# BETTER Program Evaluation Plan (Phase 3)

**September 22, 2022**

**Version 27.6**

## Table of Contents

## Change Log

Versions 1-18 covered Phase 1 and are available on Basecamp. Versions 19-24 covered Phase 2, and likewise are available on Basecamp. We begin Phase 3 with Versions 25 and upward, resetting the change log with version 25, March 28.

2022-03-28, v25

- Preface provided for major changes with Phase 3
- Standalone Abstract evaluation dropped; Abstract-enriched comparisons added for Korean Basic
- Basic events documentation updated for Phase 3 events and *money* field

2022-08-30, v26

- Incorporated Sliced Experiment for Basic as Section 2.2.9; updated Preface accordingly
- Incorporated Granular HITL task as Section 2.3.5; updated Preface accordingly
- Incorporated Basic-enriched Granular task as Section 2.3.6 ; updated Preface accordingly.
- Updated Section 1.3 (What's new for Phase 3) to include the above.
- Added Phase 3 Granular template definitions and updated corpus characteristics for Phase 3 (sections 2.3.19, 2.3.20, and 2.3.21).
- Added notifications that ETIPlates require Chinese participants to the Preface as well as to Sections 1.3.8 and 2.3.19.
- Updated the schedule to reflect new Basic evaluation dates, Granular data delivery, and "Sliced" Basic experiment.

2022-09-09, v27

- Updates for the Phase 3 IR experiments

# Preface: Summary of Phase 3 Changes

Several substantive changes have been introduced to the BETTER program for Phase 3. Many sections of the evaluation plan have been updated to reflect these changes. A summary with more details than this Preface can be found in Section 1.3. In the interest of obviating the need for performers to re-read the remainder of the plan, we catalogue the additional changes here.

**Evaluation languages (section 1.1)**

The Phase 3 evaluations will be conducted in three languages, as opposed to the single-language evaluations used for Phases 1 and 2. The Phase 3 languages are Korean, Chinese, and Russian.

**Multilingual test sets (sections 1.3 and** Error! Reference source not found.**)**

The test sets for Phase 3 will be multilingual, with documents from each of the three languages interleaved. The language used in each document will be indicated within a language ID field in the metadata. Performer submissions must accept all three languages in one configuration (it will not be acceptable to submit three monolingual models).

**Elimination of Abstract evaluation; introduction of Abstract-enriched Basic (section 2.1)**

The Abstract evaluation has been eliminated, and the Abstract data will be repurposed. Specifically, there will not be a formal evaluation on the Abstract task in any of the three languages. Instead, performers will submit paired A and B versions of their three allotted run 1 systems, with the A systems pre-trained for Basic as in Phase 1 and Phase 2, and the B systems further pre-trained with Korean Abstract data. (Run 2 configurations are unchanged.)

**Basic money field (sections 2.2.3, 2.2.6, and 6.2.1)**

Reflecting the Phase 3 analytic task focus, a *money* field has been added to Basic events. This field functions as a fourth argument beyond agent, patient, and referred-event. Performers will be required to extract money expressions and attach them to the corresponding Basic events. The money argument will be scored and will participate in the optimizing alignment.

**Basic sliced experiment (section 2.2.9)**

To assess the growth properties of their learning algorithms, performers will provide several versions of their Korean Basic system, trained with increasingly larger splits of the training data. Performers will be asked to run their systems on the Korean basic test data in-house, and submit the resulting output for scoring by T&E. This will inevitably expose the performers to the Korean test set for Basic, which will also be used or the Basic-enriched Granular evaluation (see below).

**Granular HITL task (now confirmed, section 2.3.5)**

Phase 3 will introduce a HITL subtask for Granular. As with the IR HITL evaluation, Granular HITL will notionally involve a second round of Granular evaluations. After the formal Granular runs, the Granular task definition will be extended to incorporate a few new slots. Performers will then be asked to extend their systems through some HITL means of their devising, the goal being to capture these additional Granular characteristics.

**Basic-enriched Granular evaluation (now confirmed, section 2.3.6)**

As a further add-on to the Granular evaluation, performers will be asked to submit a Basic-enriched version of their Granular systems. Towards this end, T&E will provide performers with the Korean Basic data, notionally after the main Granular evaluation; performers will be responsible for submitting an additional set of Granular systems that incorporate the Korean Basic data. While superficially similar to the Abstract-enriched evaluation for Basic, the Basic-enriched Granular evaluation actually will measure the value-added of gold standard Basic data.

**Fine-grained Granular Arguments (Section 2.3.19)**

The Phase 3 Granular ETIPlate event requires fine-grained analysis of entities. In particular, the task mandates that one or more of the participants be Chinese. The consequence of this requirement is that potential ETIPlate events should not be generated by a system, if the putative ETIPlate has no Chinese participants.

**IE-IR integration and the evaluation schedule (section 3.3)**

At issue is which extraction level would confer the most useful integration of IE with IR: Basic or Granular. For Phase 3, Basic will again serve the nexus of integration with IR: IR systems will be expected to integrate Basic event processing into the retrieval chain. To enable a more effective IE-IR integration, additional time is being provided between the Basic and IR evaluations. This reverses a prior premature decision to use Granular for IE-IR integration. An additional condition will also be available for evaluating IR without IE. enrichment.

# 1 BETTER Program Test and Evaluation Introduction

The IARPA BETTER Program is developing new methods for adaptive information retrieval (IR) and information extraction (IE) within a multi-lingual context. The IE component consists of different event extraction tasks at different levels of semantic detail, ranging from abstract and domain-independent to fine-grained (or granular) and domain-specific. The ultimate goal is to develop tools that integrate IE, IR and human-in-the-loop (HITL) techniques to help analysts find relevant documents in languages they do not speak. Multiple groups of researchers ("performers") have been funded to pursue the development of technologies that can support these goals, and a separate Test and Evaluation (T&E) team is responsible for creating the data to train the adaptive systems and to evaluate their progress. The T&E team is also responsible for implementing the infrastructure for execution and scoring of the performers' systems.

BETTER presents a new twist on the familiar theme of making foreign-language material accessible to non-speakers. Where BETTER departs from previous efforts is in a strong focus on automating the cross-lingual aspects of this activity, and consequently reducing the need for language engineering in foreign languages of interest. Central to this endeavor is the requirement that all extraction and retrieval models fielded by the performers be engineered for English only. Extraction and retrieval in the foreign texts is to be achieved through cross-lingual projection.

Performers will be provided with English training data to develop and tune their extraction and retrieval models, but aside from very small sanity-checking samples, there will be no foreign language training data. Performers will have to layer any cross-lingual capabilities onto their English systems using only generally available resources: bilingual or monolingual foreign corpora, general-purpose foreign language tools, and so forth. Further, to ensure strong language agnosticism, performer teams have been asked to avoid the use of foreign-speaking informants.

The goal of building an information extraction system for a foreign language with no task-specific development in that language shows the tremendous technological turnaround of recent language-processing technologies. This notion would have been impossible as recently as a decade ago. What makes this a reachable goal today are multilingual neural network architectures that can simultaneously be trained in multiple languages. These neural cross-lingual representations are key to the cross-lingual projection envisioned in BETTER.

## 1.1 How the evaluation will be carried out

BETTER comprises three phases with durations of 18 months, 12 months and 12 months, respectively. Each phase is driven by one or more overarching analytic tasks (or domains), as well as one or more non-English test languages. The test languages are revealed at the beginning of each phase, but in keeping with the philosophy of pursuing adaptable algorithms, the analytic tasks that drive each phase will become apparent only through the release of training data at various waypoints in each phase.

Each phase has three waypoints, each ending in an evaluation exercise. All three waypoints will involve information extraction (IE), with the final waypoint of each phase also featuring an information retrieval (IR) exercise. The information retrieval task will permit human-in-the-loop (HITL) variants not permitted at the first two waypoints. Table 1 presents an overview of the evaluation structure of the program.

| Phase | Train | Test | Task Domain | Waypoint 1 | Waypoint 2 | Waypoint 3 | | |
|---|---|---|---|---|---|---|---|---|
| **1** | English | Arabic | Protests, corruption, epidemics, terrorism | Abstract IE | Basic IE | Granular IE | IR | IR/HITL |
| **2** | English | Farsi | Disasters, migration | Abstract IE | Basic IE | Granular IE | IR | IR/HITL |
| **3** | English | Korean, Chinese, Russian | To be announced | *(Abstract IE removed from Phase 3)* | Basic IE; Korean Abstract | Granular IE | IR | IR/HITL |

Table 1. Overview of the BETTER Program phases and waypoints.

The information extraction tasks featured in each waypoint are Abstract events (Waypoint 1), Basic events (Waypoint 2), and Granular events or templates (Waypoint 3)[1]. The extraction tasks are ordered by degree of generality, with Abstract intended to apply generally to any task or domain, Basic intended to provide comprehensive coverage of specific topic areas, and Granular representing what an analytic end user might seek for a very specific information-seeking activity. This notion of specific information-seeking goals will be carried through in the IR exercises as well.

In Phase 1, the T&E team provided the relevant English IE training data with the onset of each waypoint. For Phase 2 and Phase 3, only refreshed Basic and Granular data will be provided. There is not expected to be a need for new Abstract data, since the Abstract event task is general enough that it need not be specialized for the particulars of the Phase 2 and Phase 3 analytic tasks. Also note that in keeping with the progression from general to specific, the volume of training data shrinks by around an order of magnitude from one waypoint to the next, with the

[1] The original BAA used different names for the information extraction tasks. Those original terms have been replaced as follows: "coarse-grained" is now called "abstract"; "fine-grained" is now "basic"; and "finer-grained" is now "granular."

Abstract corpus having tens of thousands of events, the Basic corpus having thousands, and the Granular corpus hundreds.

## 1.2 Some technical considerations

The overall technical goals of the BETTER program introduce several important variations on the standard approach to T&E for Information Extraction:

- *Extraction tasks are to be learned from data alone.* While the training data have been and will continue to be created with carefully constructed and refined guidelines, the annotation guidelines themselves will not immediately be shared with the performer teams. This is intended to promote approaches that acquire information extraction models entirely from data, not from some engineering exercise based on the guidelines. The data alone will be the primary driver in how the performer teams pursue their system development. This additionally rules out approaches where performer teams gain an advantage by annotating their own supplementary training data.
- *English training vs. foreign test.* While the *training data* provided to performer teams will always be in English, the *test data* presented to the performer systems will be in a different language. This accords with the expectation that most analysts are monolingual English speakers, while much of the important, real-world information on many important topics is reported in other languages. This presents one of the major challenges being posed to performer systems: to build extraction and document relevance models that can be trained on annotated data in one language, but that manage to perform well on test data in another language.
- *No foreign language extraction engineering.* As previously noted, BETTER performers must build all their information extraction models into their English language system only and perform foreign-language extraction through linguistic projection (a so-called zero-shot approach). In particular, performers are not to create or use foreign-language training data to train their non-English extraction. Performers will not be allowed to acquire target language speakers to support data annotation, acquisition or system development. Team members that coincidentally speak a BETTER target language are not to exploit that expertise to tune their systems.
- *Performers must submit systems, not just system runs.* To better understand the functional parameters of performers' approaches, the T&E team will be running performer systems directly. Toward this end, the program requires that performer teams submit *the systems themselves* as Docker images to be run by the T&E team in the evaluation environment. This is in contrast to the usual evaluation framework of distributing test data to be processed by performer systems in house, and then sent back for evaluation.

  This requirement will enable the test and evaluation team to conduct a range of experiments that control for the amount and type of training data available to the system. This also serves to emphasize that the learning capability that is required to tailor these models to specific analytic tasks should be fully encapsulated in the system itself. The goal here is to lessen reliance an expert team of engineers for system adaptation, thus bringing the technology closer to the closed-shop production environments for which it is intended. The T&E team has developed a submission server for performer teams to submit their systems. The submission process is covered in more detail in Section 4.

- *Evaluation-time retraining.* To explicitly test the ability of systems to be trainable, once uploaded into the T&E computing environment, the systems will be given another opportunity to train on additional English-only annotated training data. This will be done prior to executing the updated systems/models on the test data. As noted in the previous point, this further enables the program to demonstrate truly adaptive systems, running in an environment that is disconnected from its team of developers. It also affords additional opportunities to perform experiments that assess the relationship between training data and performance.
- *Evaluating with a human in the loop.* The third waypoint in each T&E phase will allow the systems to be executed with a "human in the loop," since an overall goal of the BETTER program is to encourage the development of systems that can be quickly tuned by human analysts to support specific analytic tasks. Performers have been asked to explore innovative ways of leveraging the human-in-the-loop (HITL) opportunity. These were exercised during the Phase 1 IR evaluation, and will continue to be explored in Phases 2 and 3.

## 1.3 Additional considerations for Phase 3

Several substantive changes to the BETTER evaluations have been added for Phase 3.

### 1.3.1 Multilingual evaluations.

Phase 3 moves from monolingual evaluations to multilingual ones. There will be three evaluation languages: Korean, Chinese, and Russian. Test data for the three languages will be freely interleaved in the evaluation sets: A Korean document might be followed by a Chinese one, and so forth. In keeping with this, performer submissions are expected to be multilingual, and be able to process all three test languages within the same Docker submission. It will not be acceptable to submit separate monolingual systems for each language.

To support multilingual processing, the language for each test document will be captured in the document's BP JSON encoding. This means that performers will not be required to identify the document language automatically. Details of this are presented in Section 6.1.2.

The multilingual emphasis also has scoring ramifications for IR. See sections 2.4.1 and 5.2.

### 1.3.2 Repurposing the Abstract task

To enable a more concerted performer focus on Basic and Granular extraction, the Abstract evaluation has been eliminated from Phase 3. The first round of evaluation will therefore be the Basic dry run, followed by the formal Basic evaluation.

Although the formal evaluation of Abstract extraction has been eliminated, there remains a BETTER program interest in understanding the utility of Abstract event identification as a pre-training step for Basic extraction. The use case under consideration is whether providing task-agnostic Abstract data for a foreign language improves extraction of more task-focused Basic events in that language. Towards that end, a variant of the Basic extraction task will take place in Phase 3 for the case of Korean.

Along with the Basic training data for Phase 3 (in English), performers will be provided with what was originally intended as the Korean Abstract test set. Performer teams will be asked to submit both an A and a B version of their three allotted Basic extraction submissions as illustrated by the diagram in Figure , below. The A version will be configured as in Phase 1 and Phase 2, trained only with English data (Basic and optionally Abstract). The B version of the submission will additionally be pre-trained with Korean Abstract data. In total, then, six system configurations will be submitted. Only the A versions of the systems will be evaluated with re-training, effectively leading to three evaluation runs:

- *Run 1:* pre-trained Basic extraction
- *Run 2:* re-trained Basic extraction (with re-training at evaluation time)
- *Run 3:* pre-trained Basic extraction, including Korean Abstract pre-training

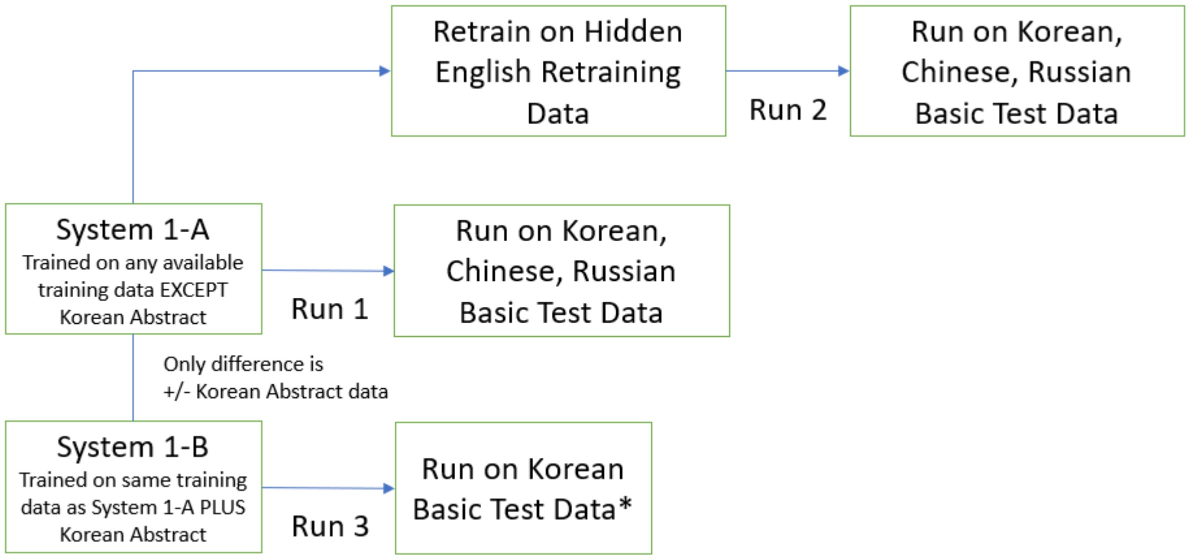

Retrain on Hidden English Retraining Data
Run 2
Run on Korean, Chinese, Russian Basic Test Data
System 1-A
Trained on any available training data EXCEPT Korean Abstract
Run 1
Run on Korean, Chinese, Russian Basic Test Data
Only difference is +/- Korean Abstract data
System 1-B
Trained on same training data as System 1-A PLUS Korean Abstract
Run 3
Run on Korean Basic Test Data*

**Figure 1. Phase 3 Basic evaluation system structure.**

For Run 3, performers will be expected to see just how far they can improve extraction by adding foreign-language Abstract data to their English-projection Basic system. The BETTER program is particularly interested in knowing whether further investments in task-independent foreign resources such as Abstract lead to concrete improvement in cross-language extraction for more task-specific cases (as in Basic). To ensure a valid A-B comparison for this exercise, performers should ensure that their A and B systems differ only in the B system's use of Korean Abstract (and the ancillary mechanisms this requires).

### 1.3.3 Money field of Basic events

Characteristics of the Phase 3 analytic tasks have moved the T&E team to make a modest extension to the Basic extraction task. The precipitating issue has to do with financial events, e.g., *fund-project*, *make-repayment*, and so forth (see Table 3 for a complete inventory). In order to make extraction of these events more useful in an analytic setting, the Basic extraction task has been extended with an additional *money* field.

For example, consider the sentence "*From 2011 to 2018, China granted Sudan an estimated $143 million in loans*," where the loan event is extracted as an instance of *fund-project*. As with Phase 1 and Phase 2, this extraction would yield agent and patient arguments, *China* and *Sudan*, respectively. In addition, the extraction should also include a money field, in this instance capturing the phrase "*an estimated $143 million*".

The money field is rarely expected to be filled outside of the context of financial events. That aside, money extractions will participate in the Basic evaluation process in just the same way as for agents and patients. Money responses will be scored as correct, missing, or spurious in the same way as for agents and patients. The money field will also guide the optimizing aligner, just as do the agent, patient, and referred event fields, meaning that the aligner will increment the value of key-to-response pairings that match on the money field, and decrement the value of those that don't. This will help ensure that performers are correctly given credit when two events are present in the same sentence that are distinguishable only by their money fields (as in "*a $10 million repayment in 2021 and a $20 million repayment in 2022*"). For further details, see sections 2.2.6 and 6.2.1).

### 1.3.4 Basic "Sliced" Experiment

At the request of BETTER program management, an additional experimental condition has been added to the Basic evaluation. The goal here is to chart the growth properties of performer learning algorithms when presented with a new task. Towards this end, performers will be given a common split of the Korean Phase 3 Basic training data into 0%, 5%, 10%, 25%, 50%, and 100% slices. Performers will re-train their systems with these increasing slices of training data and each slice variant will be evaluated individually to assess learning growth curves.

Unlike the formal Basic evaluation, performers will not be asked to submit system Dockers to T&E. Instead, performers will be provided with the unannotated Korean test data and will run their sliced systems in house. System output will then be submitted to T&E for scoring.

For further details, see Section 2.2.9.

### 1.3.5 Integration of IE and IR

With the cancellation of the Abstract evaluation, an additional emphasis naturally falls upon the Basic task. For Phase 3, Basic events are intended to remain the locus of interface with the IR components of BETTER. While some thought had been placed in recent months to have Granular serve as this locus, the greater maturity of Basic processing on the part of all performers argues for keeping Basic as the point of interaction with IR. Evaluation timing and cross-phase comparability also favored keeping Basic as the locus of IE-IR interface. Towards this end, additional time has been given in the schedule between the Basic and IR evaluations.

To demonstrate the impact of IE on IR, an option to the existing run conditions indicates whether the system should use IE during the IR ranking process. Performer dockers must support this option and T&E will run their docker in both modes.

Details of how to configure Docker modules for the IE/no-IE condition can be found in Sections 4.4.1 and 9.

### 1.3.6 Granular HITL Experiment

BETTER program management has expressed interest in assessing how readily performer systems can be updated to new tasks. For this experiment, performers will attempt a HITL approach to updating their systems. In particular, performers will not receive annotated training data for this experiment; instead, they will be provided with detailed specifications for several new slots to be added to their Granular systems. Performers will approach these slot additions through a human-centric method of their choice.

The intent of this experiment is to approximate the kind of effort that an analytic end user might undergo to extend a fielded BETTER system to encompass further functionality. Performers are free to introduce any method of their choice towards this end. The exercise will be time-limited to one week, and performers will be asked to report their level of effort in order to measure system improvement as a function of effort.

For further details, see Section 2.3.5.

### 1.3.7 Basic-Enriched Granular Experiment

This experiment is notionally the Granular counterpart of the Abstract-enriched Basic experiment mentioned above in Section 1.3.2. For this experimental condition, performers will be provided an annotated corpus of Basic data in Korean and will be asked to incorporate these data into re-trained versions of their Granular systems. The systems will be re-evaluated on the Korean Granular test set as a means of assessing the impact of including foreign language Basic data in the development of cross-lingual BETTER-class systems.

There are a few additional considerations to this experiment; for details see Section 2.3.6.

### 1.3.8 Fine-Grained Argument Selection

The ETIPlate Granular task requires fine-grained selection of participating entities. In particular, the task requires that one or another of the participants be a Chinese entity, be it a company, a bank, or the Chinese government itself. This introduces a new layer of processing to the Granular task, which to this point has not required this kind of filtering on arguments.

Details are in Section 2.3.19.

# 2 BETTER Evaluation Tasks

As laid out in the BETTER Program Broad Agency Announcement (BAA),[2] the program will carry out evaluations on three different information extraction (IE) tasks of varying abstraction and an information retrieval (IR) task that exploits the extracted information. Many of the original notions behind the BETTER BAA were informed by prior analyst-oriented event extraction projects, in particular efforts conceived within the CAMEO framework. As the program progressed with data collection and annotation, the specifics of the effort increasingly diverged from this earlier work, but nonetheless retained the original vision of IE tasks that varied on a dimension from abstract to specific.

In broad brushstrokes, the program conceives of three different levels of event extraction: Abstract, Basic, and Granular.

The level that most aligns with prior event extraction efforts is that of Basic events. Basic events are intended to be more or less atomic, meaning non-decomposable. With Basic events, we are not interested in understanding the subtle way in which one event might be a step of another event, or might enable that event as its precondition. Instead, the T&E team selected its Basic event ontology so that events would be minimally overlapping in their semantic scope. A Violence-Kill event is not intended to apply to the exact same language signal as Violence-Wound or Violence-Damage events, for example. This was done in part to give performer systems a better foundation from which to perform cross-lingual projection. In addition, it lets BETTER avoid all the difficult aspects of event coreference, a notoriously complex activity.

With Basic events, topic coverage ranges from the broad (e.g., *communicate-event*) to the generally useful (e.g., commonly noted events related to the justice system or law enforcement), to the highly domain-specific (e.g., medical events like *disease-infects* or *vaccinate*). The Basic ontology in BETTER is not intended to be a complete dictionary of all events of interest to an analyst, at least not as will be explored in the scope of this program. It is understood that for the purposes of BETTER, the ontology will have gaps, though these gaps may be addressed incrementally over the course of the program.

A foundational decision in BETTER was to keep the argument structures of Basic events simple to facilitate cross-lingual projection. While many current approaches proliferate arguments, chief among these FRAMENET, Basic events in BETTER only have an agent, a patient, and in some cases a referred event (typically for events with event complements). Crucially, the patient argument is a semantic patient: it is the entity most affected by the event, and while this is often the same as the syntactic direct object, it isn't always so.

The Abstract event level in BETTER captures a notion of "eventness" independent from the specific of what actually took place. Abstract events have agents and patients, but have no traditional event type. Instead, they are typed according to two orthogonal dimensions: materiality (with values: verbal. material, both, or unknown), and orientation (with values: helpful, harmful, or neutral). Notionally, anything event-like that can be said to have occurred in a narrative would be encodable as an Abstract event. For this reason, most Basic events have a

[2] See https://www.iarpa.gov/index.php/research-programs/better/baa and follow links to Federal Business Opportunities web site (https://www.fbo.gov/). The solicitation number for this program on the FedBizOps web site is IARPA-BAA-18-05.

corresponding Abstract event[3], but the opposite is not true: there are many events in language that can be coded by the Abstract scheme, but for which there is not a Basic event type in the BETTER ontology.

Lastly, we have Granular events. With the intent that these occupy the specific end of the abstract-to-specific dimension, many options were considered for what it would mean to have a finer level of detail than Basic events. What ultimately stuck, given that Basic events are more or less atomic, is that Granular events capture additional facets beyond the agent and the patient. In effect this makes them a little closer to FRAMENET frames or MUC-like templates.

The BETTER information retrieval (IR) evaluation is expected to benefit to a significant degree from the extracted events. The IR evaluation will be driven by sample instances of a target concept, and performer systems will be expected to infer the intended analytic concept from these samples. Underlying this to-be-inferred concept will be a repertoire of events, such that discovery of the intended concept will likely involve generalization and specialization along dimensions defined by these events.

All annotated data in the program – training data, system-generated data, and the reference data against which system output will be scored – will be in a program-specific JSON format called BP JSON, which is documented in Section 6 of this report. For Phase 1 and Phase 2, both the English and Arabic/Farsi data were drawn from news stories captured in the Common Crawl collection.

What follows in this section are the nitty-gritties of the event extraction and information retrieval tasks in BETTER. These primarily document what was decided and evaluated for Phase 1, along with the currently released details for Phase 2. For a comprehensive Phase 2 schedule that describes when data are to be distributed, when evaluations will take place, and other important items on the program calendar, see Section 3.3.

[3] In Abstract, the criteria for coding an event require that an event involve change. However, for a small subset of the Basic events, the notion of "state of affairs" was introduced in order to capture certain states of affairs that were particularly critical for the analytic focus. As a result, a small number of Basic events exist that would not have been captured at Abstract.

## 2.1 Abstract Event Extraction (revised for Phase 3)[4]

The Abstract event extraction task was performed in its full form for Phase 1. Phase 2 included a modified version of the task. While T&E continued to annotate the full Abstract task for training and test data, the *quad class* component of Abstract (as defined below) became optional for Phase 2, and systems were not penalized for failing to emit quad class labels.

For Phase 3, the Abstract evaluation will once again change. At issue is the fact that the Abstract task, by itself, does not appear to provide as much value to end users as was originally hoped. Furthermore, some BETTER performers have not found useful ways to exploit Abstract extractions to boost their downstream Basic or Granular systems. These concerns, and others, have raised questions as to the value of holding a standalone Abstract evaluation in Phase 3.

As a result, and because of the additional Phase 3 demands of evaluating on three different languages, the BETTER program leadership has decided to eliminate the standalone Abstract evaluation for Phase 3.

Abstract will only participate in Phase 3 in a diminished role, as a component of the Basic evaluation. Performers will be provided with Abstract training data in Korean at the same time that Basic training data is distributed. Along with their three normal evaluation submissions for Basic, performers will also have to submit three additional versions of their systems that somehow incorporate the Korean Abstract data.

The rationale here is that while the Abstract event scheme may not map directly to the downstream Basic scheme, some performer experience in Phases 1 and 2 suggests that Abstract may boost Basic performance through pre-training. Paired A-B comparisons will allow T&E to formally assess this effect and quantify the value proposition of Abstract data. Specifically, performers will submit paired A-B versions of their three allotted run 1 configurations (for a total of 6 Docker submissions). The A versions will be pre-trained with English data (Basic and optionally Abstract), using cross-lingual projection to extract Basic events. The B systems will extend the A systems by additionally pre-training with Korean Abstract data. Performers will be expected to make as much use as possible of the Abstract Korean data to improve their Basic extraction scores. The B systems should differ from their A versions only from the use of Korean Abstract data (and any necessary mechanisms this requires).

These paired comparisons will only be done for *run 1* of the Basic system, that is for the pre-trained runs of the systems without evaluation-time retraining. The retrained runs (run 2) will only be evaluated for the core Basic system with no built-in Korean Abstract. This is in the interest of keeping the Basic evaluation timetable to a reasonable length and reducing the evaluation's computational demands.

The T&E team is not expecting performers to have to build a stealth Abstract task into their Korean Basic systems. Rather, the idea is for performers to see just how far they can improve Basic extraction by exploiting target-language Abstract data as pre-training for Basic, and only for the case of Korean. It is not T&E's intention that this exercise rise anywhere close to the demands of the standalone Abstract evaluations in Phases 1 and 2.

[4] Although the Abstract task has a greatly diminished role in Phases 2 and 3 of BETTER, we are keeping the full description of the task in the evaluation plan to document the nature of the data and its role in Phase 1.

### 2.1.1 What are Abstract events?

One hallmark of the Abstract event extraction task is that the varied elements, or "arguments," that are usually distinguished in most domain-specific event representations, such as *who* did *what* to *whom*, *when*, and *where,* are dramatically reduced to just two: Agent and Patient. The notion of Agents and Patients is purely semantic[5] – it does not depend in any way on the particulars of how an event and its participants happen to be expressed syntactically within a sentence. Agents are those things, whether sentient or not, animate or mechanical, that have caused an event, or in some way "set it in motion." Patients, on the other hand, are those things that are most directly affected by the events.

As an example, consider the sentence below in yellow and the abstract events identified in the table below it. This sentence mentions four events that would be captured in the abstract extraction task. There are a number of things to note:

- A noun ("explosion") is sufficient to support the reporting of an event.
- Various important aspects of an event might be mentioned in the text but are left unrecorded by the abstract event annotation – in this case, the location aspect of the explosion event that was "in the factory."
- The explosion event is treated as reflexive, wherein the boiler is both the explosion's agent (causative entity) and patient (affected entity).
- The "agent" of two of the events (the melting of the pipes, the injuring of the people) is itself an event (the explosion).
- In spite of the sentence asserting that the pipes did NOT melt, the notion of the pipes melting is an event that was introduced into the discourse, and on this basis alone is worthy of being identified as an Abstract event.

"According to several witnesses, the boiler explosion in the factory injured three people, but did not melt any of the nearby fuel lines."

| Event ID | Agent(s) | Event anchor(s) | Patient(s) | Quad-Class (Phase 1 only) |
|---|---|---|---|---|
| 1 | "several witnesses" | "according" | "injure", "melt" | Verbal-Neutral |
| 2 | "boiler" | "explosion" | "boiler" | Material-Harmful |
| 3 | "explosion" | "injure" | "three people" | Material-Harmful |
| 4 | "explosion" | "melt" | "the nearby fuel lines" | Material-Harmful |

**Table 2 Events present in the highlighted sentence.**

The Abstract event task also assigns a "Quad-Class" to each event. The quad class, which was only treated as reportable in Phase 1, is a simplified event type that indicates how an event should be classified according to two orthogonal dimensions: Material-Verbal and Helpful-Harmful. Material events are those whose dominant/primary effects are physical (e.g., things moving in space, changes in the physical status of an object, such as breaking, etc.), whereas Verbal events are those whose primary impact are the informational or cognitive change that they bring about (e.g., someone announcing something, or a person learning something, or

[5] Though related, our use of the terms Agent and Patient differs somewhat from how they are used in the thematic role literature.

coming to a new conclusion). Events are categorized as Helpful or Harmful based on the impact they have (or are generally construed as having) on the patient.

The Abstract event extraction task attempts to capture attributes of events that are almost completely domain independent, and in doing so it is hoped that the extraction models derived from this domain-independent event data will prove useful across other aspects of the BETTER Program. In particular, the widespread use of pre-training for neural network models offers a potential avenue for Abstract event training to inform extraction of Basic and Granular events.

Experience in Phase 1 was mixed in this respect, with some performers finding pre-training with Abstract to be helpful, and others less so. The Abstract task will be retained for Phase 2, with some modifications. Specifically, while Farsi test data will we annotated for quad class, scoring of quad class will be optional and performers will be free not to emit quad class markup.

### 2.1.2 Characteristics of the Abstract corpus

Training data for the Abstract event extraction task are presented at the sentence level, on sentences that have been removed from their larger context and annotated in isolation. Each sentence will include annotations for every Abstract event that occurs in the sentence. (Some sentences may contain no Abstract events; many will contain more than one.) Annotations were performed by two highly trained distinct annotators with adjudication from a third where discrepancies were found.[6]

For Phase 1, approximately 5,000 English sentences annotated at the Abstract event level were provided to the performer teams to enable them to train their multi-lingual extraction models. This training corpus contained 15,416 Abstract events, divided into three official segments:

- Training: 12,390 events
- Development test: 1,499 events
- Analysis: 1,527 events

Separately, the T&E team annotated a retraining corpus of 14,793 Abstract events that was only accessible to performer systems in the evaluation environment. Performers were provided the opportunity to retrain their systems during evaluation, and most performer configurations were found to experience a small but measurable improvement from retraining.

The Arabic testing corpus for Phase 1 consisted of 7,839 Abstract events, with a small sample of 517 Arabic Abstract events provided to performers for analysis and debugging purposes. Evaluation was performed under two conditions: with a partial test set of approximately 5,000 events, and with the full test set of approximately 7,500 events. The two conditions presented only negligible score differences, which reflects positively on the stability of the scores.

For Phase 2, no new English Abstract data will be needed. The English training and retraining corpora from Phase 1 are being re-used as is, with a comparable Abstract evaluation corpus created from Farsi news sources.

[6] Historical footnote: early experimentation with cloud labor for this task proved unproductive. The T&E team originally attempted to use Amazon Mechanical Turk for cost savings, but this required cutting the entire abstract markup task into multiple sequential sub-tasks that were small enough they could be performed by cloud workers. Cloud workers tended not to be reliable enough to do this well, and even these simplified tasks ended up requiring correction by trained annotators, leading T&E to go with trained annotators for all but the earliest part of this work.

### 2.1.3 Abstract markup provided to performers

For each Abstract event the following will be provided:

- A list of distinct agents participating in the event
- A list of distinct patients participating in the event
- An *anchor* string, e.g., a word or phrase that best indicates that the event occurred
- The quad class for the event, provided as values on two orthogonal scales
    - Materiality: “material”, “verbal”, “both”, or “unk”
    - Orientation: “helpful”, “harmful”, or “neutral”

For the agent and patient arguments, the Abstract markup provides both a full phrase span, as well as the headword(s) for that span, where appropriate. The head element, defined semantically in BETTER, will be primarily of interest to performers as a locus for cross-language projection, whereas the syntactic full phrase is more of interest to potential end users of the technology. The fact that both heads and full phrases are present in BETTER acknowledges the reality that performers and end users look at the technology through different lenses.

The next few subsections cover important characteristics of Abstract events. Many of these characteristics are foundational to the entire BETTER program, and also apply directly or indirectly to the Basic event extraction task as well.

#### 2.1.3.1 Agents and patients: full spans

Full spans for English are chunk-like: they include the determiner and prenominal modifiers, and an extremely limited set of post-nominal functional arguments. Examples of the intended full span include, “the committee”, “the two committees”, “several previously-appointed ad-hoc committees”, and so forth. Full spans also include names, e.g., “the House Armed Services Committee”. In the case of person names, the full span also includes news-style titles when they occur as premodifiers, e.g., “French president Emmanuel Macron”, but not when they occur as appositives (so that “Emmanuel Macron, France’s boyish president” has two phrases, not one).

For foreign-language data, annotators have been asked to mark full phrases in as close an approximation as possible to what is included in the English full span. Getting the foreign full spans to align exactly with their English counterparts is clearly not possible in all cases, due to a vast array of linguistic differences as regards how tightly modifiers are bound to the semantic head. Nevertheless, the T&E foreign-language annotators have been able to implement full-phrase notions that work for the BETTER languages to date. It will be up to the performer systems to identify full-span like constructs in the foreign language texts.

