Generating Temporally-ordered Event Sequences via Event Optimal Transport

Bo Zhou[1,2], **Yubo Chen**[1,2], **Kang Liu**[1,2,4], **Jun Zhao**[1,2],
Jiexin Xu[3], **Xiaojian Jiang**[3], **Qiuxia Li**[3]
[1]School of Artificial Intelligence, University of Chinese Academy of Sciences
[2]National Laboratory of Pattern Recognition, CASIA
[3]China Merchants Bank, [4]Beijing Academy of Artificial Intelligence
{bo.zhou,yubo.chen,kliu,jzhao}@nlpr.ia.ac.cn
{jiexinx,jiangxiaojian,annielqx}@cmbchina.com

Abstract

Generating temporally-ordered event sequences in texts is important to natural language processing. Two emerging tasks in this direction are temporal event ordering (rearranging the set of events to correct order) and event infilling (generating an event at a specified position). To tackle the two related tasks, the existing method adopts a vanilla sequence-to-sequence model via maximum likelihood estimation (MLE). However, applying this approach to these tasks will cause two issues. One issue is that the MLE loss emphasizes strict local alignment and ignores the global semantics of the event. The other issue is that the model adopts a word-level objective to model events in texts, failing to evaluate the predicted results of the model from the perspective of event sequence. To alleviate these issues, we present a novel model to tackle the generation of temporally-ordered event sequences via Event Optimal Transport (EOT). First, we treat the events in the sequence as modeling units and explicitly extract the semantics of the events. Second, to provide event sequence-level evaluation of the predicted results of the model, we directly match events in sequences. Extensive experimental results show that our approach outperforms previous models on all evaluation datasets. In particular, the accuracy is improved by 7.7%, and the Macro F1 is improved by 7.2% on one of the datasets.

1 Introduction

Generating temporally-ordered event sequences in texts is crucial to many artificial intelligence applications, such as discourse understanding (Nie et al., 2019), dialog generation (Wu et al., 2018) and stock prediction (Ding et al., 2016). Temporal event ordering and event infilling are two challenging tasks in this line of work. The former refers to the rearrangement of an unordered sequence of events to an ordered sequence of events, and the latter refers to inferring missing events in an incomplete sequence of events. Figure 1 shows an example of the temporal event ordering task and event infilling task in (a) and (b), respectively. For the temporal event ordering, given an unordered sequence (e_3, e_2, e_4, e_1) as input, the output is ordered sequence (e_1, e_2, e_3, e_4). For the event infilling, given an incomplete sequence (e_1, e_2, e_4) as input, the output is complete sequence (e_1, e_2, e_3, e_4) with the infilled event e_3.

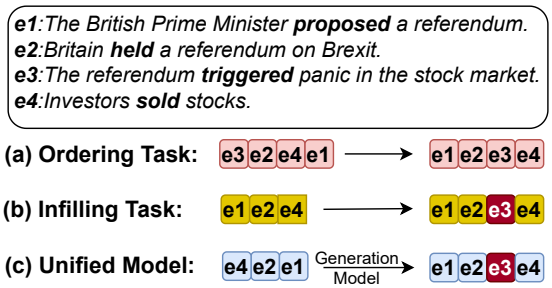

Figure 1: The temporal event ordering task, event infilling task and a single generation model handling these two tasks.

To handle these two related tasks, the currently existing method (Lin et al., 2021) employs a unified generation model which is shown in Figure 1 (c). Their model takes incomplete unordered events as input sequence and outputs a complete ordered event sequence, based on the sequence-to-sequence (Seq2Seq) model via maximum likelihood estimation (MLE), which maximizes the likelihood of the next word conditioned on its previous ground-truth words. Such an approach adopts cross-entropy loss as the objective, essentially measuring the word difference at each position of the target sequence and providing a word-level training loss. Although their model has shown good performance in handling the two tasks, there are still two issues.

First, the MLE loss emphasizes strict local alignment and ignores the global semantics of the event. For example in Figure 2, the MLE loss will give the

Proceedings of the 29th International Conference on Computational Linguistics, pages 1875–1884
October 12–17, 2022.

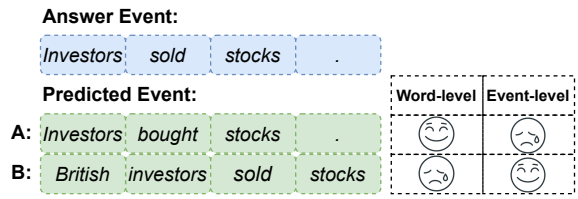

Figure 2: Comparison between word-level score and event-level score on two predicted events.

predicted event *A* a high score and *B* a low score because all words except one in event *A* are aligned with the words in the answer event yet no word event *B* is aligned with the words in the answer event. However, the loss is expected to give event *A* a low score and event *B* a high score because from the perspective of event semantics, event *A* is completely different from the answer event and event *B* is the same as the answer event. Therefore, we should consider the overall semantics of the event, rather than the strict alignment of words within the event.

