Analogous Process Structure Induction for Sub-event Sequence Prediction

Hongming Zhang[1]*, **Muhao Chen**[2], **Haoyu Wang**[2], **Yangqiu Song**[1], **& Dan Roth**[2]
[1]Department of Computer Science and Engineering, HKUST
[2]Department of Computer and Information Science, UPenn
`{hzhangal, yqsong}@cse.ust.hk`
`{why16gzl, muhao, danroth}@seas.upenn.edu`

Abstract

Computational and cognitive studies of event understanding suggest that identifying, comprehending, and predicting events depend on having structured representations of a sequence of events and on conceptualizing (abstracting) its components into (soft) event categories. Thus, knowledge about a known process such as "*buying a car*" can be used in the context of a new but analogous process such as "*buying a house*". Nevertheless, most event understanding work in NLP is still at the ground level and does not consider abstraction. In this paper, we propose an Analogous Process Structure Induction (APSI) framework, which leverages analogies among processes and conceptualization of sub-event instances to predict the whole sub-event sequence of previously unseen open-domain processes. As our experiments and analysis indicate, APSI[1] supports the generation of meaningful sub-event sequences for unseen processes and can help predict missing events.

1 Introduction

Understanding events has long been a challenging task in NLP, to which many efforts have been devoted by the community. However, most existing works are focusing on procedural (or *horizontal*) event prediction tasks. Examples include predicting the next event given an observed event sequence (Radinsky et al., 2012) and identifying the effect of a biological process (i.e., a sequence of events) on involved entities (Berant et al., 2014). These tasks mostly focus on predicting related events in a procedure based on their statistical correlations in previously observed text. As a result, understanding the meaning of an event might

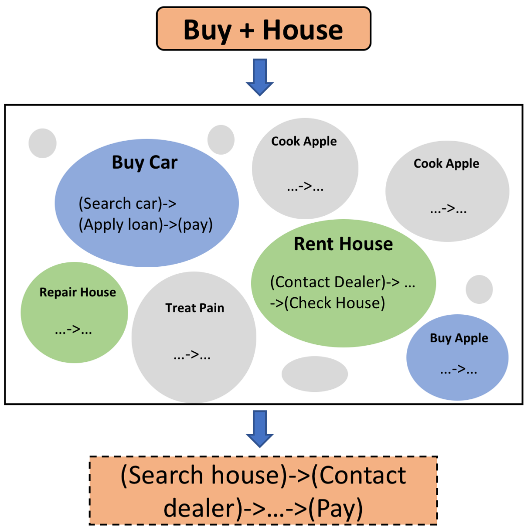


Figure 1: An illustration of leveraging known processes to predict the sub-event sequence of a new process.

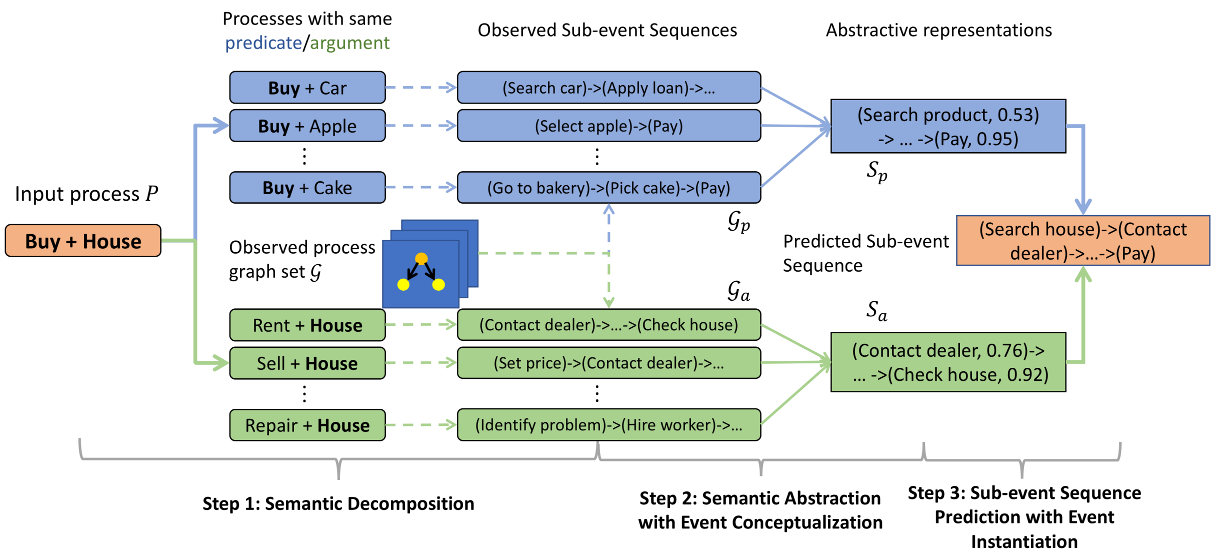
not be crucial for these *horizontal* tasks. For example, simply selecting the most frequently co-occurring event can offer acceptable performance on the event prediction task (Granroth-Wilding and Clark, 2016).

