assuming that there is just a single *aggregate* template per document (Patwardhan and Riloff, 2009; Huang and Riloff, 2011, 2012; Du et al., 2021a; Huang et al., 2021).

**Document-Level Relation Extraction** Alongside template extraction, there has been considerable recent interest within IE in various challenging document-level relation extraction objectives, beyond the longstanding and dominant focus on coreference resolution. *Argument linking* — a generalization of semantic role labeling (SRL; Gildea and Jurafsky, 2002) in which a predicate's extra-sentential arguments must also be labeled — is one notable example, and has attracted recent attention through the RAMS (Ebner et al., 2020) and WikiEvents (Li et al., 2021) benchmarks.[17] Prior benchmarks on this task include SemEval 2010 Task 10 (Ruppenhofer et al., 2010), Beyond Nombank (Gerber and Chai, 2010), ONV5 (Moor et al., 2013), and Multi-sentence AMR (O'Gorman et al., 2018). A separate line of work has concentrated on general *N-ary relation extraction* challenge tasks, in which entities participating in the same relation may be scattered widely throughout a document. Beyond SCIREX, PubMed (Quirk and Poon, 2017; Peng et al., 2017) and DocRED (Yao et al., 2019) are two other prominent benchmarks in this area.

**Imitation Learning** Our approach casts the problem of generalized template extraction as a Markov decision process. SEARN (Daumé III et al., 2009) and other related work (Ross et al., 2011; Venkatraman et al., 2015; Chang et al., 2015, i.a.) have considered structured prediction under a reinforcement learning setting. Notably, in dependency parsing, Goldberg and Nivre (2012) proposed the use of a *dynamic oracle* to guide an agent toward the correct parse (see §3.4).

[17] Argument linking also goes by the names *event argument extraction* and *implicit semantic role labeling*, though these terms are not precisely equivalent.

We also employ direct policy learning for optimization of the template extraction MDP, thus reducing the problem to one of supervised sequence learning that is amenable to gradient descent. Such treatment is reminiscent of other similar techniques in NLP. Scheduled sampling (Bengio et al., 2015), for instance, trains a sequence generator with an expert policy consisting of a mixture of the predicted token and the gold token. Relatedly, Levenshtein Transformers (Gu et al., 2019) learn to edit a sequence by imitating an expert policy based on the Levenshtein edit distance.

## 7 Conclusion

We have presented ITERX, a new model for generalized template extraction that iteratively generates templates via a Markov decision process. ITERX demonstrates state-of-the-art performance on two benchmarks in this domain — 4-ary relation extraction on SCIREX and template extraction on MUC-4 — and establishes a strong baseline on a third benchmark, BETTER Granular. In our experiments, we have also shown that imitation learning is a viable paradigm for these problems. We hope that our findings encourage future work to confront the challenge of dealing with documents that describe *multiple* complex events and relations head-on, rather than veiling this difficulty behind simplified task formulations.

## 8 Limitations

Although we believe our iterative extraction paradigm to be promising, we acknowledge that this work is not without limitations. First, ITERX features a significant number of hyperparameters. We found that these generally required some effort to tune for specific datasets, and that there was no single configuration that was uniformly the best across domains. We showed the impact of manipulating some of these hyperparameters in §5. Second, our ITERX implementation iterates over all template types in the template ontology during training and inference, which means that runtime grows linearly in the number of template types. While our framework could in principle support template *type* prediction as well (which would reduce this to $O(1)$), it does not do so in practice, and hence runtime may be long for large ontologies. However, we again stress that actual template ontologies tend to be small.

## Acknowledgments

We thank Nathaniel Weir, Elias Stengel-Eskin, and Patrick Xia for helpful comments and feedback. This work was supported in part by DARPA AIDA (FA8750-18-2-0015) and IARPA BETTER (2019-19051600005). The views and conclusions contained in this work are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, or endorsements of DARPA, IARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

## References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1171–1179.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2058–2066. JMLR.org.

