Automatic Mining of Salient Events from Multiple Documents

Anonymous ACL submission

Abstract

This paper studies a new event knowledge extraction task, *Event Chain Mining*. Given multiple documents on a super event, it aims to mine a series of salient events in a temporal order. For example, the event chain of super event *Mexico Earthquake in 2017* is {*earthquake hit Mexico*, *destroy houses*, *kill people*, *block roads* }. This task can help readers capture the gist of texts quickly, thereby improving reading efficiency and deepening text comprehension. To address this task, we regard an event as a cluster of different mentions of similar meanings. By this way, we can identify the different expressions of events, enrich their semantic knowledge and enhance order information among them. Taking events as the basic unit, we propose a novel and flexible unsupervised framework, EMINER. Specifically, we extract event mentions from texts and merge those of similar meanings into a cluster as an event. Then, essential events are selected and arranged into a chain in the order of their occurrences. We then develop a testbed for the proposed task, including a human-annotated benchmark and comprehensive evaluation metrics. Extensive experiments are conducted to verify the effectiveness of EMINER in terms of both automatic and human evaluations.

1 Introduction

Generally, a lot of unstructured text data can be regarded as a chain of salient events arranged in order (Forster, 1985; Abbott, 2020). Therefore, extracting event chain knowledge from texts is a crucial step in text understanding. Currently, various event-centric tasks has gained significant interest, such as event relation extraction (Han et al., 2019; Wang et al., 2020; Ahmad et al., 2021), salient event identification (Liu et al., 2018; Wilmot and Keller, 2021), and event process understanding (Zhang et al., 2020a; Chen et al., 2020). However, most of these studies highly rely on expert annotations. The annotation process is expensive and time-consuming. Some studies (Weber et al., 2018; Li et al., 2020) attempt to alleviate this problem under an unsupervised setting. They extract event schemas from large corpus as prior knowledge to assist downstream tasks, such as story generation (Yao et al., 2019), question answering (Reddy et al., 2019), and reading comprehension (Zhang et al., 2021a). However, there still lacks relevant research on automatic mining of event chains. Such a task can provide a brief summary and help readers to capture the skeleton of texts quickly.

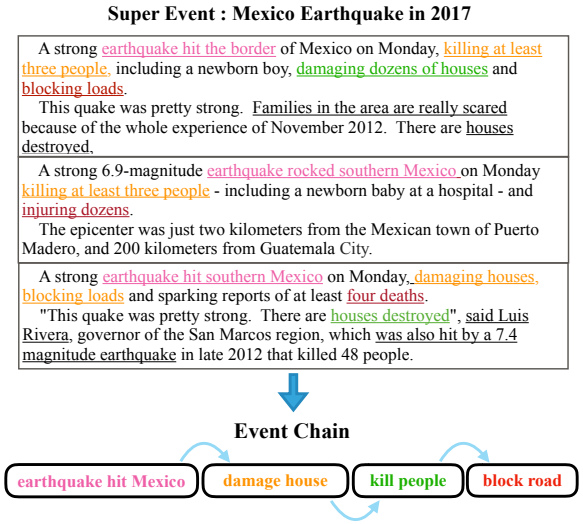

Figure 1: An example of the event chain mining task. There are three documents on Mexico Earthquake in 2017. They all mention salient events. We underline all the event mentions and mark those of similar meanings in the same color. Below is the event chain.

To this end, we propose a new task of knowledge extraction, **Event Chain Mining**. It aims to mine a series of salient events in a temporal order, which can serve as a concise highlight of texts. Specifically, given a super event[1], multiple documents usually report it from different perspectives. Moreover, these reports usually share the most salient events among their texts. For example, Figure 1

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

shows three documents on a super event, Mexico Earthquake in 2017. Most of them mention four essential events in the earthquake, including *earthquake hit Mexico*, *damage houses*, *kill people*, and *block roads*. These events can provide a sequential highlight of how this disaster occurred. With such a chain of salient events, readers can grasp the pivotal context of the text quickly, thus improving reading efficiency and deepening reading comprehension.

This observation leads to the **Event Chain Mining** problem, which poses the following challenges: (1) *variability of events*. An event can be expressed in different descriptions. For example, in Figure 1, *earthquake rocked southern Mexico* and *earthquake hit southern Mexico* are two different mentions, but actually they describe similar meanings in Mexico Earthquake. If an event chain includes both events, it will lead to information redundancy. (2) *salience inequality of events*. Not all events are equally important. Some events can be too general and contain little information, such as *say it*. Others could be too specific, not closely tied to the main points, such as *The state has 3.44 million people*. We should filter out these events and leave salient ones when mining an event chain. (3) *conflict of event relations*. Multiple documents have different narrative styles and report the development of events in different order. For example, some texts record events according to the occurrence order. Others first introduce the effect and then explain the causes. Thus, when it comes to a conflict, it become more difficult to determine the temporal relations between events.

