Augmentation, Retrieval, Generation: Event Sequence Prediction with a Three-Stage Sequence-to-Sequence Approach

Bo Zhou[1,2]**,Chenhao Wang**[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,chenhao.wang,yubo.chen,kliu,jzhao}@nlpr.ia.ac.cn
{jiexinx,jiangxiaojian,annielqx}@cmbchina.com

Abstract

Being able to infer possible events related to a specific target is critical to natural language processing. One challenging task in this line is *event sequence prediction*, which aims at predicting a sequence of events given a goal. Currently existing approach models this task as a *statistical induction* problem, to predict a sequence of events by exploring the similarity between the given goal and the known sequences of events. However, this statistical based approach is complex and predicts a limited variety of events. At the same time this approach ignores the rich knowledge of external events that is important for predicting event sequences. In this paper, in order to predict more diverse events, we first reformulate the event sequence prediction problem as a sequence generation problem. Then to leverage external event knowledge, we propose a three-stage model including augmentation, retrieval and generation. Experimental results on the event sequence prediction dataset show that our model outperforms existing methods, demonstrating the effectiveness of the proposed model.

1 Introduction

Inferring events related to a specific target is one of the capabilities pursued by natural language understanding, and it is also very helpful for other natural language processing (NLP) tasks, such as event extraction (Li et al., 2021), text summarization (Li and Zhang, 2021) and discourse understanding (Nie et al., 2019). A challenging task in this direction is event sequence prediction. Specifically, given a goal, the task aims to predict a sequence of events that fits the goal. Figure 1 shows an example of the event sequence prediction task. Given the goal *Buy a mobile phone*, it is expected to predict an event sequence, including four events *Determine the brand*, *Determine the price range*, *Select the brand series* and *Pay the bill*.

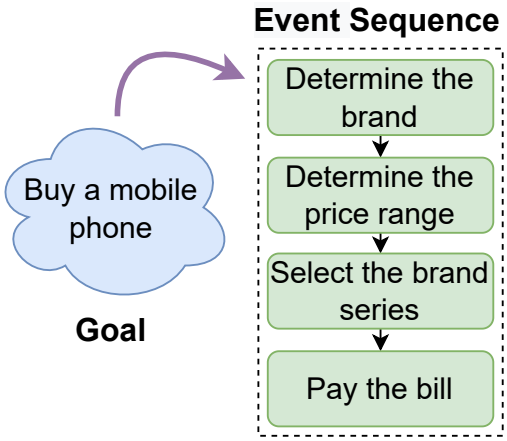

Figure 1: An example of an event sequence prediction task. Given the goal *Buy a mobile phone*, the predicted event sequence contains four events.

Although event sequence prediction is useful for many NLP tasks, it is currently understudied. Currently existing approach (Zhang et al., 2020) to this task is based on statistical model that infers new sequence of events by exploiting the similarity of the given goal to known sequences of events. Although experiments have shown that this approach is effective, it still has two shortcomings.

First, a limited variety of events are predicted. For example, when predicting the event sequence with the goal *Buying a mobile phone*, the method first collects the known event sequences with the goal *Buying a house, Buying a book, Repairing a mobile phone, Selling a mobile phone*, etc., and then predicts a new event sequence based on these event sequences. The events in the predicted sequence of events are similar to the events in the known sequence of events, or roughly, different combinations between these known events. Therefore, the types of events predicted by this method are limited.

Second, rich knowledge of external events is ignored. This method predicts a new event sequence based on known event sequences. If the

Proceedings of the 29th International Conference on Computational Linguistics, pages 1865–1874
October 12–17, 2022.

to-be-predicted event sequence differs greatly from the existing ones, it is difficult to predict a satisfactory event sequence. For example, if known sequences of events have neither Create-related nor Novel-related sequences of events, it is difficult to generate a satisfactory sequence of events given the goal *Create a novel*. At the same time, existing work (Guan et al., 2020; Li et al., 2022) shows the external event knowledge is helpful for natural language understanding and text generation, because these event knowledge bases contain rich commonsense knowledge that indicates the relationship between events. Therefore, leveraging these external event knowledge is important for event sequence prediction tasks.