Tongfei Chen, Chetan Naik, Hua He, Pushpendre Rastogi, and Lambert Mathias. 2019. Improving long distance slot carryover in spoken dialogue systems. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 96–105, Florence, Italy. Association for Computational Linguistics.

Yunmo Chen, Tongfei Chen, and Benjamin Van Durme. 2020. Joint modeling of arguments for event understanding. In *Proceedings of the First Workshop on Computational Approaches to Discourse*, pages 96–101, Online. Association for Computational Linguistics.

Nancy Chinchor. 1992. Muc-4 evaluation metrics. In *Proceedings of the 4th Conference on Message Understanding*, MUC4 '92, page 22–29, USA. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Mach. Learn.*, 75(3):297–325.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021a. GRIT: Generative role-filler transformers for document-level event entity extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 634–644, Online. Association for Computational Linguistics.

Xinya Du, Alexander Rush, and Claire Cardie. 2021b. Template filling with generative transformers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 909–914, Online. Association for Computational Linguistics.

Seth Ebner, Patrick Xia, Ryan Culkin, Kyle Rawlins, and Benjamin Van Durme. 2020. Multi-sentence argument linking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8057–8077, Online. Association for Computational Linguistics.

William Gantt, Reno Kriz, Yunmo Chen, Siddharth Vashishtha, and Aaron Steven White. 2022. On event individuation for document-level information extraction. *arXiv preprint arXiv:2212.09702*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden. Association for Computational Linguistics.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India. The COLING 2012 Organizing Committee.

Ralph Grishman and Beth Sundheim. 1996. Message Understanding Conference- 6: A brief history. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189.

Weiwei Gu, Boyuan Zheng, Yunmo Chen, Tongfei Chen, and Benjamin Van Durme. 2022. An empirical study on finding spans. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3976–3983, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kung-Hsiang Huang, Sam Tang, and Nanyun Peng. 2021. Document-level entity-based extraction as template generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5257–5269, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1137–1147, Portland, Oregon, USA. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Modeling textual cohesion for event extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 1664–1670.

Sarthak Jain, Madeleine van Zuylen, Hannaneh Hajishirzi, and Iz Beltagy. 2020. SciREX: A challenge dataset for document-level information extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7506–7516, Online. Association for Computational Linguistics.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Zeyang Lei, Yujiu Yang, Min Yang, and Yi Liu. 2018. A multi-sentiment-resource enhanced attention network for sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*,

pages 758–763, Melbourne, Australia. Association for Computational Linguistics.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 894–908, 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.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Tatjana Moor, Michael Roth, and Anette Frank. 2013. Predicate-specific annotations for implicit role binding: Corpus annotation, data analysis and evaluation experiments. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Short Papers*, pages 369–375, Potsdam, Germany. Association for Computational Linguistics.

James Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 151–160, Singapore. Association for Computational Linguistics.

Nanyun Peng, Hoifung Poon, Chris Quirk, Kristina Toutanova, and Wen-tau Yih. 2017. Cross-sentence n-ary relation extraction with graph LSTMs. *Transactions of the Association for Computational Linguistics*, 5:101–115.

Chris Quirk and Hoifung Poon. 2017. Distant supervision for relation extraction beyond the sentence boundary. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1171–1182, Valencia, Spain. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2010. SemEval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 45–50, Uppsala, Sweden. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Beth M. Sundheim. 1991. Overview of the third Message Understanding Evaluation and Conference. In *Third Message Uunderstanding Conference (MUC-3): Proceedings of a Conference Held in San Diego, California, May 21-23, 1991*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. pages 5998–6008.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 3024–3030. AAAI Press.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 Multilingual Training Corpus LDC2006T06. *Linguistic Data Consortium*.

Patrick Xia, Guanghui Qin, Siddharth Vashishtha, Yunmo Chen, Tongfei Chen, Chandler May, Craig Harman, Kyle Rawlins, Aaron Steven White, and Benjamin Van Durme. 2021. LOME: Large ontology multilingual extraction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 149–159, Online. Association for Computational Linguistics.

Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. 2019. DocRED: A large-scale document-level relation extraction dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 764–777, Florence, Italy. Association for Computational Linguistics.