Second, the goal of the model is to infer events in texts, but the sequence-to-sequence model uses a word-level objective so it fails to evaluate the result from the perspective of event sequence. For example in Figure 1, the correct predicted event sequence is (e_1, e_2, e_3, e_4). If the model predicts an event sequence (e_4, e_1, e_2, e_3), the MLE loss will give it a low score because no event is predicted correctly compared to the answer event sequence. However, the sequence (e_4, e_1, e_2, e_3) should not get a low score, because at least the orders of 3 events are predicted correctly. Therefore, how to evaluate the predicted results of the model from the perspective of event sequence is a challenging problem.

To tackle the above issues, we introduce a novel method for the generation of temporally-ordered event sequences via Event Optimal Transport (EOT). Specifically, for the first issue, we treat the events in the sequence as modeling units and explicitly extract the semantics of the events. For the second issue, we propose to use optimal transport to directly match events in sequences. The EOT allows end-to-end supervised training and acts as an effective sequence-level regularization to the MLE loss.

In summary, our contributions can be summarized as follows:

- We introduce a novel method for the generation of temporally-ordered event sequences via Event Optimal Transport (EOT), which treats the events in the sequence as modeling units and explicitly extracts the semantics of the events.
- We directly match events in sequences to provide event sequence-level evaluation of the predicted results of the model.
- Extensive experimental results demonstrate the superiority of the proposed method on all evaluation datasets. Specifically, the accuracy is improved by 7.7%, and the Macro F1 is improved by 7.2% on one of the datasets.

2 Related Work

Script Event Prediction Given context event sequence, script event prediction aims at predicting the subsequent event from a candidate list. The task is first proposed by Chambers and Jurafsky (2008), and a statistical model is proposed to learn the cooccurrence between events. Jans et al. (2012) leverages a bigram model to model the temporal order between events explicitly. The above two methods are count-based, and then researchers have proposed methods based on neural networks. Granroth-Wilding and Clark (2016) proposes a neural network based model for simultaneously learning word embedding and composition function. In Wang et al. (2017), they propose an LSTM based model to integrate order information and event relation. Li et al. (2018) treats event chain as a subgraph and leverages recurrent networks to better model relatedness between events in the candidate list with events in the graph. Lv et al. (2019) proposes a model that integrates event-level and chain-level attentions to better leverage information contained in event chain. Zhou et al. (2021) proposes a multi-task self-supervised model to cope with the problem of lack of training data in script event prediction. To incorporate event circumstances into the narrative event prediction, Wang et al. (2021) adopts the two multi-head attention to retrieve circumstances at the local and global levels. In this paper, we consider two related tasks of temporal event ordering and event infilling and leverage a single generation model to tackle these two tasks.

Optimal Transport in NLP Optimal transport has been applied in NLP for a variety of tasks recently. In Kusner et al. (2015), the author proposes the Word Mover's Distance (WMD), a

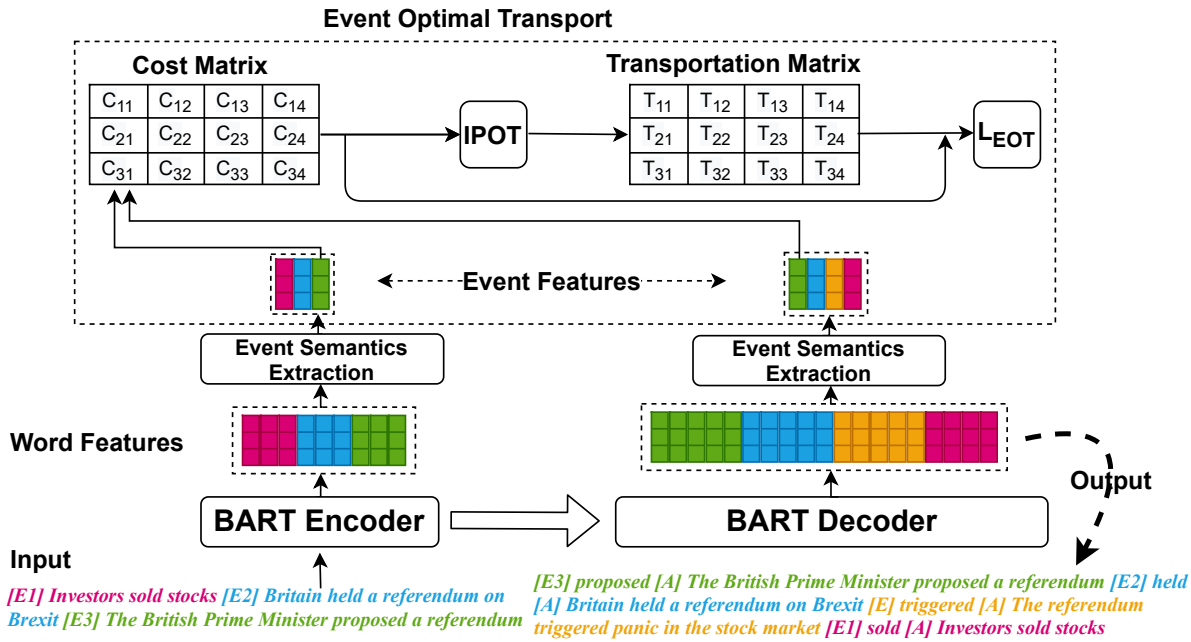

Figure 3: Overall architecture of the proposed generation model based on event optimal transport.