In this paper, we first reformulate event sequence prediction as event sequence generation for the problem of limited types of generated events. Second, we propose a three-stage generation model for the problem of ignoring external event knowledge. As shown in Figure 2, our model consists of three steps, 1) Augmentation: First we design two pre-training tasks using the external event knowledge base, and then train the generation model with these two tasks. 2) Retrieval: Secondly, we match several similar event sequences from the existing event sequences through a given goal, and then train a scoring model to select the most similar event sequence. 3) Generation: Finally, we input the given goal and the retrieved most similar event sequences into the generation model to generate event sequence that meet the given goal. The experimental results show that our method outperforms previous methods, and the ablation study also verify the effectiveness of each module. In summary, our contributions in this paper are as follows:

- We reformulate the event sequence prediction task as a sequence generation task, which can generate more diverse events than method that infers new sequence of events from known events.
- We propose a three-stage augmentation, retrieval, generation model to tackle the event sequence prediction task, which can leverage external event knowledge to better generate sequence of events.
- Experiments and detailed analysis show that our model outperforms previous methods, proving its effectiveness.

2 Our Framework

Given a goal G, the event sequence prediction task needs to predict an event sequence $(e_1, e_2, ..., e_i, ..., e_n)$ that fits the goal, where e_i is the predicted i-th event. Here both the goal G and the event e_i are mainly composed of a verb and an object. Figure 2 shows the framework of our model which consists of three modules, namely augmentation, retrieval and generation. Below we introduce these modules separately.

2.1 Augmentation

In the augmentation module, we utilize the external event knowledge base ATOMIC [1] and design two tasks to pretrain our generation model.

To benefit the event sequence prediction task as much as possible, we expect the data of the external event knowledge base to be as similar as possible to the data of event sequence prediction, and we choose ATOMIC as a result. ATOMIC is a commonsense knowledge base with everyday knowledge tuples about entities and events in the form of *(head, relation, tail)* (Hwang et al., 2021). For example, the knowledge tuple *(move towards the door, isBefore, run out of the room)* indicates that the head event *move towards the door* occurs before the tail event *run out of the room*.

Using ATOMIC, we propose two pre-training tasks that match the event sequence prediction objective as closely as possible, one is Tail Event Generation (TG) and the other is Order Recovery (OR). Specifically, given a head event e_h, the TG task aims to generate a tail event e_t corresponding to e_h. The TG task empowers the generation model to generate events related to a given event, just as the model generates events related to a given goal in event sequence prediction task. Because the TG task generates only one event, it fails to capture the relationship between different generated events. So we propose the OR task, hoping to give the model the ability to distinguish the order between events. Specifically, given a head event e_h and its corresponding tail event e_t (Here we assume that e_h happens before e_t.), regardless of whether the input is ordered pair (e_h, e_t) or reversed pair (e_t, e_h), we want the model to output ordered pair (e_h, e_t).

Then we mix the data pairs of TG task with the data pairs of OR task for training. Specifically, for the OR task, given (e_h, e_t), we take (e_t, e_h) as in-