Mahsa Yarmohammadi, Shijie Wu, Marc Marone, Haoran Xu, Seth Ebner, Guanghui Qin, Yunmo Chen, Jialiang Guo, Craig Harman, Kenton Murray, Aaron Steven White, Mark Dredze, and Benjamin Van Durme. 2021. Everything is all it takes: A multipronged strategy for zero-shot cross-lingual information extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1950–1967, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

## A Terminology

Information Extraction is rife with vague and competing terms for similar concepts, and we recognize some hazard in introducing *generalized template extraction* (GTE) into this landscape. To head off possible confusion, we highlight two important differences between this problem and the well established problem of *event extraction* (EE).

First, EE requires identifying *lexical* event triggers, whereas GTE does not, as template instances do not necessarily have one specific lexical anchor. A document describing a terrorist attack may only *explicitly* describe a series of bombings, and a document describing an epidemic may only *explicitly* state that thousands of people have contracted a particular disease. This property holds of all three datasets we focus on, and can be seen in both Figure 1 and Figure 2. Template anchors are not annotated either for MUC-4 or for SciREX. And while they are annotated for BETTER Granular, they do not factor into scoring. This contrasts with major EE datasets, such as ACE or PropBank, for which typed lexical triggers *must* be extracted.

Second, we take GTE to be a fundamentally document-level task: templates concern events described over an entire document. In practice, EE has historically referred to extraction of predicate-argument structures within a single sentence. One could conceivably argue that this usage has begun to change with the recent interest in argument linking datasets like RAMS (Ebner et al., 2020) and WikiEvents (Li et al., 2021), in which arguments may appear in different sentences from the one containing their predicate. Even so, these cross-sentence arguments are still arguments *of a particular predicate*, *in a particular sentence*. Moreover, the overwhelming majority of arguments in these datasets are sentence-local (Ebner et al., 2020). As emphasized above, templates are not necessarily anchored to particular lexical items. For this reason, they also do not necessarily exhibit the level of locality one finds in EE.

These differences are what motivate the use of CEAF-REE as an evaluation metric, in contrast to the precision, recall, and F1 scores for events and arguments that are typically reported for EE. In brief, it simply is not possible to compute these for GTE in the same way as they are computed for EE. We elaborate on this point in Appendix D.

## B Model Training and Hyperparameters

We implemented our models in PyTorch (Paszke et al., 2019) and AllenNLP (Gardner et al., 2018). We trained all our models with a single NVIDIA RTX6000 GPU. For all experiments that reproduce prior works, we trained models until full convergence under the patience settings provided in the publicly released code. For all IterX models, we trained and tuned hyperparameters under our grid's limit of 24 hours per run, with which we were able to obtain solid performance on all datasets. We performed hyperparameter search manually and report the best performing hyperparameters and the bounds we searched in Table 5, Table 6, and Table 7.

## C Dataset Details

### C.1 MUC-4

The MUC-4 dataset features a total of 1,700 English documents (1,300 for train and 200 each for dev and test) concerning geopolitical conflict and terorrism in South America. Documents are annotated with templates of one of six kinds — `Attack`, `Arson`, `Bombing`, `Murder`, `Robbery`, and `ForcedWorkStoppage` — and may have multiple templates (often of the same type) or no templates at all. All templates contain the same

| Name | Best | Search Bounds |
|---|---|---|
| Encoder | $T5^{enc}_{large}$ | $\{T5^{enc}_{base}, T5^{enc}_{large}, BART^{enc}_{base}, BART^{enc}_{large}, BERT_{base}\}$ |
| LR | $3 \times 10^{-5}$ | $\{1 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ |
| Encoder LR | $1 \times 10^{-5}$ | $1 \times 10^{-5}$ |
| $\alpha$ | 0.5 | $[0, 1]$ |
| $\beta$ | 1.0 | $\{0, 1.0, \infty\}$ |
| $\gamma$ | 1.0 | $\{1.0\}$ |
| $\pi$ Type | Joint | {Independent, Joint} |
| Max #Iteration | 10 | $\{10, 30\}$ |
| Training Spans | Gold | {Gold, Upstream} |
| Avg. training time | | 3 hrs |
| Validation Metric | | CEAF-RME |
| # of parameters* | | ~362 million |