Computational and cognitive studies (Schank and Abelson, 1977; Zacks and Tversky, 2001) suggest that inducing and utilizing the hierarchical structure[2] of events is a crucial component of how humans understand new events and can help many aforementioned *horizontal* event prediction tasks. Consider the example in Figure 1. Assume that one has never bought a house, but is familiar with how to "buy a car" and "rent a house"; referring to analogous steps in these two relevant processes would still provide guidance for the target process of "buy a house". Motivated by this hypothesis, our work proposes to directly evaluate a model's event understanding ability. We define this as the

* This work was done when the first author was visiting the University of Pennsylvania.

[1]Code is available at: `http://cogcomp.org/page/publication_view/910`.

[2]The original paper refers to the knowledge about processes and their sub-events as event schemata.



Processes with same predicate/argument
Observed Sub-event Sequences
Abstractive representations
Buy + Car
Buy + Apple
Buy + Cake
(Search car)->(Apply loan)->...
(Select apple)->(Pay)
(Go to bakery)->(Pick cake)->(Pay)
(Search product, 0.53) -> ... ->(Pay, 0.95)
S_p
$\mathcal{G}_p$
Input process P
Buy + House
Observed process graph set $\mathcal{G}$
Predicted Sub-event Sequence
(Search house)->(Contact dealer)->...->(Pay)
$\mathcal{G}_a$
S_a
Rent + House
Sell + House
Repair + House
(Contact dealer)->...->(Check house)
(Set price)->(Contact dealer)->...
(Identify problem)->(Hire worker)->...
(Contact dealer, 0.76)-> ... ->(Check house, 0.92)
Step 1: Semantic Decomposition
Step 2: Semantic Abstraction with Event Conceptualization
Step 3: Sub-event Sequence Prediction with Event Instantiation

Figure 2: Demonstration of the proposed APSI framework. Given a target process P, we first decompose its semantics into two dimensions (i.e., predicate and argument) by grouping processes that share a predicate or an argument. For each such group of processes, we then leverage the observed process graphs $\mathcal{G}$ to generate an abstract and probabilistic representation for their sub-event sequences. In the last step, we merge them with an instantiation module to produce the sub-event sequence of P.

ability to identify *vertical* relations, that is, to predict the sub-event sequence of a new process[3]. We require models to generate the sub-event sequence for a previously unobserved process given observed processes along with their sub-event sequences, which we refer to as “the observed process graphs" in the rest of this paper. This task is more challenging than “conventional" event predictions tasks, since it requires the *generation* of a sub-event sequence given a new, previously unobserved, process definition.

To address this problem, we propose an Analogous Process Structure Induction (APSI) framework. Given a new process definition (e.g., ‘buy a house’), we first decompose it into two dimensions: predicate and argument. For each of these, we collect a group of processes that share the same predicate (i.e., ‘buy-*ARG*’) or same argument (i.e., ‘*PRE*-house’), and then induce an abstract and probabilistic sub-event representation for each group. Our underlying assumption is that processes that share the same predicate or argument could be analogous to each other, and thus could share similar sub-event structures. Finally, we merge these two abstract representations, using an instantiation module, to predict the sub-event structure of the target process. By doing so, we only need a small number of analogous processes (as we show, 20, on average) to generate unseen sub-events for the target process. Intrinsic and extrinsic evaluations show that APSI outperforms all baseline methods and can generate meaningful sub-event sequences for unseen processes, which are proven to be helpful for predicting missing events.

[3]A process is a more coarse-grained event by itself. We use this term to distinguish it from sub-events.

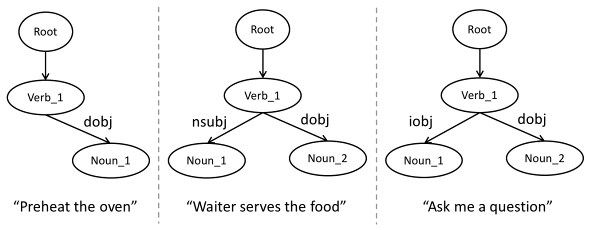


Figure 3: Examples of Sub-Event Representations.

The rest of the paper is organized as follows. Section 2 introduces the Analogous Process structure induction (APSI) framework. Section 3 describes our intrinsic and extrinsic evaluation, demonstrating the effectiveness of APSI and the quality of the induced process knowledge. We discuss related works in Section 4 and conclude this paper with Section 5.

2 The APSI Framework

Figure 2 illustrates the details of the proposed APSI framework. Given an unseen process P, a target sub-event sequence length k, and a set of observed process graphs $\mathcal{G}$, the task is to predict a k-step sub-event sequence $[E'_1, E'_2, ..., E'_k]$ for P. Each process graph $G \in \mathcal{G}$ in the input contains a process definition P^G and an n-step temporally ordered sub-event sequence $[E^G_1, E^G_2, ..., E^G_n]$. We assume that each process P is described as a combination of a predicate and an argument (e.g., 'buy+house') and each sub-event $E \in \mathcal{E}$ is given as verb-centric dependency graph as used in (Zhang et al., 2020b) (see examples in Figure 3). In APSI, we decompose the target process into two dimensions (i.e., predicate and argument). For each target process, we collect a group of observed process graphs that share either the predicate or the argument with the target process; we assume that processes in these groups have sufficient information for predicting the structure of the target process. We then leverage an event conceptualization module to induce an abstract representation of each process group. Finally, we merge the two abstract, probabilistic representations and instantiate it to generate a ground sub-event sequence as the final prediction. Detailed descriptions of APSI components are introduced as follows.