[1] https://mosaickg-graph-viz.apps.allenai.org/kg_atomic2020

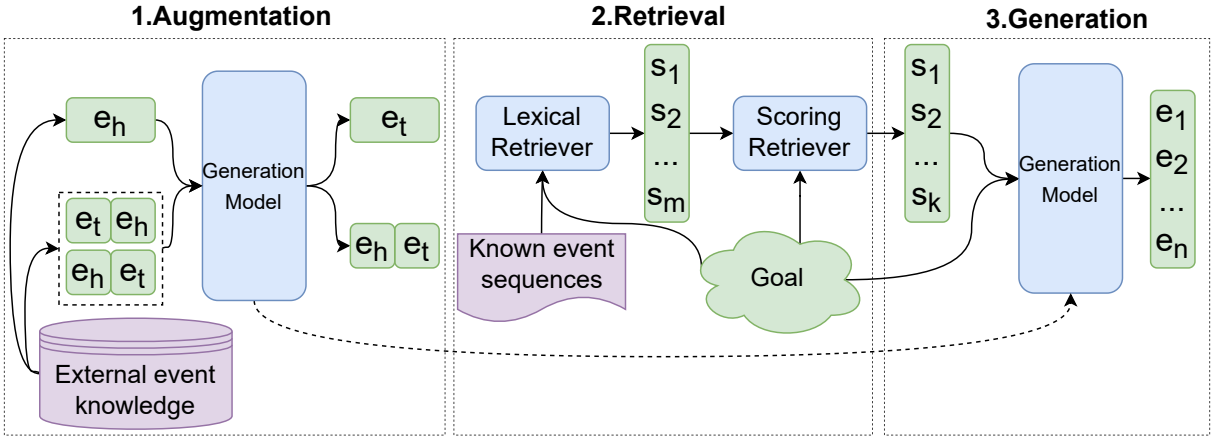

Figure 2: The framework of our model which consists of three modules, namely augmentation, retrieval and generation. Here e_h and e_t represent the head and tail events in the external event knowledge base, s_i and e_i denote the i-th event sequence and the generated i-th event, respectively.

put with probability p and (e_h, e_t) with probability $1-p$. We adopt the pre-trained encoder-decoder model T5-base (Raffel et al., 2021) as our base model and these mixed data pairs are used to train it with the following generation loss:

$$\mathcal{L}_{\mathrm{a}} = -\sum_{t=1}^{|y|} \log p\left(y_t \mid x, y_{<t}\right) \tag{1}$$

where (x, y) is the input-output pair, which may be (e_h, e_t), $[(e_t, e_h), (e_h, e_t)]$ or $[(e_h, e_t), (e_h, e_t)]$.

Because not all relations in ATOMIC are suitable for training our two tasks, we only use some relations in ATOMIC. Specifically, for the TG task, we use the *HasSubEvent* relation (The tail event is a step within the larger head event.) and the *xNeed* relation (The tail event is the condition of the head event.). For the OR task, we use the *isBefore* relation (The head event happens before the tail event.) and the *isAfter* relation (The head event happens after the tail event.). Table 1 shows the relations used by the two tasks and the corresponding examples.

At the same time, we point out that it is also possible to choose other event knowledge bases and design other pre-training tasks for event sequence prediction, which we leave as future work.

2.2 Retrieval

When predicting an event sequence that fits a given goal, it is helpful to refer to known event sequences related to the given goal. Therefore, in the retrieval module, we first use the lexical retriever to roughly retrieve m event sequences related to the given goal from the known event sequences, and then use the scoring retriever to further retrieve k most relevant event sequences from the m event sequences.

2.2.1 Lexical Retriever

Given a goal $G = (v, o)$, where v is the verb of the goal and o is the object of the goal. The lexical retriever returns event sequences whose goals contain either v or o. We denote these returned event sequences as $\{s_1, s_2, \cdots, s_m\}$, and the scoring retriever will then further select from these event sequences the ones most similar to the given goal.

2.2.2 Scoring Retriever

The scoring retriever scores each event sequence returned by the lexical retriever, and the top-k with the highest scores are used as the final retrieved event sequences. We train a BERT (Devlin et al., 2019) model as our scorer. Specifically, for each goal G, we take it and its corresponding event sequence as a positive pair. At the same time, we randomly sample an event sequence under another goal from the training set to form a negative sample pair with G.

Given a goal G and an event process s_i, we first concatenate the two as input to BERT:

$$X = [CLS]\ G\ [SEP]\ s_i\ [SEP] \tag{2}$$

here $[CLS]$ and $[SEP]$ are BERT's classification token and separation token, respectively.