To address these challenges, we regard an event as a cluster of mentions with similar meanings. By this way, there are three significant benefits. First, it naturally helps the first challenge – different expressions of the same event. Second, it enhances event semantics by with including multiple related mentions, which makes it easier to deepen event understanding and recognize salient events. Third, it enriches order information between events. By introducing multiple mentions, more clues about event relations can be obtained from the contexts. As a result, event ordering can be more convincing with the support from the majority of mentions.

Based on these, we propose a novel unsupervised event chain mining framework, EMINER, which contains four major steps. Specifically, given a set of related texts on the same super event, we first decompose them into multiple event mentions. We elaborate frequently-occurring syntactic patterns and extract all possible event mentions. Then, event mentions of similar meanings are merged into clusters as distinct events. The event mention merging problem are formulated as an online text stream clustering task without requiring a fixed number of clusters. After that, we measure the salience of events to select salient ones according to event frequency counting. Finally, those salient events are arranged in a chain according to their occurrences in original texts.

Motivated by the importance of this task, we re-annotate an existing multi-document dataset (Miranda et al., 2018) to develop a new benchmark for evaluating event chain mining systems. For multiple documents on a super event, we manually annotate salient events as a brief summary. Besides, we propose a comprehensive evaluation system, which evaluates from multiple perspectives, such as event semantics and sequential orders. Based on these, we conducted extensive experiments in terms of both automatic and human evaluations. The results verify our framework can actually produce a chain of salient events to guide people to understand texts.

Contributions. The major contributions of this paper are summarized as follows: (1) a new task, Event Chain Mining, summarizing the skeleton of texts by extracting event chains, is proposed; (2) a novel unsupervised framework, EMINER, is designed to overcome the aforementioned challenges and address this problem automatically; (3) a benchmark dataset and a comprehensive evaluation system are developed for this task; and (4) extensive experiments verify the effectiveness of the proposed framework in terms of both automatic and human evaluations.

2 Problem Formulation

In this section, we first introduce some important concepts and then present our task definition. An **event mention**, is a phrase that contains multiple words $\{w_1, w_2, \ldots, w_z\}$, where z is the number of words, and $w_1, w_2, \ldots, w_m$ are all in the vocabulary. A pair of words (w_i, w_j) in an event mention m may follow a specific syntactic relation. An **event** is a cluster of event mentions in similar meanings $\{m_1, m_2, \ldots, m_x\}$, where x is the number of mentions in event e. A **super event** refers to a more coarse-grained event described by multiple documents. A **salient event** is that can provide suf-

ficient and important information about the super event. As to an **event chain**, it stands for a series of salient events arranged in the order of occurrence.

Task Definition. Given a set of documents $\mathcal{D}$ on a super event, our task of *event chain mining* is to produce a sequence of salient events $\{e_1, e_2, \ldots, e_n\}$, which can give a brief summary of the texts. n is the number of events.

Intuitively, humans are accustomed to understanding texts in a sequential order instead of modeling a relation graph. Thus, our task aims to mine an event chain. However, notably, our proposed method can also extract the partial order relationship among events to form a relation graph (introduced in the next section) As to complex event relation graph modeling, we leave it as future work.

3 Framework

The proposed framework, EMINER, outlines the event chain mining task in four major steps: (1) event mention extraction, (2) event mention merging, (3) salient event selection, and (4) salient event ordering. The architecture is illustrated in Figure 2.

3.1 Event Mention Extraction

We adopt a lightweight method to extract event mentions in texts without relying on manually-labeled training data. It aims to decomposing texts into multiple event mentions. For example, there is a sentence about Mexico Earthquake in 2017: *A strong earthquake shook Mexico on Monday, killing at least three people and damaging dozens of buildings*. It mainly contains three event mentions, *earthquake shook Mexico*, *killing people* and *damaging buildings*. To handle the complex structure of event mentions, we elaborate frequently-occurring syntactic patterns inspired by (Zhang et al., 2020b). By pattern matching, all possible event mentions are extracted from texts based on sentence dependency tree structures.

Specifically, given a sentence, we first use a dependency parser[2] to obtain its dependency parse tree. As the centers of event mentions are verbs, we extract all verbs from each sentence. We hope to ensure that all the extracted event mentions are semantically complete and frequently occurring without being too complicated. Therefore, we elaborate 57 syntactic patterns based on those in Zhang et al. (2020b) (selected syntactic patterns are showed in Appendix). For each verb, we check its dependent words and their dependency label. If they match one of the syntactic patterns, we extract the corresponding words as an event mention. To make event mention contain more details, we give priority to more complex patterns. That is, once a pattern is exactly matched, we will no longer consider the remaining simpler ones. By such an strategy, all possible event mentions can be extracted from texts. Notably, we treat the sentences with clauses equally. So this method can decompose long sentences completely into event mentions.

