Modeling Complex Event Scenarios via Simple Entity-focused Questions

Mahnaz Koupaee[1], **Greg Durrett**[2], **Nathanael Chambers**[3], **Niranjan Balasubramanian**[1]

[1] Stony Brook University, [2] The University of Texas at Austin, [3] United States Naval Academy
[1]{mkoupaee,niranjan}@cs.stonybrook.edu
[2]gdurrett@cs.utexas.edu, [3]nchamber@usna.edu

Abstract

Event scenarios are often complex and involve multiple event sequences connected through different entity participants. Exploring such complex scenarios requires an ability to branch through different sequences, something that is difficult to achieve with standard event language modeling. To address this, we propose a question-guided generation framework that models events in complex scenarios as answers to questions about participants. At any step in the generation process, the framework uses the previously generated events as context, but generates the next event as an answer to one of three questions: *what else a participant did, what else happened to a participant, or what else happened*. The participants and the questions themselves can be sampled or be provided as input from a user, allowing for controllable exploration. Our empirical evaluation shows that this question-guided generation provides better coverage of participants, diverse events within a domain, comparable perplexities for modeling event sequences, and more effective control for interactive schema generation[1].

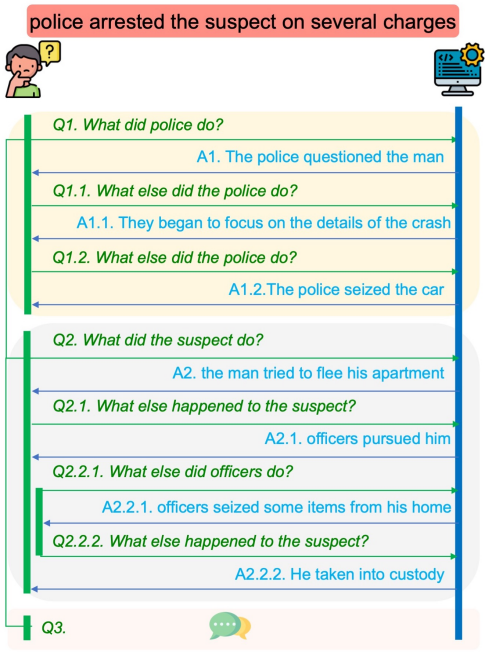

Figure 1: Question-guided event sequence generation. The user can interact with the system by asking questions regarding the entities and the system will generate corresponding events. Different questions can lead to different paths as shown with different colored boxes.

1 Introduction

Event scripts (Schank and Abelson, 1977), also known as event schemas, describe a sequence of events in a particular context. Representing and modeling such schemas is central to applications in AI such as question answering, discourse understanding, and information extraction (Balasubramanian et al., 2013). Early work used hand-crafted event schemas as a starting point (Schank and Abelson, 1977; Mooney and DeJong, 1985), but modern techniques attempt to extract these at a large scale from unlabeled data (Chambers and Jurafsky, 2008; Chambers, 2013; Pichotta and Mooney, 2016; Weber et al., 2018b).

Event language models can also be used to approximate schematic knowledge via event sequences. They can be trained to generate a sequence of events with their participating roles describing a real-life scenario. However, these scenarios can often be complex and don't always fit as simple sequences (Weber et al., 2018b). For example, suppose we have the following event: `police arrested suspect on several charges`, with **`police`**, **`suspect`** and **`charges`** as entities. The scenario can be described in multiple ways depending on which entities we want to focus on and what roles they play in the subsequent events. We may be interested in knowing what the **`police`** did, what the **`suspect`** did or what happened to the **`charges`**, each of which can be explored as its own sequence,

[1]The code is available at https://github.com/StonyBrookNLP/qa-event-lms

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2468–2483
May 2-6, 2023

yielding many interconnected sequences of events in the scenario. An event language model, however, will simply generate events conditioned on the previous events in the discourse, with no direct mechanism to guide generation towards areas of interest within the scenario. As a result, with standard decoding strategies we often end up with a sequence of events that might be relevant to the scenario, but not necessarily cover the broad set of diverse paths in the scenario; we would only know the fate of certain entities if the system samples events that included them. This lack of control will make it difficult to use the system, as one must keep sampling events until it just happens to produce events with roles one is interested in.

We propose a simple modification in which event language models are trained to also condition on the entities and the roles they play in a scenario through a set of simple questions, as illustrated in Figure 1. Given the same example event and its entities, one can then explore the scenario through various paths by asking the system to generate the events with their desired entities and roles. Such a question-guided model can be used in interactive settings to model and construct diverse paths that cover various aspects of complex scenarios.

A key challenge, however, is in creating the necessary training data at scale. We show that we can repurpose standard event sequences to create training data for question-guided models: we take a partial sequence of events as context and derive a role-based question involving an entity for which a future event in the sequence can be an answer. This allows for creating large scale training instances that are (Context, Question, Answer) triples, which then allows us to train models that can better respond to user control in the form of questions.

Our analysis shows that question-guided event language models can generate sequences with more diversity and comparable quality as an event language model. Our human evaluation of the model in an interactive setting, shows that the controllability of the question-guided model allows for generation of sequences that lead to better quality, broader-coverage schemas with fewer interactions. This interactive evaluation is a step toward constructing schematic/common-sense knowledge for analyzing events, an application in the intelligence analysis community.

In summary, this paper makes the following contributions; (i) It argues the need for control in event language models to explore complex scenarios; (ii) It provides a simple yet effective way for training event language models that can be guided to explore different aspects of complex scenarios; (iii) It provides empirical evidence showing improved control and utility via automatic and manual evaluations.

2 Related Work

Event Schema Induction Event scripts (or schemas) originally proposed by (Schank and Abelson, 1977), consist of a set of events and actors (also known as slots) playing different roles. The event schemas are capable of analyzing complex situations by encoding information from prototypical events and their participants.

Early works on scripts considered them as structured representations of events and their participants with the causal relationships between them (Schank and Abelson, 1977; Mooney and DeJong, 1985). However, the manual construction of scripts is too time-consuming and does not scale, so the scripts could only focus on specific domains of interest and have not been used more broadly.