2.1 Semantic Decomposition

Each process definition P is given as a predicate and its argument, which we term below the two "dimensions" of the process definition. We then collect all process graphs in $\mathcal{G}$ that have the same predicate as P into $\mathcal{G}_p$ and those that have the same argument into $\mathcal{G}_a$. We assume that these two sets provide the information needed to generate an abstract process representation that would guide the instantiation of the event steps for P.

2.2 Semantic Abstraction

The goal of the semantic abstraction step is to acquire abstract representations S_p and S_a for $\mathcal{G}_p$ and $\mathcal{G}_a$ respectively, to help transfer the knowledge from the grounded observed processes to the target new process. To do so, we first need to conceptualize observed sub-events in $\mathcal{G}_p$ and $\mathcal{G}_a$ (e.g., "eat an apple") to a more abstract level (e.g., "eat fruit"). Clearly, each event could be conceptualized to multiple abstract events. For example, "eat an apple" can be conceptualized to "eat fruit" but also to "eat food", and the challenge is to determine the appropriate level of abstraction. On one hand, the conceptualized event cannot be too general, as we do not want to lose touch with the original event, and, on the other hand, if it is too specific, we will not aggregate enough instances of sub-events into it, thus we will have difficulties transferring knowledge to the new unseen process. To automatically achieve the balance between these conflicting requirements and select the best abstract event for each observed sub-event, we model it as a weighted mutually exclusive set cover problem (Lu and Lu, 2014) and propose an efficient algorithm, described below, to solve it. We then merge the repeated conceptualized events and determine their relative positions.

2.2.1 Modeling Event Conceptualization

For each event E, we first identify all potential events that it can be conceptualized to. If two sub-events E_1 and E_2 can be conceptualized to the same event C, we place E_1 and E_2 into the set $\mathcal{E}_C$. To qualitatively guide the abstraction process we introduce below a notion of *semantic loss* that we incur as we move up to more abstract representations. To measure the semantic loss during the conceptualization, we assign weight to each set:

$$W(\mathcal{E}_C) = \frac{1}{\sum_{E \in \mathcal{E}_C} F(E, C)}, \qquad (1)$$

where $F(E, C)$ is a scoring function, defined below in Eq. 2, that captures the amount of "semantic details" preserved due to abstracting from E to C. With this definition, the event conceptualization problem can be formalized as finding exclusive[4] sets (such as C) that cover all observed events with minimum total weight. In the rest of this section, we first introduce how to collect potential conceptualized events for each E, how we define F, and how we solve this discrete optimization problem.

Identifying Potential Conceptualizations Assume that sub-event E contains m words $w^E_1, w^E_2, ..., w^E_m$, each corresponds to a node in Figure 3; for each of these, we can retrieve a list of hypernym paths from WordNet (Miller, 1998). For example, given the word "house", WordNet returns two hypernym paths[5]: (1) "house"$\rightarrow$"building"$\rightarrow$"structure"$\rightarrow$...; (2) "house"$\rightarrow$"firm"$\rightarrow$"business"$\rightarrow$.... As a result, we can find $\prod_{w \in E} L(w)$ potential conceptualized events for E, where $L(w)$ is the number of w's hypernyms. We denote the potential conceptualized event set for E as $\mathcal{C}_E$ and the overall set as $\mathcal{C}$.

[4]No sub-event can appear in two selected sets.

[5]We omit the synset number for clear representation.

Algorithm 1 Event Conceptualization

INPUT: Set of events $\mathcal{E}$. Each $\mathbf{E} \in \mathcal{E}$ is associated with a set of potential conceptualization events $\mathcal{C}_E$. The overall conceptualized event set $\mathcal{C}$.

```
 1: Initialize event partition set $\mathcal{P} := \emptyset$.
 2: while $\mathcal{E} \neq \emptyset$ do
 3:     for Each $E \in \mathcal{E}$ do
 4:         for Each $C \in \mathcal{C}_E$ do
 5:             $\mathcal{E}_C := \emptyset$.
 6:             Compute $F(E, C)$ using Eq. (2).
 7:         end for
 8:     end for
 9:     for Each $C \in \mathcal{C}$ do
10:         for Each $E \in \mathcal{E}$ do
11:             if $C \in \mathcal{C}_E$ then
12:                 $\mathcal{E}_C := \mathcal{E}_C \cup \{E\}$.
13:             end if
14:         end for
15:         Compute $W(\mathcal{E}_C)$ using Eq. (1).
16:     end for
17:     Select $\mathcal{E}_{C_{min}}$ with the minimum $W$ score.
18:     $\mathcal{E} := \mathcal{E} \setminus \mathcal{E}_{C_{min}}$
19:     $\mathcal{P} := \mathcal{P} \cup \{\mathcal{E}_{C_{min}}\}$.
20: end while
```

OUTPUT: Partition of n event subsets $\mathcal{P} = \{\mathcal{E}_1, \mathcal{E}_2, ..., \mathcal{E}_n\}$, where each subset $\mathcal{E}_i$ corresponds to a unique conceptualized event C_i.