[2] https://nlp.stanford.edu/software/stanford-dependencies.html

3.2 Event Mention Merging

In this step, we merge similar event mentions into the same cluster, which is indispensable for the framework. To improve generalization capability for different topics or texts, the number of clusters should not be fixed. Thus, we formulate the event mention merging as a short text stream clustering task (Yin et al., 2018; Chen et al., 2019; Kumar et al., 2020). Specifically, event mentions are regarded as a stream and each of them is processed incrementally. In each process, for a mention, we decide whether to add it to an existing cluster or create a new cluster. Then, we update the corresponding cluster to prepare for subsequent events.

For example, there are three mentions, *kill people*, *damage houses* and *destroy homes*. Initially, we create a new cluster for the first mention *kill people*. Then, when *damage houses* comes, there is an existing cluster {*kill people*}. Since this mention is not related to the cluster, we still create a new cluster for it. Thus, there exist two clusters now, {*kill people*} and {*damage houses*}. Later, for the third mention, we compare the probability of it joining these two clusters and creating a new one. This mention was decided to be grouped into the second cluster. Finally, we obtain two clusters, {*kill people*} and {*damage houses*, *destroy homes*}.

Such an evolutionary clustering can automatically increase the number of clusters with event mentions. Nonetheless, it comes to three questions: (1) how to represent and update a cluster; (2) how to estimate the probabilities of a mention belonging to existing clusters and a new cluster; and (3) how to avoid the mention order affecting the clustering process. We will solve these problems in turn.

3.2.1 Cluster Feature

We first represent an event with the cluster feature (CF) vector, which essentially is a cluster with its

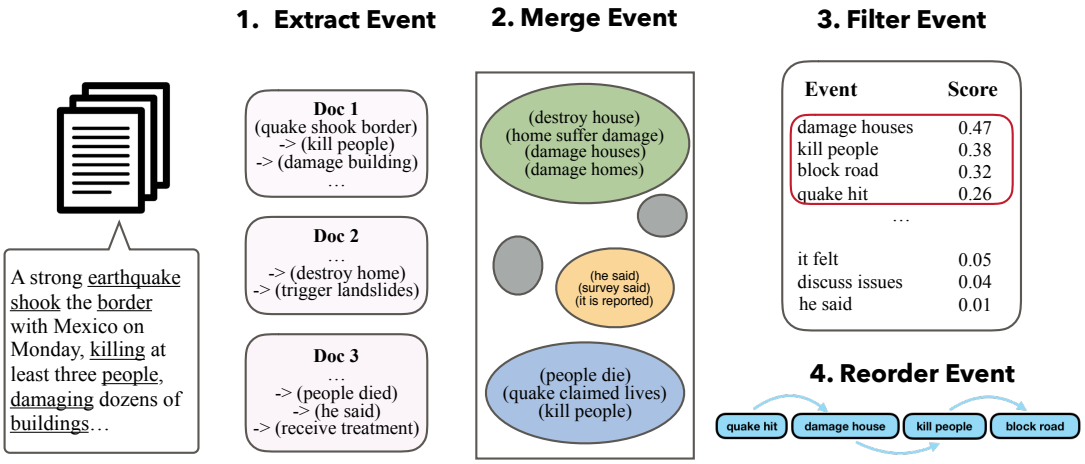

Figure 2: Architecture of EMINER. Given multiple documents on a super event, we decompose texts into event mentions and merge those of similar meanings into a cluster as an event. Then essential events are selected and arranged into a chain according to their occurrences. For the convenience of presentation, the events in the last two steps are represented by a representative mention.

event mentions. The CF vector of an event is defined as a tuple $\left\{\vec{f}_e, n_e, x_e\right\}$, where $\vec{f}_e$ contains a list of mention frequencies in event e; n_e is the number of mentions in event e; and x_e is the number of words in event e. The cluster feature vector presents desirable addition and deletion properties.

- Addition Property. A mention m can be efficiently added to cluster e by updating its CF vector as follows.

$$\begin{aligned} f_e^w &= f_e^w + N_m^w, \quad \forall w \in m, \\ n_e &= n_e + 1, \\ x_e &= x_e + N_m. \end{aligned} \tag{1}$$

- Deletion Property. A mention m can be efficiently deleted from cluster e by updating its CF vector as follows.

$$\begin{aligned} f_e^w &= f_e^w - N_m^w, \quad \forall w \in m \\ n_e &= n_e - 1, \\ x_e &= x_e - N_m, \end{aligned} \tag{2}$$

where N_m^w and N_m are the number of occurrences of word w in mention m and the total number of words in mention m, respectively, and $N_m = \sum_{w \in m} N_m^w$. Besides, f_e^w is the number of occurrence of word w in cluster e. With the addition and deletion properties, we can update the CF vectors when a cluster includes or excludes a mention.