Event schemas can also be induced automatically from text using statistical techniques in an unsupervised fashion (Chambers and Jurafsky, 2008, 2009; Balasubramanian et al., 2013). These models are easily interpretable but fail to capture long-distance complex relationships between events. Event language modeling is a type of schema induction via language modeling techniques (Rezaee et al., 2021). Given a sequence of events, the event language model predicts the probability of the next event (Manshadi et al., 2008). Framing schema learning as a language modeling problem with various ways to represent events, including word sequences annotated with predicate-argument structure (Pichotta and Mooney, 2016), OpenIE tuples (Rudinger et al., 2015; Weber et al., 2018a,b) or compositional embeddings (Modi, 2016), is another direction towards realizing large-scale schema libraries.

Graph schema induction methods (Li et al., 2020, 2021) model different relations between entities and their arguments to capture the multi-dimensionality of scenarios. Reasoning about complex relations between events requires going beyond the single dimension of event cooccurrence and capturing different types of semantic relations between events such as causal, counterfactual, etc. (Han et al., 2021). Our approach uses the standard

language models and provides them with the guidance to also produce different aspects of a scenario via simple control codes.

Controlled text generation Language models have shown promising results in text generation, however, it is not easy to have control over different aspects of generation (Keskar et al., 2019), an issue that has been studied in previous works (Dathathri et al., 2019; He et al., 2020; Lu et al., 2021; Mireshghallah et al., 2022). The key components of these methods differ in the types of controls provided and how they are provided and their applications. CTRL (Keskar et al., 2019) trains a very large language model by conditioning on texts with appended control codes that are used to guide the generation towards specific styles, contents, and task-specific behaviors. CTRLSum (He et al., 2020) uses the entity/length controls which are in forms of keywords that are automatically extracted from the text and trains a summarization system which is capable of generating summaries in an interactive manner. These approaches, however, require finetuning. Dathathri et al. (2019) and Mireshghallah et al. (2022) propose variants of controllable generation with no need to finetune or retrain the whole system.

Learning latent representations or codes from the input towards diverse generation is another direction that has been explored for machine translation (Shu et al., 2019), dialogue generation (Huang et al., 2018) and causal relations generation (Weir et al., 2020). Controllable generation to generate diverse events has been previously studied in Kwon et al. (2021), where the system uses automatically generated control codes to generate diverse preconditions of events.

In this work, we use controllable generation to model complex event scenarios. We introduce simple role-based questions about participants (agentive or non-agentive) as an effective means for control. Asking questions to get specific information about events is the focus of many existing approaches. While there has also been a body of work on semantic role labeling using QA pairs (Roit et al., 2020; Klein et al., 2020; Michael and Zettlemoyer, 2021; Pyatkin et al., 2021), the main distinction here lies in the fact that these approaches use QA pairs to identify the semantic roles, whereas our approach makes use of role-based questions to generate the next event with a specific entity playing a specific role. We show how to train for these control codes using automatically derived training sequences and demonstrate its utility for describing complex scenarios in an interactive setting.

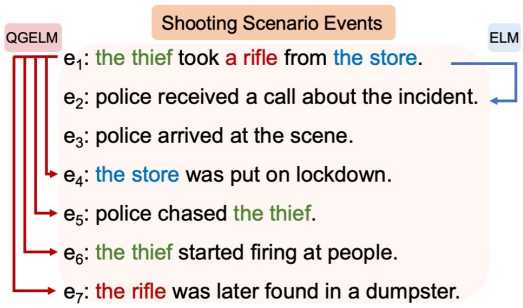

Figure 2: The shooting scenario events extracted from a news article. Typical LMs only see e_2 as the immediate next event, whereas for the question-guided LMs, any of the e_4, e_5, e_6 and e_7 involving *the store*, *the thief*, or *the rifle* can be considered a next event.

3 Question-guided Event Language Modeling

Event language models aim at predicting the probability of an event given a set of events via a conditional probability distribution. Formally, these models try to find an event $\hat{e}$ by maximizing the probability of a function parameterized by a model over a given context including a set of events:

$$\hat{\mathrm{e}} = \mathrm{argmax}_{\mathrm{e}\in\mathrm{E}} \mathrm{P}_\theta(\mathrm{e} \mid \mathrm{context})$$

where E is the set of all events. A model trained with this objective learns to generate the most probable event based on a set of cooccurring events from discourse. Suppose that shooting scenario events are extracted as shown in Figure 2. Given the first event in the sequence, a typical language model is trained to generate the next event, e_2, which does not include any entities from the prior context. What if we are interested in looking for events regarding the initial participants (*the thief*, *a rifle*, or *the store*) in this example? Given the same context, we can have multiple next events depending on the participant and the role they can have (agent or theme). If we are interested in knowing what *the thief* did, e_6 should be the next event, what happened to *the thief* is described in e_5. We can have some information about *the store* with e_4 as the next event and finally, the fate of *the rifle* is described in e_7. Controlling the model to generate the outputs based on the participants involved cannot be easily achieved with unguided (text ordered) event language models. However, we will show

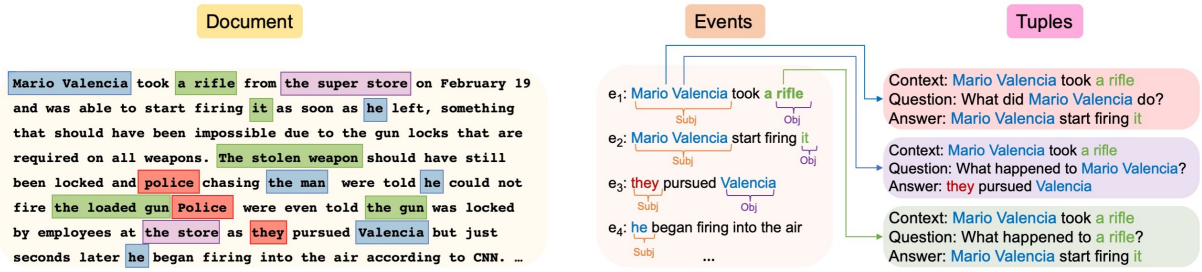

Figure 3: Data processing to create instances for a question-guided event language model. Given a document, we extract all the coreferring clusters (color-coded in the document). Next, we extract OpenIE tuples as events representations. By identifying the roles of noun phrases in the events as well as knowing which cluster they belong to, we create questions for each one of them and similarly find the events that can serve as the answers to those questions. The context can be of any length whereas the answer event is of length 1.

how the language models can be guided such that given the same context, they can directly generate events for its participants.

3.1 Problem Definition

We propose a new framing in which an event language model can be guided to generate events by not only conditioning on the events but also on specific entity-based questions of interest.