Conceptualization Scoring As mentioned above, for each pair of a sub-event E and its potential conceptualization C, we propose a scoring function $F(E, C)$ to measure how much "semantic information" is preserved after the conceptualization. Motivated by Budanitsky and Hirst (2006) and based on the assumption that the more abstract the conceptualized event is, the more semantic details are lost, we define $F(E, C)$ to be:

$$F(E, C) = \prod_{i=1}^{m} w^{D(w_i^E, w_i^C)}, \tag{2}$$

where $D(w_i^E, w_i^C)$ is the depth from w_i^E to w_i^C on the taxonomy path, and w is a hyper-parameter[6] measuring how much "semantics" is preserved following each step of the conceptualization.

Conceptualization Assignment Now we are able to model the procedure of finding proper conceptualized events as a weighted mutually exclusive set cover problem. Note that this is an NP-complete problem and requires a prohibitive computational cost to obtain the optimum solution (Karp, 1972). To obtain an efficient solution that is empirically sufficient for assigning conceptualized events with reasonable amount of instances, we develop a greedy procedure as described in Algorithm 1. For each retrieved process graph set $\mathcal{G}_p$ or $\mathcal{G}_a$, we collect all its sub-events as $\mathcal{E}$ and use it as the input for the conceptualization algorithm. In each iteration, we first compute the conceptualization score F for all the (E, C) pairs and then compute the weight score for all conceptualization sets $\mathcal{E}_C$. After selecting the set with minimum weight, $\mathcal{E}_{C_{min}}$, we remove all the events covered by it from $\mathcal{E}$ and repeat the process until no event is left. After the conceptualization, we merge sub-events that are conceptualized to the same event and represent them with the resulting conceptualized event C, whose weight is defined to be $\overline{W}(C) = \frac{1}{W(\mathcal{E}_C)}$. Compared with the naive algorithm, which first expands all possible subsets (i.e., it includes all subsets of $\mathcal{E}_C$ for all C) and then leverages the sort and filter technique to select the final subsets, we reduce the time complexity from $O(|\mathcal{C}| \cdot |\mathcal{E}|^2)$ to $O(n \cdot |\mathcal{C}| \cdot |\mathcal{E}|)$, where n is the number of conceptualized events and is typically much smaller than $|\mathcal{E}|$.

[6]In practice, we use two separate hyper-parameters w_v and w_n for verbs and nouns, respectively.

2.2.2 Conceptualized Event Ordering

After conceptualizing and merging all sub-events, we need to determine their loosely temporal order (e.g., whether they typically appear at the beginning or the end of these sub-event sequences). Let the set of selected conceptualized events be $\mathcal{C}^*$. For each $C \in \mathcal{C}^*$, we define its order score $T(C)$, indicating how likely C is to appear first, as:

$$T(C) = \sum_{C' \in \mathbf{C}^*} \theta(\sum_{E_C \in \mathcal{E}_C} \sum_{E_{C'} \in \mathcal{E}_{C'}} t(E_C, E_{C'}) - t(E_{C'}, E_C)), \tag{3}$$

where θ is the unit step function and $t(E_C, E_{C'})$ represents how many times E_C appears before $E_{C'}$ in an observed process graph.

2.3 Sub-event Sequence Prediction

In the last step, we leverage the two abstract representations we got for the predicate and argument of the target process definition to predict its final sub-events. To do so, we propose the following instantiation procedure. We are given the abstract representations S_p and S_a, for the predicate and argument, respectively. Each is a set of conceptualized events associated with weights and order scores. For each conceptualized event $C_p \in S_p$, using each event $C_a \in S_a$, we can generate a new instantiated event $\hat{C}_p$. For example, if C_p is "cut fruit" and C_a is 'buy an apple', then our model would create the new event "cut an apple". Specifically, for each $w \in C_p$, if we can find a word $\hat{w}$ such that $\hat{w}$ is a hyponym of w, we will replace w with $\hat{w}$ and repeat this process until no hyponym can be detected in C_p. We denote the generated event by $\hat{C}_p$. To account for the semantic loss during the instantiation procedure, we define the weight and order score of $\hat{C}_p$ as follows:

$$\hat{W}(\hat{C}_p) = \overline{W}(C_p) \cdot F(\hat{C}_p, C_p) \cdot \frac{\sum_{C'_a \in S_a} \overline{W}(C'_a)}{\overline{W}(C_a)} \quad (4)$$

$$\hat{T}(\hat{E}_p) = T(C_p) \cdot F(\hat{C}_p, C_p) \cdot \frac{\sum_{C'_a \in S_a} \overline{W}(C'_a)}{\overline{W}(C_a)}, \quad (5)$$

Similarly, we apply the same procedure to C_a with C_p, and denote the resulted event $\hat{C}_a$. We then repeatedly merge instantiated events by summing up their weights and averaging their order scores. In the end, we select top k sub-events based on the weights and sort them based on the order score as the sub-event sequence prediction.