3.2.2 Model Formulation

We assume the mentions are generated by the Dirichlet Process Multinational Mixture (DPMM) model (Yin and Wang, 2016). Its generative process is as follows.

$$\begin{aligned} \theta \mid \gamma &\sim GEM(1, \gamma), \\ e \mid \theta &\sim \text{Mult}(\theta), \\ \mathcal{N}_k \mid \beta &\sim \text{Dir}(\beta) \quad k = 1, \ldots, \infty, \\ m \mid e, \{\mathcal{N}_k\}_{k=1}^{\infty} &\sim p\left(m \mid \mathcal{N}_e\right). \end{aligned} \tag{3}$$

Here, when generating mention m, the model first selects a mixture component (cluster e) according to the mixture weights. Then mention m is generated by the selected mixture component (cluster e) from distribution $p\left(m \mid \mathcal{N}_e\right)$. θ is generated by a stick-breaking construction (Teh, 2010). $\mathcal{N}_k$ are also generated by a Dirichlet distribution (Teh, 2010). γ and β are two hyper-parameters.

Following Kumar et al. (2020), the probability of mention m generated by cluster e is:

$$\begin{aligned} & p\left(e_m = e \mid e, m, \alpha, \beta\right) \\ & \propto \left(\frac{n_e}{D - 1 + \alpha D}\right) \cdot \\ & \left(\frac{\prod_{w \in m} \prod_{j=1}^{f_m^w} f_e^w + \beta + j - 1}{\prod_{i=1}^{x_m} x_e + V\beta + i - 1}\right) \cdot \\ & \left(1 + \frac{1}{n_e} \sum_{i=1}^{n_e} Sim\left(m, m_i^e\right)\right). \end{aligned}$$

In the above equation, the first term represents completeness of the cluster. A new mention gives priority to clusters with more mentions. Thus, although the number of clusters can be unlimited, only a limited number of clusters will be created. Here, n_e is the number of mentions contained by the cluster e, D is the number of current mentions in the existing clusters, and α is the concentration parameter of the model.

The second term defines the term occurrence between a cluster and a mention. It is based on multinomial distribution with psuedo weight of words β. x_m and f_m^w represent total number of words and term frequency of word w in mention m, respectively. The symbol f_e^w is the term frequency of the word w in the cluster e. The current vocabulary

size of the model is represented by V, and x_e is the number of words in the cluster e.

The third term reflects semantic similarity between a cluster and a mention. For a mention m, we calculate the average semantic similarity between each mention m_i^e in the cluster e. Here, given a pair of mentions, $Sim(\cdot)$ is the function for their semantic similarity scores. Following Zhang et al. (2020c), we first use a pretrained language model (Devlin et al., 2019) to obtain contextual representations of the two mentions. Then, their similarity is computed as a sum of cosine similarities between their word embeddings.

So far, we have defined the probability of a mention choosing an existing cluster. Then we have to consider the probability for a mention to create a new cluster. By following the DPMM, the probability of creating a new cluster is as follows.

$$p\left(e_m = K+1 \mid \vec{e}, \vec{m}, \alpha, \beta\right) \propto \frac{\alpha D}{D-1+\alpha D} \cdot \frac{\prod_{w \in m} \prod_{j=1}^{f_m^w}(\beta + j - 1)}{\prod_{i=1}^{x_m}(V\beta + i - 1)},$$

where K is the number of the existing clusters; αD denotes the pseudo number of clusters related mentions, and β is the pseudo term frequency of each word (exist in mention) of the new cluster.

3.2.3 Merging Process

Following Yin et al. (2018), our merging method allows processing each event mention incrementally and updating the model accordingly. Initially, it creates a new cluster for the first mention. We initialize the cluster feature of this new cluster with the first mention. Later, for each event mention, it either belongs to an existing cluster or generates a new cluster. It depends on the corresponding probability computed with Eqs. (4) and (4). We choose the cluster with the highest probability. The CF vector of this cluster is updated according to the addition property. In this way, we can detect new clusters more naturally without a fixed number of clusters. Based on this process, we can obtain the initial clustering result.

Since all the mentions are processed one by one, their order may affect the clustering results. Therefore, to improve the robustness of the model, we then update the clustering results. For each mention, we delete it from its current cluster with the deletion property. Then, we reassign it to an existing cluster or create a new cluster for it. According to Eqs. (4) and (4), the choice with the highest probability will be made.

3.3 Salient Event Selection

In this step, we filter too general or too specific events, thereby selecting salient ones. Since each event is involved with multiple event mentions, it helps to enhance event semantic understanding. For example, *Jessica said on conference* is a general event mention with specific arguments. Existing frequency-based methods (Shen et al., 2021; Zhang et al., 2021b) might fail to handle it. However, with the help of typical general mentions in the same cluster, such as *it says*, the selection algorithm can filter it more easily.