A few final comments about English full spans: As mentioned, there are rare cases where a post-nominal element is accepted in the full span, chiefly for function words like *member* that are semantically incoherent absent their complements. In other words, for “the members of the House Armed Services Committee”, the full span ranges all the way through the of-complement. These are exceptional cases, reflecting the fact that *member*, unlike more run of the mill function nouns, bears no semantic sort on its own. This is in contrast to cases like *director*, which is also a function noun, but for which the expected denotation is a person. While *director* is person-sorted, with *member*, the sort could be persons (“members of the committee”), countries (“members of the G7”), species (“members of the hominids”) and so forth.

#### 2.1.3.2 **Agents and patients: heads**

The head field for an argument entity is *semantic*, which sometimes places it at odds with the traditional notion of the syntactic headword of a noun phrase. While the overwhelming majority of semantic heads also happen to be noun phrase heads, performers should be aware of the following head span marking conventions (presented here for English).

- Names: the full extent of the named entity is used, minus title premodifiers, so for "French president Emmanuel Macron", the head is "Emmanuel Macron".
- Compound nouns: where a noun-noun or adjective-noun construct has gained sufficient lexical prominence, it is treated as atomic for head phrase purpose. E.g., "free market", "public sector", "private interest", and the like. By lexical prominence, we mean that the compound form has its own entry in the dictionary (the T&E annotators used the Collins English Dictionary, Websters, and similar resources to determine lexical prominence).
- Sort words: occasionally, a sort word displaces an entity out of the head noun position, e.g., "the *Al Badr* mosque", or "the right of *free speech*". In such cases, the displaced entity is used as the head, not the sort word (so "Al Badr" rather than "mosque").
- Generalized quantifiers: this covers some genitive-complement constructs where the syntactic head noun acts as a quantifier over the semantic head. These include numeric forms ("fifty of the *protesters*"), group terms ("a team of 20 *people*"), or unit-inducing terms ("a plot of *land*", "a convoy of *trucks*").
- Partitives: a special case of generalized quantifiers that induces a part-whole relationship, as in "the center of the *city*" or "parts of *Tehran*".
- Member: for terms like *member*, the head phrase inherits the complement, for the same reason that it was present in the full span in the first place, semantic coherency.

#### 2.1.3.3 **Tricky cases: multiple agents, multiple patients, collectives, reflexives, dual objects**

If more than one entity participates as the agent or patient of an Abstract event, performers should report all the participants. For "Macron met Merkel in Brussels", systems would report two agents: "Macron" and "Merkel". This is true in spite of the seeming subject-verb-object form of the sentence, since *meet* is a collective verb taking a set of entities as its agent. Collective readings also hold for "Macron and Merkel met", "Merkel met with Macron", and so forth.

Because arguments are understood in semantic not syntactic terms, some verbs take on reflexive readings. In "John and Fred drove to the store", not only are "John" and "Fred" agents of the event, but they are also the patients, as they are the entities most affected by the act of driving.

For dual-object constructs, as in "John gave Fred a book", an ambiguity exists as to which of the objects should be selected as patient. Performer systems should resolve this ambiguity by anthropocentrism: the entity to be preferred is the person entity, "Fred" in this case. The same anthropocentric rule holds when one of the objects appears in a case-marked prepositional phrase, as in "John gave the book to Fred", where "Fred" remains the preferred patient.

Likewise, affected persons are preferred to instruments, even if those instruments occur in the direct object position. For example, "the protesters threw stones at the soldiers" and "the protesters pelted the soldiers with stones" both result in a Material-Harmful event with "the protesters" as agents and "the soldiers" as patients. This again reflects anthropocentrism.

#### 2.1.3.4 **Handling co-reference**

An early design decision on the part of the BETTER program was that entity and event co-reference would not be foci of the extraction tasks. In particular, BETTER would not have a stand-alone co-reference evaluation layered on top of the event extraction tasks.

Nevertheless, entity co-reference occurs even in the one-sentence pseudo-documents used for the Abstract event evaluation. Where more than one reference to an entity is available (more than one mention), performers are free to select any of the referential mentions. Performers will be scored, however, according to a specificity rubric, with name mentions preferred over non-name nominal mentions, which in turn are preferred over pronoun mentions. For example, in the case of "Mr. Smith, the broker, blames himself", "Mr. Smith" is preferred to "the broker", which is preferred to "himself".

These preferences are reflected in the scoring function. If a name mention is available, but the system selects a nominal mention, it will only receive 50% of the full value for the entity; for selecting a pronoun mention when a name is available, 25% of full value is assigned. If no name mention is available, any nominal mention receives full value, but pronoun mentions only receive 50%. If only a pronoun is available, it receives full value, if reported.

This approach is supported in the Abstract corpus by annotating multiple mention forms for each reportable entity. For example, consider "Macron spoke with Merkel; the president and the chancellor were in full agreement". The agent field for "spoke" would have one entry consisting of the pair {"Macron", "the president"}, and a second entry consisting of {"Merkel", "the chancellor"}. This is encoded through the means of *synsets* – for details, see the description of the BP JSON coding format in section 6 of this report.

#### 2.1.3.5 **Part-whole references**

Occasionally, a descriptor noun phrase applies to more than one entity. Consider, for example "The presidents, Macron and Trump, exchanged a crushing handshake". Here we have a Material-Harmful event with primary anchor "handshake", with "Macron" and "Trump" as collective agents as well as reflexive patients. But what of the descriptor "the presidents"?

The Abstract coding format considers "the presidents" to *include* "Macron" and "Trump" in its denotation. A system that reports "the presidents" as either the agent or patient of the event will receive credit for reporting both the required "Macron" and "Trump" entities. Note, however, that because of its nominal syntactic form, "the presidents" will only provide 50% of the value for "Macron" and 50% of the value for "Trump".

#### 2.1.3.6 **States of affairs are not events**

A key characteristic of the BETTER program is that states of affairs, mental states, and other static situations are not considered events. None of the following, for example, is an event:

- Macron was happy about the meeting
- John is scared of spiders
- The economy is moribund
- Boston temperatures will be in the 70's
- Susan of course wore tie-dye

Most of these non-events are easily identified by the presence of a copula verb (X *is* Y). Even when a copula is not present, however, a verb can be primarily static in intent, as for example with wearing tie-dye.

In contrast, however, we accept as events those verbs that indicate *changes* in states of affairs, mental states, or physical states. For instance, the following are considered events:

- The spider frightened John (change of mental state)
- Susan got dressed in tie-dye (change of physical state)

Notice in particular that "got dressed in tie-dye" represents a change of physical state, and is hence an event, in contrast to the state-of-affairs (and non-event) "wore tie-dye".

#### 2.1.3.7 **The special case of statement verbs**

The Abstract data set considers two different cases for statement verbs. If the statement in and of itself affects an entity, then that entity is its patient. This is often the case for verbs like "criticized", as in "the president (agent) criticized his opponent (patient)".

For statement verbs that have a sentential complement, states of affairs under the scope of the complement are not reportable, but events are. For example, with "the police chief reported that the protests had been peaceful", the copula adjective "was peaceful" indicates a non-reportable state of affairs, and thus does not participate as a patient of "said". However, "the protests" is reportable as a patient of "said" since it is an event – this is not invalidated by the fact that "the protests" is the syntactic subject of the non-reportable copula verb phrase "was peaceful".

#### 2.1.3.8 **Anchors and anchor-like constructs**

The Abstract corpus includes an anchor string for each event. Often the anchor is a main verb, as in "students *protested* in the streets". Sometimes the anchor is an agent nominalization that appears as the argument of some other event. For example, in "police dispersed the protesters", we have both a Material-Harmful event for "dispersed", and a Both-Harmful event implicitly indicated by "protesters" (protests are considered both Verbal and Material, as well as Harmful).

Systems are not required to identify event anchors – these are provided in the training data as a courtesy and aid to performer teams, but for historical reasons are not considered part of the extraction task. We recognize that performers may find anchors helpful to their approaches, perhaps even instrumental, and note that performers can count on having anchors present in both Abstract training and Abstract evaluation data.

Not all verbs anchor events. Many verbs or verb-like forms only appear in light constructs or in argument-binding roles, acting as subservient helpers to a central verb. For example:

- The students staged a demonstration ("staged" is a light verb)
- The demonstration that occurred last week ("occurred" is a helper verb)
- Some faculty participated in the demonstration ("participated" serves to bind the agent)

In such cases, where more than one verb or verb nominalization combine syntactically to predicate the occurrence of an event, the BETTER annotation standard calls for the anchor to be assigned to whichever element most completely captures the event semantics ("demonstration" in all these cases). The other verbs in these constructs typically only contribute tense or aspect, or only serve to attach an argument to the semantically central verb. In the case of these examples, neither “staged,” nor “occurred,” nor “participated” can indicate on their own that the semantical central event is a demonstration.

Although these light, helper, and argument-binding verbs are not semantically central, they may be of value to performers as they train their systems. As a courtesy to performers, and also as an aide-mémoire to annotators, the Abstract annotation tool captures these redundant verbs. In BP JSON, they are made available through the same *synset* mechanism as is used for providing alternative versions of entities. Performers can distinguish these ancillary verbs from the semantically central anchor because only the latter will be assigned as the head.

### 2.1.4 Phase 2 and beyond

For Phase 2 Abstract, systems are not required to emit Quad-Classes with their Abstract events. The “helpful-harmful” and “material-verbal” properties in BP-JSON are now optional. They may be omitted altogether.

For Phase 3, Abstract has been eliminated as a standalone evaluation. Abstract data will only participate in paired comparison evaluations of Korean Basic, with performers submitting both their normal Basic system along with a paired version of the system that incorporates Korean Abstract data by a pre-training method of the performers’ design.

## 2.2 Overview of the Basic Event Extraction Task

This section presents a short introduction to the Basic event extraction task, highlighting the ways that the task follows or differs from the preceding Abstract event extraction task. These notes are primarily intended to clarify the framework of the task. They will provide performers with enough guidance to configure their systems, but this section does not attempt to serve in any way as detailed annotation guidelines.

This is particularly pertinent as concerns the ontology of event types. While we do provide a list of all Basic event types used in Phases 1 and 2, the program's stated intent for this task is that the semantics of the event types be learned empirically from the training data.

### 2.2.1 Abstract vs. Basic, Overall

Many of the design decisions discussed previously for Abstract apply to Basic as well. In particular:

- Basic events have the same *agent* and *patient* arguments as do Abstract events.
- Argument spans are as discussed above in all aspects, including full spans, heads, …
- Arguments of the *agent* and *patient* fields can be entity-typed or event-typed.
- Performers need only report one mention of each expected argument, and where multiple mentions apply, the scorer will accept any applicable version.
- Entity co-reference is handled behind the scenes, by providing alternative string renderings of an argument with the `span-sets` BP JSON construct.
- The same scoring rules apply regarding preferred syntactic form (name over descriptor, descriptor over pronoun), and likewise the same rules apply as regards collective inclusion of multiple arguments under a single descriptor.

Where the tasks differ, in short, is as follows:

- The event type for Basic must come from a pre-selected ontology (see below).
- Basic events have an additional *referred-event* argument, primarily to handle sentential complement verbs, as in "the prosecutor *(agent)* accused Morsi *(patient)* of corruption *(referred event)*."
- Referred event arguments, as should be obvious form the name, are event-typed.
- The Basic task uses complete news stories for testing and evaluation data, as opposed to Abstract's single-sentence pseudo-documents.
- Basic events are scored locally to their enclosing sentences. The scorer will not attempt to map an event from one sentence to an event elsewhere in a document. This effectively amounts to the same scoring strategy as used in Abstract, at least as concerns events.
- Whereas arguments should only be provided if they occur within the same sentence as the event anchor, to obtain full argument score, performers should report the best mention of an argument available anywhere in the document, not just that sentence.

### 2.2.2 Event Types

Where the Basic Event Extraction task differs most significantly from Abstract, is in adopting a specific set of event types. Rather than specifying the event abstractly along the Material-Verbal and Helpful-Harmful dimensions (the so-called "quad-class" event category), Basic events have a single *event-type* field, and there is a specific, limited set of possible event types that are allowed as fillers of the field. Events that do not fall squarely within any of these pre-specified event types are ignored and not annotated. The complete set of event types in the Basic event specification are shown in **Error! Reference source not found.**, below. Most of these were already introduced in Phase 1. The table uses a ✧ symbol to point out those events that are new to Phase 2, or that have had their use expanded from Phase 1 to Phase 2.

Regarding the selection of event types: these event types were chosen to capture either very commonly occurring events (e.g., *Communicate-Event*), or to capture key facets of important domains (e.g., the law enforcement and judicial events). The resulting ontology is only specified as far as needed for the purpose of the BETTER evaluation, and will necessarily have many gaps, compared to such comprehensive efforts as FrameNet. It is expected that each BETTER phase will add incrementally to the ontology, as new analytic tasks are addressed.

### 2.2.3 Basic Event Fields

The following fields are common to all Basic events in BETTER, and have been in use since Phase 1 of the program.[7]

- *event-type,* which is filled with one of the specified event types listed in **Error! Reference source not found.**.
- *agents*, which follows the same reporting rules as for Abstract events;
- *patients,* which likewise follows the Abstract reporting rules, except as noted below for the special case of *Communicate-event*;
- *ref-events*, which captures any referred events;
- *anchors*, which is used to capture the anchor mention for the event; there may be more than one of these because of light verb constructs (just as with Abstract).
- *state-of-affairs*, which is used to indicate that a Basic event is exceptionally to be treated as a state of affairs, as in "union workers protested the economic *stagnation*".

For Phase 3, one additional field has been added to all Basic events.

- *money*, which captures any money expressions associated with an event.

This field was introduced to support the financial transactions that are featured in the event repertoire for Phase 3. The need for this additional argument reflects one of the limitations of the Basic formulation in BETTER, namely the restriction to agent and patient arguments. It is uncontroversial from a semantic perspective that transactions have an argument valence greater than two – witness FrameNet's encoding for "purchase", which has six arguments, with more arguments in the wings with the related commerce_buy frame. In the context of BETTER, however, proliferating arguments à la FrameNet would likely interfere with cross-lingual projection. Hence the choice here to approach transactions with only the lightest of extensions, e.g., an *ad hoc* argument for money phrases connected to an event.

This use of a semantically-typed extension argument is consistent with the approach some performers have used in earlier phases of BETTER. These teams used similar extensions to

[7] These field names are presented as they appear in the BP JSON encoding of Basic events. The nomenclature used here may differ at times from the field names, e.g., we may speak of *referred events* as the fillers of the *ref-event* field.

Basic to attach time and location phrases to Basic event instances, in anticipation of the need to report these facets for Granular extraction.

| | | |
|---|---|---|
| Establish-Project ☼ | Environmental-Event-or-SoA✧ | Law-Enforcement-Investigate |
| ~~Propose-Project~~ ☠ | Natural-Phenomenon-Event-or-SoA✧ | Law-Enforcement-Arrest |
| Sign-Agreement ☼ | Weather-or-Environmental-Damage✧ | Law-Enforcement-Extradite |
| Award-Contract ☼ | Rescue✧ | Law-Enforcement-Other |
| Fund-Project ☼ | Repair✧ | Judicial-Indict |
| Make-Repayment ☼ | Evacuate✧ | Judicial-Prosecute |
| Change-Repayment ☼ | Communicate-Event | Judicial-Convict |
| Employ-Workers ☼ | Coordinated-Comm | Judicial-Sentence |
| Dismiss-Workers ☼ | Conduct-Protest | Judicial-Acquit |
| Construct-Project ☼ | Conduct-Violent-Protest | Judicial-Seize |
| Interrupt-Construction ☼ | Organize-Protest | Judicial-Plead |
| Infrastructure-Operation ☼ | Corruption | Judicial-Other |
| Cyber-Crime-Attack ☼ | Bribery | Suppress-Communication |
| Information-Theft ☼ | Extortion | Suppress-Meeting |
| Information-Release ☼ | Financial-Crime | Suppress-or-Breakup-Protest |
| Interrupt-Operations ☼ | Kidnapping | Economic-Event-or-SoA |
| Cyber-Crime-Other ☼ | Violence | Business-Event-or-SoA |
| Financial-Loss ☼ | Illegal-Entry | Political-Event-or-SoA |
| Pay-Ransom ☼ | Conspiracy | War-Event-or-SoA |
| Cybersecurity-Measure ☼ | Coup | Hospitalize |
| Identify-Vulnerability ☼ | Suppression-of-Free-Speech | Vaccinate |
| Restrict-Travel✧ | Persecution | Test-Patient |
| Loosen-Travel-Restrictions✧ | Other-Crime | Treat-Patient |
| Expel✧ | Violence-Attack | Disease-Outbreak |
| Declare-Emergency✧ | Violence-Bombing | Disease-Infects |
| Famine-Event-or-SoA✧ | Violence-Set-Fire | Disease-Exposes |
| Conduct-Diplomatic-Talks✧ | Violence-Kill | Disease-Kills |
| Provide-Aid✧ | Violence-Wound | Disease-Recovery |
| Aid-Needs✧ | Violence-Damage | Monitor-Disease |
| Death-from-Crisis-Event✧ | Violence-Other | Impose-Quarantine |
| Wounding-from-Crisis-Event✧ | Change-of-Govt | Lift-Quarantine✧ |
| Missing-from-Crisis-Event✧ | Military-Declare-War | Close-Schools |
| Refugee-Movement✧ | Military-Attack | Open-Schools✧ |
| Migrant-Detain✧ | Military-Other | Restrict-Business |
| Migration-Blocked✧ | Political-Election-Event | Loosen-Business-Restrictions✧ |
| Migration-Impeded-Failed✧ | Political-Other | Require-PPE |
| Migrant-Smuggling✧ | Legislative-Action ☼ | Lift-PPE-Requirements✧ |
| Migrant-Relocation✧ | Fiscal-or-Monetary-Action | Distribute-PPE✧ |
| | Other-Government-Action | Cull-Livestock |
| *Colors show family groupings* | Conduct-Meeting | Apply-NPI |
| | Leave-Job | Conduct-Medical-Research✧ |

**Table 3**. Basic event types for Phase 3. ✧= new to Phase 2 ☼ = new to Phase 3 ☠ = unused.

Basic events may have agents and patients, just as with Abstract events. As was the case with the Abstract task, agents are the entities causing the event to occur, while patients are the entities most affected by the event. Because of their sematic specificity, Basic events differ from Abstract events in that they tend to sub-select event-specific sorts for their agents and patients. Judicial-Indict, for example will typically have justice officials as their agents and persons or organizations as their patients.

The *ref-events* field is used to capture subsidiary events that are central to the interpretation of an event. For example, Judicial-Indict will fill the ref-events field with the crime of which someone is accused:

The prosecutor (*agent*) accused (*anchor*) Morsi (*patient*) of corruption (*referred event*)

"The prosecutor" is the agent of the event (by typical linguistic analysis), while "Morsi"is the entity most affected by the event, and hence its patient. The "corruption" event is a subsidiary event that is central to the interpretation of the event overall, and is thus captured as the referred event. Note that in most cases, a referred event will stand in a raising or control relationship with one of the arguments, so that in this case Morsi is interpreted as the agent of the corruption event. From a syntactic perspective, referred events usually occur as some kind of sentential complement of the matrix event.

Continuing our conventions from the Abstract task, only occurring events are to be included as referred events. For example, "I told you that Susan wore tie-dye" does not have any *ref-events* fillers, since "wore" denotes a state of affairs, not an event.

### 2.2.4 The Special Case of Communicate-Event

A particularly common class of sentential complement verbs are statement verbs like "said," which are annotated as a Communicate-Event. Our treatment of event arguments for Communicate-Event differs somewhat from how these statement verbs were handled in the Abstract task. While the speaker of the event continues to be captured by the *agent* field, the *patient* field of Communicate-Event is only used for the affected entities, most frequently the audience of the communication, as in:

The minister (*agent*) briefed Bolsonaro (*patient*) about the outbreak (*ref-event*)

If the Communicate-Event describes other events (in this case "the outbreak"), these events are to fill the *ref*-events field.

A less common case occurs with verbs like "mock", "accuse", or "criticize": with these verbs, the *patient* field is to be filled by the target of the mocking, accusation or criticism, seeing as this target entity is arguably more affected by the event than any audience. For example:

Chavez (*agent*) mocked Bush (*patient*) in front of the UN general assembly (*not an arg*)

The opposition (*agent*) criticized Bolsonaro (*patient*) for his response (*ref-event*) to the deadly outbreak (*ref-event*)

Note that in this final case, there is more than one referred event: both "response" and "outbreak" are included in the *ref-events* field.

### 2.2.5 High-level (Generic) Events

The T&E team approached the design of the Basic event ontology with an eye towards modularity. A quick perusal of the event list shows that there are classes of events related to law enforcement, the administration of justice, violence, government actions, and so forth. The idea here is that these groupings of related events provide a modest range of predicates that can be used to capture the kind of events one might see about these topics in a typical news story. The idea here is to provide enough coverage that when these topics come up, a BETTER system will have something interesting to report to end user at a meaningful level of detail.

Over time, the intent is incrementally to add new groups of events to the ontology to capture a wider range of topics. The Phase 2 ontology, for example, improves coverage for news stories about crises and disasters.

Regarding training and evaluation data, the T&E team has mostly (but not completely) selected data that are rich in the events for which the ontology is well developed. It is not always completely possible to avoid off-topic events, however, and for these, the ontology provides a number of high-level events. These nearly generic events are given names like Environmental-Event-or-SOA, Business-Event-or-SOA, and so forth. These high-level events are invoked to complete the annotation of a story that is mostly about more fully-developed parts of the ontology. Here is an example:

> The companies paid millions in bribes (*Bribery*) to secure
> the contracts (*Business-Event-or-SOA*)

Part of the idea here is that later phases of the BETTER program may expand the ontology to cover some of these areas more fully. An analytic focus on business transactions, for example, might require the T&E team to provide more detailed predicates than just Business-Event-or-SOA. Having marked such examples as "contracts" with high-level generic events would identify those cases in previously annotated data that might be inconsistent with new events in the ontology. These cases could either be re-annotated to the expanded standard, or just passed over by performers when training their systems.

Regarding the "or-SOA" part of these generic events, we are allowing generic events, exceptionally, to also be used to mark states of affairs. This is the case, for example, with "fiscal gap" (*Economic-Event-or-SOA*) and "political turmoil" (*Political-Event-or-SOA*) in the following example:

> A surging fiscal gap and political turmoil […] will remain a challenge, analysts said.

### 2.2.6 Handling money and currency expressions (new to Phase 3)

Starting with Phase 3, performers will be responsible for identifying monetary values when these relate to a Basic event. While this is new to Phase 3 and is particularly motivated by the Phase 3 analytic tasks, this requirement applies to events from prior phases as well. Events for which this might be applicable in Phase 2 include, for example Provide-Aid, as in:

- President Biden announced an additional $800 million in security assistance to Ukraine

Here we have a Provide-Aid event with anchor "assistance", patient "Ukraine", and money field "$800 million."

The money field of Basic events is effectively a fourth argument, joining agent, patient, and referred-event. As such, it participates in scoring just as the other three arguments do, adding value to the argument subscore when correct, and contributing recall or precision errors otherwise.

Regarding the span extent of money expressions, performers will have to identify the currency designator and numeric value, as well as follow these specific guidelines:

- Include alphabetic forms like “million” or “mn”, as in “[$800 million]” or “[$800mn]”
- Include amount-changing modifiers, *e.g.*, "[approximately $5bn]", "[more than $5bn]"
- Exclude repetition and frequency indicators, *e.g.,* "[$5bn] per year", or "another [$5bn]"
- Exclude *relative* forms that lack an actual currency amount, e.g., "70% of the total cost"

In some instances, multiple mentions of the same money expressions may appear together, usually citing different currency values, for example prices in dollars alongside prices in euros.

- Brown Thomas sells Hermès handbags for as much as *10,000 euros ($13,200*)
- a bargain compared with Madrid's *3,717 euros* per square meter *($446* per square foot)

These kinds of same-valued money pairs are to be considered coreferential mentions. Consequently, performer systems should report solely one version of any paired money expressions, e.g., “”10,000 euros” or “$13,200,” but not both. As with fillers for the agent and patient arguments, coreference that applies to the fillers of the money field is captured as string-sets in the BP-encoded training data.

### 2.2.7 Scope and Scoring of Events and Arguments

Data for the Basic event task consist of entire news articles. This has a number of ramifications as concerns scoring the performance of extraction systems.

#### 2.2.7.1 Events are mapped within the scope of a sentence.

To calculate recall and precision, the scorer attempts to map reference and response events to each other. The scorer considers all possible permutations of key-to-response pairings, selecting an overall mapping that optimizes total performer score. By design of the BETTER program, these key-to-response mappings do not require the reference and response to have the same event anchor: the mappings are instead optimized on the basis of event type match and argument alignment. Earlier versions of the scorer were allowed to range across the entire document to find candidate mappings, but this occasionally resulted in response events being mapped to key events elsewhere in the document. While this did result in marginally higher scores, these long-distance mappings were found to be counter-intuitive; they also made it difficult to debug system performance.

Consequently, the scorer has since been configured to map keys and responses only within the scope of the same identical sentence. While this results in marginally lower scores, the mappings are much more intuitive and make system performance more intelligible.

The sentence scoping is guided by manually marked sentence boundaries. These sentence boundaries are provided by sentence-typed segment elements in the BP JSON encoding of a Basic news story.

#### 2.2.7.2 **Event arguments must appear intra-sententially**

Fillers of the agent and patient arguments must have a mention within the same sentence as the anchor of their matrix event. Consider the Coup event anchored on “toppling” in the following news extract (lightly edited for explanatory purposes):

> The aid showed Gulf Arab approval of the Egyptian army’s ousting of Islamist President Mohamed Morsi. […] Political turmoil followed the president’s toppling.

Taken in the full context of the document, the Coup event in the second sentence (the toppling) is understood as having the Egyptian army as its agent. However, a mention of the army would have to be explicitly present in that same sentence for the army to be reportable as the agent of the Coup. Because the army is only mentioned in the extra-sentential context, the implicit agent is not reported.

#### 2.2.7.3 **Best-mention preferences apply to the whole document**

As with the Abstract task, the Basic scorer favors name mentions over nominal mentions, and nominal mentions over pronominal ones. In a departure from Abstract’s sentence-at-a-time approach, however, the Basic task considers all mentions of an argument entity, not just those mentions present in the same sentence as an event anchor. In the previous example, a performer system would get partial credit for reporting the nominal mention of “the president” as the patient of “toppling” in the second sentence. To get full credit for that patient filler, the system would have to select the preferred name mention from the first sentence: “Islamist President Mohamed Morsi.”

In other words, while BETTER does not have a separately-scored coreference task, full-document coreference is still required on the part of performers to achieve maximum score.

#### 2.2.7.4 **Non-Symmetric Entity Co-Reference**

As with the Abstract task, Basic event annotations also capture partial co-reference in order to attempt to most accurately evaluate system output when there are multiple ways in which a set of entities has been described. For example, consider this sentence:

> At the second hearing on Sunday, Tamer el-Firgani said Mursi and 35 other Muslim Brotherhood members had revealed such secrets, and that they should be convicted of espionage.

A system should be awarded full credit if it indicates that the agent of the espionage event (of type Other-Crime) is the set [“Mursi”, “35 other Muslim Brotherhood members”], the conjoined referents of the pronoun “they” in the final clause. Systems should get some credit (but less) for indicating that the event agent was “they”. Likewise, a system should also get partial credit if only one of the two entities is indicated, such as “Mursi”. The annotated BP JSON format captures the notion that one mention can “include” another mention, but not be “identical” to it, through the “includes-relations” section of the event annotation.

#### 2.2.7.5 **Cross-sentence event coreference is not required**

The Basic task does not require performer systems to make cross-sentence judgments about event coreference. In the preceding news extract, for example, there are two references to the coup that overthrew Morsi: one for “ousting” in the first sentence and another for “toppling” in the second. While both these references denote the same coup, because they occur in separate sentences, performers should treat them as two separately reportable events.

#### 2.2.7.6 **Within-sentence event coreference is required**

As with the Abstract task, if multiple mentions of the same event are found in the same sentence, they should be reported as only one event. Examples of this include:

- Protesters rallied in Tahrir Square
- Billed as a million-patriot rally, the demonstration drew scant thousands

In the first example, the implicit demonstration event evoked by “protesters” is considered to be the same event as the event anchored on “rallied”. The annotations for this example will provide both “protesters” and “rallied” as anchors of the event, but only one event Conduct-Protest event will appear in the annotated data.

Likewise, the second sentence is also considered to have only one Conduct-Protest event, with multiple anchors on “rally” and “demonstration”.

However, if different events of the same type occur in the scope of a single sentence, they should all be reported. That two events are not the same can usually be determined by differences in time or place of occurrence, or by having different arguments.

- The protests (*event 1*) drew fewer participants than last year’s demonstrations (*event 2*)

#### 2.2.7.7 **Methodological comments**

By design, the BETTER program attempts to thread the needle on event coreference. The Abstract and Basic tasks are more than just an exercise in listing event mentions; some amount of event identity determination is required. What we seek to avoid, however, is the frustratingly hard problem of having to decompose multi-step event schemas into their component sub-events, and then determine whether a sub-event stands in an identity relationship to a matrix event – or perhaps it’s a component relationship or an enablement relationship.

BETTER attempts to finesse this problem by providing a rich enough ontology that related facets of an event can be expressed as different event types: a clash between police and demonstrators is a Violence-Attack event distinct from the protest itself (a Conduct-Protest event). Likewise, occupation of a government building is an Illegal-Entry event, which again is distinct from any Conduct-Protest that it may relate to.

The scoring rules regarding within-sentence and cross-sentence event identity emerge from the same wellspring. We want performers to go beyond just separately listing the multiple anchors that evoke the same event, be they implicit cases like “demonstrators rallied” or light-verb cases like “held a rally”. The Abstract and Basic event tasks should only be about identifying event instances, not about determining whether multiple instances across a document represent the same event. That kind of full-document event coreference is only indirectly required for some aspects of the Granular template-filling task.

### 2.2.8 Data Sources and Structural Elements of the Basic News Articles

All of the news articles annotated for Basic events have been captured from the news-specific portion of the Common Crawl.[8] These web-hosted news articles come from a great variety of news outlets from around the world.

Because they were harvested from web pages, many of these news stories contain bits of text that are not properly part of the actual story. These include text detritus associated with browser controls, as well as such structural elements of the story as section headers. These extraneous elements are treated in several ways for purpose of the BETTER evaluation.

Browser detritus includes links to related articles, web navigation links, sharing or printing buttons, and so forth. These non-content elements are identified during annotation; when the training and evaluation data are converted to BP JSON format, they are designated as sentence-like elements with the i*gnore* segment type. Performers should not process or report anything scoped inside an *ignore* segment.

In addition to browser detritus, T&E annotators also identify structural elements of a news story, most of which are not considered scorable. They include the following fields:

- Headline: The title of a story (the only scorable element).
- Dateline: When (and sometimes where) this story was written.
- Byline: The author(s) of the story.
- Story-Lead: also called lede, a telegraphic summary of the story, or list of highlights.
- Section-Header: Similar to a story title, but may occur anywhere within an article. These can be especially problematic from a processing perspective, as they often lack sentence ending punctuation, causing them to bleed into subsequent sentences.

Performers should only perform extraction of Basic events from the body of the news story and from the Headline section. All the other specially-designated segments (section headers, ledes, and so forth) should go unprocessed. This processing constraint recognizes that events and entities in the headline are often required to interpret the body of the story, while ledes and headers typically just repeat content that is already present in the body. They are thus redundant from an analytic perspective.

As with sentence markup, these structural elements are available as `segment-section` elements in the BP JSON encoding of the annotated news story.

### 2.2.9 Basic "Sliced" Experiment (new to Phase 3)

There is significant interest among BETTER program management and program partners concerning how much training data may be required to configure a BETTER system to a new task or domain. For Phase 3, T&E has added a task to address this issue directly, the Basic Sliced Experiment.

This experiment will attempt to measure the growth curve properties of the learning methods in performer systems. Towards that end, performers will be provided with slices of the Phase 3 English Training data, consisting of 0% 5%, 10%, 25%, 50%, and 100% of the training set (with each successive slice a proper superset of the previous). Performers will produce multiple

[8] https://commoncrawl.org/

versions of their three core Basic submissions, each trained with one of the six slices. These slice-specific systems will each be evaluated against the Basic test set for Korean, for a total of 18 evaluations, yielding learning growth curves for each core submission.

Some further details follow.

#### 2.2.9.1 Evaluation Runs to Occur In House

T&E and program management have decided to ask performers to run their slice-trained systems in house, using the performers' own hardware and cloud environments. Once a performer team has trained the required constellation of systems, T&E will provide them with an unannotated version of the Korean test data. The performer will be responsible for running their constellation of systems on the test data and will then return the system output to T&E for scoring. For the purpose of the sliced experiment, T&E is not releasing the answer keys for the Basic Korean test set, just the underlying text.

#### 2.2.9.2 Use the Standard Test Environment

For the purpose of the sliced experiment, performers should use the same configuration as T&E provides for Basic evaluations. In particular:

- Restrict runs to a 24-hour time limit
- Disable access to the Internet
- Follow the standard hardware configuration that T&E has separately provided

#### 2.2.9.3 Korean Only

The sliced experiment will only be run on the Korean test set. Performers will thus not be exposed to the Russian and Chinese test data.

#### 2.2.9.4 Basic Improvements are OK

Performers are free to improve their Basic systems before engaging in the sliced experiment. This is an opportunity to incorporate lessons learned from the formal Basic evaluation and fix any bugs or misconfigurations that may have been exposed during the evaluation.

In particular, performers will need to attend to the "Establish-Project hiccup" noted below in Section 2.2.9.8.

#### 2.2.9.5 Only Vary the Training Slices

Once performers have completed any changes they wish to make to their Basic systems, they should freeze the code base before starting the sliced Basic experiment. The constellation of sliced systems should differ from each other only in the size of the English training data used.

#### 2.2.9.6 Ablation Not Required

It is not necessary for performers to strip the sliced systems of Phase 1 or Phase 2 training data. BETTER program management is primarily interested in the effort required to transition a fully configured Basic system to a new task, as opposed to what may be required to bring such a system up from scratch. This necessarily means that event types introduced with the Phase 1 and Phase 2 training data will start out ahead of their Phase 3 counterparts (Communicate-Event, and so forth). It will still be possible to measure growth curves for the events new to Phase 3, which is the primary interest of the BETTER program.

#### 2.2.9.7 Measuring Variance in Training Folds

All performers are required to perform the sliced Basic experiment with the same initial set of slices released to Basecamp by T&E[9]. Some performers have expressed interest, however, in repeating the sliced experiment for more than one configuration of training slices, much as with N-fold validation. T&E will make available a number of additional slice configurations of 0% to 100% of the training data. While only the one initial configuration of slices is required, performers will be free to measure training variance by retraining their systems on these additional slice configurations. As with the required set of slices, performers will run these additional slices in house on the Korean test data, and submit the system output to T&E for scoring.

#### 2.2.9.8 The Establish-Project Hiccup

The original release of the Basic training data left out the Establish-Project event type. In particular, while no text signal was omitted, the annotations for this event type were missing from the release. The missing Establish-Project annotations will be included in the sliced version of the English Basic data, and performers will be responsible for training their systems to extract this event type.

On a related note, earlier descriptions of the Phase 3 Basic event repertoire listed a putative *Propose-Project* event type. This event type ended up being dropped, and will not be included in scoring reports for the sliced experiment.

#### 2.2.9.9 Sliced Scores are not Official

The sliced Basic experiment is considered an add-on to the formal basic evaluation. While there is significant interest in the resulting learning growth curves, the 100% condition for the sliced experiment will not replace performer scores for the formal Basic evaluation.

Performers who may have encountered score-depressing mishaps in the formal evaluation will be encouraged to address these issues before the sliced experiment, thus allowing their systems to shine in the most favorable light. Official scores, however, will remain those from the formal Basic evaluation.

[9] https://3.basecamp.com/3910605/buckets/14032182/vaults/5221394631

## 2.3 Granular Event Extraction

We have alluded to Granular events at various times in this document – in this section, we define them. What follows is documentation of the state of the Granular task up to Phase 2, inclusive. This evaluation plan will be updated to include details of the Phase 3 Granular task prior to release of the Phase 3 data.

### 2.3.1 Preliminary Comments

As previously noted, Granular events are positioned in the BETTER world view on the specific end of the abstract-to-specific continuum. There are many ways in which to consider an event to be more specific than another. In most of the semantic accounts that have played a role in language technology, this corresponds roughly to set inclusion semantics: a kind of event is more specific than another kind of event if the former's instances are a subset of the latter's.