3 Evaluation

In this section, we conduct intrinsic and extrinsic evaluations to show that APSI can generate meaningful sub-event sequences for unseen processes, which can help predict the missing events.

3.1 Dataset

We collect process graphs from the WikiHow website[7] (Koupaee and Wang, 2018). In WikiHow, each process is associated with a sequence of temporally ordered human-created steps. For each step, as shown in Figure 3, we use the tool released by ASER (Zhang et al., 2020b) to extract events and construct the process graphs. We select all processes, where each step has one and only one event, and randomly split them into the train and test data. As a result, we got 13,501 training process graphs and 1,316 test process graphs[8], whose average sub-event sequence length is 3.56.

3.2 Baseline Methods

We compare with the following baseline methods:
Sequence to sequence (Seq2seq): One intuitive solution to the sub-event sequence prediction task would be modeling it as a sequence to sequence problem, where the process is treated as the input and the sub-event sequence the output. Here we adopt the standard GRU-based encoder-decoder framework (Sutskever et al., 2014) as the base framework and change the generation unit from words to events. For each process or sub-event, we leverage pre-trained word embeddings (i.e., GloVe-6b-300d (Pennington et al., 2014)) or language models (i.e., RoBERTa-base (Liu et al., 2019)) as the representation, which are denoted as Seq2seq (GloVe) and Seq2seq (RoBERTa).
Top One Similar Process: Another baseline is the "top one similar process". For each new process, we can always find the most similar observed process. Then we can use the sub-event sequence of the observed process as the prediction. We employ different methods (i.e., token-level Jaccard coefficient or cosine similarity of GloVe/RoBERTa process representations) to measure the process similarity. We denote them as Top one similar process (Jaccard), (GloVe), and (RoBERTa), respectively.

For each process, we also present a randomly generated sequence and a human-generated sequence[9] as the lower-bound and upper-bound for sub-event sequence prediction models.

3.3 Intrinsic Evaluation

We first present the intrinsic evaluation to show the quality of the predicted sub-event sequences of unseen processes. For each test process, we provide the process name and the sub-event sequence length[10] to evaluated systems and ask them to generate a fixed-length sub-event sequence.

3.3.1 Evaluation Metric

Motivated by the ROUGE score (Lin, 2004), we propose an event-based ROUGE (E-ROUGE) to evaluate the quality of the predicted sub-event sequence. Specifically, similar to ROUGE, which evaluates the generation quality based on N-gram token occurrence, we evaluate how much percentage of the sub-event and time-ordered sub-event pairs in the induced sequence is covered by the human-provided references. We denote the evaluation over single event and event pairs as E-ROUGE1 and E-ROUGE2, respectively. We also provide two covering standards to better understand the prediction quality: (1) "String Match": all words in the predicted event/pairs must be the same as the referent event/pairs; (2) "Hypernym Allowed": the predicted and referent event must

[7]https://www.wikihow.com.

[8]We do not need a development set because the proposed solution APSI is not a learning-based method.

[9]The human-generated sequence is randomly selected from the WikiHow and excluded during the evaluation.

[10]We select the majority length of all references.

Model	String Match E-ROUGE1	String Match E-ROUGE2	Hypernym Allowed E-ROUGE1	Hypernym Allowed E-ROUGE2
Random	2.9165	0.4664	23.5873	8.1089
Seq2seq (GloVe)	5.0323	1.4965	27.8710	13.0946
Seq2seq (RoBERTa)	4.5455	0.4831	28.0032	12.8502
Top one similar process (Jaccard)	8.8589	5.1000	28.6548	14.6231
Top one similar process (GloVe)	9.8797	5.1452	29.4203	13.6001
Top one similar process (RoBERTa)	9.2599	4.7390	30.6599	15.8417
Analogous Process Structure Induction (APSI)	**14.8013**	**6.6045**	**36.1648**	**19.2418**
Human	29.0189	15.2542	50.4647	29.4423

(a) Basic Setting (for each sub-event, we only predict and evaluate the verb)

Model	String Match E-ROUGE1	String Match E-ROUGE2	Hypernym Allowed E-ROUGE1	Hypernym Allowed E-ROUGE2
Random	0.0000	0.0000	0.5104	0.0903
Seq2seq (GloVe)	0.1935	0.0534	0.9677	0.1069
Seq2seq (RoBERTa)	0.4870	0.0000	1.7857	0.2899
Top one similar process (Jaccard)	0.6562	0.2257	2.4797	0.5867
Top one similar process (GloVe)	0.8750	0.2106	2.8801	0.7372
Top one similar process (RoBERTa)	0.9479	0.3009	3.2811	0.9929
Analogous Process Structure Induction (APSI)	**3.4988**	**0.4513**	**6.1611**	**1.1885**
Human	11.6351	5.5905	18.0034	8.2695

(b) Advanced Setting (for each sub-event, we predict and evaluate all words)

Table 1: Intrinsic evaluation results of the induced process structures. On average, we have 1.7 human-generated sub-event sequences as the references for each test process. Best performing models are marked with the bold font.

have the same dependency structure, and for the words on the same graph position, they should be the hypernym of or same as each other. For example, if the referent event is "eat apple" and the predicted event is "eat fruit", we still count it as a match. The "String Match" setting is stricter, but the "Hypernym Allowed" setting also has its unique value to help better understand if our system is predicting relevant sub-events.

