Conditional Generation of Temporally-ordered Event Sequences

Shih-Ting Lin[♠] **Nathanael Chambers**[♢] **Greg Durrett**[♠]
[♠] The University of Texas at Austin
[♢] United States Naval Academy
j0717lin@cs.utexas.edu, nchamber@usna.edu, gdurrett@cs.utexas.edu

Abstract

Models of narrative schema knowledge have proven useful for a range of event-related tasks, but they typically do not capture the temporal relationships between events. We propose a single model that addresses both temporal ordering, sorting given events into the order they occurred, and event infilling, predicting new events which fit into an existing temporally-ordered sequence. We use a BART-based conditional generation model that can capture both temporality and common event co-occurrence, meaning it can be flexibly applied to different tasks in this space. Our model is trained as a denoising autoencoder: we take temporally-ordered event sequences, shuffle them, delete some events, and then attempt to recover the original event sequence. This task teaches the model to make inferences given incomplete knowledge about the events in an underlying scenario. On the temporal ordering task, we show that our model is able to unscramble event sequences from existing datasets without access to explicitly labeled temporal training data, outperforming both a BERT-based pairwise model and a BERT-based pointer network. On event infilling, human evaluation shows that our model is able to generate events that fit better temporally into the input events when compared to GPT-2 story completion models.

1 Introduction

This paper proposes a single model of events to support inferences in two seemingly different tasks: (1) temporal event ordering and (2) event infilling, or inferring unseen or unmentioned events occurring as part of a larger scenario. Figure 1 shows an example illustrating these two goals. Unlike prior approaches, we aim to address both with the same model architecture, rather than having to annotate data and build ad-hoc models for each task separately; our goal is to work towards models that capture temporal event knowledge *broadly* and support a wide range of inferences. We thus need a suitably general modeling framework to capture temporal knowledge about events, which in our case will be a BART-based (Lewis et al., 2020) model we call TemporalBART. Note that classic temporal relation extraction models, which model temporal ordering *in context* for a particular document, may chiefly learn how to use local discourse cues rather than generalizable event knowledge (Chambers et al., 2014; Ning et al., 2018b).

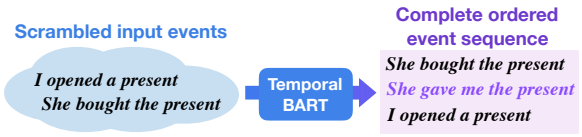

Figure 1: Diagram of our modeling setup: TemporalBART captures both temporal ordering and event cooccurrence to make various event-related inferences.

The goals in this work relate to past work on learning narrative schemas (Mooney and DeJong, 1985; Chambers, 2013; Peng and Roth, 2016; Peng et al., 2017). Our approach particularly follows a recent line of work using distributed representations of schemas (Pichotta and Mooney, 2016; Weber et al., 2018b), which support inferences about events without explicitly materializing a discrete schema library. The target tasks in this work are directly motivated by downstream applications of schema learning. Text generation tasks like story completion rely on understanding what makes narratives plausible and what events might be likely to happen before, after, and between other events (Jain et al., 2017; Yao et al., 2019), motivating our event infilling task. Answering questions about causes, effects, or what might happen next in a scenario requires knowing typical temporal orders of event sequences (Zhou et al., 2019, 2020; Ning et al., 2020), motivating our temporal ordering task.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 7142–7157
August 1–6, 2021.

Prior work has not combined traditional event cooccurrence with event temporality as we do.

We propose a conditional generation model to tackle temporal event ordering and event infilling, and train it as a denoising autoencoder over *out-of-context temporal event sequences*. As shown in Figure 1, the encoder of our TemporalBART model reads a temporally scrambled sequence of a subset of input events, obtained by corrupting a temporally-ordered sequence of events from a corpus. The decoder, which can be viewed as a conditional event language model (Kiyomaru et al., 2019; Bosselut et al., 2019; Madaan et al., 2020), then reconstructs the complete, temporally-ordered event sequence. Such denoising training has been successful exploited in many applications (Vincent et al., 2010; Lu et al., 2013; Lample et al., 2018; Lewis et al., 2020), and using seq2seq models to reorder and smooth inputs has been explored before (Goyal and Durrett, 2020), but to our knowledge we are the first to apply this in this temporal modeling setting. The conditional generation architecture of our model is flexible enough to address a variety of tasks, including our temporal ordering and event infilling tasks, by either sampling from the model or using it to score sequences. Capitalizing on the success of recent pre-trained encoder-decoder transformers (Lewis et al., 2020; Raffel et al., 2020), our model itself is based on BART, consuming and producing predicate-argument structures rendered in surface order.

Gathering large-scale high-quality labeled data with temporal annotations is often expensive and requires specially designed annotation schemes (Pustejovsky et al., 2003a; Cassidy et al., 2014; Ning et al., 2018b; Zhao et al., 2021). Here, we instead turn to a narrative documents corpus, EventsNarratives (Yao and Huang, 2018) and design an automatic method to extract the training data we need. In these documents, discourse order is loosely assumed to reflect temporal order, so events extracted from this text can directly provide training data for our models. This use of automatic annotation allows us to use broad-domain data, giving us a strong domain-independent temporal model (Zhao et al., 2021).

To evaluate how well our proposed models capture temporal knowledge and solve the two targeted tasks, we apply them on out-of domain test sets in a zero-shot manner. Specifically, for event ordering, we first extract test temporal event sequences from the CaTeRS (Mostafazadeh et al., 2016b) and MC-Taco (Zhou et al., 2019) datasets, which include the annotations on temporal relations between events. We then compare the performance of our models with two baselines: a BERT-based pairwise model and a BERT-based pointer network. For event infilling, we use the test event sequences from CaTeRS and examine the ability of our models to order unseen events and generate infilled events in comparison with GPT-2 baselines from story generation. Our BART-based models significantly outperform the baseline models on the ordering settings we consider, and human evaluation verifies that our models can generate infilled events that are better temporally-ordered with respect to the input.

2 Background and Related Work

Learning temporal knowledge to order events and generate new events as part of schemas or stories are two problems that have received significant attention, but in contrast to our work, previous work typically focuses on each in isolation.

2.1 Temporal Event Ordering

Closely related to the temporal ordering aspect of this paper is temporal relation extraction, which orders pairs of events in text *in document context* (Pustejovsky et al., 2003b; Cassidy et al., 2014; Ning et al., 2018b). This problem has been addressed as pairwise classification (Mani et al., 2006; Verhagen et al., 2007; Chambers et al., 2007; Verhagen and Pustejovsky, 2008; Cheng and Miyao, 2017; Tourille et al., 2017; Goyal and Durrett, 2019) or as a structured learning problem to enforce constraints on the output (Do et al., 2012; Ning et al., 2017, 2018a; Leeuwenberg and Moens, 2017; Han et al., 2019a,b). However, even in these latter works, the models focus on pairwise relations. In contrast, our work here views temporal event ordering as a sequence generation problem, which provides models a stronger inductive bias to capture global temporal relations between events. One recent effort (Madaan and Yang, 2020) treats this task as a graph generation problem, and so is able to predict more complex structures, but it focuses solely on ordering and is not suitable for our event infilling goals.