Based on this observation, we define the salience score for an event cluster. All the mentions of this event are taken into consideration. An event is valued high if its mentions occur frequently in the original texts and rarely exist in a general-domain background corpus. Computationally, given an event e, we define its salience as follows:

$$\begin{aligned}\text{Salience}(e) &= \frac{1}{N}\sum_{i=1}^{N}\text{Salience}(m_i) \\ &= \frac{1}{N}\sum_{i=1}^{N}(1+\log(\text{freq}(m_i))^2)\log(\frac{N_{bs}}{bsf(m_i)}),\end{aligned}$$

where m_i is a mention of event e, and N is the number of the mentions. Besides, $freq(m_i)$ denotes the frequency of mention m_i, N_{bs} is the number of background texts, and $bsf(w)$ refers to the background text frequency of mention m_i.

3.4 Salient Event Ordering

Finally, we arrange the salient events in a partial order according to original texts. We compare the relative order of describing these events in the texts. Notably, the basic unit of reordering is an event, a cluster of multiple mentions. Therefore, the relations of different mentions in different document might come into conflict. To handle this problem, a multi-document multi-mention voting mechanism is introduced. In the other word, we makes the decision supported by the majority of mentions in most texts. Specifically, the order of an event e is defined as:

$$\text{Order}(e) = \frac{1}{N_d}\sum_{i=1}^{N_d}\min_{m \in e}\text{index}(d_i, m), \quad (4)$$

where N_d is the number of related texts. d_i represents the i-th text, m is a mention for event e, and $\text{index}(d_i, m)$ refers to the index of mention m in the event sequences extracted from the text d_i. By comparing the order of each event, we can rank them in order and produce a salient event sequence.

Despite multiple mentions in multiple documents enrich order information between events,

the description style of texts still limits us. If flashbacks are frequently used in texts, we require more external knowledge to make a smarter judgement of event ordering. We leave this as future work.

4 Experiments

In this section, we conduct both automatic and human evaluations to show EMINER can mine meaningful event chains from unstructured texts, which can assist people to acquire information quickly.

4.1 Dataset

We re-annotate an existing multi-document dataset (Miranda et al., 2018) to develop a new benchmark. This benchmark involves 100 super events and approximately 2,000 articles. For each super event, there are about 20 articles describing it on average. The average number of words in an article is about 420. We manually annotate 5-10 salient events as a brief summary. For the sake of convenience, each event is described with one mention. Each mention includes less than 10 words to ensure the brevity.

Our annotation team consists of 3 graduate students in NLP. For each super event, the annotators are required to read all the related articles and write a chain of events. Annotated events should be mentioned by most of the related articles. Besides, the whole event chain should make readers understand the main plot of this super event without original texts. Each annotated chain is required to be reviewed by another annotator, after which they discuss and revise until reaching an agreement.

4.2 Baselines

Currently, there is no existing approach to solve the event chain mining task. Therefore, we replace each module in EMINER with related methods as the baselines. For mention extraction: ASER (Zhang et al., 2020b) proposes a lightweight event extractor based on syntactic pattern matching. For mention merging: (1) HDBSCAN (McInnes et al., 2017) is a hierarchical density-based spatial clustering method. (2) OSDM (Kumar et al., 2020) is an online semantic-enhanced dirichlet model for short text stream clustering. These two methods do not require fixing the number of clusters. For event selection: ETDISC provides a frequency-based salient word selector and we expand it to filter event mentions. For event ordering: SymTime (Zhou et al., 2021) is a neuro-symbolic temporal reasoning pretrained model, which can determain the order between events.

We also present a randomly produced event chain as the lower-bound for this task. Besides, due to the lead bias problem in the news domain (Zhong et al., 2019), we introduce LEAD as a strong baseline. It refers to extracting all events from the first several sentences of the texts as a chain.

4.3 Evaluation Metrics

We build a comprehensive evaluation system, which evaluates the quality of the produced event chains from multiple perspectives. Motivated by Lin (2004), we propose three kinds of event-based ROUGE F1 scores, including ERouge-1, ERouge-2, and ERouge-3. Specifically, similar to ROUGE, we evaluate how much percentage of events, event pairs, and the longest common event subsequence in the induced chains is covered by the human-provided references. Since each event contains multiple mentions, we measure the overlap of all the mention pairs in two mentions, and then take the average as the similarity of two event.

Following Zhang et al. (2020a), we provide two overlap standards of two event mentions to better understand the mining quality, "String Match" and "Hypernym Allowed". The first standard requires all words in the produced mention to be the same as the referent mention. This setting is rather strict. The second standard allows the hypernyms of words in mentions to relax the restrictions on comparison. For example, two event mentions, *damage houses* and *damage buildings*, are count as a match. This setting help check if our framework select relevant events.