These entity-aware event language models also look to find $\hat{e}$, but by maximizing the probability of a function parameterized by a model over a given context *and* a question regarding an entity:

$$\hat{\mathrm{e}} = \mathrm{argmax}_{\mathrm{e}\in\mathrm{E}} \mathrm{P}_\theta(\mathrm{e} \mid \mathrm{context}, \mathrm{question})$$

The objective now conditions on a question as well, and the goal is to use the question to train a system to generate the most probable event for a specific entity (or a noun phrase referring to that entity) in a specific role (agent or theme). As shown in Figure 1, questions can either ask about what an entity did as an agent of an action (*what did X do?*) or what happened to an entity as a theme of an action (*what happened to X?*). By conditioning on the question as well, the system will learn to generate an event with the entity in question as well as the role specified by that question.

3.2 Question-guided Training

Event language models are typically trained using event sequences extracted from documents. Our goal, however, is to train event language models to generate events as *answers* to *questions* about entities from a given *context*. To this end, we convert event sequences in text to *(Context, Question, Answer)* tuples (CQA instances) that can be used as training data for question-guided generation.

Consider a sequence $e_1, e_2, ...e_n$ of OpenIE event tuples extracted from a document D. We create *(Context, Question, Answer)* tuples for each event e_t in the sequence as outlined in Algorithm 1 in Appendix A.1 and Figure 3. The key idea behind this process is as follows: Suppose we observe an entity in an event e_i. If this entity also appears in a subsequent event e_k, then we can see this new event as an answer to a role-based question about the entity (what did the entity do or what happened to the entity), given what we know about all the events that have been observed thus far in the sequence as context. For example, as shown in Figure 3, the entity `Mario Valencia` appears in two events e_1 and e_2. Given the context e_1, we can create the question *What did Mario Valencia do?* for which the answer is e_2 i.e., *Mario Valencia started firing it*. Formally, for each entity (any noun phrase) np that appears as an argument in the event e_i, we do the following. If np appears in an agentive role in some subsequent event e_k ($k > i$), then we associate the question *What else did np do?* with the context and use event e_k as the answer to the question. If np appears in a non-agentive role, we associate *What else happened to np?* as the question and e_k as the answer. In either case, for the next step the context is extended to include e_k and the process is repeated for all arguments in e_k. To handle events e_k that introduce new entities as arguments, we use *What else happened?* as the question.

We use automatically identified coreference clusters to locate event mentions involving a specific entity. We use simple dependency-based heuristics to determine the role of an entity in an event. Given the noun phrases of an event, we use a dependency parser to identify their roles. An entity is deemed to appear in an agentive role if it appears as a sub-

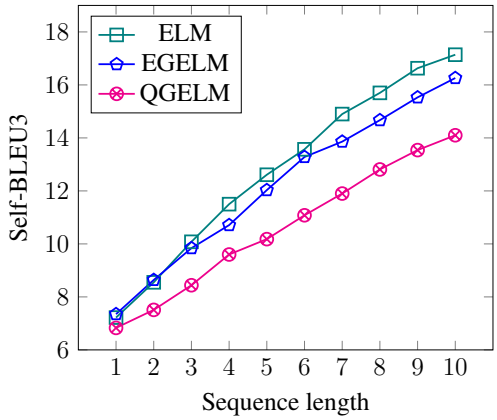

Figure 4: Diversity of generated sequences of events with varying lengths. Lower Self-BLEU scores are better as they represent more diverse sequences. QGELM is the most diverse across all sequence lengths.

Criteria	System	beam	sampling	
		Fail % ▼	Fail % ▼	Avg # samples
Any presence	ND	45.63	-	-
	ELM	79.29	26.05	17.21
	EGELM	43.39	2.73	4.47
	QGELM	**38.92**	**1.53**	**3.51**
Role specific presence	ND	69.84	-	-
	ELM	85.55	35.00	21.57
	EGELM	59.04	6.34	8.42
	QGELM	**43.03**	**2.78**	**4.54**

Table 1: Controllability Assessment: Fail % denotes the number of instances where the model fails to generate the specified entity. For sampling, we also show the average number of samples needed before an event with the specified entity is generated. ND denotes NeuroLogic decoding (Lu et al., 2021), a beam-search based controllable method with no sampling strategy.

ject and in a non-agentive role if it appears as an object. We only use these two broad categories (subject and object) to identify events as responses to specific questions. This process will automatically filter out the nonsensical questions as for such questions, no event can be found within the given sequence of events.

Note this training data has two properties that are different from standard auto-regressive training over event sequences. First, for the same conditioning context of event tuples, the model learns to generate multiple subsequent events depending on the question being asked, thus ensuring better control and diversity. Second, the model also learns to generate events that are not always adjacent to the end of the current sequence, which can be seen as a form of data augmentation shown to be effective (Koupaee et al., 2021).

4 Experimental Setup

4.1 Models

All models used in our experiments are trained using event sequences, which are OpenIE (Mausam et al., 2012) tuples extracted from articles in their discourse order.

ELM Our baseline is an **E**vent **L**anguage **M**odel trained such that given a context (a set of events), it will generate the next event. The baseline model follows the existing mechanisms used in recent prior ELM work (Manshadi et al., 2008; Rudinger et al., 2015; Pichotta and Mooney, 2016; Weber et al., 2018a,b). We train a T5 base model to learn $P(e_n \mid \text{context})$ where $\text{context} = e_1, ..., e_{n-1}$.

EGELM The **E**ntity-**G**uided **E**vent **L**anguage **M**odel generates the next event conditioned both on the context and one of its entities by learning the following probability distribution:

$$\mathrm{P}(\mathrm{e_n} \mid \text{context}, \text{entity})$$

In this setting, the system learns to maximize the probability of the next event with respect to a specific entity from the context but without considering the specific role in which the entity appears in the generated event. The training instances in this case are (Context, Entity, Answer) tuples that are obtained by replacing the question with the entity mention present in the question in the training instances described in Section 3.2.

QGELM The **Q**uestion-**G**uided **E**vent **L**anguage **M**odel generates the next event conditioned both on the context and a question regarding one of the entities in the context and a ***role*** the entity plays in the event. Here, the system learns the following probability distribution:

$$\mathrm{P}(\mathrm{e_n} \mid \text{context}, \text{question})$$

The main difference with the entity-only system is the different roles an entity can have through the question's surface form.