2.2 Schema Induction

Schema learning systems are often evaluated on their ability to predict unseen events. Initial work

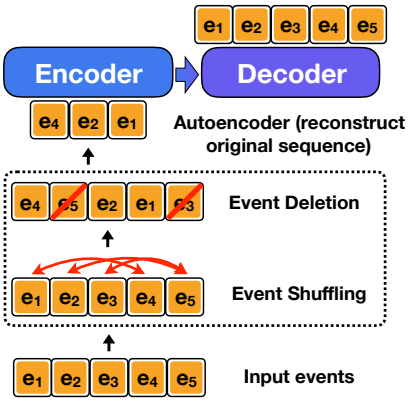

Figure 2: Our event-based denoising autoencoding training scheme used to encourage our model to learn temporal event knowledge. The input is corrupted by shuffling and deletion.

attempted to use statistical methods to derive a library of schematic information (Mooney and DeJong, 1985; Chambers and Jurafsky, 2008; Jans et al., 2012). Another thread exploits event language modeling to learn the distributions over events (Pichotta and Mooney, 2016; Peng and Roth, 2016; Weber et al., 2018b), or focuses on learning event representations (Modi, 2016; Weber et al., 2018a) rather than writing down discrete schemas. However, most of this work only models the co-occurrence between events instead of directly considering temporal information, and only represent events as a small tuple of S-V-O headwords.

Another line of work instead directly focuses on extracting coherent narratives from "story salads" (Wang et al., 2018) or more broadly generating narratives given predefined scenarios (Wang et al., 2019; Qin et al., 2020). However, without considering temporal ordering, these systems are prone to learn discourse ordering of events instead of a strong representation of temporal knowledge.

3 Method

3.1 Task Formulation and Model

Our framework involves modeling a conditional distribution $P(\mathbf{y} \mid \mathbf{x})$ over **temporal event sequences** $\mathbf{y} = \{e_1, \cdots, e_l\}$, which are sequences of events taken out of context (i.e., not represented as spans in a document) which are part of the same scenario, involve shared actors, and are temporally ordered. The input of the model is a (not necessarily temporal) sequence of events $\mathbf{x} = \{e_1, \cdots, e_m\}$ that represents incomplete information abut the scenario $\mathbf{y}$: a partial set of unordered events. Our model should learn distributions over a true underlying order of events, without obvious gaps in the event sequence, given this incomplete information. By taking events out of context rather than in the context of a document, we are encouraging the model to encode temporal knowledge between events rather than superficial cues like surface textual order or discourse connectives that might determine their order.

For the definition of events, we follow Chambers and Jurafsky (2008) where an event e is a predicate v_e along with its arguments (Palmer et al., 2005).

Our model can be formulated as a denoising autoencoder if $\mathbf{x}$ is created as a noised version of $\mathbf{y}$. Specifically, given a temporal event sequence $\mathbf{y}$ as defined above, we first corrupt it to get the required input $\mathbf{x}$ by performing two transformation functions consecutively (see Figure 2):

Event Shuffling We first perform a random shuffling of the events in $\mathbf{y}$ to produce $\mathbf{x}$. To perfectly reconstruct the original sequence $\mathbf{y}$, the model must capture the temporal relations between events.

Event Deletion We randomly delete each event in $\mathbf{y}$ with probability p to produce $\mathbf{x}$. This denoising scheme is similar to the token deletion transformation in Lewis et al. (2020). To perfectly reconstruct the original event sequence, the model needs to encode schema-like event knowledge so as to generate events not included in the input $\mathbf{x}$ and insert them at correct positions. As a result, this denoising can help the model learn event infilling.

We train our model to maximize $\log P(\mathbf{y} \mid \mathbf{x})$ on this automatically-constructed data.

3.2 Model Architecture

To leverage the power of pretrained transformers, we adopt BART (Lewis et al., 2020) as the underlying architecture for our model, and initialize our model with its pretrained weights.

The overall model, shown in Figure 3, takes a corrupted event sequence $\mathbf{x} = \{e_i\}$ as input, and outputs the true event sequence $\mathbf{y} = \{e_j\}$. To feed the event-based inputs and outputs to BART, we need to represent each event e in a textual format $\text{Repr}(e)$. We represent e with the concatenation of its predicate and all arguments. Unlike previous work which only uses the *syntactic heads* of the predicate and certain arguments (Pichotta and Mooney, 2016; Weber et al., 2018a,b), our approach preserves complex noun phrase arguments and exposes to the model arguments like temporal modifiers. We strike a balance between using

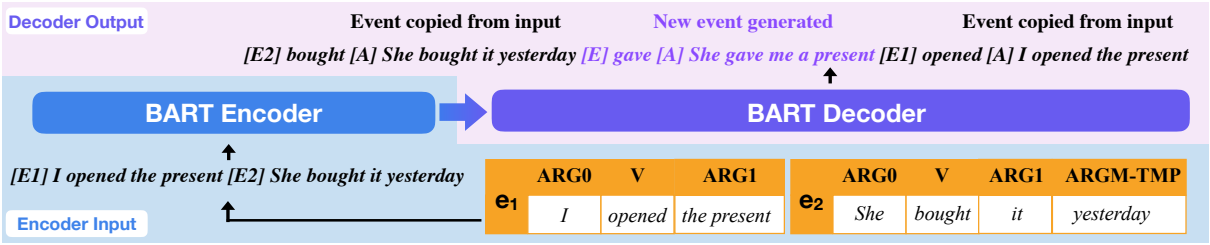

Figure 3: Model architecture of the proposed BART-based conditional generation models. TemporalBART-indexed uses indexed event tags `[Ei]` as shown in this figure. TemporalBART instead uses the single `[E]` for all events.

enough information to have meaningful event representations and not consuming entire documents (Han et al., 2019a,b), which would result in a model that overly relies on discourse clues. We then consider two variants for input and output:

TemporalBART This model first encodes each event e_i in $\mathbf{x}$ as $\text{Repr}(e_i)$, and concatenates them with a special token `[E]` prepended in front of each event. This special token can help the model identify the boundary between the input events; such placeholder tokens have been used in related tasks like entity tracking in procedural text (Gupta and Durrett, 2019). For the output, we instead prepend `[E]` v_{e_j} `[A]` in front of each $\text{Repr}(e_j)$. This setup not only provides an extra supervision signal that encourages the model to predict ordering on the basis of predicates, but also allows us to post-hoc recover an event sequence by checking the predicate part of the generation.