In BETTER, instance inclusion is one way to understand how Basic events are more specific than Abstract events. That is, any class of Basic events is by necessity a more detailed version of some class of Abstract events, so an instance of Communicate-Event, for example, would for example always be an instance of one of the Verbal Abstract classes (Verbal-Helpful, Both-Harmful, and so forth).

A more important way to understand event specificity in BETTER, however, is in terms of information need, particularly as concerns use cases in intelligence analysis. For those familiar with the political coding system CAMEO, Basic events are essentially a broadening of CAMEO to cover extensions beyond CAMEO's original political catalog. Each extension covers the general information need of a particular analytic or ontological topic area, e.g., diseases and health, the legal system, and so forth. To each topic area supported in Basic, BETTER provides a corresponding cluster of Basic event types that are intended to capture the range of events most salient to that topic. For disease and health, this includes such events as Disease-Infects or Disease-Kills, as well as such health care events as Hospitalize or Vaccinate.

In short, the idea with Basic is to capture the information needs of a topic area in a general sense, regardless of the particular analytical use case to which the events will be put. People can be hospitalized (the Hospitalize event) for any of a number of reasons that may have analytic interest: because of a disease outbreak, for example, but also as a result of war, political violence, and so forth.

In contrast, with Granular events, the idea is to capture the information needs of a specific analytic question.

### 2.3.2 Analytic Considerations for Granular Events

The BETTER T&E team explored a range of ways in which this question of specific analytical information need might play out. The guiding criterion that emerged was to imagine an end user approaching an analytic question by collating and organizing information captured in one or more Basic events. A political analyst looking at protests and political unrest, for example, would want to know several aspects of what happened at a particular protest: who participated, whether anyone was arrested, whether a facility was occupied, and so forth.

This same analyst might also want to know facets of the event of interest that are not attached to Basic events, for example time and place of occurrence. An analyst may also what to know whether some aspect of an event actually took place: a planned protest may have been thwarted by law enforcement, for example. Further, the analyst might also want to understand the ramifications of an event beyond the event itself, for example, the way in which Arab Spring protests brought down various governments.

The T&E team considered various ways in which to answer this analytic information need through a straightforward extension of Basic events. We imagined, for instance, a Basic+ approach that would look a bit more like FrameNet, where in addition to having agents and patients, an event might have slots other entities stereotypically related to the event. A putative Violence-Kills+ event might also have slots for the instrument of violence (the weapon), the date and location of the killing, and so on.

The T&E team decided against a FrameNet-like approach, however. There were technical issues related to the inconsistency between the semantically grounded notions of agents and patients, on the one hand, and the more frame-specific approach that FrameNet takes to slots. More critically, however, was the fact that the information needs of an end user might require collating elements that do not really belong in a frame.

To illustrate this point, consider again the previous example of protest events. To Granularize the Conduct-Protest event from the Basic catalogue, one could postulate that location and time ought to be slots of a Conduct-Protest+ frame, and likewise such inherent facets as (say) what was being protested against (or for). But other aspects of interest to an analytic use case cannot be seen so readily as inherent to the protest event itself. To pick some obvious cases: what about any people hurt or arrested as part of any clashes with police or counter-protesters? While clashes, arrests and the like may co-occur with a protest, it surely isn't part of the inherent meaning of conducting a protest that people are injured or arrested. Most pointedly: the agents of the protest are unlikely to have the same semantic agentive role in the co-occurring clashes, injuries, and arrests. Protesters do not injure or arrest themselves, after all. So while facets like injuries and arrests may well be of interest to analysts covering political unrest (and indeed are), it would be mixing semantic apples and oranges to handle those additional information needs through a FrameNet-like extension of the Basic Conduct-Protest event.

### 2.3.3 Granular Events as Templates: Overview

Ultimately, the T&E team decided to fill the need for an analytically focused specific event representation not as a frame-like extension of Basic events, but as complex structured events. In terms of the prior art, Granular events are most like the scenario templates of the venerable MUC evaluations. As with the templates from the MUC exercises, Granular events consist of table-like compilations of entities and facts pertinent to a particular analytic focus. The intention in BETTER is that the collation of these entities and facts is guided by events captured at the Basic level. To make this concrete, if a performer system extracts a Law-Enforcement-Arrest event in a political unrest story, the patient of this event (the arrested individuals) are available as potential fillers of the arrested slot of a Granular Protestplate event. This alignments between Basic events and Template slots is, need it be said, not accidental.

Let's first consider a few general comments about the Granular framework in BETTER. The following are key properties of Granular events:

- *Facets beyond agent and patient:* Granular events do not have the sematic agent and patient arguments of Basic events, but instead have an array of template-specific slots. The Protestplate template, for example, has an agent-like *who* slot for the protesters (often an *agent* of a Conduct-Protest event), but also slots for what is being protested, injuries that occurred, occupied facilities, arrested individuals, and so forth.
- *Multiple kinds of fillers.* Template slots can be filled in one of three ways: by entities, as for example with the *arrested* slot of a Protestplate; by events, as with the *judicial-actions* slot of a Corruplate; or by a fixed set of categorical choices, as with the *type* field of a Terrorplate (one of *bombing*, *arson*, and so on).
- *Whole-document scope:* Unlike the arguments of Basic and Abstract events, Granular slots are expected to be filled from an entire news story, not just a single sentence.
- *Time and space modifiers:* Along with template-specific slots, several of the BETTER Granular structures have designated time and location fields. Time modification can also be applied to individual facets of the event.
- *Irrealis:* Granular structures additionally provide several mechanisms for encoding assertion status (or *irrealis*), for example hypothetical or non-occurrence of events.
- *Outcome slots.* Granular templates also include slots intended to capture any analytically relevant consequences of an event that are not properly part of the event itself. For example, the prosecution of a government official for corruption (a Granular Corruplate event) could lead to protests, which would be captured in the *outcome* slot.
- *Absence of a trigger.* One of the complications with collating more than one facet of an event into a congregate template is that there is often no single precipitating event that triggers the creation of the template. For this reason, the Granular framework does not call out for a template trigger. At least, that's where we started. For more on this issue, see below.

### 2.3.4 Relationship to Scenario Extraction (MUC)

Those who are familiar with the Message Understanding Conferences (MUC), will not have failed to spot the similarities between scenario templates in MUC and Granular events in BETTER. Indeed, the MUC scenario extraction task responded to very much the same analytic information needs as Granular events in BETTER. In both cases, tables (that is, templates) are used to collate information pertinent to an analytic task. In both cases, multiple events can be found in the same news story, requiring performer systems to distinguish when one template ends and another begins.

The difference between the historical MUC and the present-day BETTER is the advent of machine learning. The wide-spread use of trainable methods was only just beginning as the MUC evaluations were winding down. Arguably, the MUC exercises came to an end in part because the black-box evaluation format didn't offer opportunities to move beyond the dominant approach at the time: marathon manual engineering efforts (often fueled by late-night coffee).

Indeed, by the time the final MUC exercise wrapped up in 1998, almost all the performer teams had converged on a common cascaded architecture:

- *Pre-process:* identify named entities, noun phrases, and verb clusters
- *Event detection:* regular-expression entity-verb patterns instantiate partial templates with appropriately filled slots (something like *e1 arrest e2 ☞ protest[arrested = e2]*)
- *Template merging:* merge adjacent partial templates that may consistently be merged

To make these architectures open to machine learning, what was missing at the time were data sets that spoke to the individual layers of the cascade. Informed by historical precedent, the BETTER program attempts in part to address this lack of supporting training data. In particular, with its explicit layer of task-independent Basic events, BETTER attempts to provide an event repertoire from which to support the construction of Granular templates. As mentioned above, it is no accident that most slots of a Granular template have roughly matching Basic events.

### 2.3.5 Granular HITL Task (new to Phase 3)

Starting with Phase 3, the Granular evaluation will be extended to include a secondary HITL evaluation exercise. This Granular HITL task would aim to elucidate how readily an existing Granular extraction system could be extended to capture additional fields of interest. Notionally, we want to approximate the steps that an informed end user would have to go through to make such an extension. The Granular HITLTask is an add-on to the Granular evaluation. While it is mandatory for performers to participate, Granular HITL will not be considered as a component of the official Granular score.

The Granular HITL task will take place shortly after systems are submitted for the Granular evaluation. Performers will build their systems to the Granular training data, just as they did with previous phases, and submit their systems as usual for the Granular evaluation. Post evaluation, the HITL exercise will give performers a limited opportunity to extend their Granular systems to a new aspect of the Phase 3 Granular task, captured as additional slots on the Granular templates. To participate in the exercise, performers will be given written definitions for the slots of interest and will then have to build up those slots during a limited time window. There will be no annotated training data for those slots, only the existing Phase 3 English release, not annotated to the HITL slots. In order to ensure Phase-to-Phase comparability, no further Granular-related data will be released (earlier discussion about the possibility of further data has since been reversed).

During the HITL exercise, performers will have the option to reconfigure their systems by whichever means is natural to their approach. Prompt-based systems, for example, might be enriched with additional prompts. At the close of the HITL period, the updated systems will be re-submitted for evaluation on test data annotated with the HITL slots.

Some further details:

#### 2.3.5.1 Applicable Templates

The Granular HITL task will only involve one of the Phase 3 templates, ETIP (Energy, Transportation, and Infrastructure Projects, see Section 2.3.19). We anticipate that the HITL task will involve no more than five distinct slots that will be added to the ETIP template. The Cyber template (section 2.3.20) will not appear in the Granular HITL exercise.

#### 2.3.5.2 What counts as HITL

It is up to each performer team to define and pursue their own HITL approach. Rounding out a repertoire of prompts is one such approach. Re-annotating the Phase 3 training data is another. The intent here is to avoid having T&E dictate an approach that may or may not work for all performers. Instead, we are looking for performers to identify an approach that naturally suits their system architecture, as this is more likely to result in a positive outcome from the HITL activity. It is also more interesting to BETTER leadership to see each performer system in its best light, so let a thousand flowers bloom (or at least four).

#### 2.3.5.3 Who is That Human in the Loop

Performers are encouraged to target their approach to subject matter experts (analysts), as opposed to engineers. This is in keeping with the notion that end users will seek to extend BETTER-class systems to answer novel information needs that arise for them. This notion meshes well with the HITL task being cast in terms of additional template slots. The end user target is not a completely hard constraint, however. For example, some engineering thought may be involved with the performers' approach, as in a SME/engineer partnership. If performers choose to go with this route, they should be prepared to illustrate how this partnership would work in practice.

#### 2.3.5.4 Scoring Granular HITL

System output for Granular HITL will be scored as usual: combined score, unitary F, argument precision and recall, etc. Scoring will be done with an expanded version of the Granular test set for ETIP, augmented by judgements for the additional HITL slots. T&E will provide several scoring profiles for the evaluation, both the whole-template score, as well as a HITL-only score that would just involve the Granular HITL slots. Performers should still generate entire templates, including the non-HITL slots, as these may be necessary to map keys to responses.

#### 2.3.5.5 Level of Effort Reporting

Beyond raw HITL scores, the BETTER program is most interested in understanding these scores as a function of HITL effort. Towards this end, performers will have to report some measure of the level of effort required to build to the Granular HITL task. At the very least, this should include person-hours as a labor measure, as well as elapsed time required to perform the task. In addition, performers are encouraged to provide any approach-specific level-of-effort metrics that more naturally fit their take on HITL, e.g., number of prompts required, number of documents annotated, and so forth.

#### 2.3.5.6 Excludable Engineering Effort

When calculating level of effort, performers may exclude from the tally the engineering time required to set their systems up for the HITL task. By setup, we mean only the minimum programming required to prepare the code for the specific characteristics HITL task, e.g., editing a Python list of slot names, changing a parameter to point to a different model, and the like. It is fine for this setup to be done by the performer's engineering team; in particular, it is not necessary for this setup to be achievable by an end user. The HITL exercise is not meant to be a stealth transition-to-end-user GUI implementation effort.

#### 2.3.5.7 Time Limit

The Granular HITL exercise will take place over a limited one-week time frame. Performers are free to not use the full time available, and if so, they should report how much of the allotted week they ended up needing. While it would be interesting for a system to be re-configurable in (say) a day or less, the BETTER program is primarily interested in assessing level-of-effort as a factor of labor expended, and not a performer's ability to pull all-nighters.

#### 2.3.5.8 Calendar Considerations

We anticipate running the Granular HITL exercise shortly after submission of the Granular dockers, specifically during the Granular run 2 retraining phase. The chronology would look something like this: (i) performers submit their dockers; (ii) T&E returns the run 1 scores; (iii) performers have several days to analyze their scores; (iv) HITL exercise begins.

#### 2.3.5.9 No Retraining

Unlike the baseline Granular task, Granular HITL will *not* have a Run 2 condition, meaning performer systems will *not* be retrained in situ by the evaluation environment. Indeed, the HITL exercise is meant to shine a light on end-user adaptability; it would be inconsistent to conflate this with automated Run 2 retraining.

#### 2.3.5.10 IR HITL is Still Required

The Granular HITL task does not replace the IR HITL task; Granular HITL is a wholly separate exercise. For phase-to-phase comparability, participation in IR HITL will still be required for Phase 3.

In summary: once again, the Granular HITL results will not become part of the official score for Granular. However, BETTER program leadership hopes that this exercise will shine a light on the question of ease of extensibility, an important concern of BETTER program partners, and a potential interest of future research programs.

### 2.3.6 Basic-Enriched Granular Evaluation (new to Phase 3)

There will be one more add-on task for the Phase 3 Granular evaluation, namely Basic-enriched Granular. For this task, performers will be provided with a corpus of Korean Basic event data, which they will use to pre-train their Granular systems. T&E will re-run these updated systems on the Korean Granular evaluation to assess the value added of pre-training on Basic data in the evaluation language.

This is superficially similar to the Abstract-enriched Basic evaluation, in that a more general data set is used to pre-train for a more specific task. The resemblances, however, end there. For this task, performers will be provided the Korean test data for Basic in order to pre-train their enriched Korean Granular systems. Because the Korean Basic test set consists of nearly the same articles as the Korean Granular test set, performers will effectively have been given gold standard Basic events for the Granular test set.

In other words, the Basic-enriched Granular task will in practice be measuring how well the Granular task can be performed with a Basic event oracle.

The BETTER program management, along with T&E, decided not to keep the Korean Basic test data hidden from performers for two reasons. To begin with, the program did not have the resources to come up with an entirely new set of Korean Basic data. These would have to have been rich in Phase 3 Basic events without involving substantially the same underlying real-world events as the existing Korean test data. In addition, T&E had already determined that the Korean test data for Basic would have to be released to performers during the Basic growth curve task.

Further details of the Basic-enriched Granular task will be provided as they become specified.

### 2.3.7 Varieties of Granular Templates

For Phase 1, the T&E team defined four Granular templates:

- *Protestplate:* these events capture protests, peaceful or otherwise, along with related events such as arrests, injuries, and the like.
- *Corruplate:* these events capture formal accusations of government corruption, and prosecutions of cases against government officials.
- *Terrorplate:* these events capture terrorist attacks, and cover much of the same information needs as are addressed by the scenario templates of MUC3 and MUC4.
- *Epidemiplate:* these events capture disease outbreaks, as measured by infections, hospitalizations, fatalities, and related fields.

In support of Phase 2, the T&E team defined two additional templates:

- *Disasterplate:* these events capture major disasters (hurricanes, etc.), including such facets as the entities affected, and related events such as rescue attempts and aid.
- *Displacementplate:* these events capture large-scale migration of refugees, resulting from war, famine, natural disasters, or other causes.

For Phase 3, the T&E Team added two more templates:

- *ETIPlate:* for Energy and Transportation Infrastructure Projects (or ETIP). These Granular events capture development projects, such as railroads and power plants, that are associated with the Chinese Belt and Road Initiative.
- *Cybercrimeplate:* these events cover cybercrime activities, typically incursions, denial-of-service attacks, and similar criminal actions.

Note that each of these template types corresponds to a particular analytic focus, with each template collating that topic's specific information needs:

It is important to note that a news story may have more than one relevant template. Sometimes these are the same kind of template, for example when a news story reports on current protests in (say) the Middle East (a Protestplate), while also referring to the Arab Spring (a different Protestplate). Often, what distinguishes one Granular event from another Granular event of the same type are such factors as time and location, as with contemporary protests versus the Arab Spring protests around 2011. Multiple same-type templates can also be called for when they apply to multiple individuals, for instance multiple politicians accused of different instances of corruption (Corruplate events).

There can also be more than one kind of template in the same news story. The Phase 1 data contained, for example, cases of protests held in response to a global disease outbreak, corruption trials held in response to protests, as well as the convers: protests held in response to the outcomes of corruption trials. Performer systems should thus be prepared to expect news stories to cover more than one analytic topic; shortcuts that attempt to treat news stories as having only a single focus are likely to underperform.

### 2.3.8 Irrealis

For Abstract and Basic, all events are reportable, even if they are asserted to not have occurred. For example, "over 100 students … were not arrested" generates an Abstract Material-Harmful event and a Basic Law-Enforcement-Arrest event, each having "over 100 students" as its patient.

Nothing further is done to record the fact that the event did not occur. With Granular events, in contrast, BETTER provides an explicit *irrealis* marker to capture non-occurrence and related properties of events, the possible values of which are "counterfactual", "hypothetical", "future", "unconfirmed", "unspecified", and "non-occurrence".

In BETTER's treatment, Irrealis is not a property of an entity or event, but of the attachment of that entity or event to a template slot (more on this below). Some examples:

- "over 100 students … were not arrested" (in a protest): the entity "over 100 students" is added to the *arrested* slot of the corresponding Protestplate, with irrealis marker *counterfactual*.
- "According to official figures, about 640 people were killed, but Muslim Brotherhood put the toll at more than 2,000" (in a protest): in this instance, the *killed* slot of the corresponding Protesplate would have two entries: "about 640 people" with no irrealis marker, and "more than 2.000" with irrealis marker *unconfirmed*.

Looking somewhat ahead, the BP JSON encoding of this second example is roughly as follows, with string set *ss-14* corresponding to "more than 2,000", and *ss-13* to "about 640 people".

```
{
  "template-id": "template-1",
  "template-type": "Protestplate",
  "killed":
    [ { "ssid": "ss-14",
        "irrealis": "unconfirmed" },
      { "ssid": "ss-13" } ],
  ... }
```

To see why irrealis is a property of slot fills, and not entities consider a fuller version of an earlier example: "over 100 students protested, but were not arrested". The same 100 students participate in a Protestplate with no irrealis as protesters (the *who* slot), but with *counterfactual* irrealis where they appear in the *arrested* slot. The BP JSON for this case would be roughly as follows, with *ss-15* being the string set for "over 100 students".

```
{
  "template-id": "template-2",
  "template-type": "Protestplate",
  "who":
    [{ "ssid": "ss-15" } ],
  "arrested":
    [ { "ssid": "ss-15",
        "irrealis": "counterfactual” },
  ... }
```

### 2.3.9 Outcome Slots

Granular templates are complex structured events that often capture multiple ad hoc related events. A key consideration with this approach is what the boundary should be between sub-events that belong as integral parts of the template, and those that do not. As a general rule, only events that are immediately related to the analytic scope of the template are fodder for becoming part of the template slot structure properly speaking. Some examples:

- Protestplates capture the events that took place during the protest: individuals getting arrested or hurt, buildings being occupied, and so forth.
- Corruplates capture the judicial actions taken against the officials being prosecuted.
- Human displacement templates capture the events precipitating the displacement, the injuries or deaths that occurred along the way, and so forth.

Often, however, there may be an interest in the outcomes of events that are not properly part of the event itself. In the case of protests, for example, an eventual consequence of widespread civil unrest may be a change of government. It is not unusual for a news story to provide this kind of linkage between an event and a long-term consequence of the event. In BETTER, these kinds of linkages are captured through a template's *outcome* slot(s).

For Phase 1, several Granular outcome slots were provided:

- *Outcome:* for events that are understood in the news story to have eventually arisen as a result of the template. E.g., Arab Spring protests leading to a change of government in Tunisia.
- *Outcome-hypothetical*: for events that the news story only presents as hypothetically following from the Granular event. E.g., a potential sentence that a political figure might be facing in a Corruplate.
- *Outcome-averted*: events that are explicitly indicated to not have occurred during the course of the Granular event. E.g., violent clashes that are said to not have occurred during a protest.

For Phase 2, this multiplicity of outcome slots is replaced with a single *outcome* slot. The distinction between actual, hypothetical, and averted outcomes is captured instead by including an irrealis marker (see above). This aligns the assertion status of outcomes align with the irrealis treatment used in the remainder of Granular slots.

### 2.3.10 Temporal modifiers

Another modality modification that may appear on the slots of Granular templates is temporal modification. This is particularly pertinent to Epidemiplates, where enumerations of infections, hospitalizations, or fatalities are often predicated by a time period. A temporal modifier is attached only in those cases where the news story clearly indicates that it applies to a particular aspect of a Granular event, and not to the Granular event as a whole. This mechanism is not frequently used, especially outside of the scope of an Epidemiplate.

### 2.3.11 The Question of Triggering Events

Both the Basic and Abstract tasks in BETTER encompass the notion of an event trigger. Because triggers are such an effective part of detecting these events, the T&E team looked at length at how to provide an analogous construct for Granular events. We found this very difficult to do.

The issue is that Granular templates are drawn from multiple loci in a news story: as a result, there isn't always a single triggering event to which the template can be tied. Going further, while our expectations are that journalistic style should dictate the use of topic sentences from which a Granular trigger might be derived, this turns out not to be always the case, especially when the event in question is not the central focus of the story.

The disease outbreak templates are a case in point. Outbreak stories often center on listings of infections or fatalities, and don't always have an explicit topic statement that would yield, say, an informative and putatively triggering Disease-Outbreak Basic event. A news story about yellow fever, for example, will not necessarily begin with a convenient topic statement such as "an epidemic of yellow fever is raging through the region". One is just as likely to see something like "yellow fever has claimed another 5 fatalities". In other words, the actual data do not support the notion of a triggering event type that predictably indicates that a Granular Epidemiplate should be instantiated.

There are similar issues with the corruption templates (there aren't always explicit Corruption events), and likewise the terrorism templates (one doesn't always get "terror attack in Cairo" or the like). Of the four original Phase 1 template types, only Protestplates might recurringly be triggered by a reasonably predictive Conduct-Protest event. For the rest of the Granular event types, however, the T&E team judged that we would too frequently be conjuring questionable triggers on an ad hoc basis, merely to ensure that the trigger slot was filled. Such an approach did not seem like it would provide a reliable and useful signal to performers, and the T&E team opted against trying to impose a trigger-based framework onto data that resisted this notion.

### 2.3.12 Repertoire of Granular Events: Preliminaries

In the following sections we present some details about the Phase 1 and Phase 2 templates, and for two of them (protest and corruption templates) we describe some of the issues that attend to capturing the information from news articles in these template structures. For the remaining Phase 1 templates (terrorism, disease outbreak), and for the Phase 2 templates (major disasters, human displacement), we simply provide a complete list of the template fields.

Analogous to event anchors in Abstract and Basic tasks, all template types have a `template-anchor` field. However, while the anchor of Abstract and Basic events can reliably be taken as a head string for the event, that generalization is not possible with Granular events. This is as a result of the fact that Granular events may not have a reliable trigger. Performers should be exceedingly wary of treating the template anchor as a stealth trigger, since this could often lead their systems astray. Regarding scoring, the template anchor is treated in an analogous way to event anchors in the Abstract and Basic tasks – *they are not considered in any way* during scoring. They are not used for determining potential system-to-reference template alignment, nor are they included in the scoring of slots described in the following sections. For any given template instance, there may be zero, one or more events that can be found mentioned in the `template-anchor` field.

### 2.3.13 The Protest template – *Protestplate*

This Granular event is intended to capture the various aspects of a protest event.

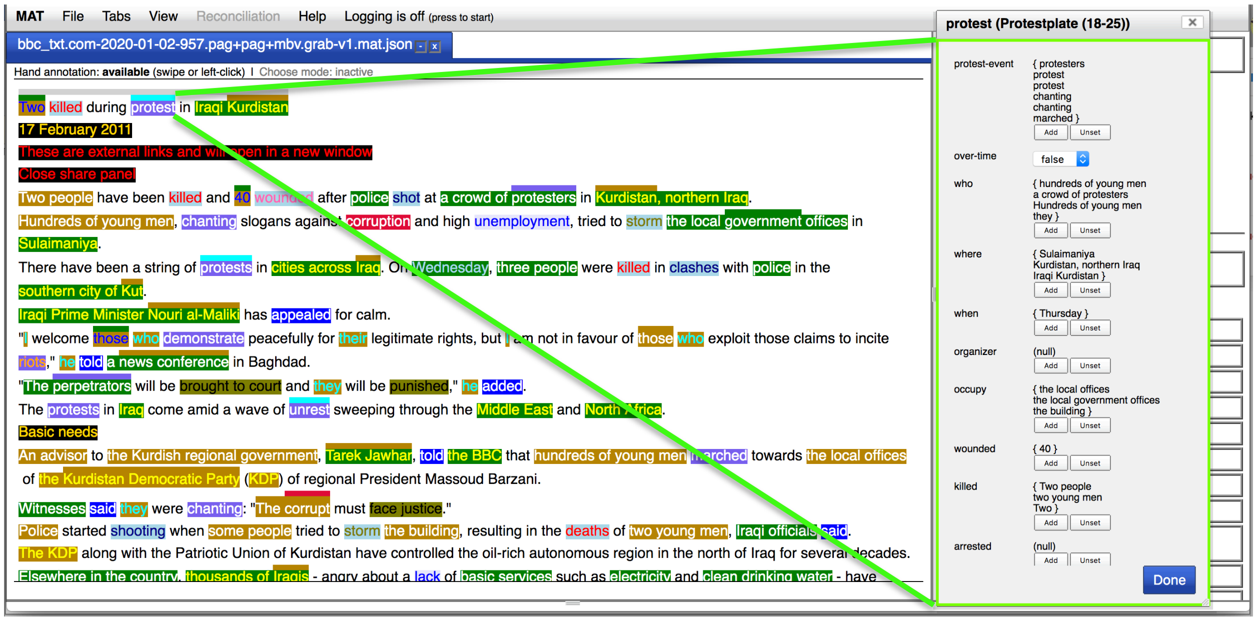
Public protests and the news stories about them will generally capture a range of different important elements of the events that take place (or have been planned or hypothesized). They are organized, so we may want to know who the organizers are; they occur (or are proposed to occur) at particular places and times; there may be acts of violence that take place either by the protesters, by counter-protesters, or by law enforcement; the protests may be intended to express support or condemnation of an idea or events or particular people; and in some cases the outcome of a protest may be indicated, whether or not these are intended. The *Protestplate* template has been developed to capture this type of information, mostly through a proliferation of slots that make explicit the role that an entity or event plays within a particular protest story.

The template is comprised of the following fields:

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `arrested` | list | Description or count of those arrested. |
| `imprisoned` | list | Description or count of those jailed. |
| `killed` | list | Description or count of those killed, *e.g.*, “two people,” “two”. |
| `occupy` | list | Any space or building taken over, *e.g.*, “the local government offices”. |
| `over-time` | Boolean | A flag indicating whether this is an individual case of corruption, or a systematic state of corruption affecting many corrupt individuals or institutions. |
| `organizer` | list | The group/individuals leading the protest, *e.g.* “the Workers Party”. |
| `outcome-averted` | list | Events that were either averted or are noted as not having occurred, *e.g.* “no injuries” (Basic events do not code negation or other realis factors). |
| `outcome-occurred` | list | Events that occurred because of the corruption. |
| `outcome-hypothetical` | list | Events that are only noted as potentially occurring because of the corruption. |
| `protest-against` | list | Any events presented as what the protest is meant to end, *e.g.* “corruption,” “unemployment” (an SOA), “a [ban] on [the washing lines],” etc. |
| `protest-event` | list | Unscored. Slot used to assist annotators. |
| `protest-for` | list | Any event presented as the aim of the protest, *e.g.*, “the corrupt must face justice,” coded as {agt ø, head “face justice,” ptt “the corrupt”}. |
| `when` | list | Date of the protest, as best identifiable: “Thursday,” “last month,” etc. |
| `where` | list | Location(s) of the protest. |
| `who` | list | References to protest participants, *e.g.* “hundreds of young men.” |
| `wounded` | list | Description of any injured participants, or a count if that is all that is available, *e.g.*, “a woman,” “40,” “several police officers”. |

**Table 4 Description of Protestplate slots.**

The following screenshots demonstrate an instance of *Protestplate* as it is being reviewed in the annotation tool (MAT) used to create the Granular data.


MAT File Tabs View Reconciliation Help Logging is off (press to start)
bbc_txt.com-2020-01-02-957.pag+pag+mbv.grab-v1.mat.json
Hand annotation: available (swipe or left-click) | Choose mode: inactive

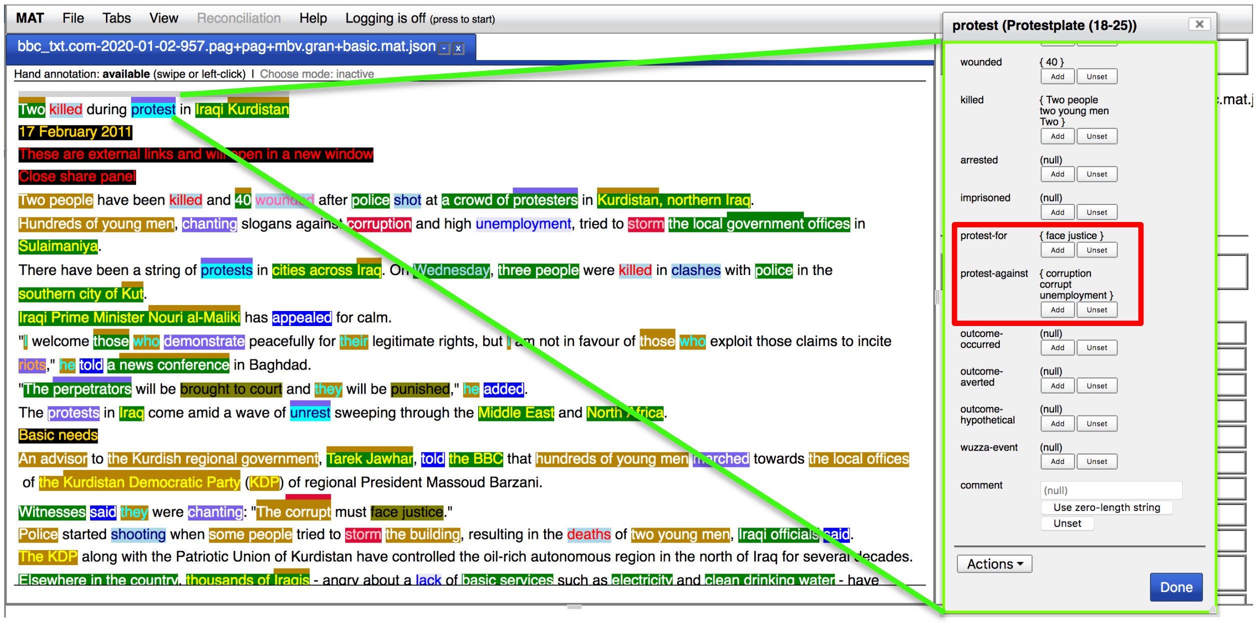
Two killed during protest in Iraqi Kurdistan
17 February 2011
These are external links and will open in a new window
Close share panel
Two people have been killed and 40 wounded after police shot at a crowd of protesters in Kurdistan, northern Iraq.
Hundreds of young men, chanting slogans against corruption and high unemployment, tried to storm the local government offices in Sulaimaniya.
There have been a string of protests in cities across Iraq. On Wednesday, three people were killed in clashes with police in the southern city of Kut.
Iraqi Prime Minister Nouri al-Maliki has appealed for calm.
"I welcome those who demonstrate peacefully for their legitimate rights, but I am not in favour of those who exploit those claims to incite riots," he told a news conference in Baghdad.
"The perpetrators will be brought to court and they will be punished," he added.
The protests in Iraq come amid a wave of unrest sweeping through the Middle East and North Africa.
Basic needs
An advisor to the Kurdish regional government, Tarek Jawhar, told the BBC that hundreds of young men marched towards the local offices of the Kurdistan Democratic Party (KDP) of regional President Massoud Barzani.
Witnesses said they were chanting: "The corrupt must face justice."
Police started shooting when some people tried to storm the building, resulting in the deaths of two young men, Iraqi officials said.
The KDP along with the Patriotic Union of Kurdistan have controlled the oil-rich autonomous region in the north of Iraq for several decades.
Elsewhere in the country, thousands of Iraqis - angry about a lack of basic services such as electricity and clean drinking water - have

**protest (Protestplate (18-25))**

| | |
|---|---|
| protest-event | { protesters<br>protest<br>protest<br>chanting<br>chanting<br>marched } |
| over-time | false |
| who | { hundreds of young men<br>a crowd of protesters<br>Hundreds of young men<br>they } |
| where | { Sulaimaniya<br>Kurdistan, northern Iraq<br>Iraqi Kurdistan } |
| when | { Thursday } |
| organizer | (null) |
| occupy | { the local offices<br>the local government offices<br>the building } |
| wounded | { 40 } |
| killed | { Two people<br>two young men<br>Two } |
| arrested | (null) |

**Figure 2 Example of a *Protestplate***

This instance of the *Protestplate* in Figure  covers the one protest in Sulaimaniya that left two demonstrators dead and 40 injured. The *Protestplate* structure is large, and the screenshot does not show the protest-for/against or outcome fields. These additional fields are shown in Figure .