4.4 Automatic Evaluation

Following Glavaš et al. (2014) and Zhang et al. (2020a), we provide two settings to make the evaluation comprehensive: (1) Basic: evaluate events based on only verbs; (2) Advanced: evaluate events based on all words. From Table 1, in these two settings, we can see the improvement of ERouge scores when adopting our proposed framework to mine event chains compared with RANDOM and LEAD. In addition, we replace the four components in our framework with similar methods. Compared with ASER, our extractor is better at processing long sentences. Based on this, EMINER show a clear advantage in the subsequent steps. In addition, although the two comparative clustering methods also work, our method is superior in all three ERouge scores. Moreover, filtering events instead of

Basic Setting						
Models	**String Match**			**Hypernym Allowed**		
	ERouge-1	ERouge-2	ERouge-L	ERouge-1	ERouge-2	ERouge-L
RANDOM	4.2500	0.0759	2.7500	12.1428	3.5298	7.9702
LEAD	10.8095	1.9076	9.4345	16.3273	4.3404	15.3630
Extraction → ASER	11.6959	2.1267	10.3818	15.8928	4.6829	14.3520
Merging → HDBSCAN	17.6547	4.2562	13.3154	22.1130	7.4998	15.1488
Merging → OSDM	14.3928	2.9125	13.7678	21.5297	5.5713	18.4761
Selection → ETDISC	15.0357	3.0548	12.9880	22.8928	7.4044	16.2440
Ordering → SYMTIME	17.4047	4.5829	13.7797	23.9166	7.6342	16.5773
EMINER	**18.3690**	**5.1104**	**14.1904**	**24.7916**	**8.5698**	**16.9880**
Advanced Setting						
Models	**String Match**			**Hypernym Allowed**		
	ERouge-1	ERouge-2	ERouge-L	ERouge-1	ERouge-2	ERouge-L
RANDOM	1.6250	0.3489	1.1250	4.5833	1.2721	3.5833
LEAD	9.6369	1.6723	8.8869	13.5654	2.9875	11.8154
Extraction → ASER	9.0315	0.9354	8.8290	13.4234	2.5079	10.3810
Merging → HDBSCAN	13.1607	3.1103	10.3214	15.4940	3.7529	13.5714
Merging → OSDM	11.7678	1.8730	11.1428	15.5654	3.0450	14.2261
Selection → ETDISC	10.2380	1.5888	9.0238	15.2083	2.7651	12.5654
Ordering → SYMTIME	13.5714	2.7329	10.9464	14.7440	3.9113	14.1190
EMINER	**14.2440**	**3.1839**	**12.1904**	**16.7916**	**4.2603**	**15.7380**

Table 1: Experimental results. Basic Setting refers to only evaluating the verb for each event while Advanced Setting refers to evaluating all the words. String Match and Hypernym Allowed are two overlap standards of two event mentions. The first requires all words to be the same and the second allows the hypernyms to relax the restrictions.

mentions can introduce more semantic information, which plays an important role in selecting salient events. Finally, although SYMTIME is a powerful pretrained temporal model, it fails to utilize rich relation information between multiple mentions. Thus, EMINER can achieve better results.

4.5 Ablation Study

We remove each component from our full framework to verify its importance for the event chain mining task. Without event mention merging, we regard each mention as an event, and then perform event selection and ordering. If event selection is detached, the merged events are ordered according to their occurrences. After removing event ordering, we directly compare the selected salient events with human references.

The experiment results are showed in Table 2. Our framework can already obtain a relatively high performance compared to the variant without merging. It reveals the significance of identifying similar event mentions, which can reduce information redundancy. Besides, removing the selection component affects the results sightly. It is supposed that, due to the lead bias problem, most salient events are arranged at the front of the chain after ordering. In addition, the obvious drop of the ERouge-L score in the fourth row reflects the important role that event ordering plays in this task.

Model	ERouge-1	ERouge-2	ERouge-L
EMINER	**16.7916**	**4.2603**	**15.7380**
- Merging	7.6785	0.8554	7.4743
- Selection	12.1011	2.5872	10.7619
- Ordering	15.5654	3.0450	14.2261

Table 2: Ablation Study (Hypernym Allowed in Advanced Setting). '-' means removing the component from the full framework.

4.6 Case Study

Table 3 shows two interesting examples with super events, event chains produced by our method and the corresponding groundtruth. Notably, for the convenience of presentation, here we show a representative event mention rather than all mentions for each event. By case study, we want to verify the effectiveness and analyze the limitations of our framework. We can see that our method can successfully discover most of salient events for a super event. For example, in the first case, the occurrence and consequences of this earthquake have been saved in the event chain. However, in the second case, it fails to determine the correct order between events. The effect, *fireman were arrested*, is arranged before the causes, *start fire* and *kill people*. Because our method orders events based on the description in the original text. When flashbacks are frequently used, it may not be able

Mexico Earthquake in 2017	
Produced Events	Groundtruth
earthquake happened	*earthquake rocked mexico*
left person dead	*damaged houses*
triggered landslide	*kill people*
destroyed houses	*trigger landslides*
blocked roads	*blocked roads*
Former Firefighter Arrested for Starting Fires	
Produced Events	Groundtruth
fireman were arrested	*fireman started fires*
fire destroyed homes	*fire destroyed homes*
killed people	*left people dead*
turned into fire	*arrested man on suspicion*
body found inside home	*fire burned miles over week*

Table 3: Case study. There are two cases including the super events (in bold), the outputs of EMINER, and the human-annotated groundtruth.

to arrange events according to their occurrences. In this case, more external knowledge is needed to assist event ordering. We leave it as future work.