TemporalBART-indexed This model, depicted in Figure 3, uses the same input and output format as TemporalBART, except the prepended special token `[E]` is instead `[Ei]` before each event e_i. For the output, if e_j is one of the input events and $e_j = e_i$, then we also change the prepended tokens e_j to `[Ei]` v_{e_j} `[A]`. Otherwise, we still use `[E]` as the special event token. Note that the model is *not* able to "cheat" using the `[Ei]` tokens to do the prediction since the input events are scrambled by the shuffling denoising training scheme described in §3.1. Compared to TemporalBART, the use of `[Ei]` here provides an extra clue for the model to associate input events to output events, which can benefit the event ordering. It also provides a potential way to focus only on modeling the ordering of the target sequence, rather than also mixing in generation decisions, many of which are copying event arguments and often affect the prediction.[1]

[1]We experiment with this method, which is denoted as "TemporalBART-indexed (tags only)", in Appendix A

Training details of these BART-based models are described in the Appendix.

3.3 Training Data Collection

For our framework, the training data we need is event sequences *in temporal order*. Note that most text data occurs in *discourse order*, which is not the same thing: human annotations of temporal relation datasets like TimeBank (Pustejovsky et al., 2003b) show that many events mentioned earlier in the text occur later in time. Existing datasets of temporal relations (Cassidy et al., 2014; Vashishtha et al., 2019) are small-scale, and annotating more data is expensive and prone to low agreement (Ning et al., 2018b). To combat this issue, we instead try to automatically gather the training data we need.

Corpus We use the English-language EventsNarratives corpus (Yao and Huang, 2018), which contains more than 200,000 *narrative-structured* documents identified from three different source domains including news articles, novel books, and blogs. Yao and Huang (2018) use a weakly supervised method to identify narrative texts, describing a sequence of events in such a way that the discourse order is very likely to reflect the temporal order. This gives us an entry point to collect temporal event sequences automatically from each document. Here we focus on documents in the novel domain as our source for temporal event sequences.

Extracting Temporal Event Sequences To obtain the training event sequences, we first use an SRL model from AllenNLP (Gardner et al., 2017) to extract verbs (events) and their arguments. Then, temporal event sequences are constructed by connecting only the events in different sentences, since the relations between events within the same sentence are unclear even in narrative documents. Here, to ensure all the events in a sequence have a strong relation with each other, we only include chains of events **that are associated with a com-**

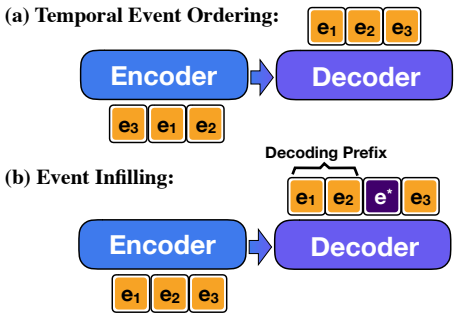

Figure 4: The two targeted tasks in this work: ordering rearranges the set of input events, whereas infilling involves hypothesizing a new event at a specified index.

mon entity (Chambers and Jurafsky, 2008), as determined by checking whether the arguments of two event have some shared non-stopword tokens. With this procedure, we are able to collect nearly 2 million temporal event sequences to train on, with nearly 70% of the sequences consisting of three or more events.

4 Target Task Formulation

Here we describe the two target tasks of our model and how they can be handled as event-based conditional generation problems. A visual of the task formulations is shown in Figure 4.

Temporal Event Ordering Given an unordered set of events $\{e_i\}$, this task's goal is to produce the temporal ordering of $\{e_i\}$, as shown in Figure 4(a). We ask the model to generate an ordered sequence of events $\{e_{f(i)}\}$ given the set $\{e_i\}$, where $f(\cdot)$ is a mapping function to determine the event to put at position i. This is a conditional generation problem that is directly solved by our proposed models.

Event Infilling The goal of event infilling is to generate inserted events at some pre-selected insertion positions in a seed event sequence (Wang et al., 2020). To simplify the evaluation, here we assume that given an event sequence $\mathbf{x} = \{e_i\}$, models will only be required to generate *one* inserted event at *one* insertion position i^*, as shown in Figure 4(b). We first feed $\{e_i\}$ as the input to our model, then generate one event e^* using $\mathbf{x}_{\text{prefix}} = \{e_i \mid i < i^*\}$ as the decoding prefix. To force our models to produce $e^* \notin \mathbf{x}$, we prevent our model from generating $\{v_{e_i}\}$ during the decoding process.

4.1 Baselines: Temporal Event Ordering

We compare against two baselines: a state-of-the-art pairwise model used for the in-context temporal ordering task and a pointer network model that directly models event sequence permutations discriminatively.

BERT-based Pairwise Model + SSVM We follow the architecture of the Deep SSVM model used in Han et al. (2019a) as our first baseline, which tackles event ordering as a pairwise classification problem. This network first exploits a BERT-based model (Devlin et al., 2019) to compute pairwise scores for e_i preceding e_j in the output $\mathbf{y}$. The final output is then obtained by solving an ILP over all the pairwise scores. The overall network is trained with the structured SVM loss so that it can learn to make joint predictions with transitivity constraint. To make this baseline more comparable to our models, we take $\text{Repr}(e_i)$ prepended with `[E]` as the event representation instead of using the sentence containing v_{e_i} as in Han et al. (2019a). Detailed formulas are in Appendix B. We denote this baseline as "Pairwise+SSVM" in the evaluations.

BERT-based Pointer Network This network first follows the BERT-based Pairwise Model + SSVM to extract the the vectorized representation $\mathbf{U}_{p_i}$ for each e_i, where $\mathbf{U}$ is the final BERT encoded matrix, and p_i is the position of the first token of e_i in the input sequence. These event representations are then instead fed into a LSTM-based pointer network to model the ordering probability by decomposing it in a sequential fashion:

$$P^{\text{seq}}(\mathbf{y} \mid \mathbf{x}) = \prod_j P(j \mid \mathbf{h}_1, \ldots, \mathbf{U}_{p_1}, \ldots) \quad (1)$$

$\mathbf{h}_t$ is the decoder hidden state in the pointer network. Compared to the above pairwise baseline, this model has a stronger inductive bias for exploiting global event relations. We train the sequential model with teacher forcing to maximize the probability of the gold ordering. We denote this baseline as "BERT-based PN" in the evaluation section.

4.2 Baselines: Event Infilling

HAQAE HAQAE (Weber et al., 2018b) is a vector quantized variational autoencoder which encodes schema knowledge with hierarchical latent variables. Since HAQAE is also an event-level seq2seq autoencoder, we can easily apply it to our setting. During training we follow Weber et al. (2018b) except that we use our narrative event sequences for training and represent each event with the predicate-argument format described in §3.2 so it is more comparable to our BART-based models.