4.2 Data Statistics

We train all the models on the 2007 portion of the Annotated NYT corpus (Sandhaus, 2008). This subset contains a total of around 38k articles spanning over a 6 month period. Following the steps

System	Perplexity ▼	NC Accuracy ▲
ELM	24.64	46.2%
EGELM	22.45	48.6%
EGELM (margin)	25.06	46.8%
QGELM	**22.11**	**49.3%**
QGELM (margin)	26.63	46.0%

Table 2: Perplexity and the narrative cloze accuracy. Lower perplexity and higher accuracy is desirable.

Metric	ELM	QGELM	change
# accepted events ▲	6.2	8.8	42% ↑
# rejected steps ▼	5.2	3.2	38% ↓
% rejected steps ▼	41.0	26.6	35% ↓
# resamples ▼	4.9	3.2	35% ↓
total steps ▲	11.3	12.0	6% ↑
tree depth ▲	5.8	8.8	52% ↑

Table 3: Quantitative analysis of schema generation using the ELM and QGELM models. With QGELM, users accepted more of its suggested events, rejected fewer steps, used fewer resamples for a given context, and produced longer event sequences. The higher the average the better a system is for metrics with ▲ whereas lower values are desired for metrics with ▼.

described earlier on these number of articles, we end up having over $700k$ instances (consisting of (context:$e_1, e_2, ..., e_{t-1}$, question:q, answer:e_t) tuples) used to train the guided systems. For more details on the data statistics and experimental settings, please refer to the Appendix A.1.1 and A.2. The code will be released upon acceptance.

5 Evaluation

We assess the utility of the question-guided event language modeling in terms of four aspects: (i) Diversity: are they able to generate diverse sets of events that relate to a scenario? (ii) Control: do they generate events involving the specific entities in desired roles? (iii) Sequence modeling ability: how well can it predict observed events?, (iv) Interactive Utility: do users generate better sequences when using the model to collect events that fit a scenario?

5.1 QGELM Improves Diversity

We want event language models to generate diverse sequences covering different aspects of a scenario. To assess diversity in generation, we first sample multiple sequences from the models. Given a context (starting with a context of length 1), we incrementally generate events by sampling one event at a time until we generate a sequence of a predefined length. We repeat this process to generate multiple sequences. We then measure the diversity of these sequences using Self-BLEU (Zhu et al., 2018), which is the average of the BLEU scores (Papineni et al., 2002) when using one of the generated sequence as the output and the rest as references.

First, we collected test instances that had a context length of one, which amounted to 938 instances. The models were used to generate five sequences of lengths one through ten (i.e., five sequences of length one, five of length two and so on). For EGELM and QGELM we randomly choose an entity/question to generate the next event. For each model, we then compute the Self-BLEU score of its five sequences of a specific length. Figure 4 shows the average Self-BLEU over the test instances when generating sequences of different lengths. Lower Self-BLEU scores represent more diverse sequences. Self-BLEU of question-guided outputs are lower compared to that of the other two models showing improved diversity. With longer sequences Self-BLEU increases for all models as there is more potential for overlap. However, the question-guided model retains higher levels of diversity compared to the rest. Standard event language modeling tends to cover the same types of events across different samples and to some extent conditioning on entities helps improve this to a small degree. Conditioning on the questions, however, yields significant gains in diversity showing promise for improved coverage of scenarios.

5.2 QGELM Controls for Entity Roles

To quantify controllability, we introduce a metric that measures how many times a system is capable of generating an event in which a specified entity of interest is present. We generate a fixed number of outputs from each model under two decoding strategies: sampling and beam decoding. We report the percentage of times the specified entity or a coreferent mention[2] of it fails to appear in the outputs. In addition to ELM, we also use NeuroLogic decoding (ND) (Lu et al., 2021), a state-of-the-art controllable generation system as a baseline.

The top block of Table 1 (Any presence) compares the models on the number of times they failed to generate an event with the specified entity within a beam of size 40 or within 40 sampling attempts

[2]A mention is considered coreferent to the input noun phrase if the mention and original noun phrase appear in the same coreference cluster extracted from the sampled sequence of events which includes the original context event.

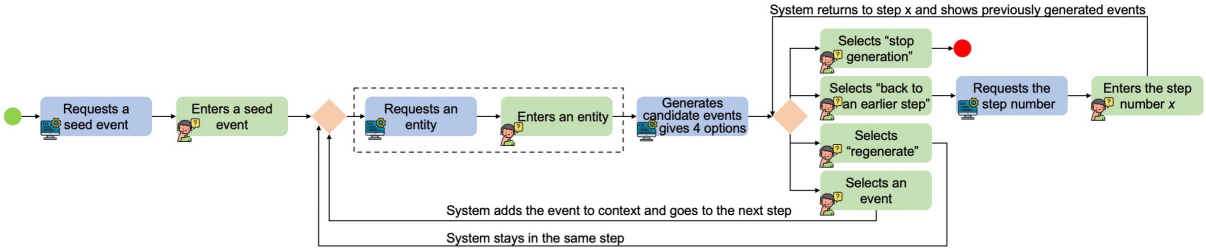

Figure 5: The overview of the interactive schema generation tool. The dashed box is only used for QGELM. The interaction starts by system asking for a seed event and the user entering an event. For QGELM, the user is asked for an entity of interest. This part is shown with a dashed box (this part is not needed for ELM). The system then samples 4 events (If QGELM, it automatically creates questions for the given entity and generates two responses per question.) These 4 events will be presented to the user where they can select an option out of 4 choices. 1. Select one of the given events. 2. Ask the system to generate a new set of events. 3. Return to an earlier step to explore a different path by entering the step number and 4. Stop the generation for the given seed.

(%Fail), and the average number of events that had to be sampled to see the entity or its coreferent mention in the output when sampling (Avg #samples). With beam decoding, ELM performs the worst since it has no control over the generation. NeuroLogic decoding does a better job at searching for events that meet the entity constraints in the model's beam. EGELM which is trained to generate events with the given entity does better even with standard beam decoding. QGELM fares even better outperforming all methods by a significant margin. With sampling, ELM works better but still fails to produce events with the input entity in more than a quarter of the cases. Both entity and question guided models respond to the input control, almost always yielding events with the input entity and much earlier in the ranked list.