novel distance function between text documents that measures the dissimilarity between two text documents as the minimum amount of distance that the embedded words of one document need to "travel" to reach the embedded words of another document. Later in Xu et al. (2018), a novel Wasserstein method with a distillation mechanism is proposed, yielding joint learning of word embeddings and topics. The cross-lingual correspondence problem is cast as an optimal transport problem in Alvarez-Melis and Jaakkola (2018), and the Gromov-Wasserstein distance is exploited for the alignment of word embedding spaces. A content-aware sparse attention module based on optimal transport is proposed in Chen et al. (2020) to deal with textual network embedding problem. Li et al. (2020) proposes student-forcing optimal transport to tackle the exposure bias problem in text-generation models trained by maximum likelihood estimation. Xu et al. (2021) formulates the quest of vocabularization – finding the best token dictionary with a proper size – as an optimal transport problem and proposes a simple and efficient solution without trial training. A time-aware optimal transport distance is introduced (Li et al., 2021) for learning the model to compress the event-graphs of news articles in an unsupervised manner. In this paper, we leverage optimal transport and propose event OT to directly match events in event sequences to provide sequence-level supervision of the predicted results.

3 Methodology

The overall structure of our model is illustrated in Figure 3. The input of the model is a sequence of events $\boldsymbol{x} = (e_1, ..., e_n)$, which is incomplete and unordered, and the output of the model is a complete and ordered sequence of events $\boldsymbol{y} = (e_1, ..., e_m)$. We represent e in the event sequence as the concatenation of its predicate with all its arguments.

3.1 Generation Model

To leverage the power of pretrained transformers, we base the underlying architecture for our model on BART (Lewis et al., 2020). BART is a denoising autoencoder for pretraining sequence-to-sequence models which can be seen as generalizing BERT (Kenton and Toutanova, 2019) (due to the bidirectional encoder), GPT (Radford and Narasimhan, 2018) (with the left-to-right decoder), and other recent pretraining schemes.

Similar to previous work (Lin et al., 2021), we then prepend a special word $[Ei]$ in front of each event in input event sequence $\boldsymbol{x}$. For the output, if e_j in $\boldsymbol{y}$ is one of the input events e_i in $\boldsymbol{x}$, then we prepend special words $[Ei]v_{e_j}[A]$ before e_j, where v_{e_j} is the predicate of event e_j. Otherwise, the special words $[E]v_{e_j}[A]$ are used in front of e_j. An example of the above is shown in Figure 3. The

use of $[Ei]$ helps the model differentiate between events in the input and output sequences, which facilitates optimal transport between events.

3.2 Event Optimal Transport

We first briefly introduce the optimal transport. Given two spaces X and Y, the Kantorovich formulation of optimal transport aims to find a probability measure γ on $X \times Y$ that attains the following infimum:

$$\inf \left\{ \int_{X \times Y} c(x, y) \mathrm{d}\gamma(x, y) \mid \gamma \in \Gamma(\mu, \nu) \right\} \quad (1)$$

where $\Gamma(\mu, \nu)$ is the set of probability measures on $X \times Y$ with marginal μ on X and ν on Y.

Now given two discrete probability measures μ and ν, which can be represented by Dirac measure as below:

$$\mu = \sum_{i=1}^{n} u_i \delta_{\boldsymbol{x}_i}, \quad \nu = \sum_{j=1}^{m} v_i \delta_{\boldsymbol{y}_j} \quad (2)$$

where $\delta_{\boldsymbol{x}}$ is the Dirac measure sitting at $\boldsymbol{x}$. The coefficients $\boldsymbol{u} = \{u_i\}_{i=1}^{n}$ and $\boldsymbol{v} = \{v_j\}_{j=1}^{m}$ satisfy the constraints $\sum_{i=1}^{n} u_i = \sum_{j=1}^{m} v_i = 1$ as both μ and ν are probability measures. Under such a setting, the objective function and the constraint in the primal Kantorovich problem are then:

$$\begin{aligned} \min_{\boldsymbol{T}} \quad & \sum_{i,j} \boldsymbol{T}_{ij} \cdot c\left(\boldsymbol{x}_i, \boldsymbol{y}_j\right) = \min_{\boldsymbol{T}} \quad \langle \boldsymbol{T}, \boldsymbol{C} \rangle \\ s.t. \quad & \sum_{j} \boldsymbol{T}_{ij} = \boldsymbol{u}_i, \quad i = 1, ..., n, \\ & \sum_{i} \boldsymbol{T}_{ij} = \boldsymbol{v}_j, \quad j = 1, ..., m, \\ & \boldsymbol{T} \in \mathbb{R}_{+}^{n \times m}. \end{aligned} \quad (3)$$

where $\boldsymbol{C}$ is the cost matrix with $\boldsymbol{C}_{ij} = c\left(\boldsymbol{x}_i, \boldsymbol{y}_j\right)$ and c is the cost function. $\langle \boldsymbol{T}, \boldsymbol{C} \rangle = \mathrm{Tr}(\boldsymbol{T}^\top \boldsymbol{C})$ denotes the Frobenius dot-product, here Tr denotes the trace of a matrix.

Because the exact solution of (3) is is highly intractable (Arjovsky et al., 2017), researchers have proposed some approximate algorithms, such as the Sinkhorn (Cuturi, 2013) and IPOT (Xie et al., 2020) algorithms. Here we choose the IPOT algorithm to approximate the minimizer of (3). The details of the IPOT algorithm are shown in Algorithm 1.

Assume the output (the last hidden state) of the BART encoder and decoder are $\boldsymbol{H}_e \in \mathbb{R}^{L_e \times d}$ and $\boldsymbol{H}_d \in \mathbb{R}^{L_d \times d}$ respectively, where L_e and L_d are

Algorithm 1 IPOT algorithm

Input: Feature vectors $S = \{\boldsymbol{x}_i\}_1^n$,$S' = \{\boldsymbol{x}'_i\}_1^m$
Parameter: Generalized stepsize $\frac{1}{\beta}$
Output: $\langle \boldsymbol{T}, \boldsymbol{C} \rangle$

$\boldsymbol{\sigma} = \frac{1}{m}\mathbf{1}_{\boldsymbol{m}}$,$\boldsymbol{T}^{(1)} = \mathbf{1}_{\boldsymbol{n}}\mathbf{1}_{\boldsymbol{m}}{}^\top$.
$\boldsymbol{C}_{ij} = c(\boldsymbol{x}_i, \boldsymbol{x}'_j)$,$\boldsymbol{A}_{ij} = e^{-\frac{\boldsymbol{C}_{ij}}{\beta}}$.
for $t = 1, 2, ..., T$ **do**
  $\boldsymbol{Q} = \boldsymbol{A} \odot \boldsymbol{T}^{(t)}$. // $\odot$ is Hadamart product
  for $k = 1, 2, ..., K$ **do**
    $\boldsymbol{\delta} = \frac{1}{n\boldsymbol{Q}\boldsymbol{\sigma}}$,$\boldsymbol{\sigma} = \frac{1}{m\boldsymbol{Q}^\top\boldsymbol{\delta}}$.
  end for
  $\boldsymbol{T}^{(t+1)} = diag(\delta)\boldsymbol{Q}diag(\sigma)$.
end for
return $\langle \boldsymbol{T}, \boldsymbol{C} \rangle$

lengths of input and output sequences. We first partition $\boldsymbol{H}_e$ into n groups where each group corresponds to features of words in an event (note that the input to our model is $\boldsymbol{x} = \{e_1, ..., e_n\}$), then features of words in a group are averaged to obtain a vector sequence $S = \{\boldsymbol{h}_i\}_{i=1}^{n}$. Note that besides average, we also tried other aggregation functions, such as max pooling, attention, etc., but they all performed worse than average. The possible reason is that there are already more complex aggregation functions such as attention in BART, so it is enough to use average as the aggregation function of the BART output.

For $\boldsymbol{H}_d$, it's first input to a linear layer:

$$\boldsymbol{H}_l = \boldsymbol{H}_d\boldsymbol{W} + \boldsymbol{b} \quad (4)$$

where $\boldsymbol{W} \in \mathbb{R}^{d \times V}$ and V is the vocabulary size. We then use the $argmax$ function over $\boldsymbol{H}_l$ to obtain the predicted indices $\boldsymbol{i} \in \mathbb{Z}_{+}^{L_d}$. We identify indices of special word $[E_i]$ in $\boldsymbol{i}$ and these identified indices are used to partition $\boldsymbol{H}_d$ into k groups where each group corresponds to features of words in an event. Note that there may be cases where there are no identified indices, which are most likely to occur at the beginning of training. To solve possible errors, we partition $\boldsymbol{H}_d$ into L_d groups when this happens. Finally after averaging, we obtain another vector sequence $S' = \{\boldsymbol{h}_j\}_{j=1}^{k}$.