GPT-2 GPT-2 (Radford et al., 2019) is a transformer-based pretrained language model that has been exploited in various generation tasks like story generation (Dathathri et al., 2020; Rashkin et al., 2020). However, one issue with the GPT-2 model is that it can only perform uni-directional generation. To apply GPT-2 to generate an inserted event e^*, we first concatenate $\{\text{Repr}(e_i) \mid e_i \in \mathbf{x}_{\text{prefix}}\}$ with periods in between, and treat it as the decoding prefix only. We then decode until another period is generated, and take the model's output as the text representation of e^*. Except where otherwise specified, we use the GPT2-medium pretrained model from HuggingFace's Transformer (Wolf et al., 2020), whose model size is comparable to BART-large.

Infilling GPT-2 To build a stronger GPT-2 baseline that doesn't only condition on the prefix events, we follow the baselines from Qin et al. (2020) to adapt GPT-2 to infilling tasks. Infilling GPT-2 generates the infilling events by "wrapping" the events after the insertion position to the front. That is, the decoding prefix fed to the infilling GPT-2 becomes the concatenation of $\{\text{Repr}(e_i) \mid i >= i^*\}$, <SEP> and $\{\text{Repr}(e_i) \mid i < i^*\}$, again with a period appended after each event. The special token <SEP> is used to help the model to differentiate the events before and after the insertion position.

5 Evaluation

5.1 Experimental Setup

All the models used in the evaluation are trained with the temporal event sequences automatically collected on EventsNarratives except GPT-2, since we want to compare the learned knowledge in GPT-2 with our proposed models. Although we are able to gather millions of sequences, for efficiency, we train on 100,000 sequences unless specified otherwise. For each sequence, we extract 2 distinct permutations from the corruption process. This results in 200,000 training examples in total.

During evaluation, all the models are evaluated on out-of-domain datasets in a zero-shot way, i.e., no fine-tuning is performed on the evaluation sets.

5.2 Temporal Event Ordering

5.2.1 Datasets

We use two out-of-domain English datasets to extract the test temporal event sequences: CaTeRS and MCTaco. As during training, two different permutations are produced from each extracted sequence.

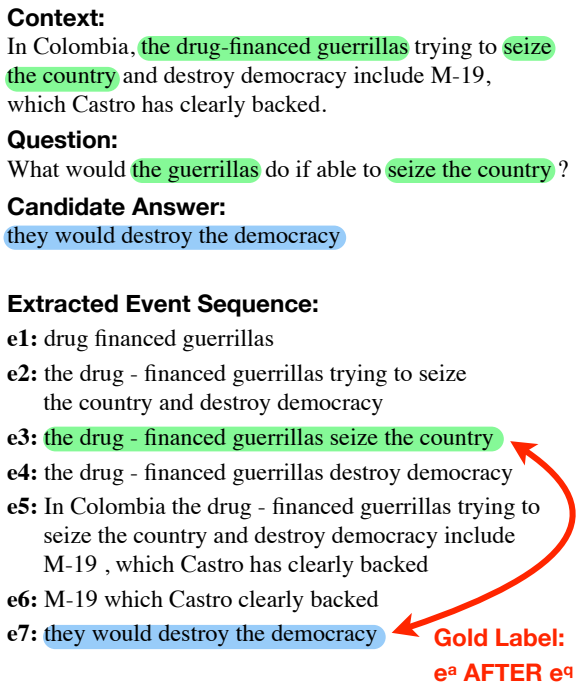

Figure 5: An example of the event sequence extracted from a context-question-answer tuple in MCTaco. e^q and e^a are highlighted with the color green and blue respectively.

CaTeRS (Mostafazadeh et al., 2016b) CaTeRS includes annotations of events and their casual and temporal relations on 320 five-sentence short stories sampled from the ROCStories corpus (Mostafazadeh et al., 2016a). To extract the evaluation data from CaTeRS, we first apply the SRL model used in §3.3 on each story. Then, a directed acyclic graph is constructed with a node being an event e whose predicate v_e can be captured by the SRL model, and an edge (e_i, e_j) indicating e_i happens temporally before e_j. Note that here we treat all types of annotated relations except "IDENTITY", "DURING" and "CAUSE_TO_END" as "BEFORE", as suggested in Mostafazadeh et al. (2016b). Test temporal event sequences are then extracted by retrieving all the path from the source nodes to sink nodes in the graph. With this procedure, we are able to gather 842 event sequences, 60% of which contain 3 or more events. With permutations, the final CaTeRS evaluation set has 1684 examples.

MCTaco (Zhou et al., 2019) MCTaco is a multiple-choice QA dataset for evaluating model understanding on 5 different types of temporal com-

Architecture	All Pairwise Acc.	Length >= 3 Pairwise Acc.
Random	50.4	50.2
Pairwise+SSVM	65.7	62.3
BERT-based PN	54.1	52.3
TemporalBART	77.1	74.7
TemporalBART-indexed	**79.7**	**78.0**

Table 1: Averaged pairwise accuracy between the gold and predicted ordering generated by each model on temporal event sequences from CaTeRS. The rightmost column is sequences with 3+ events.

Architecture	Acc.	Macro F1
Majority	90.6	47.5
Pairwise+SSVM	67.2	47.0
BERT-based PN	54.7	42.7
TemporalBART	63.9	50.1
TemporalBART-indexed	**74.9**	**55.1**

Table 2: Temporal ordering results on MCTaco sequences. Metrics are computed on the ordering between the answer event and sentence event. The test set is imbalanced, so we include macro F1. Our BART-based models outperform the baselines for macro F1.

monsense. To extract suitable test data, we focus on questions with the reasoning type of "event ordering" and their positive candidates. Each data point here consists of a sentence describing multiple events $\{e_i^c\}$, a question asking what event could happen temporally before/after a particular event $e^q \in \{e_i^c\}$, and a candidate event e^a. Critically, *the question itself* tells us whether e^a should happen before/after e^q in the temporal event sequence formed by $\{e_i^c\} \cup \{e^a\}$.

With this annotation, we evaluate our models by first feeding the randomly shuffled $\{e_i^c\} \cup \{e^a\}$ into a model, then checking the ordering between e^a and e^q in the output sequence. Here, we were able to extract 585 test sequences from MCTaco. For each sequence, $\{e_i^c\}$ and e^a are extracted with the SRL model used in §3.3. For the question, we first use a set of pre-defined regex templates to extract an event e^q and a temporal relation ("before" / "after"). We then match e^q to one of e_i^c by ROUGE-L scores. See Figure 5 for an example of the extracted data.

Compared to CaTeRS, since the sentences here are from 9 different domains in MultiRC (Khashabi et al., 2018), the types of events are more diverse. The event arguments are also more complex.