We feed X into BERT and then feed the hidden layer vector $\boldsymbol{h}$ corresponding to CLS into a feedforward network:

$$s = \sigma(W\boldsymbol{h} + b) \tag{3}$$

Pretraining Task	Relation	Head	Tail
Tail Event Generation	HasSubEvent	play joker	shuffle cards
	xNeed	maintain good health	avoid cigarettes
Order Recovery	isBefore	go bike riding	take some photos of the scenery
	isAfter	move towards the door	get up from the table

Table 1: The relations used by the two pretraining tasks and the corresponding examples.

here the σ is the sigmoid function. We train the scorer by the cross entropy loss for binary classification. The s is taken as the score for each event sequence returned by the lexical retrieval. We obtain the top-k event sequences with the highest scores as the final retrieval result, denoted as $\{s_1, s_2, \cdots, s_k\}$.

2.3 Generation

In the generation module, we utilize the given goal and the event sequences retrieved by the retrieval module to generate the event sequence that matches the given goal.

Our generation model is also based on the T5-base model, sharing parameters with the T5 model in the augmentation module. Given the target G, we concatenate it with the retrieved sequence of events $\{s_1, s_2, \cdots, s_k\}$ as the input to the generation model and the sequence of events corresponding to G as the output y, and then optimize the following generation loss:

$$\mathcal{L}_{\mathrm{g}} = -\sum_{t=1}^{|y|} \log p\left(y_t \mid G, \{s_1, s_2, \cdots, s_k\}, y_{<t}\right) \tag{4}$$

3 Experiments

3.1 Settings

3.1.1 Dataset and Metrics

For the event sequence prediction task, we use the dataset released in Zhang et al. (2020). This dataset is based on WikiHow [2], an online wiki-style publication containing many sequences of events that accomplish specific goals. Following Zhang et al. (2020), the numbers of training, validation and test sets are 12,185, 1,316 and 1,316, respectively.

For the two pre-training tasks of tail event generation and order recovery, as mentioned earlier, we use the data corresponding to the four relations *HasSubEvent*, *xNeed*, *isBefore* and *isAfter* in ATOMIC. Because pre-training does not require test data, we end up with 152,673 and 16,964 data for training and validation sets, respectively.

We use the same metrics as the work (Zhang et al., 2020): E-ROUGE1 and E-ROUGE2. Similar to the commonly used ROUGE1 and ROUGE2, E-ROUGE1 and E-ROUGE2 calculate the percentage of the events or ordered event pairs in the predicted event sequence which are covered by the reference event sequences. Two covering standards *String Match* and *Hypernym Allowed* are used to evaluate the results. The former indicates that the words in the predicted event or event pair should be the same as in the references. The latter means that the hypernyms of the words in the predicted event or event pair should be the same as the hypernyms of the words in the references. Finally, two kinds of settings are included in the evaluation. One is basic setting: evaluate events based on verbs. The other is advanced setting: evaluate events based on all words.

3.1.2 Baselines

We compare with the following baseline models, which are used in Zhang et al. (2020):

- **Random**. Given a goal, the event sequence is randomly generated. This can be regarded as a lower bound on the performance of event sequence prediction.
- **GRU** (Sutskever et al., 2014). This is based on the GRU model, but the generation unit is changed from words to events. Events are represented by the pre-trained word embedding GloVe (Pennington et al., 2014) or the language model RoBERTa-base (Liu et al., 2019), denoted as **GRU (GloVe)** and **GRU (RoBERTa)** respectively.
- **Top one process**. Given a goal, such methods take the most similar sequence of events as the predicted sequence of events. Three methods for measuring similarity are used here, namely token-level Jaccard coefficient, cosine similarity based on GloVe representations and cosine