3.3 Training and Optimization

In order to use a unified generation model to handle temporal event ordering and event infilling tasks, we first construct input-output data pairs related to these two tasks. Specifically, given an ordered

Algorithm 2 EOT algorithm

Input: Ground truth $\{(\boldsymbol{x}_1, \boldsymbol{y}_1), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$
Parameter: Batch size M

Initialize MLE model parameters.
for $epoch = 1, ..., MaxEpoch$ **do**
  for $k = 1, ..., M$ **do**
    Draw a pair of sequences $(\boldsymbol{x}_i, \boldsymbol{y}_i)$.
    Compute the outputs $\boldsymbol{H}_e$ and $\boldsymbol{H}_d$ of the encoder and decoder.
    Extract feature vector sequences of events S and S' from $\boldsymbol{H}_e$ and $\boldsymbol{H}_d$.
    Compute the cost matrix $\boldsymbol{C}$ based on S and S' via cosine distance.
    Compute the EOT loss defined in (4).
  end for
  Update model parameters by optimizing loss in (6).
end for

event sequence $\boldsymbol{y}$ as defined above, we follow previous work (Lewis et al., 2020; Lin et al., 2021) to corrupt it to obtain the required input $\boldsymbol{x}$ by two steps. The first step is to shuffle events: performing a random shuffling of the complete ordered event sequence $\boldsymbol{y}$ to produce an unordered event sequence $\boldsymbol{x}'$. The second step is to delete events: randomly deleting each event in $\boldsymbol{x}'$ with probability p to produce the incomplete unordered input event sequence $\boldsymbol{x}$.

Take Figure 1 (c) for example, the complete ordered event sequence on the right is $\boldsymbol{y}$. After performing event shuffling and event deletion, we obtain incomplete unordered input event sequence $\boldsymbol{x}$ on the left.

Now we introduce the optimization process of the model. After obtaining two vector sequences $S = \{\boldsymbol{h}_i\}_{i=1}^{n}$ and $S' = \{\boldsymbol{h}_j\}_{j=1}^{k}$ as described in the previous section, we can compute the event-level OT loss using the IPOT algorithm described above:

$$\mathcal{L}_{EOT} = \text{IPOT}(S, S') \tag{5}$$

Meanwhile, suppose the input sequence is $\boldsymbol{x} = (w_1, ..., w_L)$ and the gold output sequence is $\boldsymbol{y} = (w_1, ..., w_{L'})$, where L and L' are numbers of words. We also have the following maximum likelihood estimation (MLE) loss:

$$\mathcal{L}_{MLE} = \sum_{t=1}^{L'} \log p_\theta(w_t | w_{<t}, \boldsymbol{x}) \tag{6}$$

We then combine the two loss functions to obtain the following loss:

$$\mathcal{L} = \frac{1}{M} \sum_{n=1}^{M} (-\mathcal{L}_{MLE}(\boldsymbol{x}_n, \boldsymbol{y}_n) + \alpha \mathcal{L}_{EOT}(S_n, S'_n)) \tag{7}$$

where M is the number of training pairs. The parameters are updated by minimizing this loss and the full algorithm is summarized in Algorithm 2.

4 Experiment

4.1 Experimental Setup

Dataset Following Lin et al. (2021), the temporal event sequences are extracted from the EventsNarratives corpus (Yao and Huang, 2018). The SRL model from AllenNLP (Gardner et al., 2018) is used to extract verbs (events) and their arguments. Then, only events in different sentences are connected to construct temporal event sequences, and only event chains associated with a common entity are included. We train our model on 100,000 sequences extracted by the above procedure. Two different orders are used to scramble each sequence, resulting in a total of 200,000 training data.

For testing, two out-of-domain English datasets CaTeRS (Mostafazadeh et al., 2016) and MCTaco (Zhou et al., 2019) are used to extract the test temporal event sequences. CaTeRS includes annotations of events and their causal and temporal relations on short stories. MCTaco is a question answering dataset for evaluating the model's capability of understanding temporal commonsense. Following Lin *et al.* (2021), 842 event sequences are extracted for CaTeRS and after applying two different permutations to each sequence, 1684 CaTeRS examples are finally obtained. For MCTaco, 585 test sequences are extracted.

Training Details We choose the cosine distance as the cost function in the event optimal transport and the hyper-parameter α is set to 0.1. The learning rate of our model is 1e-5, and a polynomial decay scheduling with 500 steps of warm-up is used. We set the batch size to 64, the models are trained for 10 epochs, with 2000 updates each epoch. We set the event deletion probability to 0.15 for the deletion training strategy. The BART-large pre-trained model from Hugging-Face's Transformers library (Wolf et al., 2020) is used as the underlying structure which is the same as previous work. Beam search with the beam size 4 is used when decoding the output event sequences during the evaluation for temporal event ordering.