5.2.2 Results on CaTeRS

We first examine the temporal ordering results on CaTeRS, shown in Table 1. We compute the pairwise accuracy of the predicted event sequences, or how many pairs of events in the output are ordered correctly by a model. Note that the BART-based models can deviate from generating permutations of the input; however, we found that the most probable generated sequences were almost exact permutations of the input or easily aligned to the input using a heuristic.

Our BART-based models outperform the BERT-based pointer network by more than 20 points, a huge margin. One possible reason is that the decoder of BART can condition on the token-level embeddings of the events when generating the output events, whereas in the pointer network, the decoder is only aware of the condensed event embeddings $\mathbf{U}_{p_i}$. Our two BART-based models also outperform the BERT-based pairwise model on both all sequences and long sequences.

5.2.3 Results on MCTaco

Results on MCTaco are shown in Table 2. Here since we only know the gold temporal relation of one pair of events in the input, i.e e^q and e^a, the averaged accuracy on predicting the order of e^q and e^a is computed. In addition, since the ratio of before/after questions is significantly unbalanced in MCTaco, with 90% asking about the "after" relationship, we also compute the macro F1 score as our metric (averaging F1 across these two classes).

Our two baselines perform worse than just picking the majority label. This is possibly due to the high diversity of events in MCTaco, which makes it much harder to apply a zero-shot model. In contrast, TemporalBART achieves an F1 score about 3 points higher than the Pairwise+SSVM baseline, and TemporalBART-indexed further performs best among all.

In Appendix E, we also show that our models are able to learn temporal phenomenon not explicitly annotated in our training data, which is another demonstration of our model's ability to generalize.

5.3 Ordering Unseen Events

We evaluate our BART-based models on an additional variant of this ordering problem that better tests their capability as *generative* models. Recall that previously, BART conditions on the complete (but possibly scrambled) sequence of events. We now consider ordering an event in the decoder that the model *does not* condition on in the encoder.

Architecture	All		Length >= 3	
	EM	Top2 EM	EM	Top2 EM
Random	34.1	69.5	23.7	48.7
HAQAE	37.1	71.9	28.7	53.2
GPT-2	35.2	68.4	22.6	48.2
Infilling GPT-2	38.8	73.5	26.3	55.4
TemporalBART	57.7	83.3	48.2	70.6
TemporalBART-indexed	**58.4**	**87.4**	**50.9**	**77.4**
- event deletion	42.4	73.0	29.8	53.8

Table 3: Comparison of the ability to tackle unseen events between our BART-based models and baselines on CaTeRS. The right columns are computed on test sequences of 3 or more events.

Concretely, for each temporal event sequence in CaTeRS, we randomly select one event e^*, and treat the rest of the sequence as the seed input event sequence $\{e_1, \cdots, e_N\}$. Then we check if a model can correctly determine where to insert e^* into the input sequence. Specifically, for both the BART-based models and the GPT-2 baselines, we use the generation probability to rank event sequences $\{e_1, \cdots, e_{i^*-1}, e^*, e_{i^*}, \cdots, e_N\}$ for i^* between 1 and $N+1$ (all possible locations). If a model correctly ranks the gold candidate higher, it indicates that it can model temporal relations between seen events and new *unseen events* it may generate.

The results are shown in Table 3, where we compute the top-1 and top-2 exact match (EM): did the model rank the gold sequence 1st or 2nd highest? Our GPT-2 variants are only slightly better than random. HAQAE, also using an autoencoder framework, performs worse than infilling GPT-2, likely due to the lack of large-scale pretraining and the loss of information when compressing input into latent variables. Our BART-based models are significantly better, with TemporalBART-indexed showing the benefit of using indexed event markers to help the model capture order. We also perform an ablation of deletion during training (Figure 2). Unsurprisingly for this unseen event evaluation, not deleting events in training (setting p to 0) causes a major drop by 14 EM points. Deletion denoising is evidently critical to model new events.

5.4 Event Infilling

Now we turn to temporal event infilling: given a CaTeRS sequence, remove a random event at index i^*, and denote the resulting sequence $\{e_1, \cdots, e_N\}$. We then ask a model to generate one event e^* at position i^* so $\{e_1, \cdots, e_{i^*-1}, e^*, e_{i^*}, \cdots, e_N\}$ is temporally ordered with the new event.

Architecture	Coherence	Temporality
GPT-2	1.37	0.57
Infilling GPT-2	**1.50**	0.87
TemporalBART	1.43	**1.10**
TemporalBART-indexed	**1.50**	1.03

Table 4: Human evaluation of event infilling (0-2 scale). Data are event sequences from CaTeRS. All models fill in coherent events, but our BART-based output is more temporally ordered with respect to the input events.

We evaluate the quality of the generated (inserted) events by human evaluation on Amazon Mechanical Turk. Specifically, we randomly sample 30 examples from CaTeRS and have 5 raters judge the coherence and temporality (on a scale from 0 to 2) of the inserted event from each model. See Figure 8 in Appendix for our exact prompt. The final scores for each model on coherence and temporality are computed by taking the average of the majority rating on each prediction. Here we only include GPT-2 models as baselines since HAQAE is also using the autoencoder framework, and already performs significantly worse in §5.3.

The results of this evaluation are shown in Table 4. All models achieve reasonable coherence scores. However in terms of temporality, GPT-2 performs worst, as expected, since it can only condition on partial input event sequences while the other three consider the whole event sequence as input. Both of the BART-based models achieve better performance than infilling GPT-2. The improvements on the temporal score are significant with $p < 0.05$ according to bootstrap resampling for both TemporalBART models with respect to infilling GPT-2.

Figure 6 gives examples of infilled events generated by GPT-2, infilling GPT-2, and TemporalBART. On this specific test example, GPT-2 generates an event generally about the Apple watch, which is less relevant to the input scenario about Mike making a tree. The event generated by infilling GPT-2 is coherent with the scenario, but doesn't occur in the correct order with respect to the input events. The event generated by TemporalBART is the best in terms of coherence and temporality. More examples are in Table 7 of the Appendix.

5.5 The Effectiveness of Narrative Data

Figure 7 shows that the performance of both our models on the CaTeRS ordering task improves when increasing the amount of narrative training data. This demonstrates that the automatically ex-

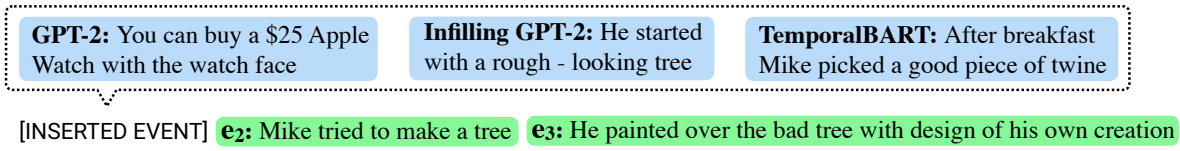

Figure 6: Real examples of infilled events generated by GPT-2, infilling GPT-2 and TemporalBART respectively. Green events are the input, and blue events are the infilled events generated by the models.