[2] https://www.wikihow.com/Main-Page

(a) Basic Setting (evaluate events based on verbs)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
Random	2.9165	0.4664	23.5873	8.1089
GRU (GloVe)	5.0323	1.4965	27.8710	13.0946
GRU (RoBERTa)	4.5455	0.4831	28.0032	12.8502
Top one process (Jaccard)	8.8589	5.1000	28.6548	14.6231
Top one process (GloVe)	9.8797	5.1452	29.4203	13.6001
Top one process (RoBERTa)	9.2599	4.7390	30.6599	15.8417
APSI	14.8013	6.6045	36.1648	19.2418
Ours	**24.9551**	**11.2535**	**46.1999**	**23.5655**
Human	29.0189	15.2542	50.4647	29.4423

(b) Advanced Setting (evaluate events based on all words)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
Random	0.0000	0.0000	0.5104	0.0903
GRU (GloVe)	0.1935	0.0534	0.9677	0.1069
GRU (RoBERTa)	0.4870	0.0000	1.7857	0.2899
Top one process (Jaccard)	0.6562	0.2257	2.4797	0.5867
Top one process (GloVe)	0.8750	0.2106	2.8801	0.7372
Top one process (RoBERTa)	0.9479	0.3009	3.2811	0.9929
APSI	3.4988	0.4513	6.1611	1.1885
Ours	**6.0443**	**1.1142**	**10.7720**	**2.4513**
Human	11.6351	5.5905	18.0034	8.2695

Table 2: Comparison of our method with the best previous methods. The best performance is shown in bold.

similarity based on RoBERTa representations, which are denoted as **Top one process (Jaccard)**, **Top one process (GloVe)** and **Top one process (RoBERTa)** respectively.

- **APSI** (Zhang et al., 2020). This is a statistical model that exploits the similarity of the given goal to known sequences of events to infer new sequence of events.

- **Human**. Given a goal, the event sequence is generated by human, which can be regarded as an upper bound on the performance of event sequence prediction.

3.1.3 Implementation Details

We use T5-base as the base model for the augmentation module and the generation module. The batch size used in both modules is 32. The optimizer used by both modules is AdamW (Loshchilov and Hutter, 2018), and the learning rate is set to 1e-5. In the augmentation module, the model is trained for 15 epochs, and the number is 30 in the generation module. We use BERT-base as the base model for the scoring retriever, AdamW is chosen as the optimizer and the learning rate is set to 5e-5. We set the batch size to 64 and train the BERT-base model for 3 epochs. We experimentally choose the number k of event sequences retrieved by the scoring retriever to be 2. Both T5-base and BERT-base models are implemented through the Huggingface Transformer library (Wolf et al., 2020).

3.2 Overall Results

The overall results are shown in Table 2, from which we can see:

(1) Our model outperforms previous methods in both basic and advanced settings, both for *String Match* or *Hypernym Allowed*. This demonstrates that our method can generate event sequences that are more in line with the given goal.

(2) The performance of all methods is inferior to human performance, but the performance of our proposed method is close to human performance. The likely reason is that our method somewhat mimics the human process of generating event sequences. Humans first learn knowledge related to

(a) Basic Setting (evaluate events based on verbs)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
T5	21.4396	8.6654	44.1595	22.3597
T5+augmentation	21.6580	9.3587	44.9007	23.0440
T5+retrieval	23.0099	9.8411	45.0442	22.4762
Ours (T5+augmentation+retrieval)	24.9551	11.2535	46.1999	23.5655

(b) Advanced Setting (evaluate events based on all words)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
T5	5.4869	0.8742	9.8168	2.0505
T5+augmentation	5.6759	0.9442	10.3454	2.1537
T5+retrieval	5.8754	1.0814	10.6658	2.2291
Ours (T5+augmentation+retrieval)	6.0443	1.1142	10.7720	2.4513

Table 3: Ablation results on the test set of event sequence prediction. We use T5 as our base model.

event sequences. When given a goal, humans first search for similar event sequences, and then use these event sequences to generate event sequence that matches the goal.

