Corpus-based Open-Domain Event Type Induction

Jiaming Shen, Yunyi Zhang, Heng Ji, Jiawei Han
Department of Computer Science, University of Illinois Urbana-Champaign, IL, USA
{js2, yzhan238, hengji, hanj}@illinois.edu

Abstract

Traditional event extraction methods require predefined event types and their corresponding annotations to learn event extractors. These prerequisites are often hard to be satisfied in real-world applications. This work presents a corpus-based open-domain event type induction method that automatically discovers a set of event types from a given corpus. As events of the same type could be expressed in multiple ways, we propose to represent each event type as a cluster of ⟨predicate sense, object head⟩ pairs. Specifically, our method (1) selects salient predicates and object heads, (2) disambiguates predicate senses using only a verb sense dictionary, and (3) obtains event types by jointly embedding and clustering ⟨predicate sense, object head⟩ pairs in a latent spherical space. Our experiments, on three datasets from different domains, show our method can discover salient and high-quality event types, according to both automatic and human evaluations[1].

1 Introduction

One step towards converting massive unstructured text into structured, machine-readable representations is event extraction—the identification and typing of event triggers and arguments in text. Most event extraction methods (Ahn, 2006; Ji and Grishman, 2008; Du and Cardie, 2020; Li et al., 2021) assume a set of predefined event types and their corresponding annotations are curated by human experts. This annotation process is expensive and time-consuming. Besides, those manually-defined event types often fail to generalize to new domains. For example, the widely used ACE 2005 event schemas[2] do not contain any event type about `Transmit Virus` or `Treat Disease` and thus cannot be readily applied to extract pandemic events.

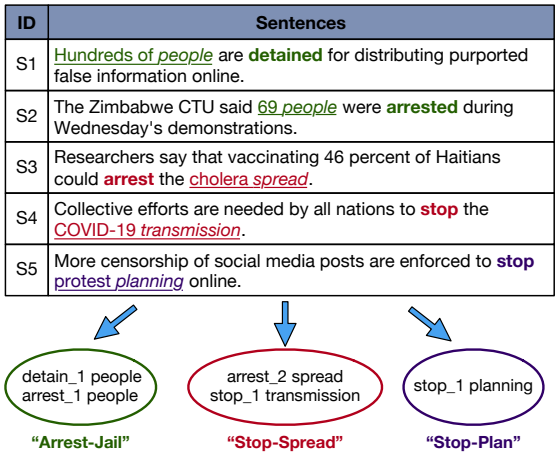

Figure 1: Motivating example sentences and induced event types. **Predicates** are in bold. Objects are underlined and *object heads* are in italics. Colors indicate event types. The suffix number followed by each predicate verb lemma indicates the predicate verb sense.

To automatically induce event schemas from raw text, researchers have studied ad-hoc clustering-based algorithms (Sekine, 2006; Chambers and Jurafsky, 2011) and probabilistic generative methods (Chambers, 2013; Cheung et al., 2013; Nguyen et al., 2015) to discover a set of event types and argument roles. These methods typically utilize bag-of-word text representations and impose strong statistical assumptions. Huang et al. (2016) relax those restrictions using a pipelined approach that leverages extensive lexical and semantic resources (e.g., FrameNet (Baker et al., 1998), VerbNet (Schuler and Palmer, 2005), and PropBank (Palmer et al., 2005)) to discover event schemas. While being effective, this method is limited by the scope of external resources and accuracies of its preprocessing tools. Recently, some studies (Huang et al., 2018; Lai and Nguyen, 2019; Huang and Ji, 2020) have used transfer learning to

[1]The programs, data and resources are publicly available for research purpose at https://github.com/mickeystroller/ETypeClus.

[2]https://www.ldc.upenn.edu/collaborations/past-projects/ace

Datasets	ACE	ERE	RAMS
# of All Event Types	33	38	138
# of Verb Triggered Event Types	33	38	133
# of Verb Frequently Triggered Event Types	28	36	124

Table 1: Statistics of verb triggered event types in three popular event extraction datasets. Event types triggered by verbs more than 5 times are considered as "Verb Frequently Triggered Event Types".

extend traditional event extraction models to new types without explicitly deriving schemas of new event types. Nevertheless, these methods still require many annotations for a set of seen types.

In this work, we study the problem of *event type induction* which aims to discover a set of salient event types based on a given corpus. We observe that about 90% of event types can be frequently triggered by predicate verbs (c.f. Table 1) and thus propose to take a *verb-centric view* toward inducing event types. We use the five sentences (S1-S5) in Figure 1 to motivate our design of event type representation. First, we observe that verb lemma itself might be ambiguous. For example, the two mentions of lemma "*arrest*" in S2 and S3 have different senses and indicate different event types. Second, even for predicates with the same sense, their different associated object heads[3] could lead them to express different event types. Taking S4 and S5 as examples, two "*stop*" mentions have the same sense but belong to different types because of their corresponding object heads. Finally, we can see that people have multiple ways to communicate the same event type due to the language variability.