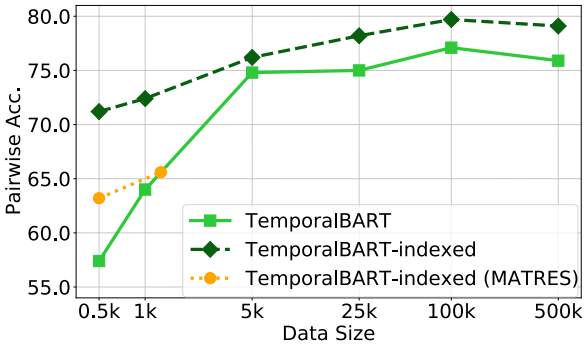

Figure 7: Pairwise accuracy of the two proposed BART-based models on temporal event ordering on CaTeRS when trained with different numbers of event sequences from narrative documents and MATRES.

tracted temporal event sequences are useful and diverse enough to help the models to learn temporal-related knowledge. The TemporalBART-indexed model is effective on surprisingly small amounts of data, but also scales well with data size; however, we observe a plateau in both models which motivated our decision to use 100k training sequences.

For comparison, we train our TemporalBART-indexed model on 1266 event sequences gathered from the MATRES dataset, a human-labeled dataset for temporal relation extraction, using the same procedure we applied to CaTeRS. However, Figure 7 shows that the resulting performance, 65.6 on MATRES, is significantly lower than the best number we get on narrative data. Even with the same size training set, using narrative data achieves over 7 points of improvement over using MATRES. This suggests that the small-scale human-labeled dataset is not enough for models to learn *generalized* temporal knowledge, and even with the same amount of data, narrative data may be a better source for general temporal knowledge.

6 Conclusion

This work presents a BART-based conditional generation model and a denoising autoencoder framework to learn temporal event knowledge, and addresses *both* temporal ordering and event infilling tasks by pretraining on automatically collected data. Our experiments demonstrate that our model is able to perform temporal ordering and infilling in a zero-shot manner, not fine-tuned on our target datasets, which suggests that it can also be applied to other settings requiring event schematic and temporal knowledge.

Acknowledgments

Thanks to Mahnaz Koupaee from Stony Brook University for providing directions on our HAQAE baseline and to the members of the UT TAUR lab for helpful discussion, particularly Yasumasa Onoe and Jiacheng Xu for suggestions on the human evaluation. Thanks as well to the anonymous reviewers for their comments. This work is based on research that is in part 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 notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory (AFRL), DARPA, or the U.S. Government.

References

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506, Baltimore, Maryland. Association for Computational Linguistics.

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, Seattle, Washington, USA. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

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

Nathanael Chambers, Shan Wang, and Dan Jurafsky. 2007. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 173–176, Prague, Czech Republic. Association for Computational Linguistics.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–6, Vancouver, Canada. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

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

Quang Do, Wei Lu, and Dan Roth. 2012. Joint inference for event timeline construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 677–687, Jeju Island, Korea. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform.

Tanya Goyal and Greg Durrett. 2019. Embedding time expressions for deep temporal ordering models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4400–4406, Florence, Italy. Association for Computational Linguistics.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Aditya Gupta and Greg Durrett. 2019. Tracking discrete and continuous entity state for process understanding. In *Proceedings of the Third Workshop on Structured Prediction for NLP*, pages 7–12, Minneapolis, Minnesota. Association for Computational Linguistics.

Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph Weischedel, and Nanyun Peng. 2019a. Deep structured neural network for event temporal relation extraction. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 666–106, Hong Kong, China. Association for Computational Linguistics.

Rujun Han, Qiang Ning, and Nanyun Peng. 2019b. Joint event and temporal relation extraction with shared representations and structured prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 434–444, Hong Kong, China. Association for Computational Linguistics.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *CoRR*, abs/1707.05501.

Bram Jans, Steven Bethard, Ivan Vulic, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL*.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Hirokazu Kiyomaru, Kazumasa Omura, Yugo Murawaki, Daisuke Kawahara, and Sadao Kurohashi. 2019. Diversity-aware event prediction based on a conditional variational autoencoder with reconstruction. In *Proceedings of the First Workshop on Commonsense Inference in Natural Language Process-*

ing, pages 113–122, Hong Kong, China. Association for Computational Linguistics.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Unsupervised machine translation using monolingual corpora only. In *International Conference on Learning Representations*.

Artuur Leeuwenberg and Marie-Francine Moens. 2017. Structured learning for temporal relation extraction from clinical records. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1150–1158, Valencia, Spain. Association for Computational Linguistics.

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

X. Lu, Y. Tsao, S. Matsuda, and C. Hori. 2013. Speech enhancement based on deep denoising autoencoder. In *INTERSPEECH*.

Aman Madaan, Dheeraj Rajagopal, Yiming Yang, Abhilasha Ravichander, E. Hovy, and Shrimai Prabhumoye. 2020. Eigen: Event influence generation using pre-trained language models. *ArXiv*, abs/2010.11764.

Aman Madaan and Yiming Yang. 2020. Neural language modeling for contextualized temporal graph generation. *ArXiv*, abs/2010.10077.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 753–760, Sydney, Australia. Association for Computational Linguistics.

Ashutosh Modi. 2016. Event embeddings for semantic script modeling. In *CoNLL*.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *IJCAI*.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016a. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California. Association for Computational Linguistics.

Nasrin Mostafazadeh, Alyson Grealish, Nathanael Chambers, James Allen, and Lucy Vanderwende. 2016b. CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the Fourth Workshop on Events*, pages 51–61, San Diego, California. Association for Computational Linguistics.

Qiang Ning, Zhili Feng, and Dan Roth. 2017. A structured learning approach to temporal relation extraction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1027–1037, Copenhagen, Denmark. Association for Computational Linguistics.

Qiang Ning, Hao Wu, Rujun Han, Nanyun Peng, Matt Gardner, and Dan Roth. 2020. TORQUE: A reading comprehension dataset of temporal ordering questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1158–1172, Online. Association for Computational Linguistics.

Qiang Ning, Hao Wu, Haoruo Peng, and Dan Roth. 2018a. Improving temporal relation extraction with a globally acquired statistical resource. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 841–851, New Orleans, Louisiana. Association for Computational Linguistics.

Qiang Ning, Hao Wu, and Dan Roth. 2018b. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

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

Haoruo Peng, Snigdha Chaturvedi, and Dan Roth. 2017. A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 173–183, Vancouver, Canada. Association for Computational Linguistics.