The bottom block of Table 1 (Role specific presence) compares the models when we are looking for events where the input entity is expected to appear in a specific role. We use the same dependency-based heuristics we described in Section 3.2 for determining the role of an entity in a given event. With beam decoding, NeuroLogic decoding (ND) is worse than QGELM with a larger margin since it can only be constrained to generate the entity but not in ***specific*** position and role. The larger gap here shows the superiority of our question control codes to not only generate an entity but also to generate it in a specific role. ELM fares even worse, with more failures and requiring even more samples to generate the entity in the specified role. Also, as expected, EGELM has a larger gap compared to QGELM which is trained to account for the roles in which the entities appear.

Note that the coreference resolution system and the dependency parser are not perfect and therefore our heuristics for deciding both when a entity is mentioned in an event and when it appears in a specific role can be faulty. A manual inspection of the outputs for 50 entities across all systems showed that the heuristic is more than 75% accurate and the mistakes are uniform across the models.

5.3 QGELM is a good event LM

How does question-guided training affect the raw ability to generate "standard" event sequences? Predicting observed events in a discourse can be seen as a downstream evaluation. To assess this, we compare the perplexities as well as the narrative cloze task accuracy of the models using the event sequences we observe in the test set. The results are presented in Table 2. One way to turn the question guided-model into a standard event language model is to marginalize its probabilities for outputs over all possible questions we can ask at every step (marked as margin in the table): $P(e_n \mid C) = \sum_{q \in Q} P(q \mid C) P(e_n \mid q, C)$, where we assume a uniform prior distribution $P(q \mid C)$ over questions that can be asked. Similarly, for the entity guided model we can marginalize over the set of noun phrases in the most recent event in the context (this setting is similar to how we created the training instances). While this allows for fair comparison as a standard event language model, this is likely a lower bound for the model's ability for its intended use as a controllable model.

We compute the average per token perplexities for the instances of the test set (including (C,Q,A) tuples) which are shown in Table 2. Although the marginalized performance is lower because the model is forced to generate the event given sub-

	Crimes	Outbreak	Disaster	Kidnapping	Cyberattck	IED	Shooting	Conflict	Overall
ELM	10.71	16.39	23.81	8.00	11.11	13.89	21.05	**30.56**	16.24
QGELM	**28.57**	**27.87**	**33.33**	**14.00**	**16.67**	**16.67**	**23.68**	25.00	**23.22**

Table 4: Percentage of the overlap between system-generated events and manually curated schemas. Overall, the QGELM system generates more diverse set of events, therefore having a higher recall (higher percentage of overlap) compared to the ELM (for all domains except one). More details can be found in Appendix A.3.4

Domain	ELM	QGELM
mass shooting	police evacuated the surrounding buildings after the shooting. the area resuscitated as soon as officials arrived. the area in search of survivors. a police helicopter carrying three officers fired at the shooting center. the police still searching for victims. the police found the bodies of four people. the bodies of two of them found in a second car.	police evacuated the surrounding buildings after the shooting. he identified as a man. a detective questioned the suspect. the police identified the gunman. the officers heard noise from the building. they spoke with him. he moved in an apartment that was recently renovated. they worked on the scene to identify the gunman. the officers involved in an investigation. he notified about the shooting. the police found no weapon in the apartment. officers searched two buildings in the immediate area near the building. the shooting began on saturday. police officers questioned him about the shooting. The officer called victims at 6:45. the officer questioned again in the area. the shooter shot two times. the shooting began with three or four shots of the gunman. he started firing slowly his gun
kidnapping	the kidnapper ambushed the target. the kidnappers shot in the arm. they shot the men. he still not identified by the authorities. the other men also gunshot wounds. the attack left six people in critical condition and his condition.	the kidnapper ambushed the target. the kidnapping triggered by an act of rebellion. he left the scene. the target captured by a surprise attack. the kidnapping of the target essentially a symbolic step in the long struggle. mr. seymour to talk about the kidnapping. the police found a handgun. the hideout an informal area of community groups in rural part of town

Table 5: Generated examples through human interaction. A mass shooting scenario in the first example, can include high level schemas such as planning, occurrence, immediate response, investigation, etc. The QGELM covered more aspects compared to the ELM. As for the QGELM, the entities can easily be tracked based on the questions asked, and therefore there are fewer ambiguous red entities (not clear what/who they are referring to.) compared to ELM in the second example. More examples can be found in Appendix A.3.6.

optimal conditions, we see that with appropriate guidance (the question from the dataset itself), the perplexity of the true event is lower. Moreover, QGELM achieves the lowest perplexity in these settings, indicating that its control is most fine-grained and allows users the highest degree of control. We also computed the accuracy of the cloze task in which each system has to correctly predict the gold output from a fixed set of events. Following Weber et al. (2018b), for each article from the test set, we randomly select an instance (context, question, answer) and then for each instance we create 5 confounding events as answers by randomly selecting instances from other documents. The task would be then picking the correct option from the given six choices (gold+confoundings). The results show that the accuracy trends align with the perplexity trends indicating that QGELM is comparable as a event language model to the baselines we build on.

5.4 Interactive Schema Generation

Event language models can be used to generate event sequences that approximate a schematic description of how events typically happen in certain scenarios (i.e. event schemas) (Weber et al., 2018a,b; Pichotta and Mooney, 2016).

Task setup We evaluate the utility of our new question-driven models when used in an interactive system, where a user collects a set of output events from the model that they think best describes a scenario. The overview of the system is depicted in Figure 5. We manually selected 35 seeds from 8 common domains and asked 7 users (graduate students from NLP and non-NLP labs) to spend 4 minutes interacting with each system. They were asked to generate sequences of events for given seeds using ELM and QGELM systems in a randomized order. For each scenario, the user is given a seed event and is tasked with collecting a set of events that *best describe the scenario*. At each step the user is presented with a set of generated events from the model and the user selects one of the events to add it to their collected set. The added events optionally become part of the conditioning context for more events. The user has the option to either regenerate events for the same context or go back and choose a subset of context from which to generate events. For the guided model, the events are generated by conditioning on questions. Since this is a timed practice, instead of asking the users to type in a full question, we only ask them to provide the system with an entity of interest. The system then automatically forms two questions (agentive, non-agentive) with the entity and outputs a mix of the events generated with all the questions. Additional details of this study including the settings, motivations for the timed version as well as instructions for the users are

listed in Appendix A.3.