	CaTeRS		MCTaco	
Model	All Pair Acc.	Longer Pair Acc.	Acc.	Macro F1
BERT-based SSVM	65.7	62.3	67.2	47.0
Pointer Network	54.1	52.3	54.7	42.7
TemporalBART†	77.1	74.7	63.9	50.1
TemporalBART-indexed†	79.7	78.0	74.9	55.1
EOT	**81.3**	**80.2**	**82.6**	**62.3**

Table 1: Temporal ordering results on dataset CaTeRS and MCTaco. For CaTeRS, All Pair Acc. means pairwise accuracy of predicted ordering for all event sequences, while Longer Pair Acc. denotes pairwise accuracy for sequences containing more than three events. For MCTaco, Accuracy and Macro F1 score are computed on the ordering between the question event and answer event. The symbol † represents the results we reproduced using public code released by Lin et al. (2021), and other two results are taken from Lin et al. (2021).

Baselines We compare our model with the following baseline methods for temporal event ordering:

- **BERT-based Pairwise Model + SSVM** (Han et al., 2019) leverages a BERT-based model (Kenton and Toutanova, 2019) to compute pairwise scores for two events in the output, and the final output is then obtained by solving an ILP over all the pairwise scores.
- **BERT-based Pointer Network** first uses BERT to extract representations for events that are fed into an LSTM-based pointer network to compute the probability for ordering.
- **TemporalBART** (Lin et al., 2021) is based on BART (Lewis et al., 2020), and special words are prepended in front of events in input and output sequences to provide extra clues.
- **TemporalBART-indexed** (Lin et al., 2021) is the same as TemporalBART except that the indices of the special words prepended before events are considered.

For event infilling, we compare our model with these extra baselines:

- **HAQAE** (Weber et al., 2018) is a vector quantized variational autoencoder with a latent space defined by a hierarchy of categorical variables which encodes schema knowledge.
- **GPT-2** (Radford et al., 2018) is a transformer-based pretrained language model which is used as the underlying structure in many generation tasks.
- **Infilling GPT-2** (Qin et al., 2020) generates the infilling events conditioned on both the prefix events and the events after the insertion position.

4.2 Results on Temporal Event Ordering

CaTeRS Dataset Experimental results on CaTeRS are shown in Table 1, pairwise accuracy is used to calculate the proportion of ordered event pairs in the predicted sequence.

Among all the approaches, our method performs best. It achieves the best performance on both all sequences and long sequences. The reasons may be that compared with the BERT-based pointer network, our model can condition the word-level embeddings of the events when generating the output events instead of condensed event embeddings. Compared with BART-based models, our model treats the events in the sequence as modeling units and explicitly extracts the semantics of the events instead of emphasizing strict local alignment of words.

MCTaco Dataset The accuracy on predicting the temporal relation of event in question and event in answer is computed, since only gold temporal relation of event in question and event in answer is known for each test sequence. The macro F1 score is also computed because the proportion of before/after questions is unbalanced in MCTaco.

Our EOT model outperforms all the baselines, in particular, our model outperforms TemporalBART-indexed by 7.7% in accuracy and 7.2% in Macro F1. We attribute the significant improvement to the direct events matching in sequences by our model. We will further demonstrate the effectiveness of direct events matching in the following experiments.

Ordering Unseen Events for CaTeRS Following Lin et al. (2021), we also evaluate our EOT model on an additional variant task of temporal

Model	All seq		Longer seq	
	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
EOT	**58.8**	**88.2**	**52.6**	**78.1**

Table 2: The results of ordering unseen events on sequences from dataset CaTeRS. Longer seq means results for sequences containing more than three events. The symbol † has same meaning as Table 1.

event ordering which better tests its capability as a generative model. Specifically, for each event sequence in CaTeRS, we first randomly delete an event e^* in the sequence, and let the remaining events be denoted as $(e_1, ..., e_N)$. We want to test whether the model can insert the deleted event e^* to its original position, thus we use the generation probability of the model to rank the $N + 1$ sequences which are obtained by inserting e^* to $N + 1$ different positions. The higher the model ranks the original sequence, the better the model is able to capture the relationship between seen events and possibly generated unseen events.

Table 2 shows the results, and the top-1 and top-2 exact match (EM) are used to evaluate the results, which calculate the proportion of gold sequences that the model ranks first and second above. Our model outperforms all the baselines on both all the event sequences and long sequences, which again demonstrates the effectiveness of considering event-level semantics and direct events matching in sequences. Another observation is that compared with the TenporalBART-indexed model, the performance of our EOT model on longer sequences is more significant than on all the sequences. One possible reason is that longer sequences have more events which are more helpful for the model to do the event-level matching.

4.3 Results on Event Infilling

Now we consider temporal event infilling and the dataset CaTeRS is used. Given an event sequence from CaTeRS, we first randomly delete an event to obtain $(e_1, ..., e_{i-1}, e_{i+1}, ..., e_n)$. Then the model should generate an infilled event e^* at position i and the new sequence $(e_1, ..., e_{i-1}, e_i, e_{i+1}, ..., e_n)$ is expected to be temporally ordered.