MAT File Tabs View Reconciliation Help Logging is off (press to start)
bbc_txt.com-2020-01-02-957.pag+pag+mbv.gran+basic.mat.json
Hand annotation: available (swipe or left-click) | Choose mode: inactive
.mat.j

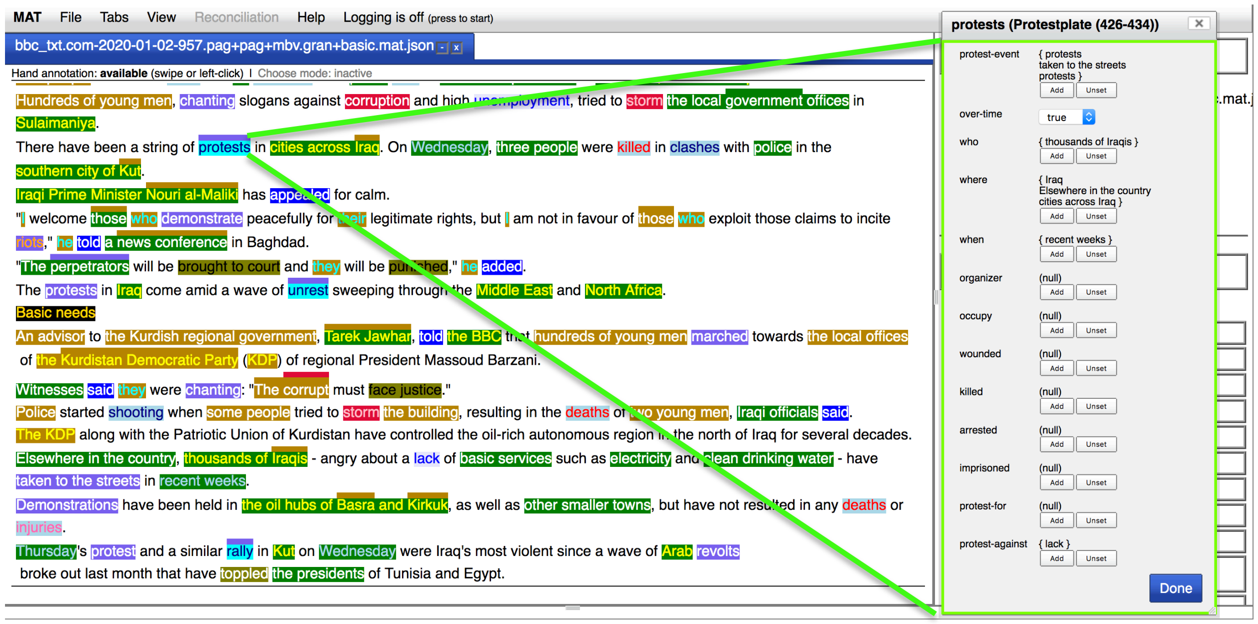
Two killed during protest in Iraqi Kurdistan
17 February 2011
These are external links and will open in a new window
Close share panel
Two people have been killed and 40 wounded after police shot at a crowd of protesters in Kurdistan, northern Iraq.
Hundreds of young men, chanting slogans against corruption and high unemployment, tried to storm the local government offices in Sulaimaniya.
There have been a string of protests in cities across Iraq. On Wednesday, three people were killed in clashes with police in the southern city of Kut.
Iraqi Prime Minister Nouri al-Maliki has appealed for calm.
"I welcome those who demonstrate peacefully for their legitimate rights, but I am not in favour of those who exploit those claims to incite riots," he told a news conference in Baghdad.
"The perpetrators will be brought to court and they will be punished," he added.
The protests in Iraq come amid a wave of unrest sweeping through the Middle East and North Africa.
Basic needs
An advisor to the Kurdish regional government, Tarek Jawhar, told the BBC that hundreds of young men marched towards the local offices of the Kurdistan Democratic Party (KDP) of regional President Massoud Barzani.
Witnesses said they were chanting: "The corrupt must face justice."
Police started shooting when some people tried to storm the building, resulting in the deaths of two young men, Iraqi officials said.
The KDP along with the Patriotic Union of Kurdistan have controlled the oil-rich autonomous region in the north of Iraq for several decades.
Elsewhere in the country, thousands of Iraqis - angry about a lack of basic services such as electricity and clean drinking water - have

**protest (Protestplate (18-25))**

| | |
|---|---|
| wounded | { 40 } |
| killed | { Two people<br>two young men<br>Two } |
| arrested | (null) |
| imprisoned | (null) |
| protest-for | { face justice } |
| protest-against | { corruption<br>corrupt<br>unemployment } |
| outcome-occurred | (null) |
| outcome-averted | (null) |
| outcome-hypothetical | (null) |
| wuzza-event | (null) |
| comment | (null) |

**Figure 3 The *protest-for* and *protest-against* values for the *Protestplate* in Figure**

The document also refers to a range of additional protests, coded as four further *Protestplates*. One corresponds to several references to a wave of protests taking place in Iraq. Note the *over-time* flag (set to true) and the *protest-against* field, which captures the Basic state of affairs (SOA) “a [*event* lack] of [*patient* basic services] such as [*patient* electricity] and [*patient* clean drinking water]”.



bbc_txt.com-2020-01-02-957.pag+pag+mbv.gran+basic.mat.json

Hand annotation: **available** (swipe or left-click) I Choose mode: inactive

Hundreds of young men, chanting slogans against corruption and high unemployment, tried to storm the local government offices in Sulaimaniya.
There have been a string of protests in cities across Iraq. On Wednesday, three people were killed in clashes with police in the southern city of Kut.
Iraqi Prime Minister Nouri al-Maliki has appealed for calm.
"I welcome those who demonstrate peacefully for their legitimate rights, but I am not in favour of those who exploit those claims to incite riots," he told a news conference in Baghdad.
"The perpetrators will be brought to court and they will be punished," he added.
The protests in Iraq come amid a wave of unrest sweeping through the Middle East and North Africa.
Basic needs
An advisor to the Kurdish regional government, Tarek Jawhar, told the BBC that hundreds of young men marched towards the local offices of the Kurdistan Democratic Party (KDP) of regional President Massoud Barzani.
Witnesses said they were chanting: "The corrupt must face justice."
Police started shooting when some people tried to storm the building, resulting in the deaths of two young men, Iraqi officials said.
The KDP along with the Patriotic Union of Kurdistan have controlled the oil-rich autonomous region in the north of Iraq for several decades.
Elsewhere in the country, thousands of Iraqis - angry about a lack of basic services such as electricity and clean drinking water - have taken to the streets in recent weeks.
Demonstrations have been held in the oil hubs of Basra and Kirkuk, as well as other smaller towns, but have not resulted in any deaths or injuries.
Thursday's protest and a similar rally in Kut on Wednesday were Iraq's most violent since a wave of Arab revolts broke out last month that have toppled the presidents of Tunisia and Egypt.

**protests (Protestplate (426-434))**

| | |
|---|---|
| protest-event | { protests<br>taken to the streets<br>protests } |
| over-time | true |
| who | { thousands of Iraqis } |
| where | { Iraq<br>Elsewhere in the country<br>cities across Iraq } |
| when | { recent weeks } |
| organizer | (null) |
| occupy | (null) |
| wounded | (null) |
| killed | (null) |
| arrested | (null) |
| imprisoned | (null) |
| protest-for | (null) |
| protest-against | { lack } |

**Figure 4: Another Protestplate from the same document**

The final three Granular events in this document are for (i) the protests in Basra and Kirkuk that did not lead to deaths or injuries (coded in the *outcome-averted* field); (ii) the violent protest in Kut, and (iii) the general references to the Arab Spring.

This document is typical of a kind of news story that includes both events that occurred in specific places and times, alongside references to broad ranges of events, such as the Arab Spring. We note that capturing these distinctions represents a technical challenge to the performers. Performer systems will need to not just distinguish events from event ranges, but also identify when two specific events are distinct, for instance the Sulaimaniya and Kut protests. This article is particularly challenging, since the narrative is not a linear presentation of separate events, but more of an interleaving: Note that the Sulaimaniya protest is referred to multiple times, with other protests mentioned in between these references.

### 2.3.14 The Corruption Template – *Corruplate*

This is a simpler template that nonetheless shares many of the same fields as the *Protestplate* above. The topic of corruption presents an annotation issue around the question of whether the mere mention of corruption is an adequate justification for triggering the generation of a Granular event. Note that templates are not generated for mere mentions of corruption that are otherwise completely lacking in detail. This is often the case in the context of protests, where corruption is one of the events about which people are protesting.

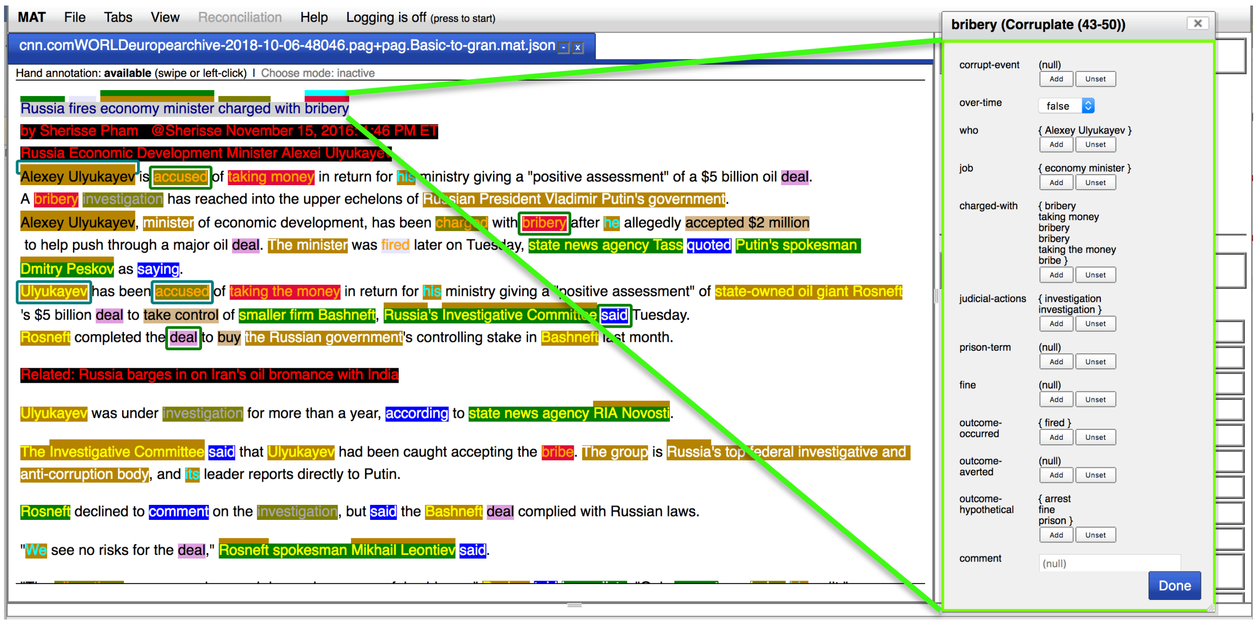
For our purposes, however, we required that there be at least one piece of further information attached to the mention of corruption. This could include an individual being cited for corruption, a reference to a judicial or law-enforcement action (for example "an anti-corruption [*event* probe]"), and so forth. A further restriction as to what we considered to fit the template: we did not instantiate this Granular event outside the realm of government. In the case of financial misappropriation, for example, government ministers may be corrupt, but bank executives are merely embezzlers.

The *Corruplate* is comprised of the following fields:

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `charged-with` | list | The crimes that the individual has been charged with, coded as Basic events "[*patient* Uyukaeve] had been caught accepting the [*event* bribe]" |
| `corrupt-event` | list | Unscored. Slot used to assist annotators. |
| `judicial-actions` | list | Investigations, trials, sentences, and so forth noted in the narrative as having taken place. |
| `fine` | list | Any monetary damages or seizures levelled as punishment. |
| `over-time` | Boolean | A flag indicating whether this is an individual case of corruption, or a systematic state of corruption affecting many corrupt individuals or institutions. |
| `outcome-averted` | list | Events that were either averted or are noted as not having occurred, *e.g.* "no injuries" (Basic events do not code negation or other realis factors). |
| `outcome-occurred` | list | Events that occurred because of the corruption. |
| `outcome-hypothetical` | list | Events that are only noted as potentially occurring because of the corruption. |
| `prison-term` | list | The duration(s) of any prison sentence mentioned as applicable (with appropriate *irrealis* status indicated as appropriate). |
| `where` | list | Location(s) of the corruption. |
| `who` | list | The individual(s) being accused of corruption. |

**Table 5 Description of corruplate slots.**

While it is generally the case that a Corruption basic event will be present to trigger generation of a *Corruplate*, this is not always the case. Figure , for example, shows a Granular event that is triggered by the Bribery basic event.



cnn.comWORLDeuropearchive-2018-10-06-48046.pag+pag.Basic-to-gran.mat.json

Hand annotation: **available** (swipe or left-click) I Choose mode: inactive

Russia fires economy minister charged with bribery

by Sherisse Pham @Sherisse November 15, 2016: 1:46 PM ET

Russia Economic Development Minister Alexei Ulyukayev

Alexey Ulyukayev is accused of taking money in return for his ministry giving a "positive assessment" of a $5 billion oil deal.
A bribery investigation has reached into the upper echelons of Russian President Vladimir Putin's government.
Alexey Ulyukayev, minister of economic development, has been charged with bribery after he allegedly accepted $2 million
to help push through a major oil deal. The minister was fired later on Tuesday, state news agency Tass quoted Putin's spokesman
Dmitry Peskov as saying.
Ulyukayev has been accused of taking the money in return for his ministry giving a "positive assessment" of state-owned oil giant Rosneft
's $5 billion deal to take control of smaller firm Bashneft, Russia's Investigative Committee said Tuesday.
Rosneft completed the deal to buy the Russian government's controlling stake in Bashneft last month.

Related: Russia barges in on Iran's oil bromance with India

Ulyukayev was under investigation for more than a year, according to state news agency RIA Novosti.

The Investigative Committee said that Ulyukayev had been caught accepting the bribe. The group is Russia's top federal investigative and anti-corruption body, and its leader reports directly to Putin.

Rosneft declined to comment on the investigation, but said the Bashneft deal complied with Russian laws.

"We see no risks for the deal," Rosneft spokesman Mikhail Leontiev said.

bribery (Corruplate (43-50))

| | |
|---|---|
| corrupt-event | (null) |
| over-time | false |
| who | { Alexey Ulyukayev } |
| job | { economy minister } |
| charged-with | { bribery<br>taking money<br>bribery<br>bribery<br>taking the money<br>bribe } |
| judicial-actions | { investigation<br>investigation } |
| prison-term | (null) |
| fine | (null) |
| outcome-occurred | { fired } |
| outcome-averted | (null) |
| outcome-hypothetical | { arrest<br>fine<br>prison } |
| comment | (null) |

**Figure 5 A *Corruplate* example.**

This template shows some interesting distinctions among the outcomes of the corruption. Note the "investigation" events in the *judicial-actions* field, as opposed to the "fired" event in the *outcomes-occurred* field. The former are placed there because they are specifically part of the prosecution of the corruption allegations, whereas the "firing" is considered here to be a non-judicial outcome of the allegations. Also note that the potential for arrests, fines and prison terms is captured in the outcome-hypothetical field, since neither is being predicated as having happened (*outcome-occurred*) or failing to have to happened (*outcome-averted*).

### 2.3.15 The Terrorism Template – *Terrorplate*

The *Terrorplate* template captures some of the key elements of the events surrounding acts of terrorism. The *Terrorplate* structure consists of the following fields:

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `blamed-by` | list | Those who are asserting the identity of the perpetrators. |
| `claimed-by` | list | Those who have claimed responsibility for the terrorist event(s). |
| `completion` | String | Whether the terrorism event(s) are considered to be completed or not. One of "planned", "thwarted", "failed", or "successful". |
| `coordinated` | Boolean | Whether the terrorism event(s) are considered to be coordinated or not. |
| `killed` | list | Mentions of those people who were killed. |
| `kidnapped` | list | Mentions of those people who were kidnapped. |
| `named-perp` | list | Those to whom the terrorist event(s) are attributed. |

|  |  |  |
|---|---|---|
| named-perp-org | list | Mentions of the organization(s) the perpetrators belong to. |
| named-organizer | list | Those to whom the planning of the terrorist event(s) are attributed. |
| over-time | Boolean | A flag indicating whether this is an individual case of terrorism, or a systematic state of terrorism |
| outcome-averted | list | Events that were prevented by virtue of the events described in this template. |
| outcome-occurred | list | Events or states-of-affairs that have actually taken place by virtue of the terrorist events. |
| outcome-hypothetical | list | Events or states-of-affairs that could have taken place by due to the terrorist events. |
| perp-captured | list | Mentions of those perpetrators of the terrorist events who were captured. |
| perp-killed | list | Mentions of perpetrators who were killed in the course of the terrorist events. |
| perp-objective | list | Mentions of events which are identified as being desired to have taken place (or states-of-affairs to have come about) by virtue of the terrorist events. |
| perp-wounded | list | Mentions of perpetrators who were wounded in the course of the terrorist events. |
| target-human | list | One or more people, named or unnamed, who are said to be the targets of the terrorist events. |
| target-physical | list | The facility or geo-political location that was being targeted by the terrorist events. |
| terror-event | list | Unscored. Slot used to assist annotators. |
| type | string | One of "arson", "assault", "bombing", "kidnapping", "murder", or "unspecified". |
| weapon | list | Mentions of the weapons or other instruments used to carry out the terrorist events. |
| when | list | Mentions of times or durations associated with the events involved. |
| where | list | Mentions of the location(s) at which the terrorist events have taken place. |
| wounded | list | Mentions of those people who were wounded. |

**Table 6 Description of Terrorplate slots.**

### 2.3.16 The Disease Outbreak Template – *Epidemiplate*

The *Epidemiplate* template captures key information elements in regards to various kinds of disease outbreaks. The *Epidemiplate* structure consists of the following fields:

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `disease` | list | Mentions of the disease that is at the heart of the disease outbreak. |
| `exposed-count` | list | Mentions of counts of people who have become exposed. |
| `exposed-cumulative` | list | Mentions of cumulative counts of people who have become exposed. |
| `exposed-individuals` | list | Mentions of people (or groups of people) who have become exposed to the disease. |
| `hospitalized-count` | list | Mentions of counts of people who have become hospitalized due to the disease. |
| `hospitalized-cumulative` | list | Mentions of cumulative counts of people who have become hospitalized due to the disease. |
| `hospitalized-individuals` | list | Mentions of people (or groups of people) who have become hospitalized due to the disease. |
| `infected-count` | list | Mentions of counts of people who have become infected. |
| `infected-cumulative` | list | Mentions of cumulative counts of people who have become infected. |
| `infected-individuals` | list | Mentions of people (or groups of people) who have become infected with the disease. |
| `killed-count` | list | Mentions of counts of people who have been killed by the disease. |
| `killed-cumulative` | list | Mentions of cumulative counts of people who have been killed by the disease. |
| `killed-individuals` | list | Mentions of people (or groups of people) who have been killed by the disease. |
| `NPI-Events` | list | Non-Pharmacologic-Intervention-Events. Interventions taken by authorities to prevent the further disease, such as closing schools, limiting travel, etc. |
| `outbreak-event` | list | Unscored. Slot used to assist annotators. |
| `tested-count` | list | Mentions of counts of people who have been tested for the disease. |
| `tested-cumulative` | list | Mentions of cumulative counts of people who have been tested for the disease. |

| | | |
|---|---|---|
| `tested-individuals` | list | Mentions of people (or groups of people) who have been tested for the disease. |
| `recovered-count` | list | Mentions of counts of people who have recovered from the disease. |
| `recovered-cumulative` | list | Mentions of cumulative counts of people who have recovered from the disease. |
| `recovered-individuals` | list | Mentions of people (or groups of people) who have recovered from the disease. |
| `vaccinated-count` | list | Mentions of counts of people who have been vaccinated against the disease. |
| `vaccinated-cumulative` | list | Mentions of cumulative counts of people who have been vaccinated against the disease. |
| `vaccinated-individuals` | list | Mentions of people (or groups of people) who have been vaccinated against the disease. |
| `when` | list | Mentions of times or durations associated with the events involved. |
| `where` | list | Mentions of one or more instances of where the disease has occurred. |

**Table 7 Description of Epidemiplate slots.**

### 2.3.17 The Major Disaster Template – *Disasterplate*

The *Disasterplate* template captures key facets of a natural or environmental disaster. This includes primarily the likes of earthquakes and hurricanes, but also covers famines.

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `major-disaster-event` | list | Unscored. Holds instances of Environmental, Natural Phenomenon, or Famine events (or SoAs), or also Weather/Environmental damage events. Slot primarily used to assist with annotation. |
| `over-time` | Boolean | A flag indicating whether this is an individual event, or an extended state of affairs. |
| `where` | list | Mentions of one or more locations associated with the disaster. |
| `when` | list | Mentions of times or durations associated with the events involved. |
| `injured-count` | list | Similar to Epidemiplates: these are mentions that enumerate the individuals injured in a disaster. |
| `killed-count` | list | As above, these are mentions that enumerate the individuals who died in a disaster. |

| | | |
|---|---|---|
| `missing-count` | list | As above, these are mentions that enumerate the individuals who have gone missing in a disaster. |
| `outcome` | event list | Events that occurred as a result of a disaster, but are not already covered in specific slots of the Disasterplate. For Phase 2, there is only one outcome slot, with no separate slot for hypothetical or averted outcomes. Outcomes with irrealis or counterfactual occurrence are now indicated with the appropriate irrealis marker in BP JSON. |
| `responders` | list | Entities designating the responders to the natural disaster. |
| `damage` | list | Entities that have been damaged in the disaster; typically these are the patients of a Weatheer-or-environmental-Damage Basic event. |
| `affected-cumulative-count` | list | For those cases where the toll of a disaster is identified over time: this slot holds entities that enumerate the cumulative count of affected individuals. |
| `individuals-affected` | list | Entities that provide descriptions of the affected individuals. |
| `rescued-count` | list | Entities enumerating the individuals rescued in the response to a disaster. |
| `rescue-events` | event list | This slot holds Basic events of the Rescue or Evacuate types (as opposed to the patients of these events, as in the fillers of the rescued-count slot). |
| `assistance-provided` | event list | This slot captures Provide-Aid events that were in response to the disaster. |
| `assistance-needed` | event list | Similarly to the previous slot, this slot captures aid-Needs events that originated in the disaster. |
| `related-natural-phenomena` | event list | Weather, Environmental, or similar events that occurred as a result of the predicating disaster. For example, floods might be related to a hurricane disaster. Because it can sometimes be difficult to determine which event should be treated as "primary," in this example, both "floods" and "hurricane" would be included in this slot. |
| `announce-disaster-warnings` | event list | Any communication events that warn of the impending disaster before it strikes. |
| `declare-emergency` | event list | Any communication events declaring the emergency (typically a Declare-Emergency Basic event). |
| `disease-outbreak-events` | event list | Disease outbreak events that follow upon the disaster, e.g., the cholera outbreak after the Haiti earthquake. Not just Disease-Outbreak, but other related epidemic event types. |

| | | |
|---|---|---|
| `repair` | event list | Repair events that are part of the response to the disaster. |
| `human-displacement-events` | event list | Any refugee movement events resulting from the disaster (including those related to detaining refugees). |

**Table 8 Fields of the Natural Disaster template**

### 2.3.18 The Human Displacement Template – *Displacementplate*

The *Displacementplate* template captures facets of human migration resulting from cataclysmic events. The precipitating events may be natural disasters or events of human origin (wars).

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `human-displacement-event` | event list | Unscored. Holds event anchors typically associated with human displacement – primarily refugee-movement events. Slot primarily used to assist with annotation. |
| `over-time` | Boolean | A flag indicating whether this is an individual event, or an extended state of affairs. |
| `origin` | list | Mentions of one or more locations from which the displaced humans are fleeing. |
| `current-location` | list | Mentions of any locations reported to be where the displaced humans' currently are. |
| `transiting-location` | list | Mentions of any locations through which the displaced humans transited on their way to their current location. |
| `destination` | list | Mentions of any locations that the displaced humans intend to continue on towards. |
| `when` | list | Mentions of times or durations associated with the events involved. |
| `total-displaced-count` | list | As with Epidemiplates, these are phrases that enumerate the total population of displaced humans. |
| `event-or-soa-at-origin` | event list | Events that precipitated the migration: these can be natural disasters, war, famine, political or economic crises, etc. |
| `settlement-status-event-or-soa` | event list | Events (of any type) that pertain to the status of the displaced humans. Events that indicate the final status of the individuals (e.g., asylum is granted, refugees are resettled, etc.). Note that applying for asylum is not a settlement status event, as an application for asylum is not an indication of where the individual(s) ultimately settled. |
| `outcome` | event list | Events that occurred as a result of the human displacement event, but are not already covered in specific slots of the Displacementplate. For Phase 2, there is only one outcome slot, with no separate slot for hypothetical or averted outcomes. Outcomes with irrealis or counterfactual |

| | | |
|---|---|---|
| | | occurrence are now indicated with the appropriate irrealis marker in BP JSON. |
| `group-identity` | list | Mentions pertaining to the group identity of the displaced humans, be that ethnic, national, religious, or otherwise. |
| `injured-count` | list | As above, mentions enumerating those displaced humans who have been injured during their migration. |
| `killed-count` | list | As above, mentions enumerating those displaced humans who have been killed during their migration. |
| `missing-count` | list | As above, mentions enumerating those displaced humans who have gone missing during their migration. |
| `detained-count` | list | As above, mentions enumerating those displaced humans who have been detained during their migration. |
| `blocked-migration-count` | list | As above, mentions enumerating those displaced humans who have been blocked from further migration. |
| `transitory-events` | event list | Events that occurred during the displaced humans' travels. |
| `assistance-provided` | event list | Assistance events provided to the displaced humans, typically Provide-Aid events. Note that the type field on these events may provide useful information. |
| `assistance-needed` | event list | Assistance request events provided to the displaced humans, typically Aid-Needs events. Note that the type field on these events may provide useful information. |

**Table 9 Fields of the Human Displacement template**

### 2.3.19 The ETIP Template (Energy and Transportation Infrastructure Projects)

The *ETIPlate* template captures development projects in the area of energy and transportation infrastructure. For this analytic task, the focus is on projects in the developing world that involve some kind of Chinese entity, as with the Belt and Road Initiative. By involvement, we mean either a Chinese-funded project, a project financed by a Chinese financial institution, a project where the work is carried out by a Chinese company, and so forth.

#### 2.3.19.1 Chinese Participation is Required

It is a definitional requirement of the ETIPlate task for Phase 3 that at least one participating entity be Chinese. Development projects that do not involve China in any way, for example EU-funded projects, are not to be reported as ETIPlates.

This aspect of the ETIPlate task introduces a need for fine-grained argument selection that has not been required in previous Granular tasks. We are highlighting this change to ensure that participants are fully alerted to the requirement.

Reflecting this requirement, the Phase 3 data collection in support of the ETIPlate task was directed at projects with Chinese involvement. Performers can expect to see a preponderance of China-linked ETIPlates in both training and test data. This does not mean, however, that the data are guaranteed not to mention development projects that do not involve China. An article that covers a Belt and Road Initiative project may also mention in passing a project with no Chinese participation, for which performers will be expected not to generate an ETIPlate.

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `etip-event` | event list | Unscored. Holds event anchors typically associated with ETIP events – primarily *Establish-Project, Construct-Project*, and the like. Other Phase 3 events may trigger the ETIPlate as well. As with prior templates, this was originally intended as an annotation aide-mémoire, though it may prove useful to performers as an event trigger. |
| `over-time` | Boolean | A flag indicating whether this is an individual event, or an extended state of affairs. |
| `when` | list | Mentions of times or durations associated with the events involved in the ETIPlate. |
| `project-name` | list | The ETIP name or project description. Most likely a descriptive phrase rather than an official name. Generally, a Patient of *Establish-project* or *Construct-project.* E.g., “Lahore Orange Line”, “Lahore Orange Line project”. |
| `project-type` | string | A fixed choice field, drawn from one of the following options: road, rail, port, energy, airport, multiple, or other. If there is no linguistic evidence for filling this slot, it does not appear in the template at all and is unscored.. |
| `overall-project-value` | list | The cost associated with the project, as captured by the *money* field of an ETIP-related Basic event such as Fund-Project, Establish-Project, and so forth. If multiple currencies are listed (“xxx KsH equivalent to yyy USD”), only add one mention to the ETIPlate slot, with other mentions handled at the Basic level (see Section 2.2.6). |
| `project-location` | list | Country or countries where the ETIP is located. Might be further specified to include particular regions, areas, or cities within the country or countries. |
| `agreement-duration` | list | When an agreement is mentioned, any duration associated with the agreed-to project. |
| `signatories` | list | The parties to an agreement, for example, any signatories to a *Sign-Agreement* Basic event. |
| `contract-awardee` | list | The entity awarded the development contract. Sometimes a signatory of the agreement, sometimes the patient of an *Award-Contract* Basic event. |
| `contract-awarder` | list | Often the host country of the development project, typically a participant in *Sign-Agreement* or *Award-* |

| | | |
|---|---|---|
| | | *Contract* Basic events. Must be explicit, and not be just the location of the project. |
| `contract-amount` | list | Amount, in some currency, that specifies the value of the contract. Typically the Financial argument of a *Sign-agreement* or *Award-contract* event at Basic. |
| `funding-source` | list | Source of the funding for an ETIP project. Typically the agent of a *Fund-Project* Basic event. |
| `funding-recipient` | list | Recipient of the funding, usually the host country of the development project or a company carrying out the work. |
| `funding-amount` | list | Amount, typically in some currency, specifies the value of funding, usually the Money argument of a *Fund-Project* Basic event. As with `overall-project-value`, if multiple currencies are mentioned, only one mention goes in the ETIPlate slot, with other mentions handled at the Basic level (see Section 2.2.6). |
| `outcome` | event list | Events that occurred as a result of the ETIP project, but are not already covered in specific slots of the *ETIPlate*. As with Phase 2, there is only one outcome slot, with no separate slot for hypothetical or averted outcomes. Outcomes with irrealis or counterfactual occurrence are now indicated with the appropriate irrealis marker. |

### 2.3.20 The Cyber Crime Template: *Cybercrimeplate*

The *Cybercrimeplate* template captures various aspects of a criminal cyber activitiy. These include incursions, denial of service attacks, theft of identity or valuables, and so forth.

| Slot Label (JSON) | Data Type | Description |
|---|---|---|
| `cybercrime-event` | event list | Unscored. Holds event anchors that can be considered to have triggered the Cybercrimeplate event – primarily *Cyber-Crime-Attack* and related Basic events. As with prior templates, this was originally intended as an annotation aide-mémoire, though it may prove useful to performers as an event trigger. |
| `over-time` | Boolean | A flag indicating whether this is an individual event, or an extended state of affairs. |
| `when` | list | Mentions of times or durations associated with the events involved in the Cybercrimeplate event. |
| `perpetrator` | list | Organization or person responsible for the crime, whether named or not (as ins “the hackers”, etc.). Often the Agent of some crime event, or Patient of *Arrest* etc. Note: software/malware is not a perpetrator, even if it is referred to as the agent of a cyber-attack. |

| | | |
|---|---|---|
| `victim` | list | Organization or person that is the victim of the crime. Might be generic or multi-valued. Acceptable victims include system owners (often the Patient of a *Cyber-Crime-Attack*), or an individual whose information is stolen (often the Patient of an *Information-Theft*). |
| `information-stolen` | list | Usually identifying information, such as social security numbers. The fillers of this field needs to be newly identified for the Granular task, since they are not expected to be captured as (say) the Patient of an *Information-Theft* event (this Patient is expected to be filled by the entity most affected by the event, namely the victim). |
| `related-crimes` | event list | Any crime events related or consequent to the cyber-crime, *e.g.*, *Information-Theft*, *Extortion*, and so forth. If there is a triggering event in the cybercrime-event field, it will also be included here. |
| `victim-impact` | event list | Effect on the victim, for example, the victim organization temporarily ceasing operations (an *interrupt-Operations* Basic event). Sometimes this is the Referred event of the triggering crime, as with *Financial-Loss* or *Pay-ransom*. |
| `response` | event list | Any Basic events that are a direct response to the cyber-crime event, for example, a software company issuing an emergency patch. This does not include cybersecurity measures not specifically resulting from the crime (e.g. scheduled software releases or cybersecurity training). |
| `outcomes` | event list | Any further outcome events that arise from the cybercrime, except as noted above. Could include legislative actions, arrests, prosecutions, identifying vulnerabilities, or financial events such as spending money on cybersecurity. As with Phase 2, there is only one outcome slot, with no separate slot for hypothetical or averted outcomes. Outcomes with irrealis or counterfactual occurrence are now indicated with the appropriate irrealis marker. |

### 2.3.21 Data Sources for the Granular Corpus

The Granular data are created by annotating files that have already been annotated for Basic events. These consist both of files that were distributed previously to performer teams for the Basic evaluation, as well as new articles not previously released. At times, a modest number of changes were made to some Basic-level annotations while preparing the Granular data.

The vast majority of Basic-marked articles also have Granular markup. For Phase 1, only a few articles from the Basic corpus were not selected for Granular annotation, chiefly off-topic stories unaligned with the Granular analytic foci. For Phase 2, the number of Basic-marked stories that were not marked for Granular is somewhat larger. All Granular files fall into the same partitions (train, devtest, analysis, hidden) as they were assigned in the Basic corpus distribution.

The resulting data counts are as follows:

- Phase 1 data provided to performers:
    - Train partition: 94 articles, 2,742 Basic events, 252 templates
    - Devtest partition: 20 articles, 560 Basic events, 47 templates
    - Analysis partition: 20 articles, 568 Basic events, 56 templates
- Phase 1 “hidden” training data: 52 articles, 1,481 Basic events, 132 templates

- Phase 2 data provided to performers:
    - Train partition: 168 articles, 4,283 Basic events, 255 templates
    - Devtest partition: 32 articles, 776 Basic events, 47 templates
    - Analysis partition: 34 articles, 809 Basic events, 57 templates
- Phase 2 “hidden” training data: 84 articles, 1,991 Basic events, 138 templates

- Phase 3 data provided to performers:
    - Train partition: 172 articles, 4,198 Basic events, 302 templates
    - Devtest partition : 37 articles, 812 Basic events, 65 templates
    - Analysis partition : 35 articles, 751 Basic events, 60 templates
- Phase 3 “hidden” training data: 97 articles, 2,175 Basic events, 147 templates

The Phase 2 and Phase 3 datasets differ in several ways from the data for Phase 1. A first difference is that the two later phases have lower template density per document: Phase 1 documents have on average 2.6 templates per document, compared to 1.6 for Phase 2 and 1.7 for Phase 3. As a result, Phases 2 and 3 required substantially more articles to match Phase 1’s template inventory of approximately 500 templates (Phase 3 went all the way to 565 templates).

In addition, because the per-document density of Basic events remains comparable across Phases, this means that Phase 2 and Phase 3 also have notably more Basic events than in Phase 1.

A further characteristic that distinguishes Phase 1 from Phases 2 and 3 is the degree to which articles tend to have more than one template type per article. In Phase 1, for example, it was not at all uncommon to have both Protestplate and Corruplate events in the same article. With the Phase 2 and Phase 3 data, inter-mixing template types is much less frequent. In Phase 3, for example, only 8% of articles exhibit more than one template type. These differences in template type distributions allow for more streamlined Granular event processing for Phases 2 and 3, wherein documents are first sorted into an ETIP or Cyber-Crime category (for Phase 3) prior to extracting Granular event templates.

### 2.3.22 Granular Scoring Metric

The Granular extraction task is described as a “template-filling” task, as it ties together the simpler, minimally structured events that have been extracted at the Basic level into a much larger data structure. In contrast to Basic events, Granular templates usually consist of a large number of slots, each of which has a very distinct meaning and role within the interpretation of the template. This structure allows for the representation of complex relationships that are considered important for a specific analytic task.

A fundamental design decision adopted for scoring Granular templates is that the score should not incorporate elements that have “already” been scored at the Basic level. That is, Granular scores should reflect only the decisions made in the course of constructing and filling the Granular templates, and not those decisions made solely at the Basic event level. Of course, a system can only fill template slots with previously recognized Basic events, so there is a strong dependency between Basic and Granular performance. However, the score for a given Basic event is not convolved with the score for the slots that it fills: all that matters in scoring an event-valued Granular slot is that it was correctly filled, not how well the filler itself was scored.

#### 2.3.22.1 **Types of slot filler values**

Each template has an attribute that indicates the type of template. The template type value is drawn from the universe of possible template types, which for Phase 1 of the BETTER program consists of: `protestplate, corruptplate, terrorplate`, and `epidemiplate`. In contrast to the role of event types in Abstract and Basic, these different template types will determine an almost completely different set of template-specific slots.

Since templates are constructed "on top of" a set of Basic events, most of the template slots are designed to be filled with entities[10] or events annotated at the Basic level. These slots can be left empty or can be filled with one, two or more entities or events. In addition to Basic-annotated entities and events there are slots that take on Boolean values (`True` or `False`) or one of a number of possible pre-defined atomic string values. For example, the `completion` slot in the Terrorist template (`terrorplate`) can take on one of these specific values: `planned, thwarted, failed`, or `successful`.

The Granular task supports the capture of the assertion status of events and entities. This occurs, for example, when an event or entity may be expected to fill a slot in the context of a story, but in fact fails to do so (e.g., demonstrators who did not show up for a planned protest). In such cases, the Granular template marks the slot value with an irrealis indicator that captures the epistemic or assertion status of the slot fill. The type of irrealis involved is captured in the "`irrealis`" field of the template slot element data structure. The types of irrealis supported in Granular are these: `counterfactual, hypothetical, future, unconfirmed, unspecified, non-occurrence`. Both events and entities can be marked with irrealis, but not Booleans or string literals.

#### 2.3.22.2 **Summary of scoring metric**

The approach to scoring Granular templates is motivated by adopting the same philosophy that has been used in the two previous levels of event extraction in the BETTER program: Abstract and Basic. The overall score is derived in the same way as each of those previous tasks:

$$GranularTemplateScore = TemplateTypeF1 * TemplateArgumentsF1$$

Exactly analogous to the scoring of Abstract and Basic events, the above F1 values are computed by counting the true positive, true negative, false positive and false negative values across all of the aligned and unaligned system and event templates, where F-Measure (F1) is defined as:

---

[10] In this document we use the term "entities" to refer to "event arguments" as defined for Abstract and Basic event extraction, which are captured as `span-sets` in the BP JSON representation. These arguments are not typed and can refer to a wide variety of types of things, whether concrete or abstract, physical or philosophical. However, given the set of events found at the Basic level, their agent and patient arguments are usually material things, and in any case are referred to in ways that are very similar to "entities" in the information extraction literature. We have adopted the term "entity" in this document because of its familiarity in the information extraction literature.

$$FMeasure = 2 * \frac{(precision * recall)}{(precision + recall)}$$

The Template Type counts are determined by whether a given system and reference template pair have the identical type or not. There is no "partial matching" of template types – they are either the same or they do not match at all and will not be aligned for scoring purposes. Template argument F1 is computed by considering all of the arguments in the system and reference templates and determining whether or not they match and how well. As in Abstract and Basic event scoring, template argument values that refer to <u>entities</u> support partial matching based on the degree of overlap of the argument strings.[11] The alignment of <u>events</u> in template slots is not determined by performing a search of all possible alignments, but instead are driven by the optimal alignments that were found when scoring the Basic events. This adoption of the Basic event alignment information is motivated by the fact that Granular template analysis is completely dependent on the events that have been extracted during Basic event extraction. It would be internally inconsistent if the system events used to fill slots in Granular templates were aligned with completely different reference events.