Haoruo Peng and Dan Roth. 2016. Two discourse driven language models for semantics. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 290–300, Berlin, Germany. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016. Learning Statistical Scripts with LSTM Recurrent Neural Networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2800–2806. AAAI Press.

James Pustejovsky, José M. Castaño, Robert Ingria, Roser Saurí, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. 2003a. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *New Directions in Question Answering*.

James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. 2003b. The timebank corpus. *Proceedings of Corpus Linguistics*.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online. Association for Computational Linguistics.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. PlotMachines: Outline-conditioned generation with dynamic plot state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4274–4295, Online. Association for Computational Linguistics.

Julien Tourille, Olivier Ferret, Aurélie Névéol, and Xavier Tannier. 2017. Neural architecture for temporal relation extraction: A Bi-LSTM approach for detecting narrative containers. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 224–230, Vancouver, Canada. Association for Computational Linguistics.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy. Association for Computational Linguistics.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 task 15: TempEval temporal relation identification. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 75–80, Prague, Czech Republic. Association for Computational Linguistics.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *Coling 2008: Companion volume: Demonstrations*, pages 189–192, Manchester, UK. Coling 2008 Organizing Committee.

P. Vincent, H. Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Mach. Learn. Res.*, 11:3371–3408.

Su Wang, Greg Durrett, and Katrin Erk. 2019. Query-focused scenario construction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2712–2722, Hong Kong, China. Association for Computational Linguistics.

Su Wang, Greg Durrett, and Katrin Erk. 2020. Narrative interpolation for generating and understanding stories. *CoRR*, abs/2008.07466.

Su Wang, Eric Holgate, Greg Durrett, and Katrin Erk. 2018. Picking apart story salads. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Brussels, Belgium. Association for Computational Linguistics.

Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. 2018a. Event representations with tensor-based compositions. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4946–4953. AAAI Press.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nathanael Chambers. 2018b. Hierarchical quantized representations for script generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3783–3792, Brussels, Belgium. 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.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:7378–7385.

Wenlin Yao and Ruihong Huang. 2018. Temporal event knowledge acquisition via identifying narratives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 537–547, Melbourne, Australia. Association for Computational Linguistics.

Xinyu Zhao, Shih-Ting Lin, and Greg Durrett. 2021. Effective distant supervision for temporal relation extraction. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 195–203, Kyiv, Ukraine. Association for Computational Linguistics.

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, A. Sabharwal, and D. Roth. 2020. Temporal reasoning on implicit events from distant supervision. *ArXiv*, abs/2010.12753.

A Scoring Orderings with TemporalBART-indexed (tags only)

TemporalBART-indexed (tags only) scores whether an output sequence $\mathbf{y}$ is temporally ordered by gathering the generation scores on the special tokens `[Ei]` only as its final ordering score:

$$P^{\text{tag}}(\mathbf{y}|\mathbf{x}) = \prod_{t \in \mathbf{I}} \text{BART}(w_t^{\mathbf{y}}|\mathbf{x}, w_1^{\mathbf{y}}, \cdots, w_{t-1}^{\mathbf{y}}) \tag{2}$$

where $\{w_t^{\mathbf{y}}\}$ is the text representation of $\mathbf{y}$ and $\mathbf{I}$ is the set of the positions of the special tokens `[Ei]` in $\{w_t^{\mathbf{y}}\}$. This allows us to make a judgment only depending on the predicted *temporal order* of the events rather than mixing in general token order. In contrast, TemporalBART scores a sequence of events $\mathbf{y}$ with the generation probability on the entire text representation of $\mathbf{y}$:

$$P^{\text{gen}}(\mathbf{y}|\mathbf{x}) = \prod_{t} \text{BART}(w_t^{\mathbf{y}}|\mathbf{x}, w_1^{\mathbf{y}}, \cdots, w_{t-1}^{\mathbf{y}}) \tag{3}$$

Since many of the generation decisions here are copying event arguments, the prediction could be largely affected by the correlation of tokens within each argument.

Architecture	Acc.	Macro F1
TemporalBART-indexed	74.9	55.1
TemporalBART-indexed (tags only)	**76.6**	**56.4**

Table 5: The comparison between TemporalBART-indexed and its (tags only) variant on temporal event ordering. The test data and metrics used here are same as in Table 2.

We evaluate "TemporalBART-indexed (tags only)" on the temporal event ordering with the procedure used for the models in Table 2. Table 5 shows that this (tags only) variant further boosts the performance of TemporalBART-indexed by 1.3 points on the macro F1. This result verifies that this setting can help prevent the ordering scores from being overly affected by the text generation probabilities, which is particularly important for MCTaco, where the arguments of events are more complex.

B Architecture of BERT-based Pairwise Model + SSVM

This network uses a BERT-based model (Devlin et al., 2019) to obtain a vectorized representation for each input event e_i in $\mathbf{x}$. As with the BART-based models, the input to the BERT model is the concatenation of $\text{Repr}(e_i)$ with `[E]` being prepended in front of each event. The vectorized representation for e_i is then extracted by $\mathbf{U}_{p_i}$, where $\mathbf{U}$ is the final BERT encoded matrix, and p_i is the position of the first token of e_i in the input sequence. Each pair of event representations, $\mathbf{U}_{p_i}$ and $\mathbf{U}_{p_j}$ are then fed into a feed-forward function g to compute a score B for e_i preceding e_j in the output $\mathbf{y}$:

$$B(e_i, e_j) = g([\mathbf{U}_{p_i}; \mathbf{U}_{p_j}; \mathbf{U}_{p_i} \odot \mathbf{U}_{p_j}]) \tag{4}$$

Finally, the final output $\mathbf{y}$ is computed by finding the best permutation over all of the pairwise scores by solving an ILP.

C Training Details of BART-based Models

We train our BART-based conditional generation models to minimize negative log likelihood of reconstructing the original event sequence. We set the learning rate to 1e-5, and use a polynomial decay scheduling with 500 steps of warm-up. All of the models are trained for 10 epochs, with each epoch being 2000 updates and the batch size being 64. For the deletion training scheme, we set the event deletion probability p to 0.15. The framework is implemented with PyTorch (Paszke et al., 2019) and AllenNLP (Gardner et al., 2017), and we use the BART-large pretrained model from HuggingFace's Transformers library (Wolf et al., 2020).

During the evaluation for temporal event ordering, we decode the output event sequences using beam search with the beam size being 4. For event infilling task, we use nucleus sampling with p set to 0.8.

D Human Evaluation

Figure 8 shows the prompt for the human evaluation described in §5.4, where we ask the MTurk raters to evaluate the coherence and temporality of the generation outputs. To help the raters ignore grammatical issues when making decisions, we first ask them to check the grammaticality, then separately judge the coherence and the temporality.