3.3.2 Implementation Details

In terms of training, we set both w_v and w_n to be 0.5 for our model. For the seq2seq baselines, we set the learning rate to be 0.001 and train the models until they converge on the training data. All other hyper-parameters following the original paper. In terms of the evaluation, we also provide two settings. (1) Basic: we follow previous works (Glavas et al., 2014) to predict and evaluate events based on verbs; (2) Advanced: we predict and evaluate events based on all words.

3.3.3 Result Analysis

We show the results in Table 1. In general, there is still a notable gap between current models' performance and human performance, but the proposed APSI framework can indeed generate sufficiently relevant sub-events. For example, if we only consider the verb. Even in the string match setting, 14.8% of the predicted event and 6.6% of the ordered event pairs are covered by the references, which is much better than the random guess and nearly half of the performance of human beings. If hypernym is allowed, 36% and 19% of the predicted event and event pairs are covered. Besides that, if we take all words in the event into consideration, the task becomes more challenging. Specifically, even human can only achieve 11.63 E-ROUGE1 and 5.59 E-ROUGE2, which suggests that low scores achieved by current models are probably due to the limitation of the current dataset (e.g., on average, we only have 1.7 references for each test process). If more references are provided, the performance of all models will also increase. In the rest of the intrinsic evaluation, we present more detailed analysis based on the advanced setting (string match) and a case study to help better understand the performance of APSI.

3.3.4 Effect of the Instantiation Module

One key step in our framework is how to leverage the two abstract representations to predict the fi-

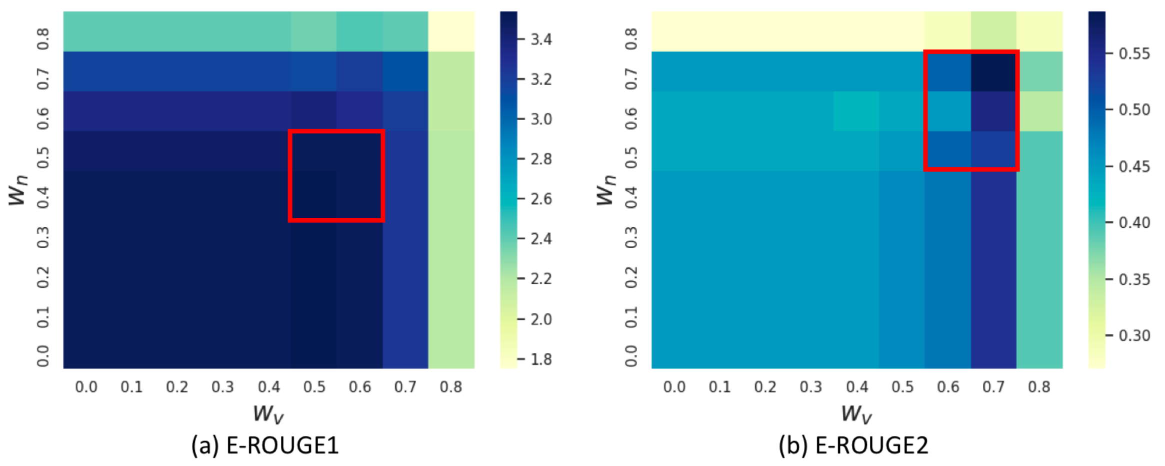


Figure 4: Hyper-parameter influence on the quality of APSI generated sub-event sequences. For both w_v and w_n, 0 indicates no conceptualization and the larger the value, the deeper the conceptualization is. Best performing ranges are marked with red boxes, which indicate that the suitable conceptualization level is the key to APSI's success.

Model	E-ROUGE1	E-ROUGE2
Simple Merge	2.5884	0.4062
Normalized	2.2238	0.3611
APSI (Instantiation)	**3.4988**	**0.4513**

Table 2: Performance of different merging methods.

Process Name:	**Treat Pain**
References:	('learn cause'->'identify symptom'->'see doctor')
	('identify cause'->'learn injury'->'recognize symptom'->`recognize symptom')
APSI Prediction:	('identify symptom'->'see doctor'->'recognize symptom'->'take supplement')

Figure 5: Case Study. We mark the covered and not covered predictions with green and red colors.

nal sub-event sequence. In APSI, we propose an instantiation module, which jointly leverages the two representations to generate detailed events. To show its effect, we compare it with two other options: (1) Simple Merge: Merge two representation and select the top k sub-events based on the weight; (2) Normalized: First normalize the weight of all sub-events based on each representation and then select the top k sub-events.