Table 5: Hyperparameters and other reproducibility information for SciREX. "LR" denotes learning rate, "$\pi$ Type" indicates which policy network architecture is used (see §3.2), "Max #Iteration" sets the maximum number of iterations that the model is allowed to perform, and "Training Spans" determines whether the training spans come from gold annotations or the intersection of gold spans and those predicted by the span finding module. *The number of trainable parameters include the parameters from the (best) encoder.

| Name | Best | Search Bounds |
|---|---|---|
| Encoder | $T5^{enc}_{large}$ | $\{T5^{enc}_{base}, T5^{enc}_{large}, BART^{enc}_{base}, BART^{enc}_{large}, BERT_{base}\}$ |
| LR | $3 \times 10^{-5}$ | $\{1 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ |
| Encoder LR | $1 \times 10^{-5}$ | $1 \times 10^{-5}$ |
| $\alpha$ | 0.6 | $[0, 1]$ |
| $\beta$ | 1.0 | $\{0, 1.0, \infty\}$ |
| $\gamma$ | 1.0 | $\{1.0\}$ |
| $\pi$ Type | Independent | {Independent, Joint} |
| Max #Iteration | 14 | $\{14\}$ |
| Training Spans | Upstream | {Gold, Upstream} |
| Avg. training time | | 20 hrs |
| Validation Metric | | CEAF-RME |
| # of parameters | | ~379 million |

Table 6: Hyperparameters and other reproducibility information for MUC-4.

slots. While the original data contains numerous slots, it has become standard practice to evaluate systems on just five of these (apart from the slot for the template's type), all of which take entity-valued fillers: `Perpetrator(Individual)`, `Perpetrator(Organization)`, `Victim`, `Weapon`, and `Target`.

### C.2 BETTER Granular

The BETTER Granular dataset contains documents spanning a number of domains, and, like MUC-4, focuses on six types of complex event, though covering different topics: protests, epidemics, natural disasters, acts of terrorism, incidents of corruption, and (human) migrations. However, Granular is substantially more difficult than MUC-4 in several ways. First, each template type is associated with a *distinct* set of slots. Second, only some of the slots take entities as fillers, whereas others take events, boolean values, or one of a fixed number of strings.

| Name | Best | Search Bounds |
|---|---|---|
| Encoder | $T5^{enc}_{large}$ | $\{T5^{enc}_{base}, T5^{enc}_{large}, BART^{enc}_{base}, BART^{enc}_{large}\}$ |
| Optimizer | AdamW | {AdamW} |
| LR | $3 \times 10^{-5}$ | $\{1 \times 10^{-5}, 3 \times 10^{-5}, 5 \times 10^{-5}\}$ |
| Encoder LR | $1 \times 10^{-5}$ | $1 \times 10^{-5}$ |
| $\alpha$ | 048 | $[0, 1]$ |
| $\beta$ | $\infty$ | $\{0, 1, \infty\}$ |
| $\gamma$ | 1.0 | $\{1.0\}$ |
| $\pi$ Type | Joint | {Independent, Joint} |
| Max #Iteration | 10 | $\{10, 30\}$ |
| Training Spans | Gold | {Gold, Upstream} |
| Avg. training time | | 24 hrs |
| Validation Metric | | BETTER combined score |
| # of parameters | | ~591 million |

Table 7: Hyperparameters and other reproducibility information for Granular.

Finally, the formal evaluation setting for Granular — which we do not adopt in this paper — is zero-shot and cross-lingual: systems trained only on English documents are evaluated exclusively on documents in a different target language.[18] The data used in our experiments is English-only and comprises the "train," "analysis," and "devtest" splits from Phase II of the BETTER program, for which the target language is Farsi.