The overall score for a given document is derived by searching for an alignment of system templates to reference templates in such a fashion that the largest score is obtained. For example, if the system has generated more templates of a certain type than those found in the reference, the scorer will align system templates with reference templates that results in the most value to the overall score, taking into account the cost of the other system templates, and all its arguments, being considered "misses." This process is carried out in exactly the same way as performed for Abstract and Basic event scoring, namely by optimizing[12] on the sum of event and argument match counts (true positive, false positive, true negative and false negatives), and thereafter computing the "combined F1" program metric (event match F1 * argument match F1).

#### 2.3.22.3 Template Argument Matching for Defined Values

As noted above, some template slots must be filled with a predefined value. This includes set-fill values, as in the Terror template's string `completion` slot or the Protest template's Boolean `Over-Time` slot. System-supplied values must match exactly the string value specified in the reference template. These slots will always add one to the set of possible hits as long as their types match (for example, strings must match strings, not Booleans), and if they match they will add a match value of 1.0, otherwise 0.0.[13]

#### 2.3.22.4 Template Argument Matching for Entities

Various Granular template slots can take on zero, one or more events, entities or a mixture of both events and entities. Each of the *entity* values in template slots are treated as a set of one or more mention strings (as captured in the BP JSON "span-sets" objects), and the scorer attempts to find the alignment of system entity span sets that maximizes the score for that template argument slot. This span alignment is handled in exactly the same way as span alignment within Abstract or Basic event arguments.

[11] Argument string overlap score is computed as follows: $S_{arg} = 1 - \frac{float(L(ref,sys))}{max(len(ref,sys))}$, where $L(ref, sys)$ is the Levenshtein distance between the two string sequences and $max(len(ref, sys))$ is the length of the larger of the two string sequences.
[12] Optimization of alignments is achieved with the Munkres assignment algorithm.
[13] Note that comparisons between values for string-literal slots are case-insensitive.

### 2.3.22.5 Template Argument Matching for Basic Events

As noted in the introduction to the Granular scoring metric, the Granular scoring of *events* in template slots does NOT revisit the scoring of the event structures themselves, such as is done at the Basic event scoring level. The goal of Granular scoring is to focus on the template-filling decisions being made at this level, not to revisit and re-evaluate the scoring carried out at the Basic level. In addition, the alignment of system to reference events should be the same alignment adopted for Basic scoring. Therefore, to score the event values for a given template slot the scorer adopts the system-to-reference event alignments adopted when scoring the Basic events. Scoring events then consists simply of identifying for each reference event whether there is a corresponding system event that had been aligned during the scoring of Basic. If there is none, then it is a miss; if there is one, the system is awarded a full point for a match to that reference event – irrespective of the degree the Basic events were matched (based on such things as agent, patient and ref-event arguments). Note that it is not possible for a reference event to be aligned with more than one system event, so the scorer does not need to handle that situation. Any unaligned reference events after this process completes confer 1 false negative. Likewise, any unaligned system events confer 1 false positive, provided that they were not part of an equivalence class already aligned to a reference event or equivalence class (see section 2.3.22.5.1 below).

#### 2.3.22.5.1 Handling Co-Referring Events Across Sentences

Basic events were annotated in such a way that any additional mentions of an identical event were only noted and annotated when those event mentions were in the same sentence. This decision was motivated by the difficulty by human annotators (and thus probably automated systems) to determine when one mention of a complex event, occurring over time and place by multiple participants (such as with protest events), is the same as another mention. But now this issue rears its head in the Granular template-filling task, because the reference data should allow for any sentence-level mention of what is actually the same event to be filled into a given template slot. To address this problem the Granular data has been annotated to include co-reference linkages ("equivalence classes") between all the sentence-level event mentions across the document. Systems are not required to recreate these linkages – they are captured on the reference data solely in order to perform the scoring function correctly. A system will get full credit for filling a given slot with *any* sentence-level event mention which has been identified as being within the co-reference set as the event found in the reference data.

This information is captured in the `template-filler-coref-events` element of the reference BP JSON data. When the scorer iterates over the reference events as indicated above, it will consider not just the system event that is aligned with that particular reference event, but it will also consider any of the other system events that are aligned with reference events that are part of the same coreference equivalence class as captured in the `template-filler-coref-events` data structure. If there is *at least one* such system event, the reference event will be considered matched and a full point will be awarded. Also, if the system happens to have generated any additional events that are ALSO in this same coreference equivalence class, they will NOT be treated as false alarms, but instead will be ignored.

### 2.3.22.6 Template Argument Matching Involving Irrealis or Time-Attachments

As noted in Section 2.3.22, Granular template slot values can capture the assertion status of arguments by using a slot element data structure that "wraps" a Basic entity or event mention.

The Granular scorer will consider the presence or absence of an irrealis or time-attachments field as an additional attribute of the slot that must be correctly indicated. This information is considered to have a collective value of 0.5 (that is, irrealis and time-attachments are each worth 0.25), so that a system can get some positive value for filling a template slot even if it fails to properly capture the irrealis information of the reference data. If, for example, a reference template slot contains an entity mention that contains some irrealis attributed to it, this counts as two separate linguistic decisions, and the system slot value must capture both of these.

Suppose a system slot element aligns with a reference slot element with an irrealis assertion and no time-attachments. If the entity mentions involved match perfectly, but the system failed to provide any irrealis information (a null value), the system will receive 0.5 points for correctly identifying the entity mentions, receive 0.25 points for correctly attributing no time-attachments, but 0 points for an incorrect irrealis assertion, for a total of 0.75 points for that slot element. If the system *did* provide an `irrealis` value, but the value of its `irrealis` did not match exactly the value in the reference `irrealis`, the system would again earn only 0.75 points out of a possible 1.0. If the `status` slot values were identical, then it would earn full credit for this aligned argument value, namely 1.0.

The time-attachments value is handled similarly to irrealis – it is an additional linguistic decision that the system should make. As with irrealis, its value is placed at 0.25, to accommodate the possibility of a system filling a slot correctly with an entity, but mischaracterizing either that value's temporal attachment and/or its irrealis characteristic, and yet still being able to extract a value of 0.50. Notably, the time-attachments value is a list of entity references. The alignment of these entity references is handled similarly to argument alignments, but the final time-attachments alignment sub-score is weighted by 0.25.

#### 2.3.22.7 An Example

Here is a greatly abridged example of a system and reference Granular template and how the Granular scoring metric applies to them.

| System template | Reference Template |
|---|---|
| **Type:** Protestplate (a:+1/1) | **Type:** Protestplate |
| **Over-Time:** True (b:0/1) | **Over-Time:** False |
| **Who:** [["Joe Smith"] (c:1.0/1.0), ["she"] (d:0.5/1.0)], (e:MISS=1) | **Who:** [["Joe Smith"/Name, "Smith"/Name], ["Susan"/Name, "she"/pronoun], ["Workers Collaborative"]] |
| **Where:** [["Aurora"] (f:0.75/1.0), ["Chicago"]] (g:FA=1) | **Where:** [{irrealis status=unconfirmed, ["Aurora"]}] |
| **Arrested:** [["Joe"]] (h:FA=1) | **Arrested: []** |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Templates | aligned: 1 | match: 1.0 | miss: 1 | false alarm: 1 | P=1.0 | R=1.0 | F=1.0 |
| Slots | aligned: 4 | match: 2.25 | miss:1 | false alarm: 2 | P=0.375 | R=0.45 | F=0.41 |
| Granular template score: 1.0 x 0.41 = 0.41 | | | | | | | |

**Table 10 Scoring a Granular template**

The numbers in parentheses show the value earned out of the possible value that could be earned for each of the templates (a) and the slots (b through h). (a) The system template is of the same type as the reference, so it earns a value of 1 out of 1 in terms of the template matching score. (b)

The system incorrectly asserts that the protest event occurred “over time” and so it earns 0 points out of a possible 1. (c, d) For two of the entity mentions within the Who slot, one (c) is an exact match with the reference argument entity, and the other entity mention (d) is a secondary value entity mention (a pronoun instead of the name mention that is available), and so it earns only half of the possible value. € The system has completely failed to produce a third option in the Who slot for the reference entity “Workers Collaborative”, earning a miss of 1. (f) In the case of “Aurora” in the Where slot, the system failed to indicate the `unconfirmed` status of the irrealis, so it only earns 0.75 points out of the possible 1. (g, h) The system has generated two false alarms, one in the Where slot, the other in the Arrested slot.

## 2.4 Information Retrieval (Document Relevance)

The Information Retrieval (IR) task involves determining whether a given document is relevant to a pre-defined information need, and also how relevant it is compared to other relevant documents (ranking). The goal is to define realistic analytic tasks that reflect the complexity of and multi-dimensional issues that arise in real-world analysis, and then explore the degree to which the various levels of information extraction can improve retrieval for such information needs. The "user" for the IR task speaks English only, however, the documents being searched are in the target evaluation languages (Chinese, Korean, Russian).

An analytic task will have an overarching information need, for example, Russian anti-corruption protests in 2018. The task will include a narrative statement of need which will be about the length of a short paragraph. Each task will have around 10 individual sentence-length information requests, for example, "Identify individual protests by their location and date, and if possible determine how many people were reported to attend," corresponding to specific aspects of the analytic task that the analyst wants to compile information into their report. The task statement and requests are in English. See the example analytic task below.

The key interaction modality for BETTER IR is *query by example*. The analyst using the BETTER system will not key in queries but rather will provide examples of documents and events that are relevant to their task. To model this, the task author will include with each analytic task and information request two example relevant documents from the English training corpus. These documents will not be graded on the relevance scale (below) but will at least be at the level of "specific information" (2 in the relevance scale on page 65). Each example will include a highlighted passage that led the task author to choose that example. The passage will be annotated for Basic events and (if applicable) Granular templates.

The set of Basic events[14] annotated in the highlight passage together form a set of *critical extractions* that the analyst expects to find in relevant documents. Notionally, we imagine that a BETTER IR system would detect and highlight these elements in retrieved documents, and the user analyst could use them to skim for content, or to quickly judge if a document is of interest or has covered already seen content. (In the alpha-nDCG metric (see section 5.2), the gain of relevant documents is discounted for repeating critical extractions from documents retrieved previously.). The extractions are not verified against any external source of truth, and may not include all slots in the basic or granular schema, only those that are mentioned in the passage. Any dates mentioned in the passage or critical extractions are from the document are not normalized or constrained in any way. The critical extractions should be considered as examples and not as structured search queries.

The example documents, highlight passages, and critical extractions are specified in BP JSON within the tasks and requests. An example analytic task and request follows:

[14] In Phase 1, this section also included Granular events as critical extractions. Since the IR evaluation will happen before Granular in Phase 2, we will not consider Granular templates as critical extractions. This is what was done in Phase 1.

```
[{
   "task-num": "DR-T4",
   "task-title": "Flint Water Crisis",
   "task-stmt": "Examine the causes and responses to the Flint water crisis that exposed residents to high levels of lead
poisoning",
   "task-link": "https://en.wikipedia.org/wiki/Flint_water_crisis",
   "task-narr": "The Flint water crisis is a public health crisis that started in 2014, after the drinking water source for the
city of Flint, Michigan was changed from treated Detroit Water and Sewerage Department water (sourced from Lake
Huron and the Detroit River) to the Flint River. As a result, the Flint residents were exposed to elevated levels of lead.",
   "task-in-scope": "",
   "task-not-in-scope": "",
   "task-docs": {"8af0b51a-cede-4822-941c-5dc32fcc5317": {
      "doc-id": "8af0b51a-cede-4822-941c-5dc32fcc5317",
      "entry-id": "8af0b51a-cede-4822-941c-5dc32fcc5317",
      "segment-type": "highlight",
      "segment-text": "...",
      "annotation-sets": {
         "basic-events": {"..."},
          "granular-events": {"..."},
      },
   },
   # ...
   }},
   "requests": [{
      "req-num": "DR-T4-r1",
      "req-text": "Identify the events in 2014 leading to increased lead in Flint's drinking water.",
      "req-docs": {"77a50dff-ba25-4041-a05c-cf107fdaec99": {
         "doc-id": "77a50dff-ba25-4041-a05c-cf107fdaec99",
         "entry-id": "77a50dff-ba25-4041-a05c-cf107fdaec99"
         "segment-type": "highlight",
         "segment-text": "the lead contamination that began in April 2014, when Flint began using the Flint River as its
water supply to save money.",
         "annotation-sets": {{
            "basic-events": {
               "events": {
                  "ev-1": {
                     "eventid": "ev-1",
                     "event-type": "Other-Government-Action",
                     "agents": ["ss-1"],
                     "patients": ["ss-2"],
                     "anchors": "anch-1",
                     "ref-events": []
                  },
                                    # ...
         }}},
      }, "8af0b51a-cede-4822-941c-5dc32fcc5317": {
         "doc-id": "8af0b51a-cede-4822-941c-5dc32fcc5317",
         "entry-id": "8af0b51a-cede-4822-941c-5dc32fcc5317",
         "segment-type": "highlight",
         "segment-text": "the state of Michigan ran Flint\u2019s day-to-day operations through an emergency manager,
who prioritized balancing the city\u2019s budget through a cost-cutting measure: switching Flint\u2019s water source in
April 2014 from Lake Huron, which serviced the city for more than 50 years, to a local river.",
         "annotation-sets": {
            "basic-events": {
               "events": {
                  "ev-1": {
                     "eventid": "ev-1",
                     "event-type": "Other-Government-Action",
                     # ...
                                    },
```

**Figure 6: IR task file JSON example**

Documents can be relevant to an analytic request on a graded scale:

0. The document is not at all relevant
1. The document is topically relevant to the analytic task and/or request
2. The document contains specific information that contributes to an understanding of the analytic request.
3. the document contains a direct answer to the analytic request.
4. the document is "decisional" -- it will drive a direct decision on the situation giving rise to the analytical task. It is a “home run” document, it is a primary citation in the report, it is the “smoking gun,” pick the metaphor of your choice.

Note that relevance is not a function of whether a document contains a critical extraction element, but only how useful the document is in answering the analytic requests and task. Relevance assessments will be done post-hoc, and during the relevance assessment process annotators will identify additional critical extractions in order to fill out an IR-specific extraction evaluation set.

The metrics (see section 5.2) map the relevance levels to gains for the nDCG metrics. For retrieving a document with relevance levels (irrelevant) or 1 (topically relevant), the system gets zero gain. For level 2 (specific information), the system receives a gain of 4. For level 3, the gain is 8, and for level 4, the gain is 20.

### 2.4.1 Evaluation Procedure

Information retrieval will be evaluated in three stages (plus an IE stage, see below): automatic, human-in-the-loop (HITL), and an “automatic HITL” condition. In each stage, the system is packaged into a Docker container and uploaded to the MITRE evaluation platform as discussed in detail later in this document. That system may spend up to five days processing the target language evaluation corpus, processing the analytic tasks and requests, and retrieving up to 1000 documents per request. The system also has access to the English training corpus while inside the MITRE evaluation platform. In Phase 2, the IR dockers supported a mode to run IE on a provided set of documents; these documents were pooled from performer retrieval runs and combined into a single set for a retrieval-focused IE evaluation. In Phase 3, there will be a task option which indicates that the system should do retrieval without the benefit of IE during the process. This option can be present in any condition (auto, auto-HITL, or HITL). Participation in the IR-without-IE experiment is required.

There will be a 5TB “scratch disk” available that systems can use to store indexes or other data from the automatic run, for use by the auto-HITL and HITL submissions. If teams wish the scratch disk wiped between submissions, they will need to notify MITRE.

In the automatic evaluation, systems will be given the task- and request-level example documents and the critical extractions, and perform a fully automatic search for each analytic request in turn. The tasks and requests are not revealed to performers, but only to systems running within the evaluation environment. The systems see the task-level documents and critical extractions, and then for each request the request-level documents and critical extractions within each task.

For HITL, performer teams will be allowed to read the analytic task text (statement, narrative, relevant documents, and critical extraction elements), as well as two analytic requests for that task (with their documents and critical extractions), and may spend up to 15 minutes per analytic task interactively working with their system on the English training corpus to "tune" it to each analytic task. The 15 minute limit applies to user interaction time only. The 15 minute block should be one complete session of interaction; it is ok to break for unplanned interruptions, but not to insert intervals for computation during the interaction. Do not, for example, have five minutes of interaction followed by three hours of computation prior to the next ten minutes. The tuned system is then packaged into a new Docker container and run within the evaluation environment over all the analytic tasks and requests. The HITL-tuned system sees the analytic task text and text for two requests, and only example documents and critical extractions for the remaining requests. The HITL user should not have a technical understanding of how the system works, since we expect the end users of the BETTER system to be analysts and not IR or NLP researchers.

The following table captures salient characteristics of the HITL task from system and performer perspectives.

| | System has access to… | Performer team members have access to… |
|---|---|---|
| During the automatic eval run | Task-level and request-level example documents, highlight sections, and critical extractions; the English training corpus, and the Target language eval corpus. | Nothing; system is running inside MITRE eval environment. |
| During the HITL tuning | Same as above, plus human input for tuning, using the English training corpus. | Task narrative, sample request statement, example documents and extractions for the task and the sample requests. |
| During the HITL eval run | Task-level and request-level example documents, highlight sections, and critical extractions; the English training corpus, and the Target language eval corpus, PLUS task narrative sections and sample request statements. | Nothing; system is running inside MITRE eval environment. |
| During the automatic HITL run | Task-level and request-level example documents, highlight sections, and critical extractions; the English training corpus, and the Target language eval corpus, PLUS task narrative sections | Nothing; system is running inside MITRE eval environment. |

| | | |
|---|---|---|
| | and sample request statements | |

**Table 11: HITL characteristics**

Since having the user key in from the narrative and examples would be a reasonable baseline HITL approach, there will also be an "automatic HITL" condition. In auto-HITL, systems will have access to the text sections of the analytic tasks (task-title, task-stmt, task-narr, task-in-scope, task-not-in-scope) and the request text for two requests, just as the HITL user does in the HITL condition. The system may make use of those text sections to formulate queries, as long as the processing happens in an automatic way. Note that as in HITL, all remaining requests past the first two are expressed to the system only by example.

For all IR tasks, each system will produce four top-1000 document rankings for each analytic request. The four rankings are one monolingual ranking for each evaluation language, and one ranking over the full collection. This format is described below in the section on the submission system.

Because of scheduling constraints between the IR, HITL, Basic, and Granular evaluations and dry runs, there will only be a single submission permitted from each team for each of automatic, auto-HITL, and HITL. In the event of substantive errors, teams will be permitted to submit one (but only one) replacement submission per unsuccessful "ready for scoring" submission. Minor errors (e.g., pathname errors) will not count against the quota. MITRE will determine minor vs. substantive errors with input from the T&E team.

There will be an IR dry run using a small, 5,000-document multilingual collection and the Phase 2 evaluation tasks.

### 2.4.2 IE-on-IR

After the IR tasks have run for all performers, NIST will pool the results for relevance assessment and annotation. After relevance assessment, a selection of relevant documents will be sentence-segmented and packaged as a test set for Basic. T&E will run a docker for Basic extraction over that test set. (This docker may be submitted up to two weeks after the HITL docker deadline, to incorporate any improvements to IE based on IR HITL.)

### 2.4.3 IR-without-IE

In the IR-without-IE experiment, systems may not use information extraction during the retrieval process. This is an end-of-program effort to benchmark "baseline retrieval" without IE for contrast with the main automatic, auto-HITL, and HITL conditions. Systems are permitted to use the annotations in the example documents.

# 3 Evaluation Structure and Schedule

## 3.1 Evaluation Structure

The four evaluation tasks described in the previous section will be explored in the context of several target languages, where the training data will be in English and the test data will be in one or more other languages. The target languages will be explored sequentially in *phases*, with each phase consisting of English for training, and one or more other languages for test data.

Each phase will also have a particular set of information needs that will be posed against the target language through the information extraction and information retrieval tasks. In Phase 1, for instance, the foreign language was Arabic, and the information needs covered a range of geopolitical events of consequence, in particular protests, government corruption, terrorist attacks, and epidemics. For Phase 2, the foreign language is Farsi, and the analytical events of interest will be announced at some date in the near future.

Within each phase the three different information extraction tasks of varying abstraction will be introduced sequentially, starting with the most abstract representation first and proceeding to add detail. The separate evaluations of each of these three increasingly fine-grained IE tasks are referred to as *waypoints* within a phase. The third and final waypoint within a phase will be combined with the evaluation of the IR task. In addition, the amount of data provided per IE task is to decrease roughly by an order of magnitude from one task to the next, as illustrated by the diagram in Figure , below.

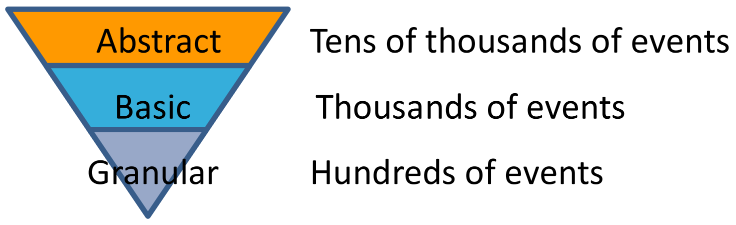


**Figure 7. Amount of training data for each IE task.**

These event counts do not necessarily correspond linearly to document counts. For the Abstract task, the training and test data provide pseudo-documents consisting of one single sentence. The Basic and Granular tasks share the exact same corpus of several hundred documents. This actually represents an expansion of the initial estimate for the amount of Basic data to be provided to performers; it turns out additional Basic annotation was required in order to have an adequate corpus of Granular events.

Since the BETTER program wishes to promote research into the rapid training or tuning of these tasks to meet what might be very specific interests of a small group of analysts, the program is interested in exploring mechanisms for human-in-the-loop (HITL) system refinements that could further quicken the pace of model customization, focused on their use for the Information Retrieval task. Hence, IR is broken into two stages: an automatic baseline and a human-in-the-loop stage where performers may customize their systems to the analytic tasks given a small amount of analyst-provided context.

## 3.2 Specific Roles of the Training and Test Data

All of the data provided to performer teams for training and development will be divided into three distinct sets, each of which is to be treated differently by the performer teams. The T&E team will provide a free-standing scorer that can be used to assess development performance.

- **Training**. Seventy percent of the data provided will be identified as training data. These data can be used to feed the model building procedures adopted by the performer teams. They can be analyzed manually and automatically in any way that suits the performers.
- **Analysis**. Fifteen percent of the data provided will be identified for use in performer teams analyzing the behavior of their models on held-out data. This data can be examined manually by the performer team developers but should not be included in the model training regimen.
- **Dev-Test**. Fifteen percent of the data provided will be restricted for use as blind test data for performer team use. These data are NOT to be examined manually or statistically by the performer teams but instead used to internally evaluate the performance of their system's on held out data, and to provide the Program Manager with performance reports at regular intervals.

*Hidden Training.* As mentioned in the first section, some amount of the training data will *not* be provided to performer teams but will instead be retained within the T&E computing environment. These data will be made available to the systems themselves at the time of evaluation, allowing each of the performer systems to carry out additional model development prior to running on test data.

*Topic and event coverage.* Performers can expect that the topics covered by the English training data will be roughly (but not identically) reflected in the foreign test data. This is especially true for the Basic and Granular tasks, which will be refreshed with new training and test data in Phase 2 and Phase 3. Aside from topic coverage, however, no attempt will be made by T&E to impose any pre-determined distributions of event types. Performers should expect that Abstract and Basic events, in particular, will occur according to their "natural" distributions for the topics covered by the data (experience to date suggests that these distributions are Zipfian).

*Release of test data.* After the completion of each phase, and after final analysis of performer output by the T&E team, the foreign test data will be released to performers. The expectation is that performers might be able to use the former test data to improve their multi-lingual language models. The hidden English training data will remain hidden, however.

## 3.3 Overall Evaluation Schedule

As of April 2022, the BETTER program is entering Phase 3 of three planned phases. Phase 1 ran through March 2021, with Phase 2 starting up in April 2021. Phase 2 ran for 12 months, through March 2022. Phase 3 will have several salient events: a demo day in June 2022, the Basic evaluation in July, the IR/HITL evaluation in October and November, and the Granular evaluation in February 2023.

Phase 3 will not include any evaluation of the Abstract task. This is a change from Phase 1 and Phase 2, whose first rounds of evaluation were for Abstract. Phase 1 had the full Abstract task, and Phase 2 had a simplified version that dropped the event type subtask. Earlier versions of this evaluation plan detail the evaluation procedure for Abstract in Phase 1 and Phase 2 (which see). Briefly, these earlier Abstract evaluations followed essentially the same procedure as documented here for Basic. Performers submitted their systems as Docker containers which were run and scored by the T&E team (run 1). The T&E team also re-trained the submissions with hidden training data (run 2). In addition, for Phase 1 Abstract, the T&E team assessed sensitivity of results to the test corpus size by running both a full and an abbreviated version of the test corpus (this was only done for Abstract, not for the other IE evaluations).

| Event | Date | Deliverables |
|---|---|---|
| | | Performer Team Deliverables |
| | | T&E Deliverables to Performers |
| **Phase 3 Kickoff** | 2021/04/01 | Target languages for Phase 3 |
| **Basic IE Data Delivery** | Mid-April 2022 | Basic Phase 3 data (English), split into training, analysis, and dev test; Phase 3 Korean, Chinese, and Russian dev test mini-samples |
| **Abstract IE Data Delivery** | Mid-April 2022 | Korean Abstract training data, to be used for training A-B Basic comparison |
| **Kickoff Meeting** | 2022/04/25 | Briefing charts and eval plan |
| **Demo Day** | 2022/06/01 | Demo presentations to government partners; College Park, MD |
| **Basic Dry Run** | 2022/07/07 | Performer Basic IE containers uploaded to T&E web site |
| **Basic Dry Run Results** | 2022/06/30 | Basic IE dry run results returned to performers |
| **Basic Eval** | 2022/07/28 | Performer Basic IE containers uploaded to T&E web site |
| **Basic Eval Results** | 2022/08/15 | Results of Basic IE evaluation reported to performer teams |
| **Basic "Sliced" Experiment Training Sets Delivery** | 2022/08/11 | T&E delivers training set size splits of English Basic training data |
| **Granular IE English Data Delivery** | 2022/08/22 | T&E delivers Granular English training set and Granular task |
| **Granular IE Foreign Sample Delivery** | 2022/08/31 | T&E delivers small samples of Korean, Chinese, and Russian |
| **Basic "Sliced" Experiment Outputs Due** | 2022/09/30 | Outputs of in-house runs for "sliced" experiment delivered to T&E for scoring |
| **IR/HITL Data Delivery** | Fall 2022 | T&E delivers English training set, and small Korean, Chinese, and Russian samples to performers |
| **IR Dry Run (T&E environment) begins** | 2022/09/20 | Performer IR systems uploaded to T&E web site |
| **IR Dry Run results** | 2022/10/04 | Results of IR dry run reported to performer teams |
| **IR/Auto Systems delivered** | 2022/10/31 | Performer teams upload Automatic IR systems to Submission Server |
| **IR/HITL Analytic Task Information to Teams** | 2022/11/01 | Analytic Task Information (task description, analytic requests, example relevant docs, critical extractions) delivered to performer teams |
| **IR/HITL Tuning in Performer Computing Environment** | 2022/11/01 – 2022/11/07 | Performer teams perform Human-in-the-Loop (HITL) IR tuning on Analytic Task info + English Data |
| **IR/HITL Systems Delivered** | 2022/11/08 | Software containers of performer IR/HITL systems uploaded to T&E web site |
| **Granular IE Dry Run** | 2023/01/13 | Software containers of performer Granular IE systems uploaded to T&E web site |
| **Granular IE dry run results** | 2023/01/27 | Results of Granular IE dry run reported to performer teams |
| **Performer teams upload Granular IE Systems** | 2023/02/09 | Software containers of performer Granular IE systems uploaded to T&E web site |
| **Granular Eval Results** | 2023/03/15 | Results of Granular IE eval reported to performer teams |
| **Demo Day/Final PI meeting** | (TBD) | Demo presentations to government partners; Demos address feedback from first Demo Day event |

**Table 12. Detailed timeline for evaluation events during Phase 3**

### 3.3.1 Materials Previously Provided for Phase 1 and Phase 2

Over the course of Phase 1, performers were incrementally provided with English training data for the Abstract, Basic, and Granular tasks. In addition, small samples of target language test data were provided to verify end-to-end functionality of performer systems. After the final evaluations for Phase 1, performers were provided with the full target language test corpus. The T&E team also provided a larger representative corpus of unannotated "clean" news data (collected from Common Crawl) in both English and the target language.

Similarly, performers were provided new topic-relevant Basic and Granular training data (in English) over the course of Phase 2. No new Abstract data were provided for Phase 2. Following the Phase 2 evaluations, the Farsi test sets were released to performers as well.

### 3.3.2 Program Schedule for Phase 3

Table 2 above provides a detailed timeline of Phase 3 activities. Phase 1 and Phase 2 followed much the same schedule, with somewhat different time spans between waypoints, and with Phase 1 additionally having a longer start-up period. Phase 3 differs from the two preceding phases in dropping the formal Abstract evaluation, and in introducing some variants to the Basic and Granular evaluations. All phases are planned to have staggered data deliveries for each task, followed by evaluation dry runs and then actual evaluations.

#### 3.3.2.1 Phase 3 Kickoff Dates

Phase 3 officially begins on April 1, 2022, with the virtual kickoff meeting to be held on April 25. The first release of Phase 3 data is expected to occur in the second week of April, and will consist of Basic training data in English. This will be soon followed by the release of Korean Abstract data, to be used for A-B pre-training comparisons in the Korean Basic evaluation.

The Phase 3 Basic data join the other data sets that have already been shared with performers during Phase 1 and Phase 2. The Phase 3 corpus will expand upon the earlier Basic event corpora in two ways: (i) it will contain additional event types that were not included with Phase 1 and Phase 2, and (ii) it will expand the implicit topics of the training corpus, covering more of the analytic concerns of Phase 3. As with Phase 1 and Phase 2, a very small Basic development test set will also be provided for each of the evaluation languages.

#### 3.3.2.2 Demo Day

A one-day demo day will take place on June 1. The event is open to USG employees, BETTER T&E teams, and BETTER performers. Performers will have the opportunity to demonstrate their work to government parties with an interest in BETTER-related technologies. IARPA hopes this event will generate interest in the program and generate successful collaborations and transitions. Partner feedback will inform preparations for a second PI meeting/Demo Day event to be held at the end of the program. The BETTER program may make available data sets of relevance to program partners. Performers would have the opportunity to demonstrate their systems on these data sets.

#### 3.3.2.3 Basic Event Extraction Evaluation

As with prior phases, the Phase 3 Basic evaluation will take place over two months, with a dry-run in mid-June 2022, the actual evaluation in mid-July, and results reported back to performers by mid-August. Performer systems will be submitted as Docker containers, which will be run and scored by the T&E team within the T&E environment (submission and evaluation are described in Section 4 below). Performers will have the opportunity to submit three different system configurations, with results reported back for all three.

Docker containers will be expected to run either as pre-trained by performers (run 1), or as retrained by the T&E team during the evaluation cycle (run 2). The pre-trained systems will incorporate the Basic training data distributed with the Phase 3 kickoff, and the re-trained systems will have access to both the original distribution and an additional corpus of hidden training data, available only within the T&E evaluation environment. Re-training will be performed on all three Docker submissions, yielding six sets of evaluation scores.

New to Phase 3, for Korean only, the Basic evaluation will require additional submissions that will help assess the utility of Abstract annotation. Specifically, performers will be asked to provide their three standard Basic extraction configurations (the A systems), as well as versions of the same systems for which performers will have had access to Korean Abstract data with which to pre-train their models (the B systems). T&E will run the B systems (run 3) on the same test data as for run 1 and run 2, thus providing an A-B comparison of Abstract pre-training in the target language. The B systems will only be used in their pre-trained configuration; there will be no B system re-training in the T&E environment.

As with the other evaluation tasks in Phase 3, Basic submissions will be expected to operate on all three evaluation languages out of the same Docker container.

#### 3.3.2.4 IR Evaluations

As noted in prior versions of this evaluation plan, BETTER management and T&E originally planned to hold the Phase 3 IR evaluations after the Granular one. The idea at the time was to emphasize Granular extraction as the locus of the IE/IR interface. Results from the Phase 2 Basic and Granular evaluations, however, suggest that Basic extraction is at a more point in technology development, and is more likely to offer synergy opportunities with IR. The IR evaluations for Phase 3 have thus been left where they currently are, after Basic and before Granular.

To provide performers with more opportunities to develop a synergistic IE/IR integration, we expect to increase the time between the Basic and IR evaluations. In Phase 2, the schedule only allowed for two weeks between the posting of Basic evaluation results and the Docker submissions for the first round of IR evaluation. For Phase 3, we expect this interval to be closer to two months.

In addition, the IR-without-IE experiment allows teams to highlight the impact of IE in their IR systems.

#### 3.3.2.5 Granular evaluation

The Phase 3 Granular evaluation will take place in January and February of 2023. This places the evaluation after the completion of the IR evaluations; as noted above, this is reverses earlier notional plans to sequence Phase 3 Granular ahead of Phase 3 IR.

As with prior Granular evaluations, the templates for Phase 3 will speak directly to a number of specific analytic topics. Granular events in the training and evaluation corpora will primarily, though not exclusively, consist of instances of these Phase 3 event types. As with previous evaluations, performers can expect to also encounter non-Phase 3 event types defined in Phase 1 and Phase 2. In addition, performers can expect to encounter a number of off-topic documents that have no Granular events at all. Off-topic documents were included in the Phase 1 data sets, but were inadvertently left out of Phase 2.

For Phase 3, T&E and BETTER management are considering two additional subtasks of the Granular evaluation: (i) Basic-enriched Granular (Korean only), for which the Korean Basic event test set would be made available for pre-training; and (ii) a HITL variant of Granular. These remain notional extensions to the Granular evaluation; details will be forthcoming if they become actual components of the Phase 3 Granular evaluation.

# 4 Using the Submission Server

The T&E team has developed a “submission server” web site where performer teams will upload their IE/IR systems for scheduled T&E, after independently developing them in their own computing environments. This can be done at any time before an evaluation, at which time the site is locked. During the evaluation itself MITRE will be executing the Dockerized performer systems against the battery of test data and as a function of a variety of control variables, such as whether or not additional “hidden” training data is made available to those systems (but unseen by the development team). The submission server also allows the results of each evaluation to be posted, so performer teams can privately access their results.

To enable execution of what might be very diverse systems the program requires that the trainable performer system has been containerized in a Docker[15] image, meeting specific requirements as described below. MITRE will then execute the performer-developed encapsulated systems in the T&E environment using controlled training data and blind test data. The Docker images are required to generate JSON outputs that conform to the “BP JSON” format, described elsewhere in this document.

Starting with Phase 2, the performer teams received local versions analogous to the T&E environment to facilitate any troubleshooting with their Docker images prior to submission for T&E evaluation.

## 4.1 Overview of the Submission Server User Interface

The frontend of the submission server has been designed to permit a simple, intuitive, and secure interface for performer teams to submit systems for testing and to review their results. The site (https://warp.mitre.org/bpeval-submissions/) also supports short program or system status announcements (as shown in Figure ), and performer access to documentation specifying submission requirements.