From the result in Table 2, we can see that due to the imbalanced distribution of the two representations, simply choosing the most weighted sub-events is problematic. On average, for each predicate, we can collect 18.04 processes, while we can only collect 1.92 processes for each argument. As a result, the sub-events in the predicate representation typically have a larger weight. Thus if we simply merge them, most of the predicted sub-events will come from the predicate representation. Ideally, the "normalized" method can eliminate the influence of such imbalance, but it also amplifies the noise and achieves worse empirical performance. Differently, the proposed instantiation module uses events in one representation as the reference to help instantiate the events in the other one. As a result, we jointly use these two representations to generate a group of detailed events, and then we can select the top k generated new events. By doing so, we do not only go detailed from the abstract representation but also avoid the imbalanced distribution issue.

3.3.5 Hyper-parameter Analysis

In APSI, we use two hyper-parameters w_v and w_n to control the conceptualization and instantiation depth we want over verbs and nouns respectively. 0 means no conceptualization and the larger value indicates more conceptualization we encourage. We show the performance of APSI with different hyper-parameter combinations in Figure 4, from which we can see that a suitable level of conceptualization is the key to the success of APSI. If no conceptualization is allowed, all the predicted events are restricted to the observed sub-event, thus we cannot predict "search house" after seeing "search car" and some events about the house. On the other hand, if we do not restrict the depth of conceptualization, all the sub-events will be conceptualized to be too general. As a result, even with the instantiation module, we could not predict the detailed sub-event as we want.

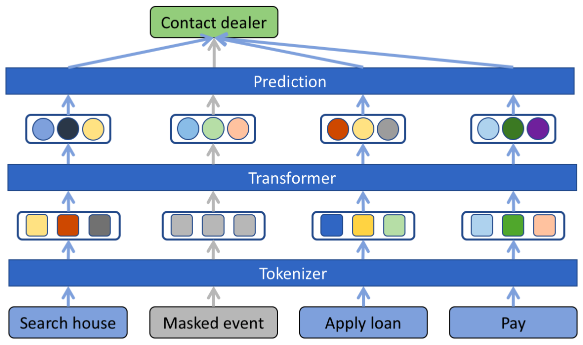


Figure 6: Demonstration of the event masked LM. Pre-trained language models are trained to predict the masked event given other events as the context.

Model	Accuracy	Δ
RoBERTa-based Event LM	73.59%	-
+ Seq2seq (GloVe)	73.06%	-0.53%
+ Seq2seq (RoBERTa)	72.33%	-1.26%
+ Top1 similar (Jaccard)	72.76%	-0.83%
+ Top1 similar (GloVe)	74.14%	0.55%
+ Top1 similar (RoBERTa)	74.16%	0.57%
+ APSI	74.78%†	1.19%
+ Human	76.97%‡	3.38%

Table 3: Results on the event prediction task. † and ‡ indicate the statistical significance over the baseline with p-value smaller than 0.01 and 0.001 respectively.

3.3.6 Case Study

Figure 5 shows an example that we use to analyze the current limitations of APSI. We can see that APSI can successfully predict events like "identify symptoms", but fails to predict event "identify causes". Instead, it predicts "take supplements". This is because APSI learns to predict such sequence from other processes like "treat diarrhea" or other diseases in the observed process graphs. Treating those diseases typically does not involve identifying the cause, which is not the case for treating pain. And, treating diseases often involves taking medicines, which can be conceptualized to "take supplement". As no events about pain helps instantiate "supplement", APSI just predicts it.

3.4 Extrinsic Evaluation

As discussed by (Rumelhart, 1975), the knowledge about process and sub-events can help understand event sequences. Thus, in this section, we investigate whether the induced process knowledge can help predict the missing events. Given a sub-event sequence, for each event in the sequence, we can use the rest of the sequence as the context and ask models to select the correct event against one negative event example. To make the task challenging, instead of random sampling, we follow Zellers et al. (2019) to select similar but wrong negative candidates based on their representation (i.e., BERT (Devlin et al., 2019)) similarity. We use the same training and test as the intrinsic experiment and as a result, we got 13,501 training sequences and 7,148 test questions.

The baseline method we are comparing with is the event-based masked language model[11], whose demonstration is shown in figure 6. We use pre-trained RoBERTa-base (Liu et al., 2019) to initialize the tokenizer and transformer layer and all sequences of training processes as the training data. To show the value of understanding the relationship between process and their sub-event sequence, for each sub-event sequence in the test data, we first leverage the process name and different structure prediction methods to predict sub-event sequences and use them as additional context to help the event masked LM to predict the missing event. To show the effect upper bound of adding process knowledge, we also tried adding the process structure provided by human beings as the context[12], which is denoted as '+Human'. All models are evaluated based on accuracy.

From the results in Table 3, we can make the following observations. First, adding high-quality process knowledge (i.e., APSI and Human) can significantly help the baseline model, which indicates that adding knowledge about the process can help better understand the event sequence. Second, the effect of process knowledge is positively correlated with their quality as shown in Table 1. Adding a low-quality process structure may hurt the performance of the baseline model due to the introduction of the extra noise. Third, the current way of using process knowledge is still very simple and there is room for better usage of the process knowledge, as the research focus of this paper is predicting process structure rather than applying it, we leave that for the future work.