### C.3 SciREX

The 4-ary SciREX relation extraction task seeks to idenfity entity 4-tuples that describe a `metric` used to evaluate a `method` applied to an ML `task` as realized by a specific `dataset` — e.g. (*span F1*, *BERT*, *SRL*, *ACE 2005*). The challenge of SciREX lies not only in these pieces of information tending to be widely dispersed throughout an article, but also in the fact that only tuples describing *novel* work presented in the paper (and not merely *cited* work) are labeled as gold examples. Following Huang et al. (2021), we frame this as a template extraction task, treating each 4-tuple as a template with four slots.

## D Model Evaluation Details

A key consideration that arises in evaluating generalized template extraction is the need to align predicted and reference templates: a given predicted template may be reasonably similar to multiple different reference templates, and one must decide on a *single* template to use as the reference for each predicted one. Generalized template extraction is similar in this respect to coreference resolution, in

[18] Systems are permitted to use machine-translated versions of this documents, but gold data in the target language is prohibited.

which predicted *entities* may (partially) match multiple reference entities, and one must determine a ground truth alignment. Importantly, this consideration also renders metrics that are traditionally reported for *event extraction* — namely, event and argument precision, recall, and F1 — inappropriate. This is because event extraction is fundamentally a span labeling problem, and the identity of the appropriate reference span is always clear for a given predicted span: either a reference span with the same boundary and type exists or it does not. By contrast, the mapping from prediction to reference for templates is only this transparent in cases of perfectly accurate predictions.

All the evaluation metrics presented in this appendix are, at base, minimal extensions of precision, recall, and F1 to cases where event alignments are both necessary and non-trivial. For CEAF-REE in particular, the various versions of the metric that we discuss (CEAF-REE, CEAF-RME, and Du-CEAF-RME) merely reflect differences in how this alignment should be performed and whether the template type should be treated in the same way as slot types for reporting purposes.

### D.1 MUC-4

MUC-4 evaluation presents a special challenge, owing to its long and complicated history, and to terminological confusion.[19] Here, we discuss CEAF-REE (Du et al., 2021a), the current standard metric for MUC-4 evaluation. We begin with definitions, following with a discussion of some of its problems, and conclude with an extended presentation of our CEAF-RME variant, introduced in §4.

#### D.1.1 CEAF and CEAF-REE: Definitions

The CEAF-REE metric, introduced by Du et al. (2021a), has since been adopted as the standard evaluation metric for MUC-4 (Du et al., 2021b; Huang et al., 2021). To our knowledge, no official scoring script has ever been released for MUC-4, although the metrics used as part of the original evaluation are described in detail in Chinchor (1992). CEAF-REE does not attempt to implement these original metrics, but is rather a lightly adapted version of the widely used CEAF metric for coreference resolution, proposed in Luo (2005).[20] CEAF computes an alignment between reference ($\mathcal{R}$) and system-predicted ($\mathcal{S}$) entities, with each entity represented by a set of coreferent mentions, and with the constraint that each predicted entity is aligned to *at most* one reference entity. This is treated as a maximum bipartite matching problem, in which one seeks the alignment that maximizes the sum of an entity-level similarity function $\phi(R, S)$ over all aligned entities $R \in \mathcal{R}$ and $S \in \mathcal{S}$ within a document. In principle, CEAF is agnostic to the choice of $\phi$, though it is generally desirable that $\phi(R, S) = 0$ when $\nexists x \in S$ such that $x \in R$ and that $\phi(R, S) = 1$ when $R = S$, for reasons described in Luo (2005). In practice, the $\phi_4$ similarity function is most commonly used, defined as the Dice coefficient (or F1 score) between $R$ and $S$:

$$\phi_4 := \frac{2|R \cap S|}{|R| + |S|} \tag{9}$$

The version of CEAF that uses $\phi_4$ is sensibly denoted $\text{CEAF}_{\phi_4}$, or sometimes $\text{CEAF}_e$ ($e$ for *entity*). CEAF-REE differs from $\text{CEAF}_{\phi_4}$ in the following two ways:

- All entities are aligned *within role, conditioned on matching template type*. E.g. only predicted entities for the `Victim` slot in `Bombing` templates would be considered valid candidates for alignment with entities filling the `Victim` slot in the reference templates of the same type.
- A binary similarity function $\phi_{\text{REE}}$ is used, defined as follows:

$$\phi_{\text{REE}}(R, S) := \begin{cases} 1, & \text{if } S \subseteq R \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

#### D.1.2 CEAF-REE: Problems and Solutions

Our principal concerns with CEAF-REE lie with how it has so far been reported and implemented, and with challenges in extending it to the full template extraction task, in which multiple templates of the same type may be present in a document. We elaborate on two issues discussed briefly in §4 and also introduce a third.

First, previous work that reports CEAF-REE treats the template type merely as another slot, with template type labels treated as special "entities" that may fill this slot. This is not necessarily problematic in itself: template type-level metrics are valuable for evaluating system performance. However, it is problematic when reporting (micro

[19] Early writing on MUC-4 used the term *entity* to refer to what the IE community would now call a *mention*. We suspect this is the source of a great deal of confusion.

[20] Luo's motivations for proposing CEAF actually derive in large part from observed shortcomings with the original MUC-4 F1 score. See Luo (2005) for details.

or macro) average CEAF-REE figures *across* slots, as these works do. This is because incorporating the scores for template type into the average elides the distinction between roles (slots) and the kind of event being described (the template type). Moreover, the alignment between slot-filling entities is also *already* conditioned on a match between the template types. There are thus two distinct ways in which information about a system's predictive ability with respect to template type end up in a slot-level average CEAF-REE score. This results in reported values that are very difficult to interpret, and potentially misleading to the extent that these features of CEAF-REE implementations are not made apparent in writing.

Second, the constraint that at most one predicted entity be aligned to each reference entity — stipulated in the metric definition (CEAF-REE$_{\text{def}}$) — is not enforced in the implementation (CEAF-REE$_{\text{impl}}$). Practically, this means that the alignment shown in Figure 6 would receive full credit, whereas it ought to receive a precision score of only 0.75, as Du et al. (2021a) describe. As we argue in §4, we believe this constraint to be overly strict. But this point aside, the discrepancy between definition and implementation is clearly troubling in itself.

Third, full template extraction introduces a *second* maximum bipartite matching problem, which requires aligning predicted and reference *templates* of the same type, and which CEAF-REE (either CEAF-REE$_{\text{def}}$ or CEAF-REE$_{\text{impl}}$) is not natively equipped to handle, given that it operates at the level of slots. Du et al. (2021b) reports CEAF-REE for GTT under an optimal template alignment, but this is obtained via brute-force, enumerating and evaluating every possible alignment, including those between templates of different types. The similarity function, (call it $\phi_{\text{TEMPLATE}}(T_R, T_S)$) that they use for the template alignment problem is *itself* the cross-slot average CEAF-REE$_{\text{impl}}$ score for predicted template $T_S$ and reference template $T_R$. This brute-force template alignment, in conjunction with the hierarchical maximum bipartite matching problem, results in prohibitively long scorer execution times in cases where there is only a modest number of predicted or reference templates of the same type.[21]

[21] Only CEAF-REE$_{\text{def}}$ requires solving a two-level maximum-bipartite matching problem. Since CEAF-REE$_{\text{impl}}$ does not enforce the entity alignment constraint, these alignments will not necessarily be bipartite.

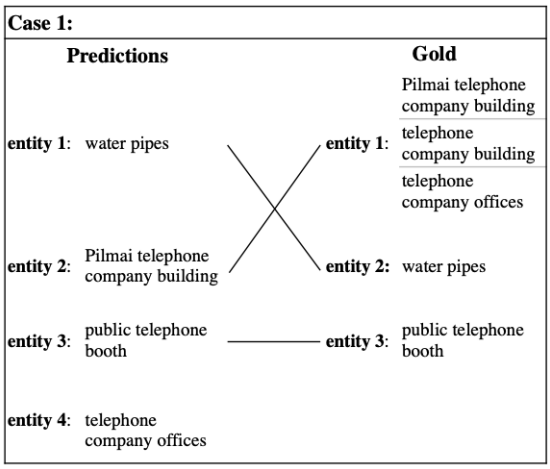


Figure 6: An example alignment between predicted and reference entities from Du et al. (2021a). In past implementations of CEAF-REE, this alignment would receive full credit, rather than being penalized for precision ($P = 0.75$).