(3) When using *Hypernym Allowed* instead of *String Match*, performance is greatly improved in both settings. The reason is that it is easier to predict similar words than to predict the same words to the answer. The better performance of our model over the previous methods on both *String Match* and *Hypernym Allowed* shows that our method is more able to generate accurate events, or at least, our model is more able to generate events similar to the answer than the previous methods. We also observe that the performance of the model in the advanced setting is inferior to that in the basic setting, suggesting that it is easier for the model to correctly predict the verb in an event than to correctly predict the entire event.

3.3 Ablation Study

Here we conduct ablation experiments to investigate the effect of various modules of our method. The results are shown in Table 3, from which we can observe:

(1) Our model (T5+augmentation+retrieval) outperforms the base T5 model on all metrics in both basic and advanced settings, indicating that the introduction of augmentation and retrieval modules can help improve the performance of event sequence prediction.

(2) When the basic model T5 is added with the augmentation module, the performance of the model in various metrics is also improved. This shows that the introduction of external event knowledge is helpful for event sequence prediction. By endowing the model the ability to generate relevant events and capture the relationship between events by the two pre-training tasks, we successfully inject external knowledge into the model.

(3) After adding the retrieval module, the performance of the base model T5 is improved on all metrics. This illustrates the importance of referring to known similar event sequences when generating event sequence that meets a given goal. Although large-scale pre-trained language models have been shown to possess some general world knowledge, it is necessary to introduce specific event knowledge by referring to known event sequences when generating new event sequence.

3.4 Effect of Pretraining Tasks

Here we study the effect of the two pre-training tasks Tail Event Generation (TG) and Order Recovery (OR). The results are shown in Table 4, from which we can see:

(1) Whether adding only Tail Event Generation or only Oder Recovery pre-training tasks, the performance of the model improves over models that do not utilize pre-training tasks. This suggests that both tasks can endow the model with the ability to facilitate event sequence prediction by implicitly injecting external event knowledge into the model.

(2) When two pre-training tasks are added at the same time, the performance of the model is further improved. This shows that the two pre-training

(a) Basic Setting (evaluate events based on verbs)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
No pretraining	23.0099	9.8411	45.0442	22.4762
Only Tail Event Generation	24.2118	10.8155	46.0440	23.5396
Only Order Recovery	24.9415	9.9171	45.0969	22.7561
Both	24.9551	11.2535	46.1999	23.5655

(b) Advanced Setting (evaluate events based on all words)

Model	String Match		Hypernym Allowed	
	E-ROUGE1	E-ROUGE2	E-ROUGE1	E-ROUGE2
No pretraining	5.8754	1.0814	10.6658	2.2291
Only Tail Event Generation	5.9299	1.1011	10.6725	2.3762
Only Order Recovery	6.0424	1.0905	10.6745	2.4341
Both	6.0443	1.1142	10.7720	2.4513

Table 4: The effect of the two pre-training tasks Tail Event Generation (TG) and Order Recovery (OR). No pre-training means only the base model with the retrieval module is used.

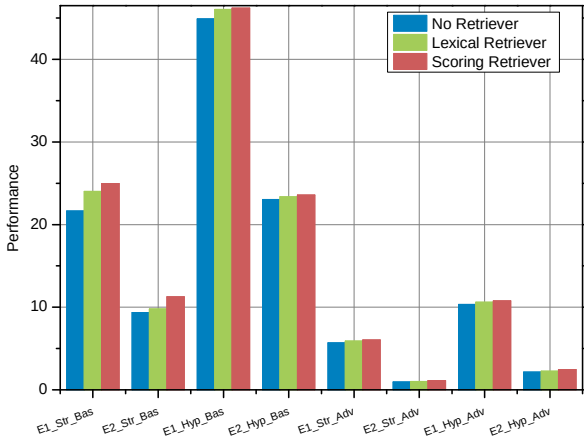

Figure 3: The effect of the lexical retriever and scoring retriever. The *E1_Str_Bas* represents E-ROUGE1 of String Match in the basic setting, and *E2_Hyp_Adv* represents E-ROUGE2 of Hypernym Allowed in the advanced setting. Other abbreviations have similar meanings.

tasks can coexist harmoniously, and at the same time endow the model with the ability to generate related events and capture the relationship between events, thereby jointly helping the event sequence generation.