[11]On our dataset, the RoBERTa based event LM model outperforms existing LSTM-based event prediction models.

[12]We randomly select another sub-event sequence that describes the same process from WikiHow, which could be different from the currently tested sequence. As a result, adding such sequence cannot help predict all missing events.

4 Related Works

Throughout history, considering the importance of events in understanding human language (e.g., commonsense knowledge (Zhang et al., 2020a)), many efforts have been devoted to define, represent, and understand events. For example, VerbNet (Schuler, 2005) created a verb lexicon to represent the semantic relations among verbs. After that, FrameNet (Baker et al., 1998) proposed to represent the event semantics with schemas, which has one predicate and several arguments. Apart from the structure of events, understanding events by predicting relations among them also becomes a popular research topic (e.g., TimeBank (Pustejovsky et al., 2003) for temporal relations and Event2Mind (Rashkin et al., 2018) for causal relations). Different from these *horizontal* relations between events, in this paper, we propose to understand event *vertically* by treating each event as a process and trying to understand what is happening (i.e., sub-event) inside the target event. Such knowledge is also referred to as event schemata (Zacks and Tversky, 2001) and shown crucial for how humans understand events (Abbott et al., 1985). One line of related works in the NLP community is extracting super-sub event relations from textual corpus (Hovy et al., 2013; Glavas et al., 2014). The difference between this work and them is that we are trying to understand events by directly generating the sub-event sequences rather than extracting such information from text. Another line of related works is the narrative schema prediction (Chambers and Jurafsky, 2008), which also holds the assumption that event schemata can help understand events. But their research focus is using the overall process implicitly to help predict future events while this work tries to understand events by knowing the relation between processes and their sub-event sequences explicitly.

5 Conclusion

In this paper, we try to understand events *vertically* by viewing them as processes and predicting their sub-event sequences. Our APSI framework is motivated by the notion of analogous processes, and attempts to transfer knowledge from (a very small number of) familiar processes to a new one. The intrinsic evaluation demonstrates the effectiveness of APSI and the quality of the predicted sub-event sequences. Moreover, the extrinsic evaluation shows that, even with a naive application method, the process knowledge can help better predict missing events.

Acknowledgements

This research is supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the BETTER Program, and by contract FA8750-19-2-1004 with the US Defense Advanced Research Projects Agency (DARPA). The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This paper is also partially supported by Early Career Scheme (ECS, No. 26206717), General Research Fund (GRF, No. 16211520), and Research Impact Fund (RIF, No. R6020-19) from the Research Grants Council (RGC) of Hong Kong.

References

Valerie Abbott, John B Black, and Edward E Smith. 1985. The representation of scripts in memory. *Journal of memory and language*, pages 179–199.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of COLING-ACL 1998*, pages 86–90.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Proceedings of EMNLP 2014*.

Alexander Budanitsky and Graeme Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. *Comput. Linguistics*, 32(1):13–47.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL 2008*, 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 NAACL-HLT 2019*, pages 4171–4186.

Goran Glavas, Jan Snajder, Marie-Francine Moens, and Parisa Kordjamshidi. 2014. Hieve: A corpus for extracting event hierarchies from news stories. In *Proceedings of LREC 2014*, pages 3678–3683.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? event prediction using a compositional neural network model. In *Proceedings of AAAI 2016*, pages 2727–2733.

Eduard H. Hovy, Teruko Mitamura, Felisa Verdejo, Jun Araki, and Andrew Philpot. 2013. Events are not simple: Identity, non-identity, and quasi-identity. In *Proceedings of EVENTS@NAACL-HLT 2013*, pages 21–28.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations 1972*, pages 85–103.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *CoRR*, abs/1810.09305.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of Text Summarization Branches Out 2004*, pages 74–81.

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. *CoRR*, abs/1907.11692.

Songjian Lu and Xinghua Lu. 2014. An exact algorithm for the weighed mutually exclusive maximum set cover problem. *CoRR*, abs/1401.6385.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP 2014*, pages 1532–1543.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, page 40.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 909–918.

Hannah Rashkin, Maarten Sap, Emily Allaway, Noah A. Smith, and Yejin Choi. 2018. Event2mind: Commonsense inference on events, intents, and reactions. In *Proceedings of ACL 2018*, pages 463–473.

DE Rumelhart. 1975. Notes on a schema for stories language, thought, and culture. *Representation and understanding*, pages 211–236.

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

Karin Kipper Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NeurIPS 2014*, pages 3104–3112.

Jeffrey M Zacks and Barbara Tversky. 2001. Event structure in perception and conception. *Psychological bulletin*, 127(1):3.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of ACL 2019*, pages 4791–4800.

Hongming Zhang, Daniel Khashabi, Yangqiu Song, and Dan Roth. 2020a. Transomcs: From linguistic graphs to commonsense knowledge. In *Proceedings of IJCAI 2020*, pages 4004–4010.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020b. ASER: A large-scale eventuality knowledge graph. In *Proceedings of WWW 2020*, pages 201–211.