4.7 Human Evaluation

To better understand the model performance, we also conduct human evaluation. Specifically, we ask 10 graduate students to rank six different event chains (produced by our framework, its variants, and groundtruth) according to three metrics: relevance, informativeness, and coherence to the texts. Ranking first means the best performance on this metric. We randomly select 20 samples from our dataset for evaluation. The results are provided in Table 4. From the perspective of relevance, our framework can output more relevant event chains to the super events. Compared with Selection → ETDISC, other methods can mine more relevant and salient events thanks to event-based selection introducing more semantic information. In terms of the informativeness metrics, our framework substantially extract distinct events and reduce information redundancy when comparing to other baselines. The capacity of grasp different events in similar meanings is largely responsible for this improvement. However, Merging → HDBSCAN performs poorly on the informativeness because it lacks semantic knowledge to identify synonyms in the mentions. Coherence depends on whether event chains can reflect the plot of texts smoothly. Benefit from rich event relationships, event-based ordering can obtains high scores. However, the performance of all automatic models is still far from the human-annotated answers.

Model	Relev.	Infor.	Cohen.
Merging → HDBSCAN	4.14	4.54	3.21
Selection → ETDISC	4.66	3.28	3.52
Ordering → SYMTIME	2.84	3.53	4.37
EMINER	**2.13**	**2.56**	**2.85**
Reference	1.23	1.09	1.05

Table 4: Results of human evaluation by ranking. Relev., Infor., and Cohen. represent relevance, informativeness and coherence to original texts, respectively. Reference refers to the human-annotated event chains.

5 Related Work

Considering the importance of events in understanding unstructured texts, many efforts have been devoted to represent and understand events. Han et al. (2019); Wang et al. (2020); Ahmad et al. (2021) pay attention on event relation extraction and predicting. Salient event identification is also a popular research topic. (Liu et al., 2018; Jindal et al., 2020; Wilmot and Keller, 2021). Apart from these, there have been recent interests in event process understanding (Zhang et al., 2020a; Chen et al., 2020). However, most of these studies highly rely on expensive expert annotations.

Some studies (Weber et al., 2018; Li et al., 2020) alleviate this problem under an unsupervised setting. The pioneer work (Chambers and Jurafsky, 2008) induces event chains as a new representation of structured knowledge. Chambers and Jurafsky (2008) and Radinsky and Horvitz (2013) extended such event chain modeling for news prediction and timeline construction. Berant et al. (2014) extracted events and their relationships in biological processes for biological reading comprehension. More recently, Zhang et al. (2021a) models salience-aware event chains for narrative understanding. These studies extract event schemas from a large amount of texts as prior knowledge to assist downstream tasks (Yao et al., 2019; Reddy et al., 2019). However, there still lacks relevant research on automatic mining of event chains.

6 Conclusion

In this paper, we propose a new task, Event Chain Mining, to summarize the skeleton of texts by extracting event chains. To address it automatically, a novel unsupervised framework EMINER is suggested. Besides, we develop a benchmark dataset and a comprehensive evaluation system for this task. Extensive experiments verify the effectiveness of the proposed framework and the quality of the produced event chains.

References

H Porter Abbott. 2020. *The Cambridge introduction to narrative*. Cambridge University Press.

Wasi Uddin Ahmad, Nanyun Peng, and Kai-Wei Chang. 2021. GATE: graph attention transformer encoder for cross-lingual relation and event extraction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12462–12470. AAAI Press.

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 the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 789–797. The Association for Computer Linguistics.

Junyang Chen, Zhiguo Gong, and Weiwen Liu. 2019. A nonparametric model for online topic discovery with word embeddings. *Information Sciences*, 504:32–47.

Muhao Chen, Hongming Zhang, Haoyu Wang, and Dan Roth. 2020. What are you trying to do? semantic typing of event processes. In *Proceedings of the 24th Conference on Computational Natural Language Learning, CoNLL 2020, Online, November 19-20, 2020*, pages 531–542. Association for Computational Linguistics.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Edward Morgan Forster. 1985. *Aspects of the Novel*, volume 19. Houghton Mifflin Harcourt.

Goran Glavaš, Jan Šnajder, Parisa Kordjamshidi, and Marie-Francine Moens. 2014. Hieve: A corpus for extracting event hierarchies from news stories. In *Proceedings of 9th language resources and evaluation conference*, pages 3678–3683. ELRA.