In addition to our implementation of CEAF-RME (see below), we also present the first *correct* implementation of CEAF-REE$_{\text{def}}$ that fully addresses the first two points above: template types are no longer treated as additional slots and the entity-level alignment constraint is enforced. On the third point, our implementation efficiently computes optimal template alignments using the Kuhn-Munkres algorithm (Kuhn, 1955; Munkres, 1957). However, even with this efficient implementation, solving such hierarchical maximum bipartite matching problems is computationally intensive.

#### D.1.3 Coreference and CEAF-RME

As CEAF was designed for coreference, it is perhaps unsurprising that coreference considerations introduce a further wrinkle for CEAF-REE. None of the three models described in this work (including ITERX) performs entity coreference. This clearly presents a problem because CEAF-REE$_{\text{def}}$ is an entity-level metric. One way to score these models is simply to treat each extracted mention as a singleton entity and use CEAF-REE$_{\text{def}}$ exactly as defined, and we report these scores in the main text. However, reporting *only* CEAF-REE$_{\text{def}}$ would be undesirable for several reasons:

- It would render our results incomparable to past work, which reports only CEAF-REE$_{\text{impl}}$.
- It would put our work at odds with the overwhelming majority of the template extraction literature, where evaluation criteria focus on string

matching between predicted and reference *mentions*. (The original MUC-4 evaluation only required systems to extract a single representative mention for each entity — not to identify all such mentions.)

- The constraint that at most one predicted entity be aligned to a given reference entity would yield punishingly low scores for systems that are highly effective at extracting relevant spans, but that simply do not perform the additional step of coreference. It is also not especially relevant for certain benchmarks, like SciREX, where coreference is less central to the task.

For these reasons, we disfavor a template extraction metric that *requires* template extraction systems to do coreference. These considerations motivate our introduction of CEAF-RME (*role-filler mention extraction*) — that makes a minimal modification to CEAF-REE$_{\text{def}}$ to address (1) and (2) above. CEAF-RME treats system-predicted mentions as singleton entities, but *deliberately* relaxes the alignment constraint, potentially allowing multiple predicted singletons to map to the same reference entity, effectively pushing the burden of coreference into the metric. We believe CEAF-RME is consistent with what template extraction research has *in fact* historically cared about (identifying *mentions* that fill some slot) while correcting implementation problems with CEAF-REE that produce misleading results.

The micro-average CEAF-RME results that we report on MUC-4 in the main body of the paper are micro-average CEAF-RME scores *under an optimal template alignment* (using CEAF-RME as the template similarity function), which is efficiently obtained using the Kuhn-Munkres algorithm.

### D.2 SciREX

We use the same CEAF-RME implementation for scoring SciREX as we use for MUC-4. We simply treat the SciREX 4-tuples as 4-slot templates, following (Huang et al., 2021). Since SciREX 4-ary relations are untyped, all SciREX "templates" are similarly untyped for scoring purposes.

### D.3 BETTER Granular

Evaluation for the BETTER Granular task bears some core similarity to CEAF-REE$_{\text{def}}$ in that relies on obtaining the alignment between system and reference templates that maximizes some similarity function that decomposes over slot fillers. And just as with (our corrected implementation of) CEAF-REE$_{\text{def}}$, this is achieved via the Kuhn-Munkres algorithm. However, Granular scoring differs from CEAF-REE$_{\text{def}}$ in four key respects. First, the overall system score — referred to as the *combined score* — incorporates both a slot-level F1 score *and* a template-level F1 score:

$$\text{CombinedScore} := \text{TypeF1} \times \text{SlotF1}$$

Only exact matches between system and reference templates types are awarded credit. It is worth noting that because this score does not decompose over template pairs, it cannot be optimized directly using Kuhn-Munkres. In practice, what is optimized is *response gain* — the number of correct slot fillers minus the number of incorrect ones — which provably yields alignments that optimize the combined score within a probabilistic error bound.