E Learning Timex Knowledge

The temporal ordering and event infilling tasks correspond to information that we might expect to be

Architecture	Year		Month		Weekday		Hour:Minute (24)		Hour:Minute (12)	
	EM	Pairwise	EM	Pairwise	EM	Pairwise	EM	Pairwise	EM	Pairwise
Random	26.0	53.0	21.0	51.0	18.0	52.0	18.0	50.7	13.0	52.7
GPT-2	18.0	49.0	14.0	45.3	18.0	55.3	12.0	43.3	15.0	46.7
GPT-2 Large	15.0	47.3	16.0	47.7	19.0	57.7	9.0	41.7	11.0	48.7
TemporalBART	**93.0**	**96.7**	83.0	88.7	67.0	78.0	**88.0**	**93.3**	**67.0**	**78.7**
TemporalBART-indexed	81.0	91.3	**85.0**	**90.0**	**71.0**	**80.7**	84.0	90.7	65.0	78.0

Table 6: The results of temporal event ordering on the events anchored with various types of timex. The test data used here are length-3 sequences artificially made up with "die" events for the "Year" timex, and 3 typical daily events as shown in Figure 9 for other types of timex. The timexes of type "Year" are randomly sampled from 1000 to 2100. Our BART-based models significantly outperform the GPT-2 and random baselines, showing that they can capture useful timex-related knowledge.

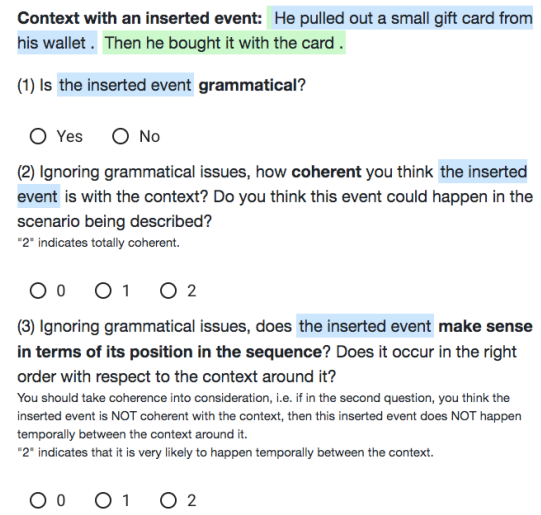

Figure 8: A screenshot of prompt for the human evaluation described in §5.4, which includes the 3 questions the raters are asked to judge the event infilling outputs from each model. The input events are highlighted with the color green, and blue for the inserted events.

encoded by our model pre-training. To test whether our models generalize to slightly more distant temporal phenomena, we examine whether they are able to capture the temporal relationships between timexes. This knowledge has been shown to be hard to learn in temporal relation extraction models (Goyal and Durrett, 2019).

E.1 Evaluation Setup

The timexes we examine here include years, months, weekdays, 24-hour clock time in "hour:minute" format and 12-hour clock time in "hour:minute am/pm" format. We evaluate the ability of our models to order events that are anchored with a timex in their arguments. To prepare the test input event sequences of a given type of timex, we first artificially make up a template event sequence with 3 typical daily events that have no

Event 1 Durer went to a supermarket **at 7:52 pm**

Event 2 Durer bought a book at a shop **at 5:16 am**

Event 3 Durer took a photo in front of a museum **at 3:10 pm**

Figure 9: An example of test input event sequences for timex evaluation. The appended timexes in each event, which are 12-hour clock time here, is highlighted.

temporal order relations. We then randomly sample 3 different timexes, e.g "June", "May", "July" for "Month", and append each of them to the events in the template sequence respectively with proper prepositions. At the end, 100 examples are created with this process for each type of timex. More concrete examples are shown in Figure 9. For the baselines, here we use GPT-2 models to do the ordering by using the generation probability to rank all permutations of the input events.

E.2 Results

The results are shown in Table 6. First, we examine the results of the GPT-2 models. In general both the unsupervised GPT-2 (the medium model) and GPT-2 large perform worse than the random baseline, indicating that they have a limited ability to order timexes. Our BART-based models achieve stronger results. The results are strongest on years. For 12-hour clock time, even though the model has to make a challenging link between the temporal knowledge on "am' and "pm" and numerical comparisons, both of the BART-based models still performs significantly better the random baseline.

F Examples for Event Infilling

In Table 7, we demonstrate more examples of the infilled events generated by GPT-2, infilling GPT-2 and TemporalBART given the seed event sequences from CaTeRS. In general, while the events output

#1	**Scenario**: [INSERTED EVENT] he tell me Happy Birthday		
	GPT-2: A New England University study reveals the effect of the American tax code on college tuition .	**Infilling GPT-2**: He then turned to me and his mom and said, " Good morning .	**TemporalBART**: He looked at me in the most amazing of ways, as if he were wishing me a Happy Birthday .
#2	**Scenario**: [INSERTED EVENT] Then he bought it with the card .		
	GPT-2: A federal judge dismissed a lawsuit Wednesday from the family of an elderly man who died after inhaling a toxic gas during an air show at the Lincoln Memorial .	**Infilling GPT-2**: He put the card in his wallet .	**TemporalBART**: He pulled out a small gift card from his wallet .
#3	**Scenario**: Kim hunkered down at home as a hurricane blew through town . [INSERTED EVENT] She went to sleep hoping all would be well . She hoping all would be well .		
	GPT-2: It wasn't until his brother and mother returned from vacation that he found out that a hurricane had struck, bringing the death toll from Hurricane Andrew in 2012 up to 24 and leaving many people without electricity for weeks .	**Infilling GPT-2**: Kim slept soundly .	**TemporalBART**: Kim turned the TV to catch the latest news .
#4	**Scenario**: Tony needed to buy his grandma a birthday present . He went to her favorite bakery . The owner told Tony how to make it himself . [INSERTED EVENT] His elated grandma couldn't tell the difference .		
	GPT-2: Tony got mad and left the bakery .	**Infilling GPT-2**: Tony went to the grocery store .	**TemporalBART**: make a loaf that looked like grandma's .
#5	**Scenario**: [INSERTED EVENT] He decided to use the batteries in his fire detector . He use the batteries in his fire detector .		
	GPT-2: I'm an independent developer who's worked for both big and small companies .	**Infilling GPT-2**: He find a place to charge the fire detector batteries .	**TemporalBART**: He see the batteries in his alarm clock were dead .

Table 7: More examples of the infilled events generated by GPT-2, infilling GPT-2 and TemporalBART respectively. Scenarios are the temporally-ordered input events fed into the models, with the events separated by periods, and the insertion position specified by [INSERTED EVENT] in this figure. The second row in each example shows the infilled event generated by each model.

by TemporalBART are coherent and temporally-sensible, those from the GPT-2 models has a worse quality in terms of the temporality. Note that the nature of the event representation does not necessarily guarantee a grammatical sentence when the event is rendered in surface order.