3.5 Impact of Retrievers

Here we study the effect of the lexical retriever and scoring retriever in the retrieval module. When only the lexical retriever is used, we randomly select k event sequences from the results returned by the lexical retriever as the final retrieval result. The experimental results are shown in Figure 3, from which we can observe:

(1) When only the lexical retriever is used, the performance is improved compared to without the retrieval module. This shows that when generating event sequences, it is helpful to introduce specific event knowledge, i.e., existing event sequences, even if these event sequences are only related to a certain extent.

(2) When a scoring retriever is added, the performance of the model is further improved compared to using only the lexical retriever. This illustrates the necessity of using a scorer to select the sequences of events most similar to a given goal from the related sequences of events. So how to design a better scorer is an important issue, which we leave as future work.

3.6 Case Study

Figure 4 lists three examples of event sequences generated by our model and their corresponding answer event sequences. For the first example, two events *Open App* and *Delete Contact* in the sequence generated by our model are the same as the corresponding events in the answer sequence. Another event *Tap Icon*, although not the same as event *Select Contact* in the answer, expresses a similar meaning. This shows that our model can generate new event types. A similar situation exists for the other two examples.

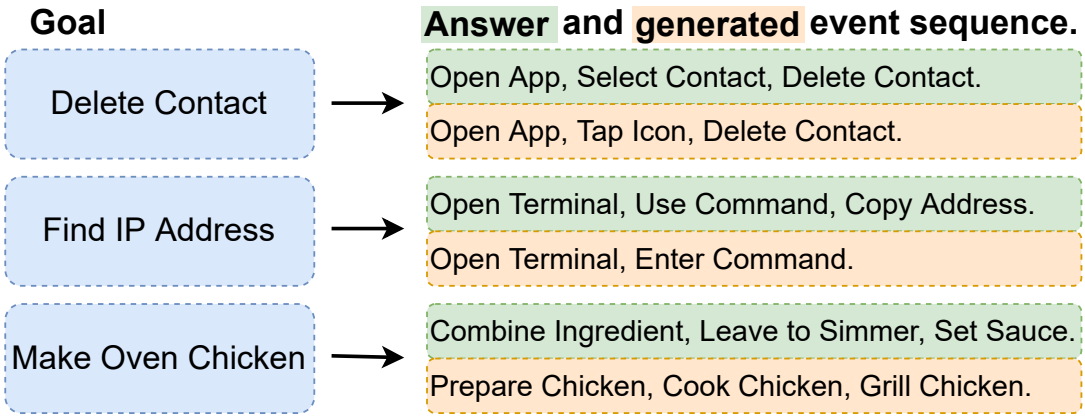

Figure 4: A comparison of the answer event sequence with the event sequence generated by our model.

4 Related Work

4.1 Script Event Prediction

Script event prediction aims to predict a correct subsequent event from a candidate list, given an ordered event sequence. The task is first proposed by Chambers and Jurafsky (2008), with a statistical model to predict the subsequent event by learning the cooccurrence between events. Neural networks are used in recent studies for script event prediction. In Pichotta and Mooney (2016), they shows that LSTM-based model outperforms previous cooccurrence-based models for script event prediction. A neural network is proposed in Granroth-Wilding and Clark (2016) which can learn word embedding and composition function simultaneously. To better model relatedness between events, Li et al. (2018) treats event chain as a sub-graph and leverage recurrent networks to capture the relatedness for predicting the event. The model in Lv et al. (2020) integrates external event knowledge and designs three methods to predict the subsequent event. A Transformer-based model is proposed in Bai et al. (2021) which integrates deep event-level and script-level information for script event prediction. To deal with the data insufficiency problem, Zhou et al. (2021) proposes a multi-task self-supervised model for script event prediction. Compared to script event prediction, the event sequence prediction considered in this paper is more challenging because multiple events need to be predicted instead of one, and there is no candidate list of events like script event prediction.