From the above observations, we propose to represent an event type as a cluster of ⟨predicate sense, object head⟩ pairs (P-O pairs for short)[4]. We present a new event type induction framework **ETypeClus** to automatically discover event types, customized for a specific input corpus. ETypeClus requires no human-labeled data other than an existing general-domain verb sense dictionary such as VerbNet (Schuler and Palmer, 2005) and OntoNotes Sense Groupings (Hovy et al., 2006). ETypeClus contains four major steps. First, it extracts ⟨predicate, object head⟩ pairs from the input corpus based on sentence dependency tree structures. As some extracted pairs could be too general (e.g., ⟨say, it⟩) or too specific (e.g., ⟨document, microcephaly⟩), the second step of ETypeClus will identify salient predicates and object heads in the corpus. After that, we disambiguate the sense of each predicate verb by comparing its usage with those example sentences in a given verb sense dictionary. Finally, ETypeClus clusters the remaining salient P-O pairs into event types using a latent space generative model. This model jointly embeds P-O pairs into a latent spherical space and performs clustering within this space. By doing so, we can guide the latent space learning with the clustering objective and enable the clustering process to benefit from the well-separated structure of the latent space.

We show our ETypeClus framework can save annotation cost and output corpus-specific event types on three datasets. The first two are benchmark datasets ACE 2005 and ERE (Entity Relation Event) (Song et al., 2015). ETypeClus can successfully recover predefined types and identify new event types such as `Build` in ACE and `Bombing` in ERE. Furthermore, to test the performance of ETypeClus in new domains, we collect a corpus about the disease outbreak scenario. Results show that ETypeClus can identify many interesting fine-grained event types (e.g., `Vaccinate`, `Test`) that align well with human annotations.

Contributions. The major contributions of this paper are summarized as follows: (1) A new event type representation is created as a cluster of ⟨predicate sense, object head⟩ tuples; (2) a novel event type induction framework ETypeClus is proposed that automatically disambiguates predicate senses and learns a latent space with desired event cluster structures; and (3) extensive experiments on three datasets verify the effectiveness of ETypeClus 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. A **corpus** $\mathcal{S} = \{S_1, \ldots, S_N\}$ is a set of sentences where each sentence $S_i \in \mathcal{S}$ is a word sequence $[w_{i,1}, \ldots, w_{i,n}]$. A **predicate** is a *verb mention in a sentence* and can optionally have an associated **object** in the same sentence. We follow previous stud-

[3] Intuitively, the object head is the most essential word in the object such as "*people*" in object "*hundreds of people*".

[4] Subjects are intentionally left here because (Allerton, 1979) finds objects play a more important role in determining predicate semantics. Also, many P-O pairs indicate the same event type but share different subjects (e.g., "*police capture X*" and "*terrorists capture X*" are considered as two different events but belong to the same event type `Capture Person`. Adding subjects may help divide current event types into more fine-grained types and we leave this for future work.

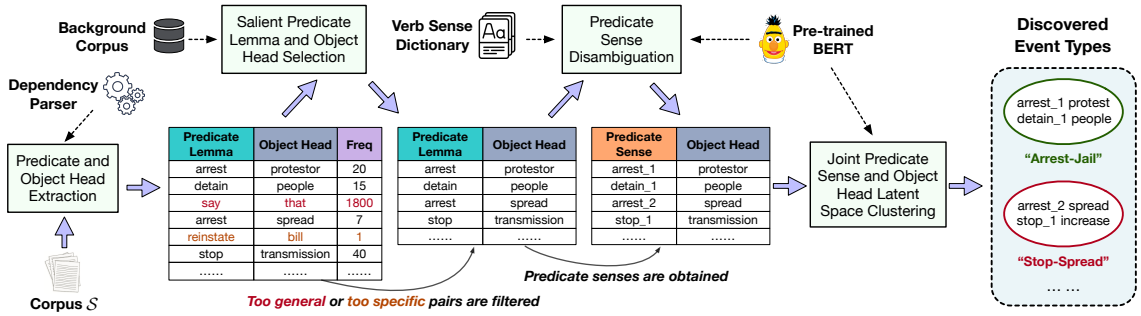

Figure 2: Our ETYPECLUS framework overview.

ies (Corbett et al., 1993; O'Gorman et al., 2016) and refer to the most important word in the object as the **object head**. For example, one predicate from the first sentence in Figure 1 is "*detain*" and its corresponding object is "*hundreds of people*" with the word "*people*" being the object head.