At each step the user selects the best event that meets the following criteria: (i) *Sensibleness*: whether the generated event is grammatically correct, sensible and easy to understand. (ii) *Uniqueness*: events do not duplicate each other and describe different subevents. (iii) *Relatedness*: events are related to the domain. (iv) *Typicality*: the events are quite common for things in this domain and not too niche. For each event, the users make a binary judgment on whether each criteria is satisfied. If any of the criteria is not satisfied then the event is not selected. If no event meets all these criteria then the user either regenerates from the same context or moves to an earlier context. Each user interaction with the system results in a sequence of events in the form of a tree, as users might have explored different paths by selecting different events at each step. Table 5 shows example outputs.

Analysis Table 3 shows that with the QGELM based interactive system, users accept more of the system suggested events, which means that more events meet the criteria we set for good events. They ask for fewer resampling steps, require fewer returns to earlier steps and thus having fewer reject steps or wasted steps in their interaction. They also produce longer descriptions of the scenarios with higher tree-depth i.e. the length of the longest sequence they generated within a domain.

We further analyze the output generated using the interactive system to assess their utility in creating complex schematic knowledge. The seeds we use come from 8 different domains relevant to the intelligence analysis community. For each domain we have access to manually curated schemas created by ten language experts in collaboration with the intelligence analysis experts over multiple days. An example of such schema is presented in Table 9 in Appendix A.3.5. We used these schemas as references and compared the percentage of overlap between system generated events selected by the users (within the four minute interaction) and the events in the reference schemas. The results in Table 4 show that both automatic systems can generate events that are expected to describe certain scenarios, however, the events generated by QGELM tend to have higher recall compared to the ELM (in 7 out of 8 domains).

6 Conclusion

Controlling event language models to generate events with respect to the participants is not trivial. We propose a simple yet effective question-guided approach that learns to generate events by not only conditioning on the events but also on specific entity-based questions of interest. Our empirical analysis shows that this approach can be used to generate more diverse sequences with better coverage and controllability allowing for better modeling of complex scenarios.

7 Limitations

One of the limitations of our proposed approach is the coverage of the entity roles. We have used two broad categories of roles, mainly agentive (subject) and non-agentive (object) roles, however, there can be more fine-grained semantic roles for the participating entities in the events such as agent, patient, theme, manner, etc. Considering this taxonomy of semantic roles can lead to finer-grained questions which might lead to even richer descriptions of the scenarios. Also, the human evaluation setting is limited since it is timed and users can not explore the models to their fullest extent. However, our analysis of the systems, when not timed, shows even a higher margin in terms of performance with the QGELM model.

8 Ethics Statement

The models presented in the paper make use of the existing pretrained systems that train on large collections of data and are known to inherit biases that are existent in the training data. The event language models we train are also susceptible to these biases, which can result in generation of event sequences with these biases.

Acknowledgments

We thank the anonymous reviewers for their insightful feedback and suggestions. This material is based on research that is supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes.

References

Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni, et al. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1721–1731.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

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

Rujun Han, I-Hung Hsu, Jiao Sun, Julia Baylon, Qiang Ning, Dan Roth, and Nanyun Peng. 2021. ESTER: A machine reading comprehension dataset for reasoning about event semantic relations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7543–7559, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Junxian He, Wojciech Kryściński, Bryan McCann, Nazneen Rajani, and Caiming Xiong. 2020. Ctrl-sum: Towards generic controllable text summarization. *arXiv preprint arXiv:2012.04281*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Chenyang Huang, Osmar Zaïane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 49–54, New Orleans, Louisiana. Association for Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ayal Klein, Jonathan Mamou, Valentina Pyatkin, Daniela Stepanov, Hangfeng He, Dan Roth, Luke Zettlemoyer, and Ido Dagan. 2020. QANom: Question-answer driven SRL for nominalizations. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3069–3083, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Mahnaz Koupaee, Greg Durrett, Nathanael Chambers, and Niranjan Balasubramanian. 2021. Don't let discourse confine your model: Sequence perturbations for improved event language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 599–604, Online. Association for Computational Linguistics.

Heeyoung Kwon, Nathanael Chambers, and Niranjan Balasubramanian. 2021. Toward diverse precondition generation. In *Proceedings of* SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 160–172.

Manling Li, Sha Li, Zhenhailong Wang, Lifu Huang, Kyunghyun Cho, Heng Ji, Jiawei Han, and Clare Voss. 2021. Future is not one-dimensional: Graph modeling based complex event schema induction for event prediction. *arXiv preprint arXiv:2104.06344*.

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 684–695, Online. Association for Computational Linguistics.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4288–4299, Online. Association for Computational Linguistics.

Mehdi Manshadi, Reid Swanson, and Andrew S Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *FLAIRS Conference*, pages 159–164.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CONLL)*.

Julian Michael and Luke Zettlemoyer. 2021. Inducing semantic roles without syntax. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4427–4442, Online. Association for Computational Linguistics.

Fatemehsadat Mireshghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generationusing energy language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 401–415, Dublin, Ireland. Association for Computational Linguistics.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 75–83.

Raymond J Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*, pages 681–687.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Karl Pichotta and Raymond J Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Valentina Pyatkin, Paul Roit, Julian Michael, Yoav Goldberg, Reut Tsarfaty, and Ido Dagan. 2021. Asking it all: Generating contextualized questions for any semantic role. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1429–1441, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mehdi Rezaee, Francis Ferraro, et al. 2021. Event representation with sequential, semi-supervised discrete variables. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Paul Roit, Ayal Klein, Daniela Stepanov, Jonathan Mamou, Julian Michael, Gabriel Stanovsky, Luke Zettlemoyer, and Ido Dagan. 2020. Controlled crowdsourcing for high-quality QA-SRL annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7008–7013, Online. Association for Computational Linguistics.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1681–1686.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Roger C Schank and Robert P Abelson. 1977. Scripts, plans, goals, and understanding: an inquiry into human knowledge structures.

Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni, et al. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534.

Raphael Shu, Hideki Nakayama, and Kyunghyun Cho. 2019. Generating diverse translations with sentence codes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1823–1827, Florence, Italy. Association for Computational Linguistics.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018a. Event representations with tensor-based compositions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018b. Hierarchical quantized representations for script generation. *arXiv preprint arXiv:1808.09542*.

Nathaniel Weir, João Sedoc, and Benjamin Van Durme. 2020. COD3S: Diverse generation with discrete semantic signatures. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5199–5211, Online. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. *SIGIR*.