4.2 Commonsense Knowledge

Commonsense knowledge is an important resource for artificial intelligence, and it is also helpful for many natural language processing tasks, such as reading comprehension, question answering, and text generation. For reading comprehension, Mihaylov and Frank (2018) introduces a neural reading comprehension model, which leverages a key-value memory to integrate external commonsense knowledge. In Yang et al. (2019), attention mechanism is employed to select knowledge from external knowledge bases, which is then fused with BERT to do knowledge-aware predictions. For question answering, Ma et al. (2021) proposes a neuro-symbolic framework for zero-shot question answering, to transform knowledge resources into an effective form for pretraining models. A model containing relevance scoring and joint reasoning is proposed in Yasunaga et al. (2021) to form a joint graph connecting the QA context and KG, whose representations are updated through a graph neural network. For text generation, Guan et al. (2020) proposes a knowledge-enhanced model based on multi-task learning for commonsense story generation. Li et al. (2022) proposes a two-stage method to explicitly arrange the ensuing events in open-ended text generation. In this paper, we design two pre-training tasks to implicitly inject external knowledge into the model to help the model generate event sequence that meets a given goal.

5 Conclusion

In this paper, we first reformulate the event sequence prediction task as an event sequence generation task, which can generate a wider variety of events. We then propose a three-stage model including augmentation, retrieval, and generation in order to leverage an external event knowledge base to generate event sequences. Finally, exper-

imental results show that our model outperforms existing methods, demonstrating the effectiveness of the proposed method.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No.U1936207, No.61976211, No.62176257). This work is supported by the Key Research Program of the Chinese Academy of Sciences (Grant NO.ZDBS-SSW-JSC006), the independent research project of National Laboratory of Pattern Recognition (No.Z-2018013) and the Youth Innovation Promotion Association CAS.

References

Long Bai, Saiping Guan, Jiafeng Guo, Zixuan Li, Xiaolong Jin, and Xueqi Cheng. 2021. Integrating deep event-level and script-level information for script event prediction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9869–9878.

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

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.

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.

Jian Guan, Fei Huang, Zhihao Zhao, Xiaoyan Zhu, and Minlie Huang. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, 8:93–108.

Jena D Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. On symbolic and neural commonsense knowledge graphs.

Qintong Li, Piji Li, Wei Bi, Zhaochun Ren, Yuxuan Lai, and Lingpeng Kong. 2022. Event transition planning for open-ended text generation. *arXiv preprint arXiv:2204.09453*.

Quanzhi Li and Qiong Zhang. 2021. Twitter event summarization by exploiting semantic terms and graph network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 15347–15354.

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

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.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Shangwen Lv, Fuqing Zhu, and Songlin Hu. 2020. Integrating external event knowledge for script learning. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 306–315.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Yonatan Bisk, Eric Nyberg, and Alessandro Oltramari. 2021. Knowledge-driven data construction for zero-shot evaluation in commonsense question answering. In *35th AAAI Conference on Artificial Intelligence*.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 821–832.

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.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Karl Pichotta and Raymond Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2021. Exploring the limits of transfer learning with a unified text-to-text transformer (2019). *arXiv preprint arXiv:1910.10683*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, 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.

An Yang, Quan Wang, Jing Liu, Kai Liu, Yajuan Lyu, Hua Wu, Qiaoqiao She, and Sujian Li. 2019. Enhancing pre-trained language representations with rich knowledge for machine reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2346–2357, Florence, Italy. Association for Computational Linguistics.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qa-gnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1541–1550.

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.