We measure the quality of generated events through human evaluation. Given a sequence with a generated event at some position, 3 raters are asked to score this sequence in terms of the coherence (How coherent the generated event is with the context?) and temporality (Does the generated event occur in the right order concerning the context?) for the generated event and both scores are from $\{0, 1, 2\}$. The final scores are the majority scores of the 3 raters for both coherence and temporality. We then randomly sample 15 sequences from CaTeRS and the averaged scores are took as the metric.

The result is shown in Table 3, and our EOT model achieves better performance than all the baseline models in terms of coherence and temporality. The reason may be that by explicitly matching events in input and output sequence, the model can generate an event that is more relevant to the scenario of events in the input sequence.

Model	Coherence	Temporality
GPT-2	1.27	0.60
Infilling GPT-2	1.53	0.80
TemporalBART	1.13	0.87
TemporalBART-indexed	1.53	1.07
EOT	**1.67**	**1.13**

Table 3: The human evaluation result for event infilling on dataset CaTeRS.

We show two examples of events generated by different models in Figure 4. As we can see in Figure 4 (a), the event generated by infilling GPT-2 is less relevant to the context events. The order of the event generated by TemporalBART-indexed is inappropriate, although it's coherent with the scenario of the input events. The event generated by our model EOT is both coherent and temporally ordered. Another example is shown in 4 (b).

4.4 Further Demonstration of EOT's Effectiveness

To further demonstrate the effectiveness of our event optimal transport, we compare EOT with two extra models. Recall that in our model, the output (the last hidden state) of the BART encoder and decoder are $\boldsymbol{H}_e \in \mathbb{R}^{L_e \times d}$ and $\boldsymbol{H}_d \in \mathbb{R}^{L_d \times d}$ respectively, then two vector sequences $S = \{\boldsymbol{h}_i\}_{i=1}^n$ and $S' = \{\boldsymbol{h}_j\}_{j=1}^k$ corresponding to events in input and output are extracted to construct event OT loss. Now, we directly treat vectors in $\boldsymbol{H}_e$ and $\boldsymbol{H}_d$ as S and S' respectively to construct word-level loss, and this model is called Word Optimal Transport (WOT). We also test the model which com-

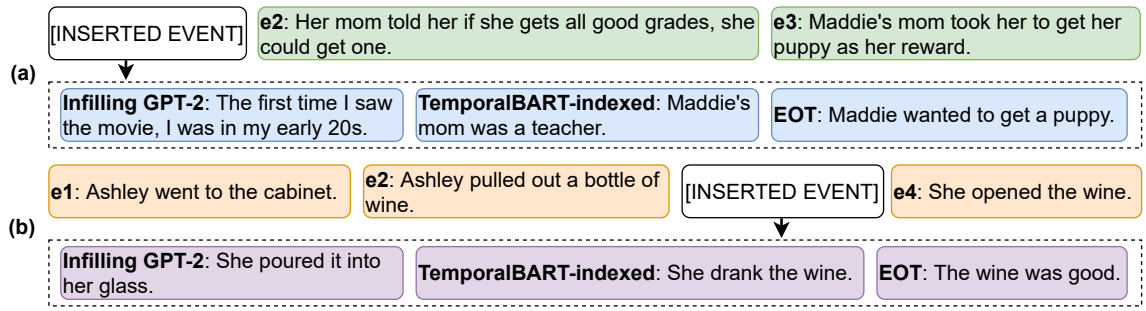

Figure 4: Two examples of events generated by infilling GPT-2, TemporalBART-indexed and our model EOT. The input events are in green and orange, while events generated by the models are in blue and purple.

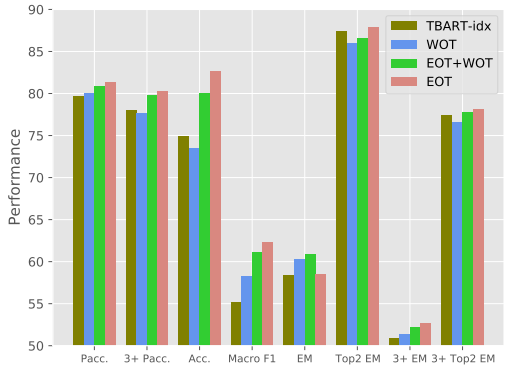

Figure 5: The performances of the 4 models on the 8 evaluation metrics which correspond to Table 1 and 2. The TBART-idx represents the TemporalBART-indexed model. Pacc means Pairwise Accuracy and 3+ means sequences with 3 or more events.

bines event and word-level OT loss and is called EOT+WOT.

Experimental results are shown in Figure 5, from which we can make the following observations:

(1) The WOT model achieves comparable results with the TemporalBART-indexed model and performs better on some metrics. This shows that imposing global sequence-level guidance via new supervision is effective, although the effect is not obvious, because it did not take into account the event-level matching.