A Appendix

A.1 Data Processing Details

QGELM uses the data in the form of (Context, Question, Answer) tuples. The details of the data processing is outlined in Algorithm 1.

Algorithm 1 Data Processing

Input: D, Document, **Output:** T, List of (Context, Question, Answer) tuples
extract OpenIE event tuples E from D
find co-referring clusters $Clusters$ from D
for e_i in E:
  find all nps N_{e_i}
  for each np in N_{e_i}:
    generate Q_{np}
    for each q_j in Q_{np} for np with role r_j
      find an event e_k $(k > i)$ with np with role r_j
      add $(e_1..e_i, q_j, e_k)$ to T

We use AllenNLP (Gardner et al., 2018) for coreference resolution to find the clusters in a document and the spaCy (Honnibal et al., 2020) dependency parser to identify all the noun phrases and their roles within an event.

A.1.1 Data Statistics

We initially extract Open IE event tuples from $37,924$ articles from the 2007 portion of the NYT Annotated Corpus using Ollie (Schmitz et al., 2012). All the extracted events from a single document are concatenated to form a single event sequence for that document. Therefore, we end up having $37,924$ event sequences with average length of 27 events.

Then, we run the data processing algorithm on the extracted sequences to generate (Context, Question, Answer) tuples. Table 6 shows the data statistics of the dataset.

split	Total	Q1	Q2	Q3
All data	762,004	466,757	188,300	106,947
Train	752,004	460,573	185,582	105,579
Dev	5,000	3,091	1,212	697
Test	5,000	3,093	1,236	671
Common Test	18953	11,712	4,412	2,829

Table 6: Data Statistics of the (Context, Question, Answer) tuples for different types of questions. Q1 refers to *what else happened?*, Q2 is *what else did np do?* and Q3 is *what else happened to np?*

A.2 Experimental Settings

A.2.1 Input/Output format

For the ELM, the input will be the context and the output will be the next event. For EGELM and QGELM, the input will be the concatenation of the context and the entity/question, separated by [SEP] token. Since the input can be more than 512 tokens (in case of long contexts), we need to truncate the input. Truncating the input from its end (for EGELM and QGELM) will result in input sequences without entity/question. To avoid this, we instead truncate from the beginning of the input sequence by removing the earlier events in the context. We remove events and not tokens from the context until its length is within the model input size. The length of the output is also fixed at 50 tokens.

A.2.2 Systems Details

Our systems finetune a pre-trained T5-base model and tokenizer with the implementation from Huggingface library (Wolf et al., 2020). Adam optimizer (Kingma and Ba, 2014) is used with an initial learning rate of $6.25e-5$.

We use a batch size of 4. Each training epoch takes almost 24 hours to run. We use the dev set for early stopping. All the systems will converge after 3 epochs.

A.3 Evaluation

A.3.1 Interactive Evaluation Setup

The details of the settings of the interactive evaluation are presented in Table 7.

A.3.2 Interactive Evaluation Design Choices

Some design choices for this evaluation are constrained by practicalities that would ensure a fair comparison. We initially had used an untimed version where users could interact with the system indefinitely. However, we found that users spent different amounts of time working, making it near impossible to do fair comparisons across systems. Further, users found it difficult to retain focus over longer periods of time.

As for using an entity of interest instead of typing questions, typing questions at every stage induces a burden on the user and introduces variance because of typing speeds. But note that even though they only select an entity, the roles are used as part of the questions we generate for the entity. Half of the generated answers are with the entity in the subject role and the other half are in the object role. Users can pick whatever role they want to explore.

Parameters	Values
number of seed events	35
number of users	7
number of seeds per worker	5
allotted minutes	4
number of generated events per step	4
number of domains	8
domains: disease outbreak, cyberattack, ied, international conflict, kidnapping, disaster, mass shooting, financial crimes	

Table 7: The parameters of the human generation task.

Domain	# seeds	# gold events
Crimes	4	28
Outbreak	6	61
Disaster	5	42
Kidnapping	4	50
Cyberattck	3	54
IED	3	36
Shooting	5	38
Conflict	5	36

Table 8: The statistics of the human evaluation in terms of number of seeds and the number of gold sequence events.

A.3.3 Instructions for Users

Below, you can find the guidelines that were provided to the users prior to starting the task.

User Manual This study is aimed at evaluating the capabilities of the event language models in generating a sequence of events with their participating arguments that can be used to describe a scenario. For each scenario, a seed event will be given. Using the seed event, you can start generating the sequence incrementally by selecting the best event at each step based on the following set of criteria:

At each step select an event that is: grammatical/understandable non-redundant or unique: events do not duplicate each other and describe different subevents on-topic: events are related to the domain and not unrelated typical: the events are quite common for things in this domain and not too niche (for example, earthquake could cause a nuclear reactor to meltdown but it's not common)

You will use two different systems to do this task. For one system you only need to select the events at each step while for the other, at each step you initially type an entity of interest and then pick the best event. This task will be timed, and you keep interacting with the system for 4 minutes, exploring different paths and entities. Once the time is over, the generation stops automatically.

User Interface: Once you run the commands, the system will load the pretrained models which will take a few seconds and then it will ask you to enter the seed event (The seeds will be given to you). Then you need to follow the prompts at each step for the allotted time. At each step there will be 4 actions you can take: Choose a preferred event generated at that step. You will be shown a set of events from which you can pick the best one according to the above criteria. Regenerate events for the last step. If you feel none of the generated events satisfy the criteria, you can choose this option so that the system will generate a different set of events for this step. Please use this option if NONE of the generated events satisfy the criteria. Choose an earlier step to return to. If you get stuck in one path and cannot generate events, you can choose to go back to an earlier step and continue the generation from a different path. Once you choose to return, the current set of events will be saved. Stop generation for the given seed event. If you think the system has generated enough events to describe the scenario or if it is no longer generating good events, then you can decide to stop the generation even if the allotted time is not yet over.

Notes regarding the entities: If you are asked for an entity, you can have the following things in mind: You are given an initial set of entities that are relevant to the given scenario. You can pick entities from this set, think of other entities that might be relevant, pick entities from already generated events or just select 'none' if you cannot think of an entity. You do not need to use all the given entities and you can choose the same entity if you think that is a main entity in the scenario or if you are interested in knowing more about that entity. You can generate a sequence which is centered around one specific entity. For instance, you can have an event sequence like this for the earthquake scenario where 'earthquake' is an argument in all the generated events: earthquake struck city, earthquake magnitude measured on scale, earthquake killed people, earthquake injured people, earthquake damaged buildings, earthquake disrupted services,...