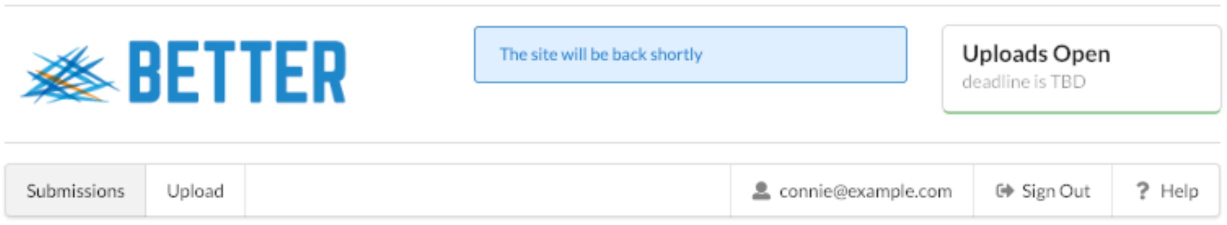
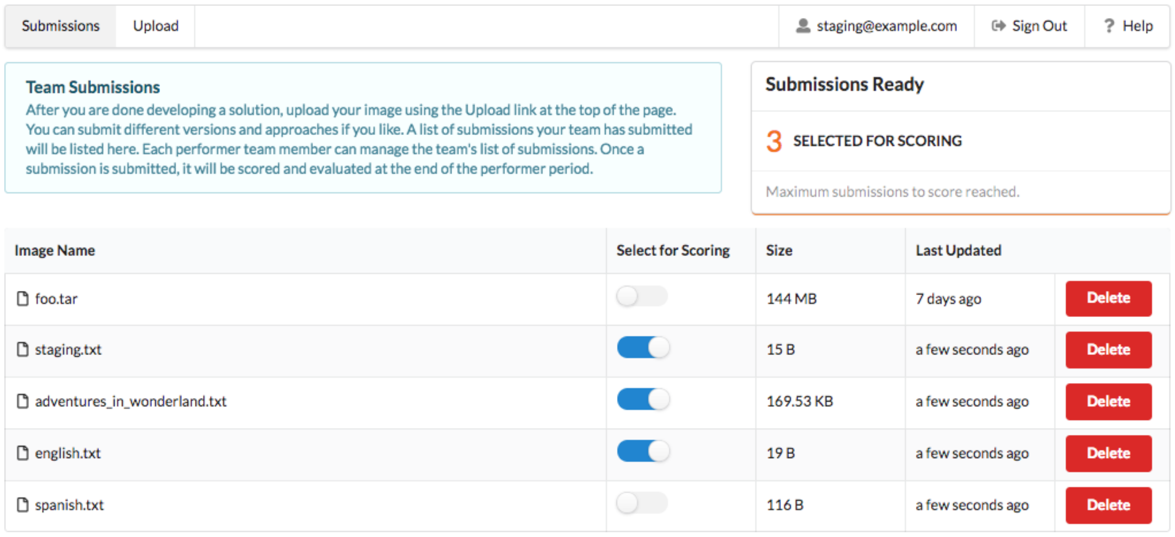


**Figure 8. A site-wide short announcement.**

Performer teams develop their system on their own machines. When ready, they export and upload their Docker image to the site. The image is stored in S3 by the backend, which uses a naming convention that includes their team and user id as a simple prefix, so that the T&E team can easily identify the provenance of Docker images. The performer teams can review their various submissions using a file listing feature on the site – see Figure , below.

[15] https://www.docker.com/



Submissions | Upload | staging@example.com | Sign Out | Help

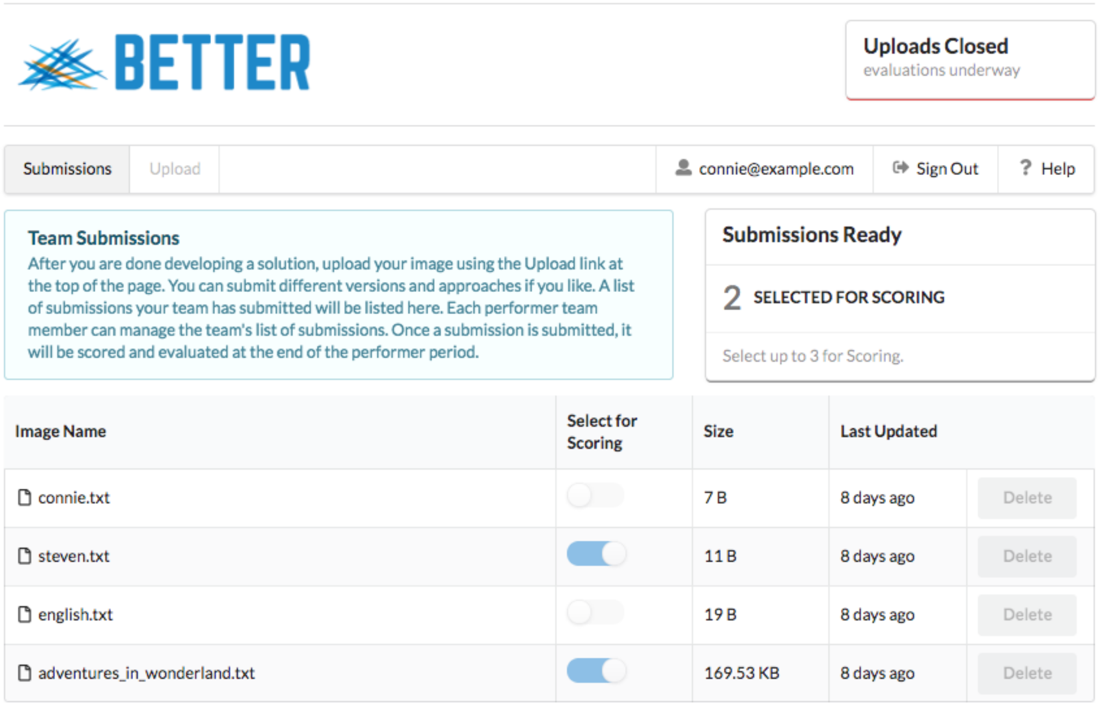
**Team Submissions**
After you are done developing a solution, upload your image using the Upload link at the top of the page. You can submit different versions and approaches if you like. A list of submissions your team has submitted will be listed here. Each performer team member can manage the team's list of submissions. Once a submission is submitted, it will be scored and evaluated at the end of the performer period.

**Submissions Ready**
3 SELECTED FOR SCORING
Maximum submissions to score reached.

| Image Name | Select for Scoring | Size | Last Updated | |
|---|---|---|---|---|
| foo.tar | | 144 MB | 7 days ago | Delete |
| staging.txt | ✓ | 15 B | a few seconds ago | Delete |
| adventures_in_wonderland.txt | ✓ | 169.53 KB | a few seconds ago | Delete |
| english.txt | ✓ | 19 B | a few seconds ago | Delete |
| spanish.txt | | 116 B | a few seconds ago | Delete |

**Figure 9. Performer team views of submitted systems and/or models.**

When the submission deadline is reached, the site is switched to read-only using a backend task. Figure  shows the same file view once the site has been “frozen” prior to an evaluation.



BETTER

**Uploads Closed**
evaluations underway

Submissions | Upload | connie@example.com | Sign Out | Help

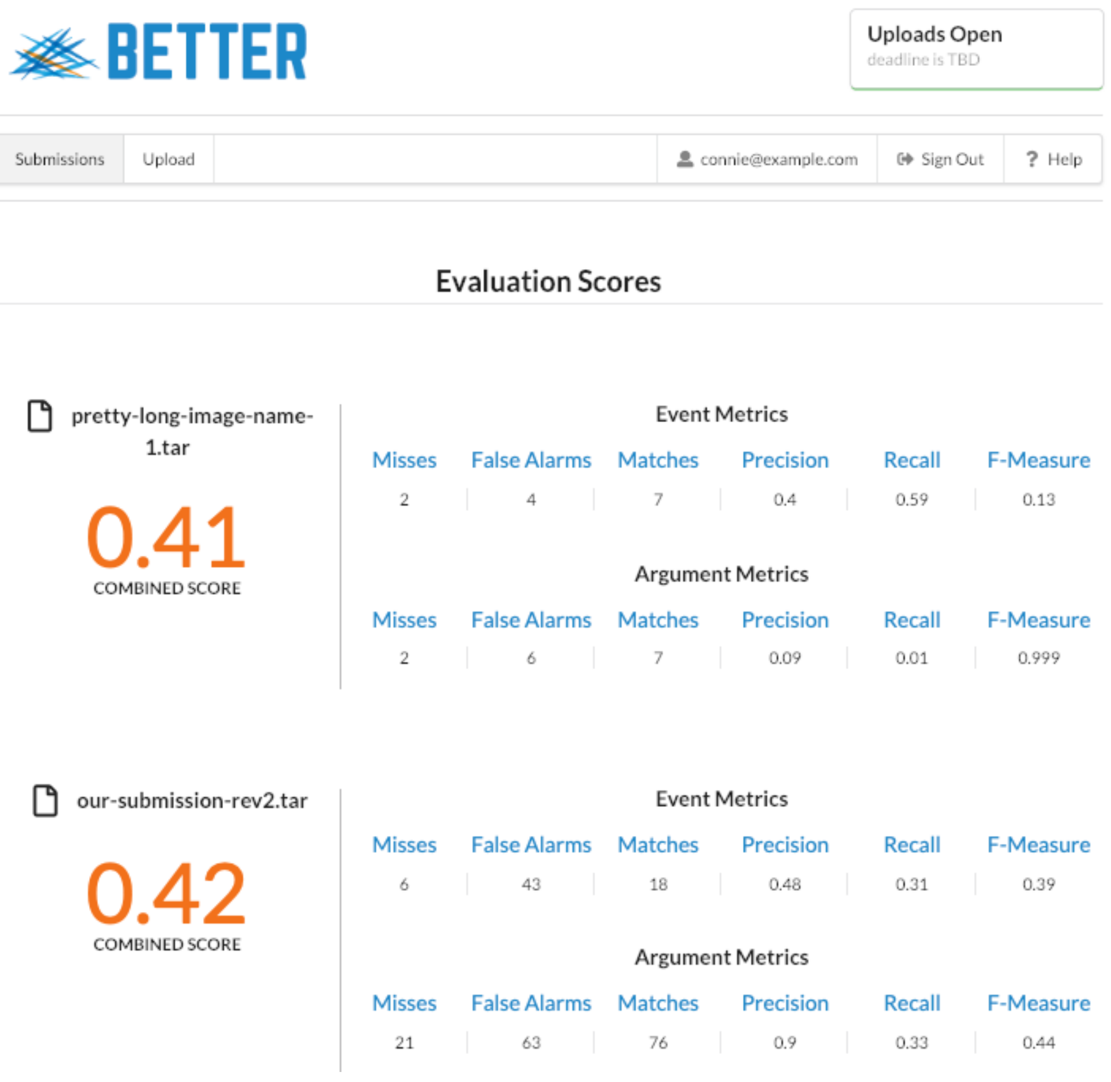
**Team Submissions**
After you are done developing a solution, upload your image using the Upload link at the top of the page. You can submit different versions and approaches if you like. A list of submissions your team has submitted will be listed here. Each performer team member can manage the team's list of submissions. Once a submission is submitted, it will be scored and evaluated at the end of the performer period.

**Submissions Ready**
2 SELECTED FOR SCORING
Select up to 3 for Scoring.

| Image Name | Select for Scoring | Size | Last Updated | |
|---|---|---|---|---|
| connie.txt | | 7 B | 8 days ago | Delete |
| steven.txt | ✓ | 11 B | 8 days ago | Delete |
| english.txt | | 19 B | 8 days ago | Delete |
| adventures_in_wonderland.txt | ✓ | 169.53 KB | 8 days ago | Delete |

**Figure 10. File listing "frozen" prior to evaluation.**

When the MITRE T&E team performs an evaluation, results are organized into BP JSON result documents and placed in a directory where the backend can read it.  By design, performer teams can access only their own results, as shown in Figure ; there is no public scoreboard.



Evaluation Scores

pretty-long-image-name-1.tar

0.41 COMBINED SCORE

Event Metrics

| Misses | False Alarms | Matches | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 2 | 4 | 7 | 0.4 | 0.59 | 0.13 |

Argument Metrics

| Misses | False Alarms | Matches | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 2 | 6 | 7 | 0.09 | 0.01 | 0.999 |

our-submission-rev2.tar

0.42 COMBINED SCORE

Event Metrics

| Misses | False Alarms | Matches | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 6 | 43 | 18 | 0.48 | 0.31 | 0.39 |

Argument Metrics

| Misses | False Alarms | Matches | Precision | Recall | F-Measure |
|---|---|---|---|---|---|
| 21 | 63 | 76 | 0.9 | 0.33 | 0.44 |

**Figure 1. Team access to their own detailed score reports.**

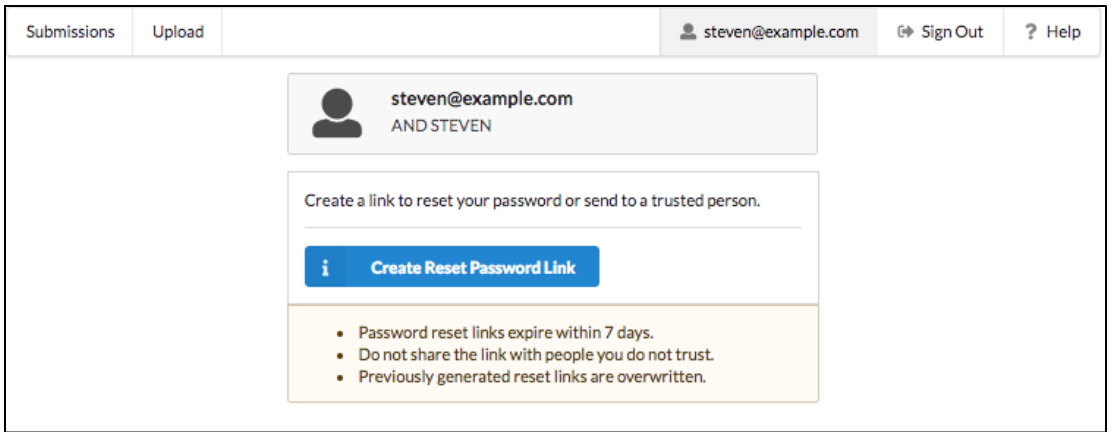
Teams can only see images uploaded by their own team members. Images are in an S3 bucket that only the backend (or an administrator) can read, and the only way to gain access to the backend is with a token which identifies a user. Results posted in the team-specific areas of the site are protected in the same way.

Administrators need `ssh` access and keys, and superuser (`sudo`) permissions on the backend machine to administer the system.

The submission server supports a password reset system that works as follows:

1. An administrator creates the users and the teams.
2. No initial password is created but instead a password reset URL is generated that uses a token.
3. The user is emailed the URL and asked to reset their password.
4. If successful, the reset link is invalidated.
5. If not used, the reset link expires in 7 days.

Users can then reset their passwords using the password reset screen shown in Figure .



steven@example.com
AND STEVEN

Create a link to reset your password or send to a trusted person.

- Password reset links expire within 7 days.
- Do not share the link with people you do not trust.
- Previously generated reset links are overwritten.

**Figure 2. A user resetting their password.**

## 4.2 "Zip-splitting" Docker images prior to upload

As noted earlier and in separate discussions, we have not yet resolved a problem in uploading files larger than 5GB into our AWS storage. For this reason, we require that each performer team split their Docker images into a set of files that are no larger than 4GB (to be safe). The process for doing this is described here. We expect this issue to be resolved before the next evaluation.

The format of Docker submissions when using docker-compose should look like this:

```
submission.zip           # any name you want
├─ docker-compose.yml    # this will run the entire system
├─ image-1.tar           # exported tar file from docker save
├─ image-2.tar           # another docker saved image
└─ README.md             # file containing instructions and notes
```

Performer teams should export each Docker image as image-1.tar with a command like this:

```
    `docker save [image-name:tag] -o image-1.tar`
```

This should be repeated for every image in the composition of systems. If the Docker submission is a single process or can run serially, then one does not need to follow this structure exactly, but the final submission format still needs to be a single zip. The submission files should include instructions or a script and/or additional indications as to how the system should be run within a Linux operating system.

A submission needs to be split into <5GB chunks using a zip split tool. We recommend aiming for 4GB just to be safe. Here is an example zip split command you can use:

```
zip -s 4g submission.zip [files]
# where submission.zip is any name you want to identify it

# an example using the -r flag to grab a whole directory:
zip -r -s 4g dry-run-test.2020.05.20.zip dry-run-test/
```

This will create the following files for you with each file being no larger than 4GB.

```
dry-run-test.2020.05.20.z01
dry-run-test.2020.05.20.z02
dry-run-test.2020.05.20.zip
```

Where the last split file will be the smallest in size (this is ok). unzip (on Linux) does not handle unsplitting nicely. The MITRE T&E team has been using 7zip to unsplit them. 7zip supports the zip format too. If you want to create a zip file using 7zip, that's fine.

Here's a sanity test that requires that p7zip has been installed on the system being used.

```
7z l dry-run-test.2020.05.20.z01
# note that you might not be able test the last zip file like this
```

Each of these parts should be uploaded to the Submission Server and the T&E team will assemble them. While it is possible see which performer team has uploaded individual files, it would be helpful if the names of the files could clearly indicate the team name and other aspects of the system(s) being uploaded.

## 4.3 T&E Environment Hardware

The evaluation box has

- 4x Tesla V100-SXM2-16GB cards.
- 32x Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz
- 240GB of memory

## 4.4 System Behaviors for Running in T&E Environment

The final form of the "Dockerized" system (that is, the Docker image in which the performer system has been containerized) must conform to a small set of simple constraints – namely, that the Docker image has been developed to access four predefined logical filenames that are command-line arguments to the image. Docker provides a facility that will allow these named files to be replaced at execution time with alternative file paths, which will be those that are specific to the test and evaluation computing environment. While the performer teams may have tested their systems using some local development versions of these four directories and files, MITRE will be running the Docker images during evaluation against a set of local files and directories that contain the data used in the evaluation. The four named logical files and their functions are listed below.[16]

### 4.4.1 `/app/tasks.json`

This is a JSON file that is intended to be opened for reading by the Dockerized performer system, and which is used to inform the performer system what task or tasks it is being asked to perform. The set of expected tasks for each "waypoint" of the program will already be very clear, but as the number and variety of tasks grows over the course of the program, it will be useful to control which of the specific tasks are to be carried out by the performer system for

[16] This description of performer system behaviors does not include addressing what might be needed to support HITL, which, as noted earlier, must be proposed by performer teams, and developed by MITRE and in collaboration with the performer teams.

some specific evaluation purpose. This file will also provide other metadata regarding the nature of the experimental conditions, which should be passed on through to be included in the results file (`/app/results.json`, see below) written at the conclusion of performer system execution. (There is a field within the results file in which the information provided in the `/app/tasks.json` can be placed, verbatim.) An example of what this file is expected to look like is shown below[17].

The set of tasks to be performed are specified in the "task-set" field, which is a table indexed by the various types of tasks within the program.[18] At the moment these are defined as one of these task identifiers:

- `extract-abstract-events`
- `extract-basic-events`
- `extract-granular-events`
- `find-relevant-docs.automatic`
- `find-relevant-docs.automatic-hitl`
- `find-relevant-docs.hitl`

If an entry for one of these task types is missing from the "task-set" table, then it is not to be performed. If it is present, then the performer system should find the table that is indexed by this task identifier. If the table indexed by a task name has a field called `"perform?"` that is set to `true`, then then the system should perform that task on the test data. For any task that is to be performed, there may be additional parameters specified within the table indexed by the task name. In the example below, we see that an additional parameter has been provided for the `extract-abstract-events` task, stipulating that the system should use the available additional "hidden" training data to improve its model(s) before attempting to perform the extraction task.

For the IR and HITL tasks, the task table will have additional fields:
- `corpus-location`: the path to the evaluation corpus.
- `scratch-storage`: the path where scratch space is mounted. Systems may use this space to store indexes or other data precomputed from the evaluation data.
- `ie-allowed`: if 'true', the system may use IE in order to improve the IR results. If 'false', then the system runs without using IE in the retrieval process.

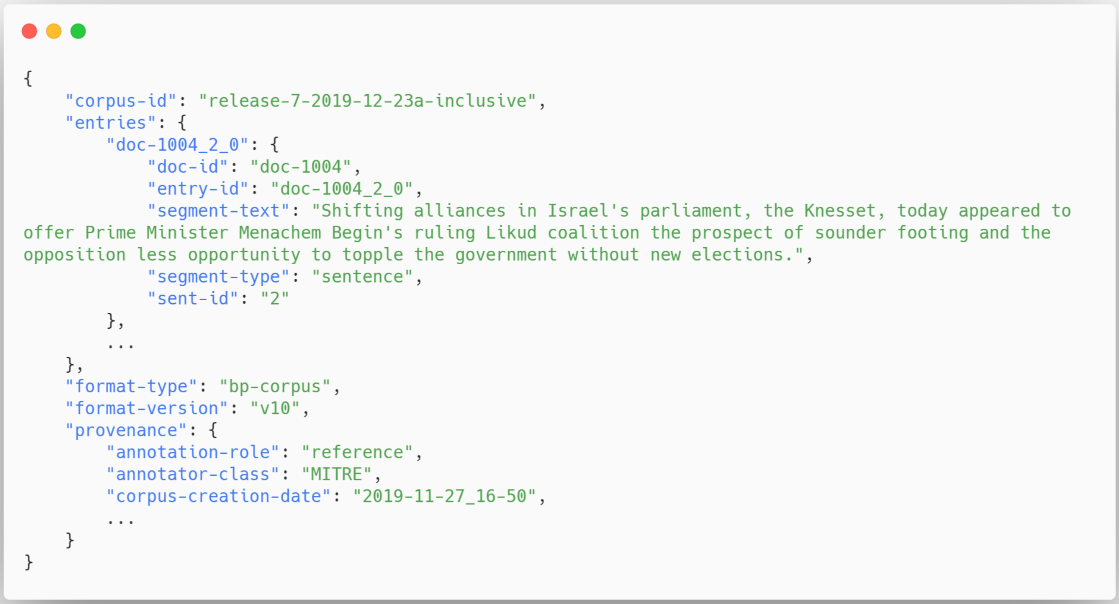
### 4.4.2 `/app/train_data.bp.json`

This is a file that will be used to provide additional training data to performer systems for the IE tasks. The format of these data will be BP JSON[19], identical to the training data provided previously to performer teams. Under some experimental conditions this file may not contain any additional training entries, although it will still be a "well-formatted" BP JSON file (simply lacking any "entries"). In such a situation the performer system will be expected to carry out an extraction or document relevance task without the benefit of any additional training data.

[17] The format of this file was changed slightly after version 0.6 of this Evaluation Plan document (dated November 2019).
[18] For Phase 2 the IE task in IR will re-use the 'extract-basic-events' task.
[19] The BP JSON format is described in more detail in Section 6.

### 4.4.3 `/app/test_data.bp.json`

This is the file that will be used to indicate all the data on which the system is expected to perform the indicated IE task(s). For the various information extraction tasks (extract-abstract events, extract-basic-events and extract-granular-events), the BP JSON file will contain a list of entries that contain text content (in the "segment-text" field) in the form of either a single sentence (indicated with the field "segment-type" set to "sentence", such as is used for abstract event extraction) or a full article (indicated with the field "segment-type" set to "document", for basic and granular). The identifiers associated with each entry (in the "entry-id" field) must be used by the system when generating its output, which will also be in BP JSON format and which is described next. An example of what the test data file will look like for the Abstract evaluation task is shown here:



```
{
    "corpus-id": "release-7-2019-12-23a-inclusive",
    "entries": {
        "doc-1004_2_0": {
            "doc-id": "doc-1004",
            "entry-id": "doc-1004_2_0",
            "segment-text": "Shifting alliances in Israel's parliament, the Knesset, today appeared to
offer Prime Minister Menachem Begin's ruling Likud coalition the prospect of sounder footing and the
opposition less opportunity to topple the government without new elections.",
            "segment-type": "sentence",
            "sent-id": "2"
        },
        ...
    },
    "format-type": "bp-corpus",
    "format-version": "v10",
    "provenance": {
        "annotation-role": "reference",
        "annotator-class": "MITRE",
        "corpus-creation-date": "2019-11-27_16-50",
        ...
    }
}
```

**Figure 3 A test data file**

### 4.4.4 `/app/ir-tasks.json`

This is the file that will be used to indicate the IR and HITL analytic tasks. This JSON format differs from BP JSON, but where IE elements are indicated, those elements follow the BP JSON structure. An example of what the analytic tasks file will look like for the Automatic IR evaluation task is shown on page 65.

### 4.4.5 `/app/results.json`

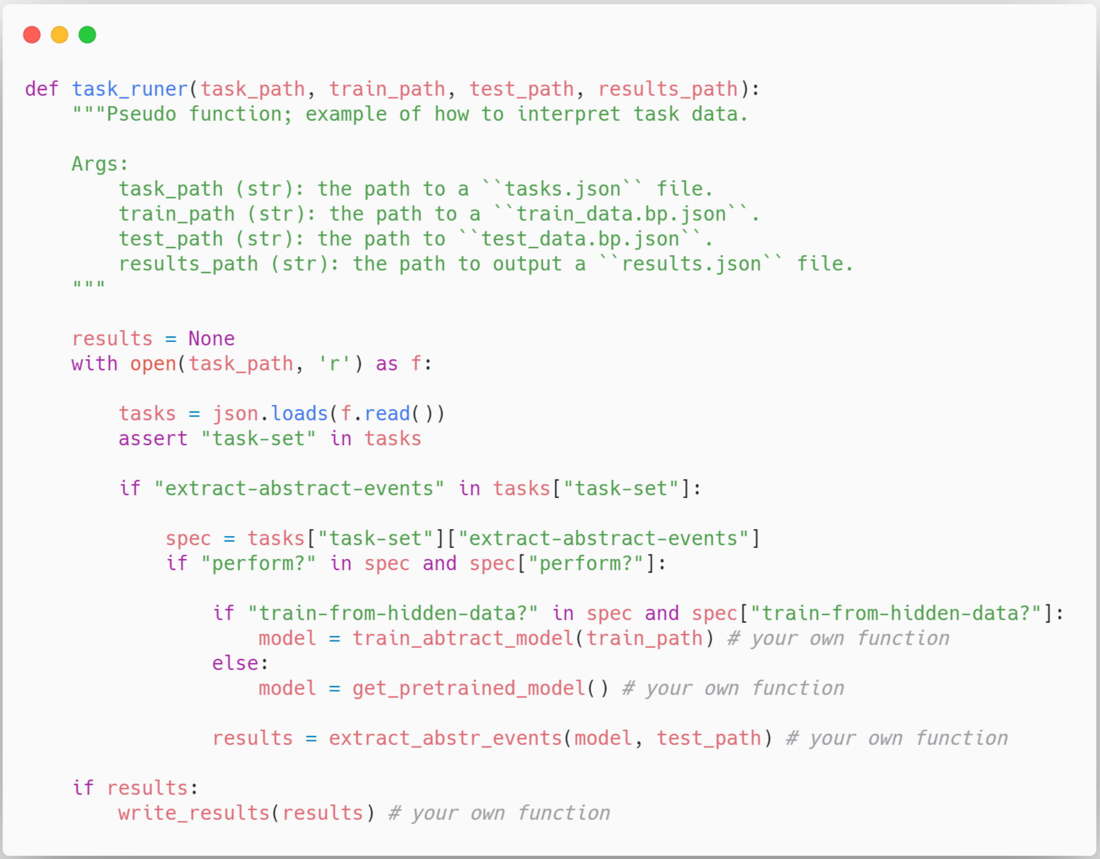
This is the name of a file that is expected to be written by the Dockerized performer system at the completion of its execution. It will report out all the results of event extraction in the BP JSON format. For IR ranking, it will report search results for all analytic tasks and information requests. For the post-IR Basic IE run (see section 2.4.2 on page 67), it will report out results as for the Basic evaluation.

```
{  "format-type": "ir-results",
   "format-version": "phase-3",
   "corpus-id": "release-1",
   "search-results": {
     "IR-T1": {
       "task": "IR-T1",
       "requests": {
         "IR-T1-r1": {
           "request": "IR-T8-r5",
           "rankings": {
             "rus": [
               { "docid": "321931", "score": 100000 },
               { "docid": "820146", "score": 99999 },
               # ...
             ],
             "kor": [
               { "docid": "123456", "score": 100000 },
               { "docid": "111222", "score": 99999 },
               # ...
             ],
             "zho": [
               { "docid": "654444", "score": 100000 },
               { "docid": "765432", "score": 99999 },
               # ...
             ],
             "combined": [
               { "docid": "987651", "score": 100000 },
               { "docid": "135790", "score": 99999 },
               # ...
           ],
```

**Figure 4: results.json from an IR submission with retrieval results only**

## 4.5 Pseudo System Example

In order to illustrate the constraints above using a concrete example, consider the "pseudo" system presented below, written in Python. This code expects four command line arguments, corresponding to those described in the previous section. After reading in the task specification JSON file, the system probes for the presence of an "`extract-abstract-events`" entry in the "`task-set`" table, and if present, retrieves the table (dictionary) indexed by that task name. It then probes that dictionary as to whether a "`perform?`" field is present and if it has been set to true. If that is the case, it starts to execute the top-level function devoted to that task. This function, in turn, checks whether it is expected to train from additional hidden training data or not, and if so invokes the training module prior to invoking the function that kicks off the abstract extraction processing, which we presume will finish by writing a BP JSON file into the file named by the invocation from the `task_runner()` function.



```python
def task_runer(task_path, train_path, test_path, results_path):
    """Pseudo function; example of how to interpret task data.

    Args:
        task_path (str): the path to a ``tasks.json`` file.
        train_path (str): the path to a ``train_data.bp.json``.
        test_path (str): the path to ``test_data.bp.json``.
        results_path (str): the path to output a ``results.json`` file.
    """

    results = None
    with open(task_path, 'r') as f:

        tasks = json.loads(f.read())
        assert "task-set" in tasks

        if "extract-abstract-events" in tasks["task-set"]:

            spec = tasks["task-set"]["extract-abstract-events"]
            if "perform?" in spec and spec["perform?"]:

                if "train-from-hidden-data?" in spec and spec["train-from-hidden-data?"]:
                    model = train_abtract_model(train_path) # your own function
                else:
                    model = get_pretrained_model() # your own function

                results = extract_abstr_events(model, test_path) # your own function

    if results:
        write_results(results) # your own function
```

**Figure 5 Pseudo Abstract system example.**

# 5 Evaluation Scoring

## 5.1 Information Extraction Scoring

The material to be scored will come in the form of a pair of BP JSON files, one from the performer system (the "system" input) and one developed by the T&E team (the "reference" input). The BP JSON format is described in Section 6 of this report. Events at all three levels of analysis in the BETTER Program (Abstract, Basic and Granular) are characterized by some kind of *type* indicator (Abstract event attributes at the Abstract level, and event types (at different levels of detail) for the Basic and Granular levels), and two or more "arguments" (or "fields", "slots"). The two arguments common to all three levels of event extraction tasks are *agent* and *patient*. The *agent* of an event is often found as the subject of a typical active voiced sentence describing an event, such as "Dr. Smith" in the sentence "A passerby, Dr. Smith, applied cardiopulmonary resuscitation to the accident victim." The agent is the person or other entity that is performing the action that is happening in an event. The *patient* is the thing (or things) to which the event is happening. Not all events have patients, but many do. In the sentence "Dr. Smith performed the necessary treatment", there is no patient argument. In the previous example sentence, the event's *patient* is "the accident victim."

The general approach to scoring event extraction in the BETTER program consists of combining the performance of *event type categorization* with the performance for correctly identifying the various *event argument strings* of those events. Determining the event categorization score consists of computing the recall, precision, and F-Measure[20] for the events taken as atomic elements, where the only requirement for a system event to match a reference event is whether or not they are of the same type. Types are specified in different ways for each of the different levels of event extraction being developed for this program (Abstract, Basic and Granular), so for this overview these distinctions are being ignored.

The scorer is responsible for counting the matches, misses and false alarms between system and reference. A *match* occurs when a system-generated event's type matches a reference event's type. A *miss* occurs when the system data has **not** provided an event that is able to match up with one or more of the available reference events. A *false alarm* occurs when there is a system-generated event for which there is no matching reference event. Note that this counting process assumes a particular alignment or "mapping" between system and reference events (or to nothing, in the situation that the system has either generated too many or too few events for a given input). The scoring software finds an optimal mapping of system-reference event pairings and uses that as the score for a given set of system events for a sentence. From these counts the recall, precision and F-measure can be derived.

The second element that contributes to a system's overall score relates to the degree to which the argument strings specified within a given system-generated event record can be lined up and matched with a counterpart argument string in the reference event to which it has been aligned (or "mapped"). The event arguments can have zero, one or multiple "entities" specified. And for each entity specified, that entity may have more than one way by which it is referred in the text, which is called an entity "mention." (We use the term entity here, but in fact, as noted

[20] Precision = match / (match + false-alarms); Recall = match / (match + misses); F-Measure = 2.0 * ((precision * recall) / (precision + recall)).

earlier, agent and patient arguments need not be "entities" *per se*, but can be events or states-of-affairs or other things that properly fill the agent or patient role as described earlier.) For example, in the example sentence mentioned earlier, the agent of the event is described via the phrases "A passerby" and "Dr. Smith". A system will be counted as having successfully provided a mention of this entity as an event argument if *either one* of these two mentioned strings is provided in the system event record. The scoring software is responsible for counting the matches, misses and false alarms that occur when considering all the entities and their many mentions that are specified in all of the arguments in an event, and from these counts are computed precision, recall and F-Measure. A pronoun mention will be accepted with a weight of 0.25 when either a name ("Dr. Smith") or a nominal ("A passerby") is available., or with a weight of 0.5 if only a nominal is available. A nominal will be accepted at a 0.5 weight if a name is available.

The overall score for a particular alignment of system events and reference events in a document is computed by multiplying the F-Measure derived from the system-to-reference event categorization (type matching) task and the F-Measure derived from the system-to-reference event argument strings matching task.

For a given set of system-generated events and a given set of reference-provided events derived from the same article, there is often going to be more than one way to align the system events to reference events based on their meeting the minimum criterion of having matching (identical) types. But the degree to which the argument string values match each other can be significantly different and could lead to very different overall score results. For example, a given text source (of any length, from single sentence to long article) might mention a number of events that have the same type. Consider the sentence "Dr. Smith helped the driver exit the car, after he had stabilized the passenger." This sentence describes two events which, in the Abstract event extraction task, have identical *types (material-helpful)*. But the patient entity strings for these two events are obviously quite different ("the driver" and "the passenger"), so the argument matching score for alignments of system-to-reference events would result in very different scores, based on which is aligned with which.

The scorer algorithm considers all possible system-reference event pairings, and it searches through all of these alignments to find one with the maximum score available across all of them. It can often be the case that there are multiple pairings that achieve the same maximum score value, in which case the scorer will select one arbitrarily – the one which it happens to have encountered first in its search. The scorer will report the specific system-to-reference pairing that was chosen based on its being the maximum possible score.

Optimizing the alignment of reference and response is unexpectedly tricky. In particular, it is not possible to optimize the official combined value metric directly, since the metric violates the linearity assumptions of the optimizer algorithm. In brief: if the response gets the type field wrong in a response-to-reference alignment, this zeroes out the scoring contribution of the response's arguments, making it impossible to distinguish a response that gets many arguments right from one that gets just one argument right.

As a result of this non-linearity, instead of attempting to optimize combined value directly, the scorer actually optimizes response gain (elements correct minus elements incorrect). This has been shown both theoretically and empirically to produce alignments that maximize the official combined value score, within a probabilistically diminutive error bound.

### 5.1.1 Word Segmentation and Scoring

Punctuation aside, the BETTER training and test data make no attempts at any kind of sub-lexical tokenization. Punctuation, however, is not included in annotation at the beginning and ends of words; this includes initial and final quote marks, parentheses, periods, and similar typical uses of punctuation. In addition, for the English training data, possessive apostrophe-s forms are not included in cases like “Mubarak’s brother” (“Mubarak” is marked, not “Mubarak’s”). Because *errare humanum est*, the scorer enforces these conventions by modifying spans that were mistakenly annotated with an inappropriate bit of punctuation. Likewise, the scorer will remove any inappropriate initial or final punctuation from system output (as well as any mistakenly included whitespace). This effectively normalizes punctuation during scoring, thus making performer scores insensitive to trivial variations in the handling of punctuation.

No further tokenization is performed beyond this punctuation handling. The absence of further tokenization matters in particular for agglomerative languages like Arabic, where the definite determiner and other syntactic markers may become attached to a lexeme. Neither annotators nor performer systems are expected to remove these markers in reporting an entity or anchor span. The span is to be reported with no sub-lexical analysis, and with no characters stripped off of tokens (punctuation aside).

Once punctuation is normalized, the official IE scoring algorithm relies on character-by-character comparisons of spans. Note in particular that this does not require that performers report precisely the same instance of a span as the annotators selected. If the agent or patient of an event is annotated as “Mubarak”, for example, it only matters that the system reports the string “Mubarak”. The scorer is insensitive to whether the system identified the entity from the same location in the texts as the annotators did, or from some different mention.

## 5.2 Information Retrieval Scoring

For Information Retrieval tasks, systems will return a ranked list of documents in standard TREC format, and a BP JSON file of extractions from those documents.