The remaining three key differences relate to the calculation of the slot-level F1. For one, Granular slots are not exclusively entity-valued, but may also be event-, (mixed) event-and-entity-, boolean-, and (categorical) string-valued, and different similarity functions must be employed in these different cases. For another, where CEAF-REE defines mentions by their string representation, the Granular score defines mentions based on document offsets. Finally, Granular also requires extraction of temporal and irrealis information for slots, and this in turn impacts the SlotF1 score.

Borrowing terminology from the discussion of MUC-4 above, we describe below how $\phi(R, S)$ is calculated for some generic reference slot filler $R$ and system-predicted slot filler $S$ for slots of different types.

**Boolean and Categorical Values** For boolean- and categorical-string valued slots (i.e., slots taking on one of a predefined set of values). $\phi(R, S) = 1$ if there is an exact match between the system and reference fillers and is 0 otherwise.

**Entities** Unique among the three tasks discussed in this paper, Granular features an explicit preference for *informative* arguments in its scoring structure. In particular, (proper) *name* mentions of an entity are worth more than *nominal* mentions, which in turn are worth more than *pronominal* ones.[22] Thus, if Barack Obama were represented by the reference entity $\{Obama, the\ former\ President, he\}$,

[22] This is precisely the hierarchy described for the *informative argument extraction* task in Li et al. (2021).

full credit would be awarded for returning only the mention *Obama*, less credit for *the former President*, and still less for *he*. Exact point values depend on the mentions present in the reference entity:

- Correct name mentions always receive full credit ($\phi(R, S) = 1$)
- Correct nominal mentions receive half-credit ($\phi(R, S) = 0.5$) if the reference entity additionally contains a name mention, and receive full credit otherwise.
- Correct pronominal mentions receive quarter-credit ($\phi(R, S) = 0.25$) if the reference entity additionally contains *both* a name and a nominal mention, and half-credit if only a nominal mention is featured. (Note that entities will never feature only pronominal mentions.)

**Events** Some Granular slots require *events* as fillers. Like entities, events are represented as sets of mentions (event *anchors* or *triggers*). Unlike entities, there is no informativity hierarchy for events. Furthermore, while event coreference is not a part of the Granular task, annotations for event coreference are nonetheless provided for scoring purposes: $\phi(R, S) = 1.0$ iff $S$ contains only mentions belonging to events in the set of gold coreferent events $R$, and is 0 otherwise, akin to $\phi_{\text{REE}}$.

**Mixed Entities and Events** Some slots may take a mix of events and entities as fillers. Since systems must indicate whether predicted mention clusters are entity- or event-denoting, the same similarity criteria for events and entities as described above are used to compute $\phi$ for events and entities that fill these slots.

**Temporal and Irrealis Information** One of the features of Granular that makes it decidedly more difficult than either MUC-4 or SciREX is the requirement to extract information relating to the time and irrealis status of an event when such information is available in the document. This information is encapsulated in special `time-attachments` and `irrealis` fields associated with each slot-filling entity or event. The former is given as a set of temporal expressions that describe the time at or during which the filler satisfied the role denoted by the slot (e.g. when individuals filling the `tested-count` slot in the Epidemic template were tested for the disease). The latter is given as one of a set of strings that describe whether or how the filler satisfied the role denoted by the slot: `counterfactual`, `hypothetical`, `future`, `unconfirmed`, `unspecified`, and `non-occurrence`. `time-attachments` and `irrealis` are each worth 0.25 points, where exact matches are required for full credit on either and where zero points are awarded otherwise. For slots for which `time-attachments` and `irrealis` are required, the value of $\phi$ appropriate to its filler type is scaled by 0.5 such that the maximum overall score $\phi(R, S)$ for a given filler — factoring in `time-attachments`, `irrealis`, *and* event or entity similarity — is 1.