(2) When event-level matching is considered, the EOT model achieves performance improvements on almost all metrics compared to the WOT model, and the EOT model outperforms the TemporalBART-indexed model on all metrics. This demonstrates that incorporating event-level matching can further improve the performance of optimal transport and outperform the baseline models.

(3) When we combine word-level with event-level OT, the performance is expected to get better. Unfortunately, the performance of EOT+WOT decreases compared with EOT. One possible reason is that word-level matching and event-level matching conflict with each other to some extent: two events may match well, but the words in the two events may not match each other.

5 Conclusion

In this paper, we consider a single generation model which can support inferences in the two related tasks. We introduce a novel method for the generation of temporally-ordered event sequences via Event Optimal Transport (EOT). Compared with the MLE-based Seq2Seq model, our approach has two advantages: (i) we treat the events in the sequence as modeling units and explicitly extract the semantics of the events; (ii) we directly match events in sequences to provide event sequence-level evaluation of the predicted results of the model. Experimental results show the superiority of our model on all evaluation datasets. Specifically, the accuracy is improved by 7.7%, and the Macro F1 is improved by 7.2% on one of the datasets.

Acknowledgements

This work is supported by the National Key Research and Development Program of China (No.2020AAA0106400), the National Natural Science Foundation of China (No.62176257, 61976211, 61922085). This work is also supported by the Strategic Priority Research Program of Chinese Academy of Sciences (Grant No.XDA27020200), the Youth Innovation Promotion Association CAS, and Yunnan Provincial Major Science and Technology Special Plan Projects (No.202202AD080004).

References

David Alvarez-Melis and Tommi Jaakkola. 2018. Gromov-wasserstein alignment of word embedding spaces. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

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

Liqun Chen, Guoyin Wang, Chenyang Tao, Dinghan Shen, Pengyu Cheng, Xinyuan Zhang, Wenlin Wang, Yizhe Zhang, and Lawrence Carin. 2020. Improving textual network embedding with global attention via optimal transport. In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 5193–5202. Association for Computational Linguistics (ACL).

Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2016. Knowledge-driven event embedding for stock prediction. In *Proceedings of coling 2016, the 26th international conference on computational linguistics: Technical papers*, pages 2133–2142.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

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

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR.

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.

Jianqiao Li, Chunyuan Li, Guoyin Wang, Hao Fu, Yuhchen Lin, Liqun Chen, Yizhe Zhang, Chenyang Tao, Ruiyi Zhang, Wenlin Wang, et al. 2020. Improving text generation with student-forcing optimal transport. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9144–9156.

Manling Li, Tengfei Ma, Mo Yu, Lingfei Wu, Tian Gao, Heng Ji, and Kathleen McKeown. 2021. Timeline summarization based on event graph compression via time-aware optimal transport. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6443–6456.

Zhongyang Li, Xiao Ding, and Ting Liu. 2018. Constructing narrative event evolutionary graph for script event prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4201–4207.

Shih-Ting Lin, Nathanael Chambers, and Greg Durrett. 2021. Conditional generation of temporally-ordered event sequences. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7142–7157, Online. Association for Computational Linguistics.

Shangwen Lv, Wanhui Qian, Longtao Huang, Jizhong Han, and Songlin Hu. 2019. Sam-net: Integrating event-level and chain-level attentions to predict what happens next. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6802–6809.

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

Allen Nie, Erin Bennett, and Noah Goodman. 2019. Dissent: Learning sentence representations from explicit discourse relations. In *Proceedings of the*

57th Annual Meeting of the Association for Computational Linguistics, pages 4497–4510.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Bras, Antoine Bosselut, and Choi Yejin. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. pages 794–805.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners.

Shichao Wang, Xiangrui Cai, Hongbin Wang, and Xiaojie Yuan. 2021. Incorporating circumstances into narrative event prediction. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4840–4849.

Zhongqing Wang, Yue Zhang, and Ching Yun Chang. 2017. Integrating order information and event relation for script event prediction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 57–67.

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

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 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.

Xianchao Wu, Ander Martinez, and Momo Klyen. 2018. Dialog generation using multi-turn reasoning neural networks. 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 2049–2059.

Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. 2020. A fast proximal point method for computing exact wasserstein distance. In *Uncertainty in Artificial Intelligence*, pages 433–453. PMLR.

Hongteng Xu, Wenlin Wang, Wei Liu, and Lawrence Carin. 2018. Distilled wasserstein learning for word embedding and topic modeling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 1723–1732.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. 2021. Vocabulary learning via optimal transport for neural machine translation. In *Proceedings of ACL 2021*.

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.

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.

Bo Zhou, Yubo Chen, Kang Liu, Jun Zhao, Jiexin Xu, Xiaojian Jiang, and Jinlong Li. 2021. Multi-task self-supervised learning for script event prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 3662–3666.