There will be four primary metrics for the Information Retrieval tasks. The first is *normalized discounted cumulative gain* (nDCG)[21], a commonly-used ranked retrieval metric:

$$\mathrm{DCG}_p = \sum_{i=1}^{p} \frac{gain_i}{log_2(i+1)}$$

$$\mathrm{IDCG}_p = \sum_{i=1}^{|rel|_p} \frac{gain_i}{log_2(i+1)}$$

$$\mathrm{nDCG}_p = \frac{DCG_p}{IDCG_p}$$

Each relevant document retrieved provides an amount of *gain* (highly skewed towards the “decision relevant” and “decisive” levels). That gain is *accumulated* from each document retrieved. However, the gain is *discounted* as we move through the ranked list. Finally, the

[21] Kalervo Järvelin, Jaana Kekäläinen: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems 20(4), 422–446 (2002)

discounted cumulative gain is *normalized* by the DCG of the ideal ranking which places all the relevant documents at the top, in decreasing gain order. nDCG is measured to the depth corresponding to the number of documents labeled as Specific Information or better (so-called *nDCG@R*).

The gain values for each level of relevance will be:

| **Relevance level name** | **Relevance code number** | **Gain value** |
|---|---|---|
| Not relevant | 0 | 0 |
| Topically relevant | 1 | 0 |
| Specific information | 2 | 4 |
| Direct answer | 3 | 8 |
| Decisive | 4 | 20 |

**Table 13: Gain assigned to relevance levels**

The second metric will be the precision and recall (and combined F1 score) of all Basic events and Granular template extractions in the top *N* retrieved documents. *N* will be small and will be determined by assessment time and budget constraints, since retrieved documents will have to be exhaustively annotated for this (and the next) metric. This metric provides a retrieval-biased measure of extraction performance.

The third metric will be the precision and recall (and combined F1 score) of analytic task and informational request critical extractions in the top *N* documents. This measure of extraction performance is biased both by retrieval and by relevance.

The last metric will be alpha-nDCG, which is a *diversity ranking version* of nDCG.[22] We consider the analytic request to have $\alpha$ "subtopics" of interest. The subtopics are defined as the critical extraction: a document is considered relevant to the subtopic if it contains that critical extraction, correctly extracted, in the highlighted relevant portion of the document. The 'alpha' term is an additional discount applied to repeat retrievals of the same critical extraction. This variant of nDCG will benefit systems that retrieve documents that cover all the critical extractions rather than focusing on only a few.

The equations for alpha-nDCG are the same as for nDCG, except that the gain value for a document is multiplied by an additional discount factor:

$$gain_{\alpha nDCG} = g * \alpha^{r_{i,k-1}}$$

Where alpha is a critical extraction discount factor (0.5 in BETTER), and the exponent is the number of relevant documents retrieved before this document with the same critical extraction. A relevant document might of course have several critical extractions mentioned in it; in that case, the gain is divided evenly across all extractions, and the alpha discount applied to each.

[22] Charles L. A. Clarke et al, "Novelty and Diversity in Information Retrieval Evaluation" Proceedings of the 2008 ACM Conference on Research and Development in Information Retrieval (SIGIR 2008), Singapore.

| Document rank | Relevance | Critical Extractions | Gain | Alpha discount | Rank discount | DCG | a-DCG |
|---|---|---|---|---|---|---|---|
| 1 | 3 | A | 8 | 1 | 1 | 8 | 8 |
| 2 | 0 | | 0 | | | | |
| 3 | 2 | B | 4 | 1 | 2 | 10 | 10 |
| 4 | 1 | | 0 | | | | |
| 5 | 2 | A, C | 4 | 0.5; 1 | 2.584 | 11.55 | 11.16 |
| 6 | 4 | C, E | 20 | 0.5; 1 | 2.807 | 18.67 | 16.5 |
| 7 | 3 | A, B | 8 | 0.25, 0.5 | 3 | 21.34 | 17.5 |
| 8 | 0 | | 0 | | | | |
| 9 | 2 | * | 4 | 1 | 3.321 | 22.54 | 18.7 |
| 10 | 1 | F | 0 | | | | |

**Table 14: nDCG and alpha-nDCG example**

Table 14 above gives an example ranking of documents. The second column is the relevance label, and the third column indicates what critical extractions were mentioned in the highlighted portion of each relevant document. The table presumes that the system has identified the critical extractions correctly; in practice, the system only gets credit when it does so.

The fourth column gives the gain value corresponding to that relevance label. The fifth column gives the alpha discount, which is multiplied by the gain in alpha-nDCG.

For documents with multiple critical extractions, the gain is split across them and the alpha for each extraction is applied respectively: the document at rank 5 mentions critical extractions A and C. The gain of 4 is split into 2 for A and 2 for C, but since A was already mentioned in document 1, it only gets half credit for that extraction, so the alpha-discounted gain is 3 (that is, 2*0.5 + 2*1). For document 6, C was already mentioned in the document at rank 5, so the alpha-discounted gain is 15 (that is, 10*0.5 + 10*1).

When gainful documents do not mention any critical extractions, such as document 9, they are assigned to a "catch-all" critical extraction with its own alpha discount; hence, more documents without critical extractions quickly lose value.

Not shown in the table is the ideal gain vector used for normalizing DCG (or a-DCG). For the nDCG case, the ideal ranking would have the gains in order (20, 8, 8, 4, 4, 0…). For a-nDCG, the sequence is (20ce, 8ab, 8a, 4ac, 4b) where the letters disambiguate which 8-gain and 4-gain documents we mean. If the 8-gain documents had only repeat critical extractions, a 4-gain document may have actually been worth more than they were.

The matching of critical extractions to events mentioned in the document is done heuristically: if an event in the highlighted portion of a relevant document matches a critical extraction event's type, the type of any referred event, and whether the event(s) is/are states of affairs events or not, then they match for purposes of alpha-nDCG scoring.

All of these metrics, along with other diagnostic metrics to be determined, will be reported as averages across analytic requests. We may also report analytic-task-level averages for diagnostic and illustrative purposes only.

The main score will be computed over the combined ranking, but language-specific ranking scores may be computed to illustrate language-specific trends.

# 6 Description of BP JSON

The primary data interchange format for the BETTER Program is BP JSON (for BETTER Program JSON). All annotated training and evaluation data are captured as standoff markup in BP JSON, as are input files to performer systems, as well as performer system output. BP JSON covers all three event extraction tasks, Abstract, Basic, and Granular. This section of the BETTER evaluation plan documents the details BP JSON, starting with general characteristics of BP JSON, followed by the specifics set out for Abstract, and then the details of Basic and Granular, which share some aspects of the markup scheme.

First, however, some background about the annotation process we used for BETTER. At the onset of the program, the T&E team experimented at length with cloud labor for the creation of what was then our first data set, English Abstract. We abandoned this approach for reasons of excessive complexity and annotator unreliability, eventually settling on hand-picked annotation teams at MITRE, and later at ARLIS. These teams rely on the MAT tool for data annotation (MAT is the MITRE Annotation Toolkit). MAT does not work natively with BP JSON, however, but uses its own JSON-coded file format, MAT JSON.

After annotation is complete, a series of data transformation scripts convert the MAT JSON standoff annotation into BP JSON. These scripts make several changes to the encoding.

- *Enforcement of head defaults:* in the interest of efficiency, annotators were allowed to leave heads unmarked when they were mechanically retrievable, e.g., the rightmost token of a noun phrase in English. This step inserts head markup for such cases.
- *Marking of non-annotatable regions:* annotators are instructed to mark certain regions of a news story as unmarkable, e.g., links to related articles. This step populates the "`segment-sections`" BP JSON key with these annotator judgments. Performers are only expected to process those segments that bear the `sentence` and `headline` types. All other segments (*e.g.,* `byline`) should not be annotated and will not be scored.
- *Sentence insertion:* this semi-automated process was only used for early versions of the English Basic data. These data had not been annotated for sentence boundaries; later Basic data sets have all been manually annotated for sentences, and future Basic data will be manually annotated as well
- *Removal of MAT detritus:* various fields that are only of value to annotators are left off during the MAT JSON to BP JSON conversion.
- *Introduction of metadata:* these fields support the T&E team with error analysis, computing inter-annotator reports, and tracing data though various stages of processing.[23]

These data transformations all occur before any annotated corpora are delivered to performers. Performers will only see complete and final BP JSON data. In addition, to aid performer teams in working with BP JSON encodings, MITRE will provide the teams with scripts and APIs to aid in the creation, parsing and syntax checking of BP JSON data.

[23] Examples of such additional information that might be specified in reference annotated data include: the text being annotated, the event trigger strings, metadata about the source of the data and the annotations, etc.

## 6.1 General characteristics of BP JSON

For expository purposes, most of the following discussion presents general characteristics of BP JSON using examples from the Abstract event extraction task. Later sections of this report cover such details as are specific to only Basic and Granular.

The BP JSON format consists of two main parts – (1) a set of metadata fields that *describe* various attributes of the corpus, and (2) a particular field, the "entries" field, whose value is *the data that constitutes* the corpus. The corpus data can consist of raw data (for example, the text of a sentence, paragraph or whole document), annotations (at any of the various types already mentioned, such as abstract events, basic events, etc.), or a combination of those two (where the annotations are derived from the raw (text) data). In the description of this format below, optional and required fields are identified. In general, it is expected that performer systems will only bother to generate the required data fields, while the reference and training data will usually include all of the optional fields.

While the BP JSON format critically capture annotations, it also organizes "raw" or unannotated data, and performer systems will be required to read and process such unannotated data as part of the evaluation. For example, when systems are asked to perform automatic Abstract event extraction, this level of analysis is performed on individual sentences, so it will be useful to provide performer systems with gold-standard sentence annotations. This will enable system output to unambiguously identify the text segments from which annotations are derived.

### 6.1.1 Corpus Metadata

In the top-level metadata fields that describe the corpus there are a small set of required fields, and a larger number of optional fields. Figure  shows an example of the top-level fields in a possible BP JSON corpus, but without showing the specifics of any individual corpus entry.

```
{
    "format-type": "bp-corpus",
    "format-version": "v10",
    "corpus-id": "some-id-string",
    "entries": {
        "doc-1004_2_0": {...},
        "doc-1004_7_0": {...},
      ...
    }
}
```

**Figure 6. The top-level fields that provide various metadata about a corpus**

The required fields are as follows:

- `format-type`: which must be the string "bp-corpus"
- `format-version`: a string, currently "v10"
- `corpus-id`: a string, ideally unique to this corpus
- `entries`: a table of corpus entries indexed by each corpus entry identifier; the structure of each corpus entry value is described below.

### 6.1.2 Corpus Entries

Entries in a BP JSON corpus (the value of the `entries` field) capture the corpus' units of analysis. For training and evaluation on the Basic and Granular tasks, corpus entries will usually be an entire document or news story. The exception to this are the critical extraction entries for the IR evaluations: these are typically paragraph-scale elements that were selected for saliency by IR evaluation analysts.

For the case of Abstract data, the units of analysis are sentences. These are generally unrelated sentences that were excerpted from multiple documents while compiling the Abstract corpus. This compilation process dropped repetitive or topically-redundant sentences so as to provide a maximally diverse and informative data set.

Each corpus entry, e.g., each value of the `entries` field in the BP JSON schema, is a dictionary of JSON dictionary elements, each indexed by a unique `entry-id` string. These elements are associated with an identified region of text (document, paragraph, or sentence – see above), and are intended to capture all the annotations relevant to that text segment. Only annotations that are derived from the identified segment will be found in that segment's entry – these entries can be considered independent from any annotations found in other entries elsewhere in the corpus.

```
{
  "entry-id": "some-id-string", // required
  "doc-id": "some-id-string",   // required
  "sent-id": "some-id-string",  // the rest are optional
  "lang": "eng", // new in Phase 3: ISO 639-3 code
  "segment-type": "sentence",
  "segment-start": 42,
  "segment-end": 84,
  "segment-text": "Ellen opened the door to welcome the chef.",
  "annotation-sets": {
    "abstract-events": {...},
    "basic-events": {...}
  }
}
```

**Figure 7. An individual entry in the BP JSON corpus format (Phase 3).**

The value of each corpus entry is itself a dictionary, in which there are only two required fields – the `entry-id` and `doc-id` fields. For training and reference purposes, however, it will be useful to include either the text itself (in the `segment-text` field), information on where to find the raw text (in the `source`, `segment-start` and `segment-end` fields), and annotations related to the raw text (`annotation-sets`). The example in Figure  illustrates some of the optional fields that can be specified in a corpus entry object:

- source (indicates the source of the text data being annotated, usually a file name)
- segment-text (which will contain a copy of the text being annotated)
- segment-type (specifies the level of granularity of this text segment – one of "sentence" or "document" – critical extraction paragraphs are typed as "document").

Starting with Phase 3, entries also may have an additional `lang` field to capture the language of the entry. The value of this field is the ISO 639-3 code for that language, which for the Phase 3 languages are: `eng` (English), `kor` (Korean), `zho` (Chinese), and `rus` (Russian).

Note that in Phase 1 and Phase 2, the lang field was not used, with the language of entries implicit in the corpus purpose: that is, training data were all English, and evaluation data were either Arabic (Phase 1) or Farsi (Phase 2). The `lang` field is technically optional for backward compatibility with these earlier phases, but all training and evaluation data for Phase 3 will have the language field set.[24]

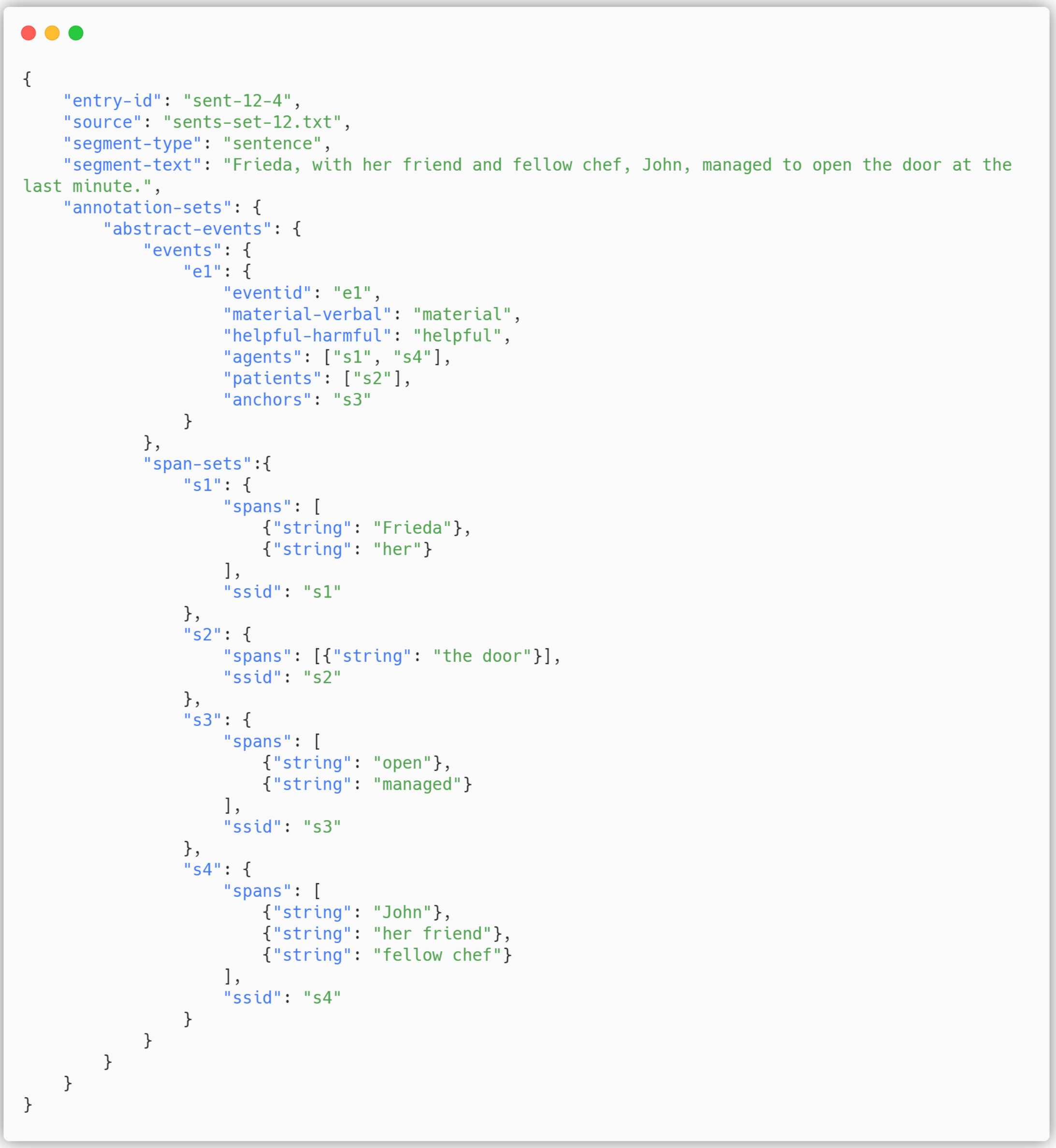
### 6.1.3 Annotation Sets

A corpus entry captures annotations through the `annotation-sets` field, a JSON dictionary indexed by the type of annotation. The keys that can appear as indices of the dictionary are either `abstract-events` (for the Abstract task) or `basic-events` (for both the Basic *and* Granular tasks) See example Figure , above. A corpus entry can in principle have both `abstract-events` and `basic-events` fields, but this will not occur in practice because the corresponding Abstract and Basic/Granular tasks range over different units of analysis.

Structures that fill the `abstract-events` or `basic-events` fields are themselves JSON dictionaries, whose two principal fields are `span-sets` and `events`. Instances of `events` structures capture the event type, its arguments, and an anchoring reference in the text signal (more on this below). The span-sets field captures the anchoring references for an event and for event arguments. Figure , below, show a simple example of this for the Abstract task.

The example continues where we had left off with the sentence in Figure : "Ellen opened the door to welcome the chef." Note how in the example there are two event structures, one (event `e1`) for the helpful-material event corresponding to Ellen opening the door, and one (event `e2`) for the helpful-verbal event corresponding to Ellen welcoming the chef. These events reference the `span-sets` entries `s1`through `s5`). Several of these capture the agent and patient arguments of the events, e.g. Ellen is `s1`; others capture the anchoring expressions for the events, e.g., the door opening is anchored to span-set `s3`, for "opened". For readability, the event arguments are specified via a span set identifier (effectively like an entity identifier), and the various strings (spans) that are co-referring are listed separately.

[24] The remaining examples of corpus entries in this document do not mention the optional language field. These examples reflect the state of the BP JSON definition as of Phases 1 and 2, for which a language field was not used. Because the field is optional in the Phase 3 specification, all these examples are valid instances of BP JSON, but would be considered incomplete in Phase 3.



```
{
    "entry-id": "sent-12-4",
    "source": "sents-set-12.txt",
    "segment-type": "sentence",
    "segment-text": "Frieda, with her friend and fellow chef, John, managed to open the door at the
last minute.",
    "annotation-sets": {
        "abstract-events": {
            "events": {
                "e1": {
                    "eventid": "e1",
                    "material-verbal": "material",
                    "helpful-harmful": "helpful",
                    "agents": ["s1", "s4"],
                    "patients": ["s2"],
                    "anchors": "s3"
                }
            },
            "span-sets":{
                "s1": {
                    "spans": [
                        {"string": "Frieda"},
                        {"string": "her"}
                    ],
                    "ssid": "s1"
                },
                "s2": {
                    "spans": [{"string": "the door"}],
                    "ssid": "s2"
                },
                "s3": {
                    "spans": [
                        {"string": "open"},
                        {"string": "managed"}
                    ],
                    "ssid": "s3"
                },
                "s4": {
                    "spans": [
                        {"string": "John"},
                        {"string": "her friend"},
                        {"string": "fellow chef"}
                    ],
                    "ssid": "s4"
                }
            }
        }
    }
}
```

**Figure 8. An Abstract event with multiple agents and their mentions.**

Going beyond the details of the example, as the name `span-sets` suggests, these structures can have multiple entries for each entity or event that they anchor. This mechanism captures co-reference, as required for evaluation purposes. In particular, the representation of event arguments needs to capture all the different mentions of the argument that might be mentioned in a text segment: these mentions are all collated into the same span set entry. The BETTER program is not explicitly evaluating co-reference resolution, so system output will be judged correct if it identifies *at least one* of the co-referring mentions of an event argument (see further discussion of this earlier in this document, in Section 2.1.3.4). For a detailed example, see Figure , above, and for further details about the fields of spa-set elements, see Figure  and the related commentary in Section 6.2, below.

In addition, note that events can have multiple agents and/or patients, and each of those may have multiple mentions somewhere within the body of the text. The BP JSON in Figure illustrates how this is handled for reference data. The event being annotated is again a door getting opened, but in this instance there are two agent arguments ("Frieda" and "John"). These in turn have multiple mentions, with "John" for example also having mentions "her friend" and "fellow chef". The reference data needs to account for all of these valid mentions of "John", although system-generated output would only need to mention one of the valid mentions.

### 6.1.4 More about Abstract events in BP JSON

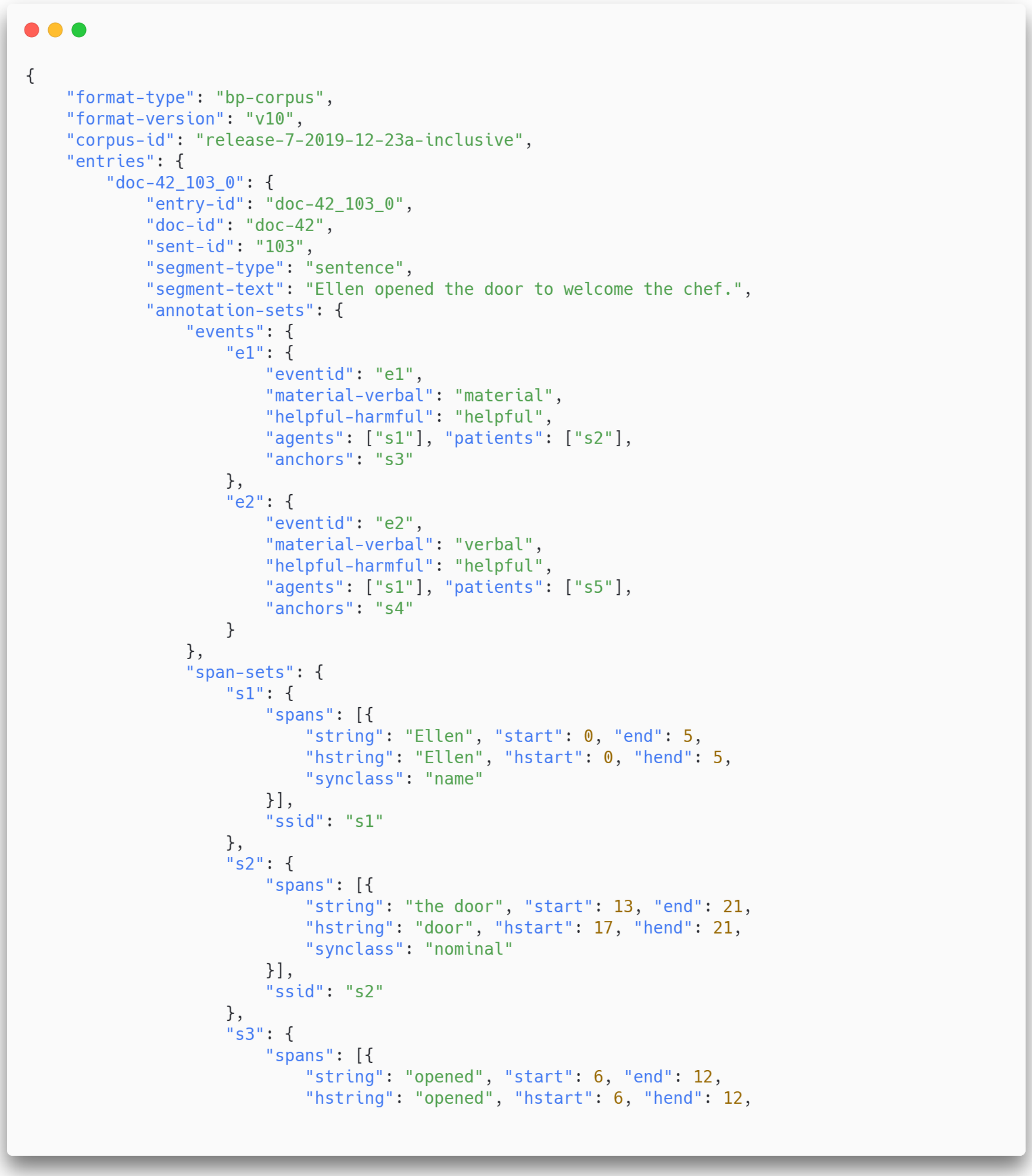
We have glossed over some details of these examples, in particular the event type specification of events. In BP JSON, the Quad Class of an Abstract event is provided as the value of two fields, `material-verbal` and `helpful-harmful`. The agent, patient, and anchor span-set references are captured by the correspondingly named fields (`agents`, `patients`, and `anchors` respectively).

The `anchors` field is optional in system output, though it will always be provided in both training and reference data. The `eventid` field is required to uniquify event references.

A final note about Quad Class: for Phase 2, systems are only optionally being scored for Abstract event type. This is accomplished at the level of the scoring algorithm, not at the level of the Abstract event annotation. The Farsi test data for Phase 2 Abstract do in fact have event type annotations; they just don't participate in the score by default. For Phase 3, the details of whether Abstract test data will or will not have event types will be announced at a later time.

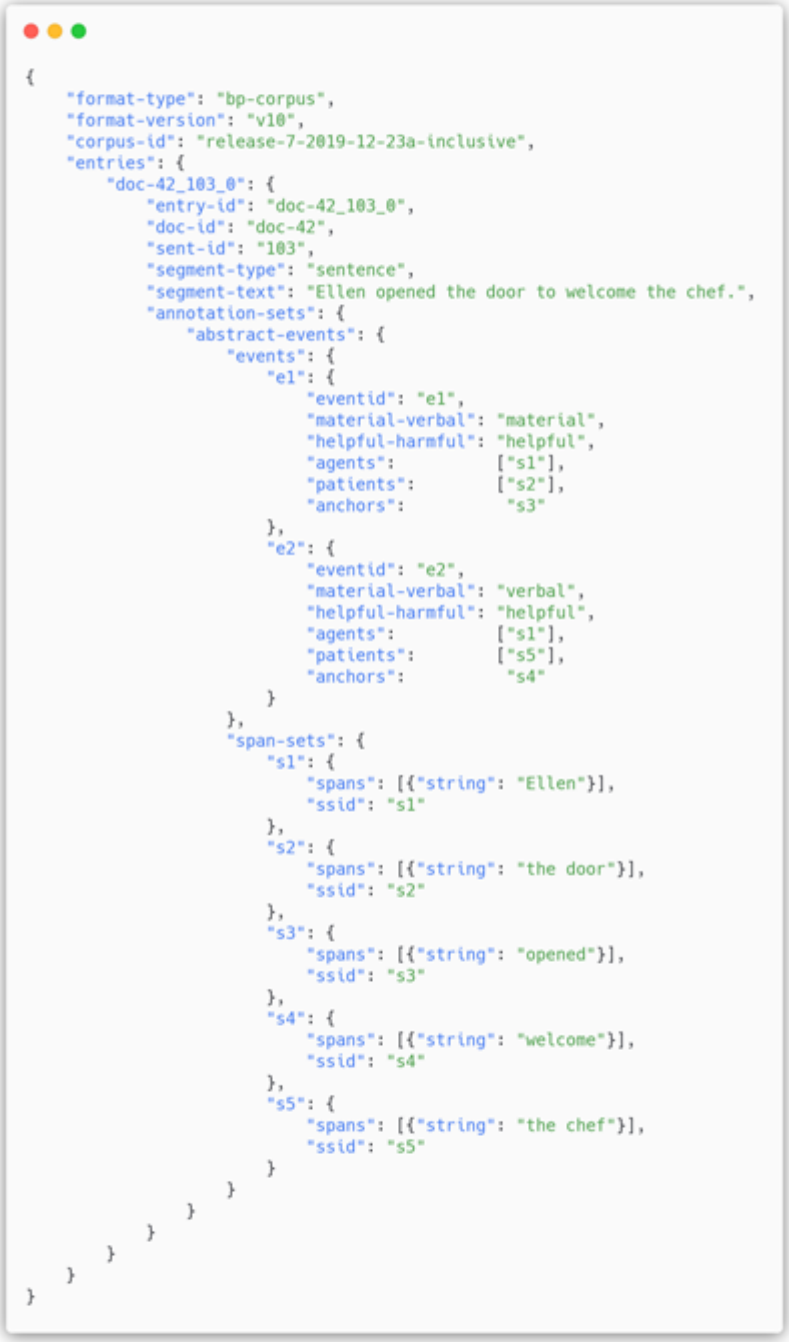
### 6.1.5 Complete Examples of Abstract events

Figure shows the previous elements of the BP JSON corpus format merged together into a single complete example. Figure is an example of reference data, since it includes optional fields that are supported in BP JSON. It should be emphasized that system-generated output is expected to be much more minimal, focusing on only those elements that are required for processing and scoring. Figure shows these same data as one would expect it to appear when generated by an automated extraction system. Notice the absence of any offsets into the sentence text.



```
{
    "format-type": "bp-corpus",
    "format-version": "v10",
    "corpus-id": "release-7-2019-12-23a-inclusive",
    "entries": {
        "doc-42_103_0": {
            "entry-id": "doc-42_103_0",
            "doc-id": "doc-42",
            "sent-id": "103",
            "segment-type": "sentence",
            "segment-text": "Ellen opened the door to welcome the chef.",
            "annotation-sets": {
                "events": {
                    "e1": {
                        "eventid": "e1",
                        "material-verbal": "material",
                        "helpful-harmful": "helpful",
                        "agents": ["s1"], "patients": ["s2"],
                        "anchors": "s3"
                    },
                    "e2": {
                        "eventid": "e2",
                        "material-verbal": "verbal",
                        "helpful-harmful": "helpful",
                        "agents": ["s1"], "patients": ["s5"],
                        "anchors": "s4"
                    }
                },
                "span-sets": {
                    "s1": {
                        "spans": [{
                            "string": "Ellen", "start": 0, "end": 5,
                            "hstring": "Ellen", "hstart": 0, "hend": 5,
                            "synclass": "name"
                        }],
                        "ssid": "s1"
                    },
                    "s2": {
                        "spans": [{
                            "string": "the door", "start": 13, "end": 21,
                            "hstring": "door", "hstart": 17, "hend": 21,
                            "synclass": "nominal"
                        }],
                        "ssid": "s2"
                    },
                    "s3": {
                        "spans": [{
                            "string": "opened", "start": 6, "end": 12,
                            "hstring": "opened", "hstart": 6, "hend": 12,
```

**Figure 9. Complete BP JSON Abstract encoding for a single sentence (closing braces elided).**

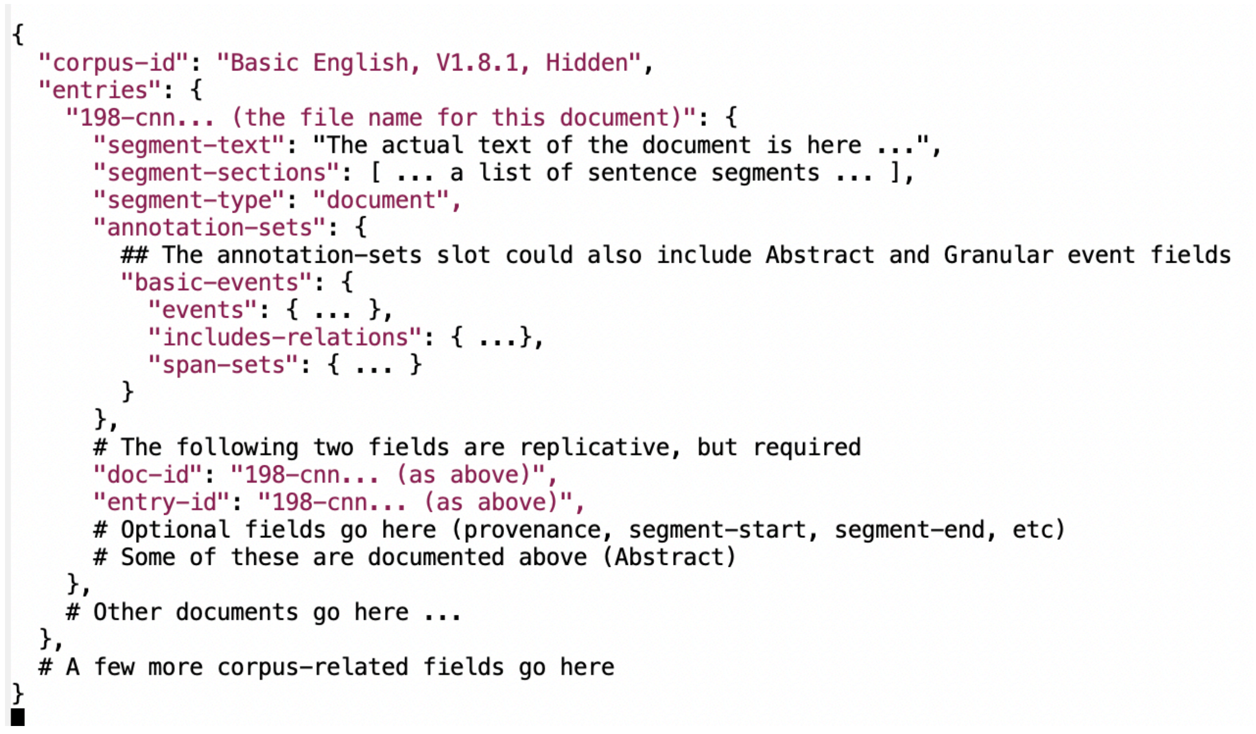


```json
{
    "format-type": "bp-corpus",
    "format-version": "v10",
    "corpus-id": "release-7-2019-12-23a-inclusive",
    "entries": {
        "doc-42_103_0": {
            "entry-id": "doc-42_103_0",
            "doc-id": "doc-42",
            "sent-id": "103",
            "segment-type": "sentence",
            "segment-text": "Ellen opened the door to welcome the chef.",
            "annotation-sets": {
                "abstract-events": {
                    "events": {
                        "e1": {
                            "eventid": "e1",
                            "material-verbal": "material",
                            "helpful-harmful": "helpful",
                            "agents":          ["s1"],
                            "patients":        ["s2"],
                            "anchors":          "s3"
                        },
                        "e2": {
                            "eventid": "e2",
                            "material-verbal": "verbal",
                            "helpful-harmful": "helpful",
                            "agents":          ["s1"],
                            "patients":        ["s5"],
                            "anchors":          "s4"
                        }
                    },
                    "span-sets": {
                        "s1": {
                            "spans": [{"string": "Ellen"}],
                            "ssid": "s1"
                        },
                        "s2": {
                            "spans": [{"string": "the door"}],
                            "ssid": "s2"
                        },
                        "s3": {
                            "spans": [{"string": "opened"}],
                            "ssid": "s3"
                        },
                        "s4": {
                            "spans": [{"string": "welcome"}],
                            "ssid": "s4"
                        },
                        "s5": {
                            "spans": [{"string": "the chef"}],
                            "ssid": "s5"
                        }
                    }
                }
            }
        }
    }
}
```

**Figure 20. An example of a system-generated BP JSON file that captures the Abstract event annotations on a single sentence.**

## 6.2 Basic event encoding in BP JSON

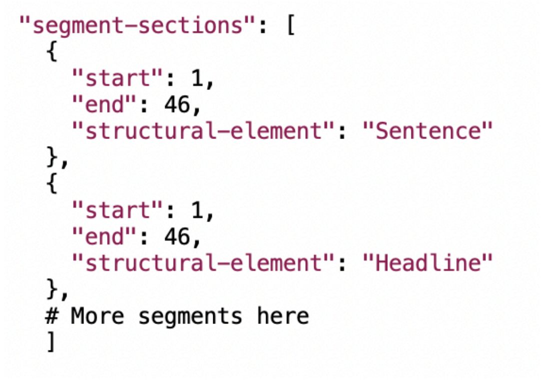
The BP JSON encoding of Basic events follows the same overall layout as for Abstract events. As before, a corpus object consists of metadata and a table of entries, with each entry containing a dictionary of annotation sets. See Figure  below (note that we've arranged the fields of the object for expository simplicity – the JSON pretty-printer actually puts them alphabetical order).



```
{
  "corpus-id": "Basic English, V1.8.1, Hidden",
  "entries": {
    "198-cnn... (the file name for this document)": {
      "segment-text": "The actual text of the document is here ...",
      "segment-sections": [ ... a list of sentence segments ... ],
      "segment-type": "document",
      "annotation-sets": {
        ## The annotation-sets slot could also include Abstract and Granular event fields
        "basic-events": {
          "events": { ... },
          "includes-relations": { ...},
          "span-sets": { ... }
        }
      },
      # The following two fields are replicative, but required
      "doc-id": "198-cnn... (as above)",
      "entry-id": "198-cnn... (as above)",
      # Optional fields go here (provenance, segment-start, segment-end, etc)
      # Some of these are documented above (Abstract)
    },
    # Other documents go here ...
  },
  # A few more corpus-related fields go here
}
```

**Figure 10: Top-level structure of a corpus containing Basic events**

The corpus and entry object structures accept the same fields as were documented above for Abstract (including Phase 3's optional `lang` field), but entries add a `segment-sections` field to encode the structural subdivisions that are pertinent to the Basic task. In particular, where the Abstract corpus is constructed from pseudo-documents, all one sentence long, a Basic data set is made up of conventional documents with multiple sentences per document. The sentence elements are available as entries in the segment-sections list, as in Figure , below.



```
"segment-sections": [
  {
    "start": 1,
    "end": 46,
    "structural-element": "Sentence"
  },
  {
    "start": 1,
    "end": 46,
    "structural-element": "Headline"
  },
  # More segments here
  ]
```

**Figure 11: Segment elements of a Basic document entry**

Each segment entry is itself a JSON dictionary structure with `start` and `end` fields that index the segment into the text of the document, itself kept under the `segment-text` field of the document entry. Additionally, segment entries have a `structural-element` field, which indicates whether the segment corresponds to a sentence or to some structural component of the document. In the example, the span from character positions 1 through 46 corresponds both to a sentence and a headline. For more discussion of sentences and related structural elements, see the discussion in Section 2.2.8, above.