A.3.4 System-generated events VS Manually-curated schemas

We tried to show the plausibility of the system-generated events by comparing the system outputs with schemas that are curated by a group of experts for different domains. We initially used a number of seed events from these domains and provided the users with these seeds to interact with the system. Then for all seeds from a single domain, we grouped all the generated events and measured the amount of overlap with human-written schemas. The statistics of this experiment is presented in Table 8. An event is considered to have an overlap with an event from the gold set if it either shares the exact predicate or a predicate with similar meaning. To do this, we provided a user with the list of system-generated events (not knowing which system this is coming from) and asked them to find the mappings between the gold set and the generated set. We then counted the number of events that are considered as overlapping with the gold events.

A.3.5 Manually curated schemas

Real-life scenarios can be described with a sequence of events and their relations. Manually curated schemas represent the events that can unfold a scenario in a hierarchical structure. The events in this structure can be either primitive or non-primitive, depending on whether they can be further expanded into additional events. We use the term "schema" to refer to the non-primitive events. Table 9 shows an example of such schemas for the disaster domain. As can be seen, the events are represented in multiple levels. For the sake of the evaluation conducted in this work (to make the comparison of the system-generated sequences with curated schemas more compatible), we consider the flattened representations of the schemas which consists of concatenating all the primitive events into a single sequence of events.

We also did the comparison on predicate level. The reason is that the system-generated events are instantiated events with specific arguments as the models are trained on news articles whereas the curated schemas are generalized forms of events.

A.3.6 System generated event sequences

Table 10 shows examples of generated event sequences through users interaction with the interactive tool described in section 5.4.

Schema name	Level	Events
natural disaster progression	0	detection and tracking* preparations* damages* immediate responses* rescue organization* rescue* rescue outcome* economic assistance* rebuild damaged property*
detection and tracking	1	scientists detect warning signs scientists track progress scientists assess threat scientists warn public media broadcasts information
preparations	1	government announce order people buy supplies preparations outcome*
damages	1	disaster hurts person/people disaster kills person/people disaster destroys buildings disaster damages infrastructure disaster causes food/water shortage disaster causes power outage
immediate responses	1	government counts deaths government sends equipment government estimates damage government requests aid person obtains aid person sheltered
rescue organization	1	government hold session to plan rescue government command rescue teams rescue teams travel
rescue	1	government hold rescuers locate survivors rescuers assess rescuers triage survivors
rescue outcome	1	victims found dead recovery*
economic assistance	1	government request help government estimate loss government make assistance policies people apply for assistance government gifts aid government gives loans
rebuild damaged property	1	people contact insurers people repair damage people rebuild
recovery	2	rescuers save survivors rescuers transport survivors rescuers treat survivors
preparations outcome	2	evacuation* people shelter in place
evacuation	3	people hear evacuation order rescuer evacuate people people leave affected area

Table 9: Manually curated schemas. Each domain is represented through a set of high-level schemas which in turn are expanded into multiple events. Here you can see the events of the disaster schema which are represented at different levels. Events with * are non-primitive events that are expanded.

Domain	Seed	System	Generated event sequence
international conflict	country agreed to sign the accord.	ELM	he country and several members of its foreign relations committee negotiated a peace agreement as part of the agreement. the agreement called for more cooperation. the terms of the agreement announced at the start of a yearlong talks. the talks took place in 2002. the deal still subject to approval. the agreement formally announced on saturday
		QGELM	peace take a major step. the accord a sign of change. the accord not necessarily end war. the agreement calls for a gradual suspension of all military combat and police operations. the accord also requests immediate withdrawal of troops
cyberattack	the cyberattack disrupted services	ELM	the network crashed after a new attack. the virus affected internet connections. the virus infecting computers that have computer processors. two new attacks launched on thursday. the virus also infects networks that used the same basic operating system. two new attacks triggered an attack that affected 1 million internet users on tuesday. both attacks exposed another email containing information about a security company. one email message posted by an employee of a security company. security engineers investigating the e-mail messages. the other attack not affect the internet connection
		QGELM	the attack also infected some computer systems. at least 60 computers affected by the cyberattack. the attacks continued across an industrial network of 8. the attack started in friday. the attacks hit at least five businesses and four hospitals. the network failed in this attack. the attack attributed to an increase in unreported customer data from two years earlier to more widespread attacks. one problem the nature of these attacks. the attacks linked to an "antiviral attack. the attackers focused to eliminate part of microsoft's core operating system that would detect the attack, and to prevent it from becoming more sophisticated. ... ... the attacks aimed at businesses that are vulnerable to any type of attack. the attacks caused significant disruptions for internet service providers. they trying to improve the security. the attack involved a variety of devices. the hackers asked for information. the security department declined to comment about the attack.
disaster	the heavy flooding disrupted the services.	ELM	the damage, which is estimated to cost about $2.5 billion annually due to flooding and rainy weather largely a product of human error. the department of agriculture forecasts the flood. two million people killed in the flooding. a few roads halted on the west side. the power grids closed down in some areas. the flooding also damaged roads and bridges. the department of transportation and emergency management put forward a plan. most of the roads closed in new york. the department also announced at least 17 routes halted in the same area as they were closed on oct. 26
		QGELM	the flooding resulted in a series of delays. the flooding also caused problems. the federal emergency management agency and the emergency maintenance team also monitoring damage from the flood. the flooding continued as far as tuesday. the flood already brought over from the south. the new york city fire department dispatched officers and engineers to help in the flooding. the flooding worsened on the west side. the flood occurred along the west side. about 3,000 people killed by the flooding. the flooding occurred north of manhattan around 1:55 p.m.. the water caused by another flooding a series of conditions. the flooding followed an investigation of a suspected bribery. the flooding still also affected the emergency management system. officials not expecting much rain until tuesday. the firefighters remained on manhattan. firefighters and crews sent out to assist residents with their tasks. the officials not identify a cause of the flooding. the flooding came after a storm that had swept up from manhattan. the rain caused by a river of soil. firefighters began an investigation into possible obstructions. the emergency crews working in all locations. they expected about eight feet of water to be there by this morning. the flood affected at least seven other areas. the flooding caused the agency to cut off access to the subways. the agency attributed much of the flooding to human error

Table 10: Generated examples through human interaction with the system.