Rujun Han, I-Hung Hsu, Mu Yang, Aram Galstyan, Ralph M. 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 2019, Hong Kong, China, November 3-4, 2019*, pages 666–106. Association for Computational Linguistics.

Disha Jindal, Daniel Deutsch, and Dan Roth. 2020. Is killed more significant than fled? A contextual model for salient event detection. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 114–124. International Committee on Computational Linguistics.

Nikhil Ketkar. 2017. Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer.

Jay Kumar, Junming Shao, Salah Uddin, and Wazir Ali. 2020. An online semantic-enhanced dirichlet model for short text stream clustering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 766–776. Association for Computational Linguistics.

Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare R. Voss. 2020. Connecting the dots: Event graph schema induction with path language modeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 684–695. Association for Computational Linguistics.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Zhengzhong Liu, Chenyan Xiong, Teruko Mitamura, and Eduard H. Hovy. 2018. Automatic event salience identification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1226–1236. Association for Computational Linguistics.

Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205.

Sebastião Miranda, Arturs Znotins, Shay B Cohen, and Guntis Barzdins. 2018. Multilingual clustering of streaming news. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4535–4544.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Sixth ACM International Conference on Web Search and Data Mining, WSDM 2013, Rome, Italy, February 4-8, 2013*, pages 255–264. ACM.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. Coqa: A conversational question answering challenge. *Trans. Assoc. Comput. Linguistics*, 7:249–266.

Jiaming Shen, Yunyi Zhang, Heng Ji, and Jiawei Han. 2021. Corpus-based open-domain event type induction. *CoRR*, abs/2109.03322.

Yee Whye Teh. 2010. Dirichlet process.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020. Joint constrained learning for event-event relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 696–706. Association for Computational Linguistics.

Noah Weber, Leena Shekhar, Niranjan Balasubramanian, and Nate Chambers. 2018. Hierarchical quantized representations for script generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3783–3792. Association for Computational Linguistics.

David Wilmot and Frank Keller. 2021. Memory and knowledge augmented language models for inferring salience in long-form stories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 851–865. Association for Computational Linguistics.

Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 7378–7385. AAAI Press.

Jianhua Yin, Daren Chao, Zhongkun Liu, Wei Zhang, Xiaohui Yu, and Jianyong Wang. 2018. Model-based clustering of short text streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 2634–2642. ACM.

Jianhua Yin and Jianyong Wang. 2016. A model-based approach for text clustering with outlier detection. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 625–636. IEEE Computer Society.

Hongming Zhang, Muhao Chen, Haoyu Wang, Yangqiu Song, and Dan Roth. 2020a. Analogous process structure induction for sub-event sequence prediction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1541–1550. Association for Computational Linguistics.

Hongming Zhang, Xin Liu, Haojie Pan, Yangqiu Song, and Cane Wing-Ki Leung. 2020b. ASER: A large-scale eventuality knowledge graph. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 201–211. ACM / IW3C2.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020c. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Xiyang Zhang, Muhao Chen, and Jonathan May. 2021a. Salience-aware event chain modeling for narrative understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 1418–1428. Association for Computational Linguistics.

Xiyang Zhang, Muhao Chen, and Jonathan May. 2021b. Salience-aware event chain modeling for narrative understanding. *CoRR*, abs/2109.10475.

Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2019. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1049–1058. Association for Computational Linguistics.

Ben Zhou, Kyle Richardson, Qiang Ning, Tushar Khot, Ashish Sabharwal, and Dan Roth. 2021. Temporal reasoning on implicit events from distant supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 1361–1371. Association for Computational Linguistics.

A Appendix

A.1 Example of Event Mention Patterns

To better understand our event extraction approach, we list some specific patterns and examples in Table 5.

A.2 Implementation details

We implement EMINER using PyTorch (Ketkar, 2017). The experiments are conducted on 8 NVIDIA TITAN Xp GPUs. For event mention extraction, we give priority to matching the responsible patterns to extract as many details as possible. The extracted events will not overlap. For

Pattern	Example
n_1-$nsubj$-v_1	people die
n_1-$nsubj$-v_1-$dobj$-n_2	earthquake hit Mexico
n_1-$nsubj$-v_1-$xcomp$-a	residents felt scared
n_1-$nsubj$-v_1-$xcomp$-v_2-$dobj$-n_2	he wants to drink water
n_1-$nsubjpass$-v_1	people was injured

Table 5: Several event mention patterns and the corresponding examples. ('v' stands for verbs, 'n' stands for nouns, and 'a' stands for adjectives. 'nsubj', 'dobj', 'xcomp', and 'nsubjpass' are syntactic relations)

event mention merging, we set α = 0.3, β = 0.03, and the number of iterations to 10. All the extracted events are grouped. We do not manually de-duplicate events. For salient event selection, we select those events with salience scores ranked in the top 20, which can cover the main content of the texts. For salient event ordering, we score and rank each salient event. The number of events finally output is the same as the groundtruth.