The Basic events themselves are kept in a table indexed under the `basic` keyword in the annotation set dictionary, with the notional path being

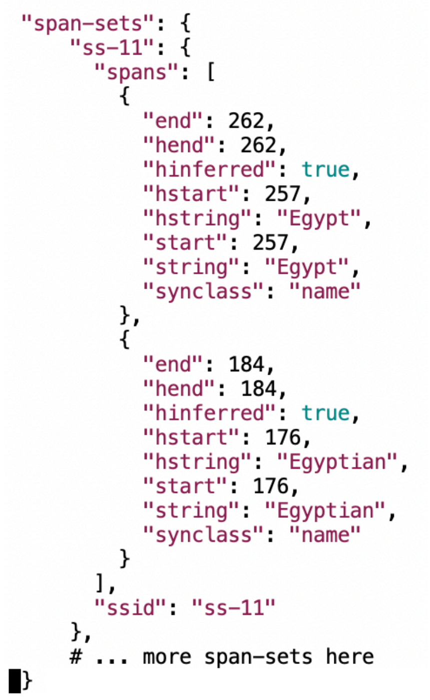
> *corpus_obj*`.entries.`*entry_id*`.annotation-sets.basic`

where *corpus-obj* is the top-level BP JSON corpus object, and *entry-id* ranges over the document entries in the corpus.

As is also the case for Abstract, the table of Basic events is itself a JSON dictionary with several elements:

- `events` is a table of Basic events (actually a JSON dictionary indexed by event ID)
- `span-sets` is a table of span sets, just as with Abstract (and also, more properly, a JSON dictionary indexed by span set ID)
- `includes-relations` is a JSON dictionary that encodes any part-whole relations holding between arguments or events in the text segment

The `span-sets` entries are exactly as we saw with Abstract: they consist of mention alternatives for the arguments or anchors of events.



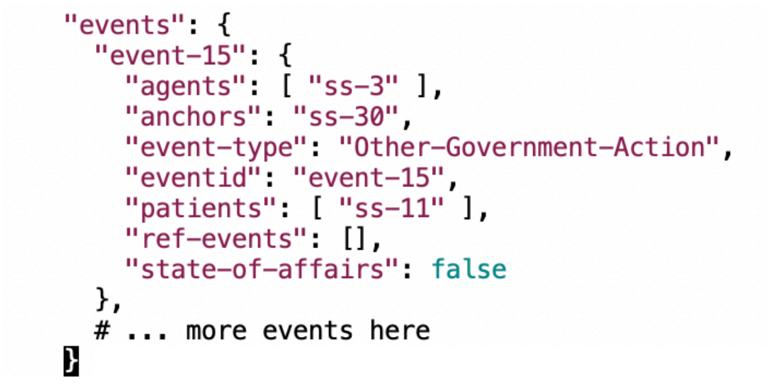
```
"span-sets": {
    "ss-11": {
      "spans": [
        {
          "end": 262,
          "hend": 262,
          "hinferred": true,
          "hstart": 257,
          "hstring": "Egypt",
          "start": 257,
          "string": "Egypt",
          "synclass": "name"
        },
        {
          "end": 184,
          "hend": 184,
          "hinferred": true,
          "hstart": 176,
          "hstring": "Egyptian",
          "start": 176,
          "string": "Egyptian",
          "synclass": "name"
        }
      ],
      "ssid": "ss-11"
    },
    # ... more span-sets here
}
```

**Figure 12. A span set with two mentions.**

In particular, the span-sets field is a JSON dictionary indexed by span-set ID: each ID-indexed entry is itself a JSON structure with exactly two fields: `spans`, whose value is a list of span elements, and `ssid`, whose value reproduces the indexing span-set ID. For example, the span set indexed by `ss-11` in Figure  has two mention spans, one for the name “Egypt,” and another for its adjectival form “Egyptian”. Each mention span in a span set includes the following fields:

- start, end: the start and end character positions of the mention in the text signal
- string: the string in the signal spanning the start and end character positions
- hstart, hend: the start and end positions of the head of the mention
- hstring: the head string spanned by the head start and end positions
- hinferred: a flag indicating whether the head was determined by a default rule (e.g., the rightmost token in the case of English)
- `synclass`: the syntactic form of the mention, either `name`, `nominal`, or `pronoun` (for argument spans), or `event-anchor` (for event mentions)

As with Abstract, the event entries for Basic refer to span set elements to identify their arguments. As a Basic event may have more than one agent, patient, or referred event, the corresponding slots in the BP JSON encoding are list-valued, consisting of lists of zero or more span sets. A simple example follows in Figure , where the patient of an Other-Government-Action event is the same `ss-11` span set we saw above, whose acceptable mentions are both “Egypt” and “Egyptian”. The example only shows single span-set fillers for the agents and patients (an no referred events), but in general there can be more than one agent span set for the agents (“*Merkel* and *Macron* met”), the patients (“they occupied *the library* and *student union*”), or the referred events (“he was accused of *corruption* and *money laundering*”).



```
"events": {
  "event-15": {
    "agents": [ "ss-3" ],
    "anchors": "ss-30",
    "event-type": "Other-Government-Action",
    "eventid": "event-15",
    "patients": [ "ss-11" ],
    "ref-events": [],
    "state-of-affairs": false
  },
  # ... more events here
}
```

**Figure 13: A simple BP JSON example of a Basic event**

As was the case with the `span-sets` field, the `events` table of a Basic annotation set is more properly a JSON dictionary indexed by event ID. Each ID-indexed entry in the table is the BP JSON coding of that event; Figure  shows this for `event-15`, one event out of many. The full logical pseudo-path from corpus root to an individual event is thus of form:

*corpus_obj*.`entries`.*entry_id*.`annotation-sets.basic.events`.*event_id*

where once again *corpus-obj* is the top-level BP JSON corpus object, *entry-id* ranges over the document entries in the corpus, and *event_id* is the ID for the individual event.

The fields of a BP JSON event element are as follows, listed here in logical order, as opposed to the alphabetical order into which they are usually pretty-printed by the BP JSON renderer.

- event-type: the Basic event type of the event; see **Table 3**
- anchors: a span set ID of anchor alternatives for the event
- agents: a list of span set IDs, with each span set presenting alternative mentions for a single agent argument
- patients: a list of span set IDs for the patient arguments of the event
- ref-events: a list of span set IDs for the event's referred event arguments
- money: new in Phase 3, a field capturing money values for transaction events (see below)
- state-of-affairs: a rarely-used Boolean field with value true for Basic events that actually encode states of affairs (which in turn are sometimes needed as fillers of the ref-events slot)
- eventid: the event's indexing ID, which replicates the index in the enclosing JSON dictionary

The final element of the basic annotation-set element is the includes-relations table. At the Basic event level, this is almost only required for capturing set inclusion among mentions, for example as in the two mentions italicized in:

"*19* are injured, including *eight police officers*"

For Basic events, the implications are primarily related to scoring arguments. In this case, the Violence-Wound event anchored on "injured" has the mention "19" as its patient (span set ss-6), but partial credit would be awarded for providing "eight police officers" as the patient (span set `ss-7`). This is captured by the includes relation between `ss-6` and `ss-7` in Figure  below.

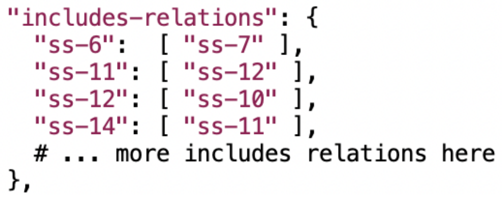
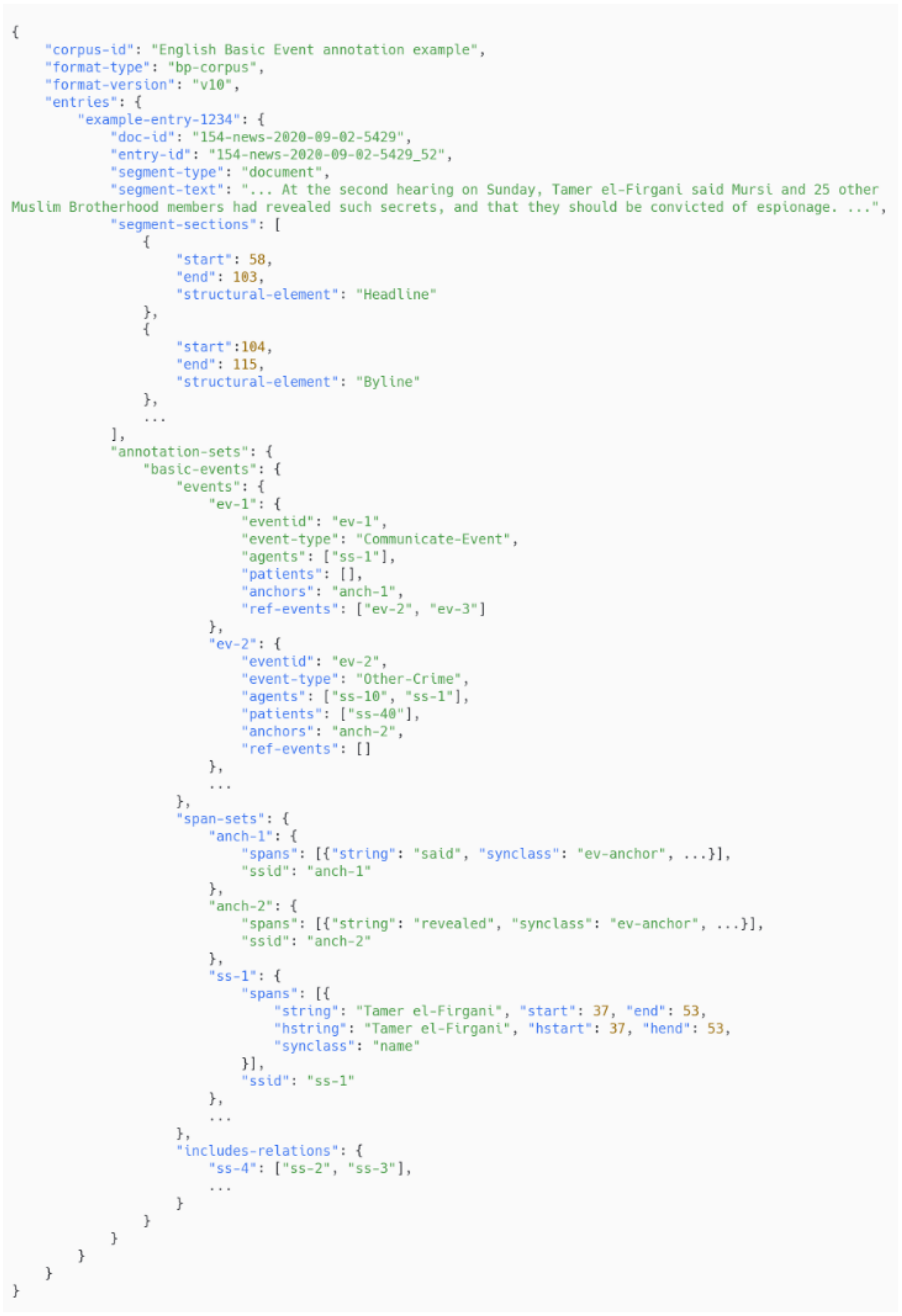

```
"includes-relations": {
  "ss-6":  [ "ss-7" ],
  "ss-11": [ "ss-12" ],
  "ss-12": [ "ss-10" ],
  "ss-14": [ "ss-11" ],
  # ... more includes relations here
},
```

**Figure 14: span-set inclusions for a Basic document**

As Figure  demonstrates, the inclusion relations are indexed by the larger component of the relation, with the subset accessed as the value of the indexing element.

In addition to set-subset inclusion among arguments, the `includes-relations` construct also captures geographical inclusions, for example between `ss-11` (Egypt) and `ss-12` (Cairo), and then between `ss-12` and `ss-10` (Tahrir Square). This is primarily of interest for the `location` field of Granular templates.

Putting all these pieces together, Figure  below shows a complete example of a Basic event corpus.



```
{
    "corpus-id": "English Basic Event annotation example",
    "format-type": "bp-corpus",
    "format-version": "v10",
    "entries": {
        "example-entry-1234": {
            "doc-id": "154-news-2020-09-02-5429",
            "entry-id": "154-news-2020-09-02-5429_52",
            "segment-type": "document",
            "segment-text": "... At the second hearing on Sunday, Tamer el-Firgani said Mursi and 25 other
Muslim Brotherhood members had revealed such secrets, and that they should be convicted of espionage. ...",
            "segment-sections": [
                {
                    "start": 58,
                    "end": 103,
                    "structural-element": "Headline"
                },
                {
                    "start":104,
                    "end": 115,
                    "structural-element": "Byline"
                },
                ...
            ],
            "annotation-sets": {
                "basic-events": {
                    "events": {
                        "ev-1": {
                            "eventid": "ev-1",
                            "event-type": "Communicate-Event",
                            "agents": ["ss-1"],
                            "patients": [],
                            "anchors": "anch-1",
                            "ref-events": ["ev-2", "ev-3"]
                        },
                        "ev-2": {
                            "eventid": "ev-2",
                            "event-type": "Other-Crime",
                            "agents": ["ss-10", "ss-1"],
                            "patients": ["ss-40"],
                            "anchors": "anch-2",
                            "ref-events": []
                        },
                        ...
                    },
                    "span-sets": {
                        "anch-1": {
                            "spans": [{"string": "said", "synclass": "ev-anchor", ...}],
                            "ssid": "anch-1"
                        },
                        "anch-2": {
                            "spans": [{"string": "revealed", "synclass": "ev-anchor", ...}],
                            "ssid": "anch-2"
                        },
                        "ss-1": {
                            "spans": [{
                                "string": "Tamer el-Firgani", "start": 37, "end": 53,
                                "hstring": "Tamer el-Firgani", "hstart": 37, "hend": 53,
                                "synclass": "name"
                            }],
                            "ssid": "ss-1"
                        },
                        ...
                    },
                    "includes-relations": {
                        "ss-4": ["ss-2", "ss-3"],
                        ...
                    }
                }
            }
        }
    }
}
```

**Figure 15: A complete example of a Basic event corpus**

### 6.2.1 Handling financial transactions (new to Phase 3)

The BETTER program adopted a minimalist argument structure for the Basic task. As noted earlier (prologue of Section 2), in order to simplify cross-lingual projection, Basic restricts the argument structure of events to two traditional fields, *agent* and *patient*, along with a subordinate *referred event*. As previously noted, these fields are more semantic than syntactic, all in the hope of making Basic events more language-agnostic and more supportive of cross-lingual projection.

This minimalist formulation is in contrast to such current approaches as FrameNet, which proliferate arguments to cover aspects of an event beyond the agent and patient. One place where the Basic representation most shows its shortcoming relative to FrameNet is with events that are realized in English as ditransitive verbs. Chief among these are financial transactions. With transactions, the seller (typically agent) and buyer (typically seller) are readily captured, but the item being sold and its value have no corresponding arguments.

The Phase 1 and Phase 2 data sets swept this issue under the rug by largely avoiding documents that were rich in transactions. With Phase 3, this is no longer possible, as the analytic topics have a significant interest in transactions, monetary values in particular. To address this need, the T&E team decided not to revamp the whole Basic representation to make it more like FrameNet, but rather to special-case the monetary value associated with some Basic events. This is handled in Phase 3 by providing an additional `money` argument.

In BP JSON, the `money` field is handled in much the same way as are `agent` and `patient`. As with these other arguments, the money field takes as value a list of string sets corresponding to the monetary value (or values) associated with an event. There is typically only one such value, the chief exception being cases where an explicit currency conversion is provided, such as "*10,000 euros ($13,200*)".

Figure , for example, shows the BP JSON representation for the sentence "*From 2011 to 2018, China granted Sudan an estimated $143 million in loans*," where granting a loan is captured as a *fund-project* event.

```
{
  "eventid": "ev-1",
  "event-type": "fund-project",
  "agents": ["ss-2"],   // China
  "patients": ["ss-3"], // Sudan
  "money": ["ss-4"],    // an estimated $143 million
  "anchors": ["ss-10","ss-11"], // loans and light verb granted
}
```

**Figure 16: BP JSON for a Basic event with a money field**

## 6.3 Granular event encoding in BP JSON

The BP JSON data format used to convey the Granular information is an extension of that used for the Basic-level event annotation. Specifically, the "`events`", "`span-sets`" and "`includes-relations`" fields are identical in form and function as they were in the Basic BP JSON distribution, with the only addition being that the "`synclass`" field of the "span" elements have some additional possible values: `template-anchor, time-mention` and `duration-mention`.

All of the new information added through the annotation of Granular event templates is captured in a "`granular-templates`" field that falls within the scope of the "`basic-events`" field. This field is parallel to the `events` field under `basic-events`: where `events` enumerates Basic event annotations, `granular-templates` enumerates Granular events. The value of this field consists of a table of template data structures, each indexed by the template's unique identifier. The template data structures differ based on their type (specified in the "`template-type`" field), but generally follow the same pattern, with a field for each of slot of the Granular event, as defined in Sections 2.3.12 and following.

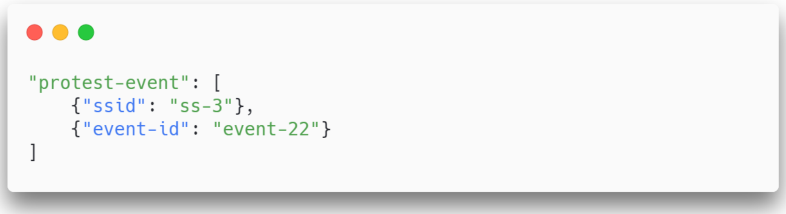
In their BP JSON encoding, the fields of a template fall into two general categories – those that take other annotations (events or span-sets) as arguments, and those that take specific categorical (string) values.

For fields that take span-sets or events as values, the span-set or event will always be specified within a table (dictionary) JSON data structure, with the span-set indicated via its identifier in the "`ssid`" field, and with the event indicated via its identifier in the "`event-id`" field. An example is shown in Figure .

The reason that the identifiers are "wrapped" within a dictionary is to handle the situation when either temporal or *irrealis* contextual information is to be captured. For example, if an event is specified in the "protest-against" field as occurring in the future, this would be captured as shown in Figure .

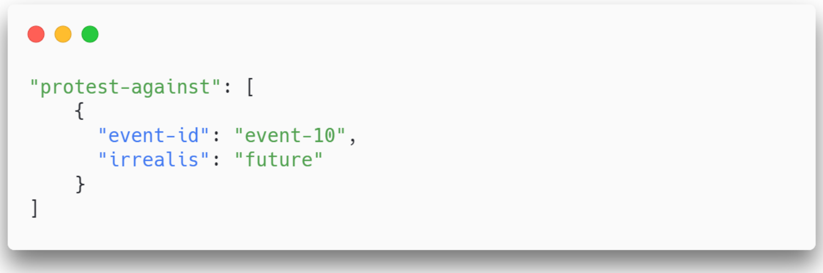
An example of a temporal attachment being applied to event is shown in Figure , where the "ss-22" span set would refer to a temporal expression.

Figure  below provides a small snippet of BP JSON that depicts how the Granular event template information is integrated into the pre-existing Basic event information.



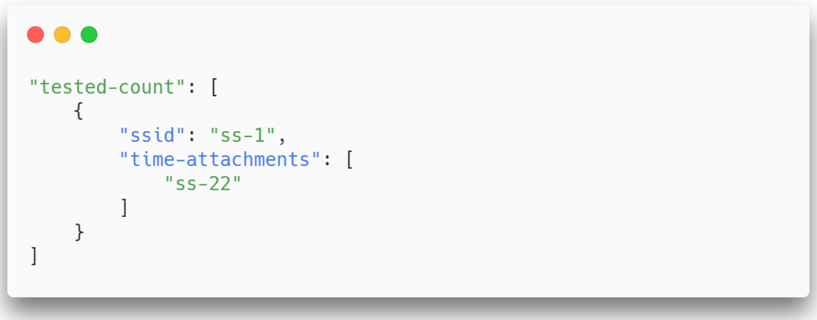
```
"protest-event": [
    {"ssid": "ss-3"},
    {"event-id": "event-22"}
]
```

**Figure 17 A sample *protest-event* field for a *Protestplate***



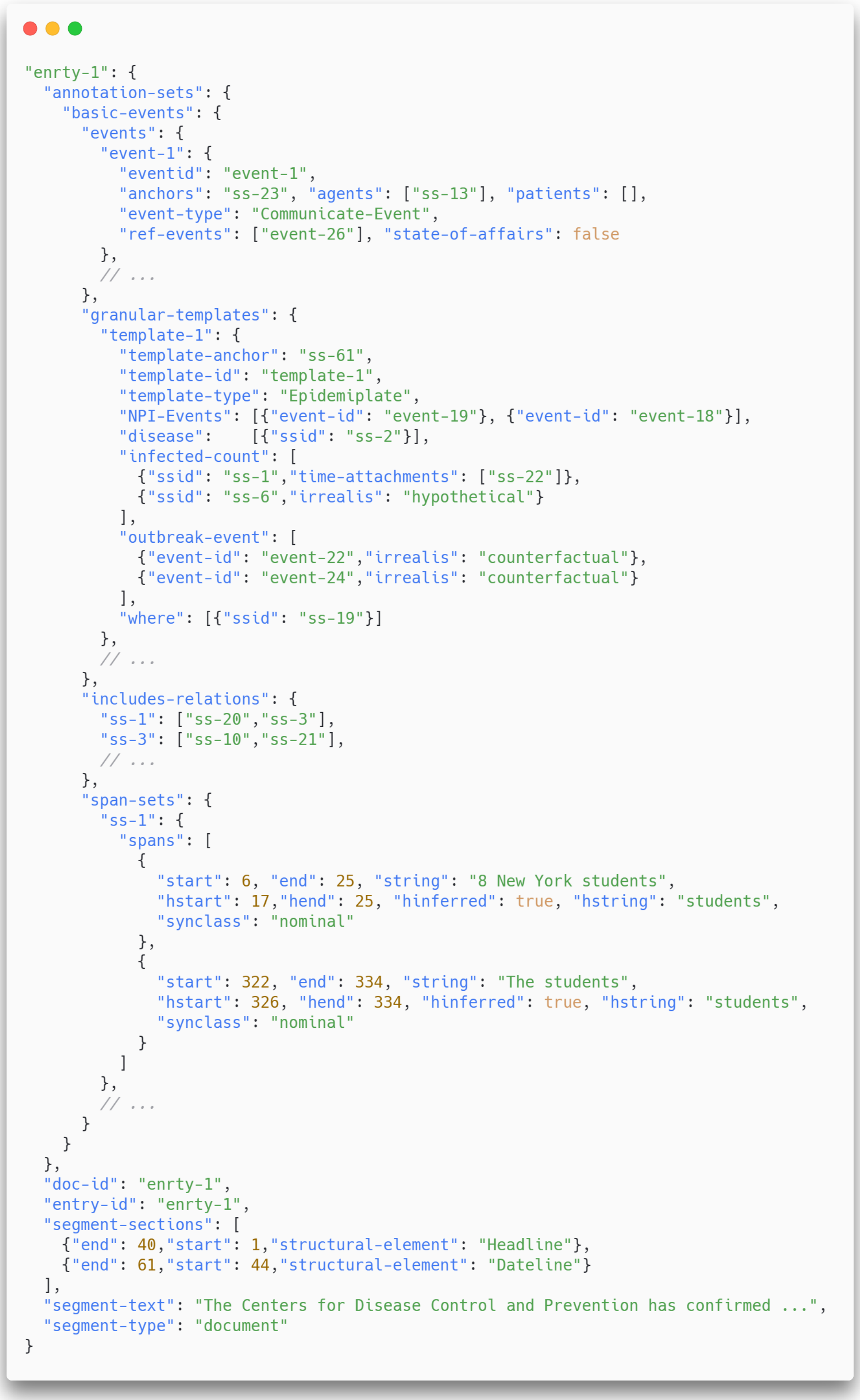
```
"protest-against": [
    {
      "event-id": "event-10",
      "irrealis": "future"
    }
]
```

**Figure 18 A *protest-against* field for a *Protestplate*, demonstrating how *irrealis* properties are captured in template fields**



```
"tested-count": [
    {
        "ssid": "ss-1",
        "time-attachments": [
            "ss-22"
        ]
    }
]
```

**Figure 30 A *tested-count* field for an *Epidemiplate*, demonstrating how *time-attachment* properties are captured in template fields**



```
"enrty-1": {
  "annotation-sets": {
    "basic-events": {
      "events": {
        "event-1": {
          "eventid": "event-1",
          "anchors": "ss-23", "agents": ["ss-13"], "patients": [],
          "event-type": "Communicate-Event",
          "ref-events": ["event-26"], "state-of-affairs": false
        },
        // ...
      },
      "granular-templates": {
        "template-1": {
          "template-anchor": "ss-61",
          "template-id": "template-1",
          "template-type": "Epidemiplate",
          "NPI-Events": [{"event-id": "event-19"}, {"event-id": "event-18"}],
          "disease":    [{"ssid": "ss-2"}],
          "infected-count": [
            {"ssid": "ss-1","time-attachments": ["ss-22"]},
            {"ssid": "ss-6","irrealis": "hypothetical"}
          ],
          "outbreak-event": [
            {"event-id": "event-22","irrealis": "counterfactual"},
            {"event-id": "event-24","irrealis": "counterfactual"}
          ],
          "where": [{"ssid": "ss-19"}]
        },
        // ...
      },
      "includes-relations": {
        "ss-1": ["ss-20","ss-3"],
        "ss-3": ["ss-10","ss-21"],
        // ...
      },
      "span-sets": {
        "ss-1": {
          "spans": [
            {
              "start": 6, "end": 25, "string": "8 New York students",
              "hstart": 17,"hend": 25, "hinferred": true, "hstring": "students",
              "synclass": "nominal"
            },
            {
              "start": 322, "end": 334, "string": "The students",
              "hstart": 326, "hend": 334, "hinferred": true, "hstring": "students",
              "synclass": "nominal"
            }
          ]
        },
        // ...
      }
    }
  },
  "doc-id": "enrty-1",
  "entry-id": "enrty-1",
  "segment-sections": [
    {"end": 40,"start": 1,"structural-element": "Headline"},
    {"end": 61,"start": 44,"structural-element": "Dateline"}
  ],
  "segment-text": "The Centers for Disease Control and Prevention has confirmed ...",
  "segment-type": "document"
}
```

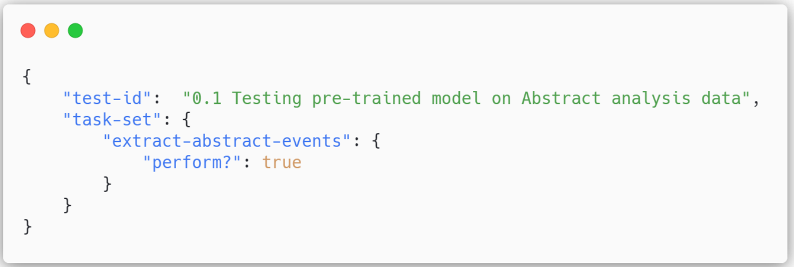
**Figure 19 An example snippet of Granular data in BP JSON format**

# 7 Information Extraction Evaluation “Dry Runs”

Evaluations will begin with an initial “dry run” of the evaluation procedures. The goal of this exercise is for all parties in the BETTER program to be confident that the Docker images that have been constructed and tested in their original development environment can carry out the required behaviors in the T&E environment. The goal is for each performer team to upload one or more systems that will be executed within two runs. One run is designed to test identical execution on a known test data set, while the second run will ensure that each of the systems can perform machine learning on new data and will also give an early indication of how the “re-trained” models perform on “hidden” held out data – in this case, English data rather than the eventual test data. Performer teams will be able to see their score results and ensure that they are identical to what they find when running in their original development environment. In both cases all parties will also be able to test that their Docker images are able to run and properly process the input files provided within the test and evaluation computing environment.

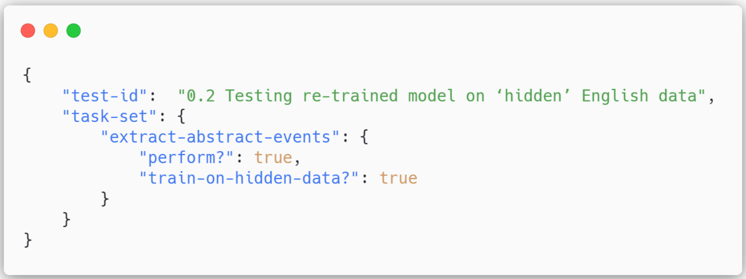
The outline of the Dry Run is as follows:

- Performer teams will upload their Docker images to the Submission Server.
- The T&E team will put back together the zip-split parts of the submitted Docker images and each system will be executed twice, using one of these two task configurations:
  - Configuration 1 will involve no model training and will execute the system’s pre-trained model on the analysis data set.



```
{
    "test-id":  "0.1 Testing pre-trained model on Abstract analysis data",
    "task-set": {
        "extract-abstract-events": {
            "perform?": true
        }
    }
}
```

  - Configuration 2 will involve model training followed by the application of the trained model on a held-out portion of the “hidden” training set.



```
{
    "test-id":  "0.2 Testing re-trained model on ‘hidden’ English data",
    "task-set": {
        "extract-abstract-events": {
            "perform?": true,
            "train-on-hidden-data?": true
        }
    }
}
```

- The results file from executing each system, in BP JSON format, will be scored against the reference annotations.
- For the Dry Runs we will also allow systems to generate some system diagnostic information that is written to the "standard output" stream. The T&E team will capture and manually convey these outputs back to the appropriate performer team. This diagnostic information should not indicate any specific information about the "hidden" data that the system encountered during training or testing, since the T&E staff will be reviewing and manually sending to the performer teams.

The submission server is running in the Amazon Web Services (AWS) computing cloud, and so there are real costs associated with executing performer systems on the GPU-enabled large systems made available for this evaluation. It is our expectation that the execution of performer systems on the first run, which will involve no additional training, will not require particularly intensive computing resources. However, it is difficult for us to predict how much time (or other compute resources) will be used by performer systems when addressing the second run that involves machine learning on additional training material, especially in light of the possible use of model selection approaches, which can add considerable amounts of compute cycles. This is one of the reasons that running the Dry Runs will be helpful for us to learn what is needed. For this reason, we impose an upper bound of 24 calendar hours for each run.

Details about the Submission Server and how to use it are presented in Section 4.

## 8 Information Extraction Evaluation Policies

Teams will be scored on a maximum of 3 system submissions. Teams are allowed to upload more than 3 submissions but will mark a maximum of 3 as "ready for scoring" prior to the evaluation deadline. In the event that a submitted system does not run successfully in the T&E environment, MITRE will provide information about error messages observed but will not provide in-depth debugging support. In the event of errors, performers will have the opportunity to submit one (but only one) replacement submission per unsuccessful "ready for scoring" submission. No further development work may be done on the replacement submission. Replacement systems must be submitted within 48 hours of notification by MITRE of errors with the original submission.

In the event that an unsuccessful run is caused by a failure of the T&E environment or a problem with testing materials that affects all teams, this won't be counted against teams' replacement submission limit. The categorization of any failure will be determined by IARPA.

## 9 Information Retrieval Evaluation "Dry Runs"

There will also be dry runs for the IR Evaluation, with the same goal that program performers can be confident that their evaluation Docker images will run in the T&E environment. Each performer team may upload one system that will be executed in the automatic, auto-HITL, HITL, and IE invocations.

The Phase 3 IR dry run uses the following resources:

1. The Phase 3 English Training Corpus
2. A multilingual 5,000-document set, taken from the same population as the evaluation documents.
3. Two analytic tasks from the Phase 2 IR evaluation.

The outline of the Dry Run is as follows:

- Performer teams will upload their Docker images to the Submission Server.
- The T&E team will put back together the zip-split parts of the submitted Docker images and each system will be executed four times:
    - Configuration 1 is an automatic run. Systems should retrieve 10 documents for each request in each task, and return only the document ranking, not extraction output. If your docker supports running in the IR-without-IE mode, for this configuration it can use IE. Keep in mind that there is no scoring in the dry run;

your system can output any 10 documents per request that it likes.

```
{
  "test-id": "IR automatic evaluation run",
  "task-set": {
    "find-relevant-docs.automatic": {
      "perform?": true,
      "corpus-location": "/corpus/path/here",
      "scratch-storage": "/scratch/path/here",
      "ie-allowed": true
    }
  }
}
```

- Configuration 2 is an auto-HITL run, with the same outputs as for automatic (10 documents per request, ranking only). If your docker supports running in the IR-without-IE mode, for this configuration it should **not** use IE.

```
{
  "test-id": "IR HITL evaluation run",
  "task-set": {
    "find-relevant-docs.auto-hitl": {
      "perform?": true,
      "corpus-location": "/corpus/path/here",
      "scratch-storage": "/scratch/path/here",
      "ie-allowed": false
    }
  }
}
```

- Configuration 3 is a HITL run (again, outputting 10 documents per request, ranking only). If your docker supports running in the IR-without-IE mode, for this configuration it can use IE.

```
{
  "test-id": "IR HITL evaluation run",
  "task-set": {
    "find-relevant-docs.hitl": {
      "perform?": true,
      "corpus-location": "/corpus/path/here",
      "scratch-storage": "/scratch/path/here",
      "ie-allowed": true
    }
  }
}
```

- Configuration 4 is an information extraction run. This will run on a test-data.bp.json file containing 10 documents. Again, recall that the dry run is not scored, so your system should only execute as much of its extraction process as is needed to ensure a successful dry run.

```
{
  "test-id": "IR automatic extraction run",
  "task-set": {
    "extract-basic-events": {
      "perform?": true,
    }
  }
}
```

- For the Dry Runs we will also allow systems to generate some system diagnostic information that is written to the "standard output" stream. The T&E team will capture and manually convey these outputs back to the appropriate performer team. This diagnostic information should not indicate any specific information about the "hidden" data that the system encountered during training or testing, since the T&E staff will be reviewing and manually sending to the performer teams.

The submission server is running in the Amazon Web Services (AWS) computing cloud, and so there are real costs associated with executing performer systems on the GPU-enabled large systems made available for this evaluation. It is our expectation that the execution of performer systems should be very quick, since only one condition is performing extraction, and the document collections are quite small, and full output is not required in any step. The first run, which will involve no additional training, will not require particularly intensive computing resources. However, it is difficult for us to predict how much time (or other compute resources) will be used by performer systems. This is one of the reasons that running the Dry Runs will be helpful for us to learn what is needed. For this reason, we impose an upper bound of 24 calendar hours for each run.

Details about the Submission Server and how to use it are presented in Section 4.