As predicates with the same lemma may have different senses, we disambiguate each predicate verb based on a **verb sense dictionary** $\mathcal{V}$ wherein each verb lemma has a list of candidate senses with example usage sentences. One illustrative example of our verb sense dictionary is shown in Figure 3. We refer to the sense of predicate verb lemma as the **predicate sense**.

Task Definition. Given a corpus $\mathcal{S}$ and a verb sense dictionary $\mathcal{V}$, our task of *event type induction* is to identify a set of K event types where each type T_j is represented by a cluster of ⟨predicate sense, object head⟩ pairs.

Arrest; 3 senses

Sense 1: Catch and take into custody
Example 1: He was **arrested** when customs officers found drugs in his bag.
Example 2: The police **arrested** her for drinking and driving.
Example 3: Airport officials were **arrested** after a major heist.

Sense 2: Stop or interrupt something
Example 1: The treatment has so far done little to **arrest** the spread of the cancer.
Example 2: The look in his eyes **arrested** him on the spot.
Example 3: The mechanism will **arrest** the motion of the flywheel.

Sense 3: Take a hold and capture suddenly
Example 1: An astonishing sight **arrested** our attention.
Example 2: The musician had **arrested** his interest at first glance.

Figure 3: One example in verb sense dictionary $\mathcal{V}$.

3 The ETYPECLUS Framework

The ETYPECLUS framework (outlined in Figure 2) induces event types in four major steps: (1) predicate and object head extraction, (2) salient predicate lemma and object head selection, (3) predicate sense disambiguation, and (4) latent space joint predicate sense and object head clustering.

3.1 Predicate and Object Head Extraction

We propose a lightweight method to extract predicates and object heads in sentences without relying on manually-labeled training data. Specifically, given a sentence S_i, we first use a dependency parser[5] to obtain its dependency parse tree and select all non-auxiliary verb tokens[6] as our candidate predicates. Then, for each candidate predicate, we check its dependent words and if any of them has a dependency label `auxpass`, we believe this predicate verb is in passive voice and find its object heads within its syntactic children that occur before it and have a dependency label in SUBJECT label set[7]. Otherwise, we consider this predicate is in active voice and identify its object heads within its dependents that occur after it and have a dependency label in OBJECT label set[8]. Finally, we aggregate all ⟨predicate, object head⟩ pairs along with their frequencies in the corpus.

3.2 Salient Predicate Lemma and Object Head Selection

The above extracted ⟨predicate, object head⟩ pairs have different qualities. Some are too general and contain little information, while others are too specific and hard to generalize. Thus, this step of ETYPECLUS tries to select those salient predicate lemmas and object heads from our input corpus.

We compute the salience of a word (either a predicate lemma or an object head) based on two criteria. First, it should appear frequently in our corpus. Second, it should not be too frequent in a large general-domain background corpus[9]. Computationally, we follow the TF-IDF idea and define

[5] We use the Spacy `en_core_web_lg` model.

[6] A token with part-of-speech tag `VERB` and dependency label not equal to `aux` and `auxpass`.

[7] {`nsubj(pass)`, `csubj(pass)`, `agent`, `expl`}

[8] {`dobj`, `dative`, `attr`, `oprd`}

[9] We use the English Wikipedia 20171201 dump as our background corpus.

the word salience as follows:

$$Salience(w) = \big(1 + \log(freq(w))^2\big) \log(\frac{N\_bs}{bsf(w)}), \quad (1)$$

where $freq(w)$ is the frequency of word w, N_bs is the number of background sentences, and $bsf(w)$ is the background sentence frequency of word w. Finally, we select those terms with salience scores ranked in top 80% as our salient predicate lemmas and object heads. Table 2 lists the top 5 most salient predicate lemmas and object heads in three datasets. The first two datasets contain news articles about wars and thus terms like "*kill*" and "*weapon*" are ranked top. The third dataset includes articles about disease outbreaks and thus most salient terms include "*infect*", "*virus*", and "*outbreak*".

3.3 Predicate Sense Disambiguation

As verbs typically exhibit large sense ambiguities, we disambiguate each predicate's sense in the sentence. Huang et al. (2016) achieves this goal by utilizing a supervised word sense disambiguation tool (Zhong and Ng, 2010) to link each predicate to a WordNet sense (Miller, 1995) and then mapping that sense back to an OntoNotes sense grouping (Hovy et al., 2006). In this work, we propose to remove such extra complexity and present a lightweight sense disambiguation method that requires only a verb sense dictionary.

The key idea of our method is to compare the usage of a predicate with each verb sense's example sentences in the dictionary. Given a predicate verb v in sentence S_i, we compute two types of features to capture both its *content* and *context* information. The first one, denoted as $\mathbf{v}^{emb}$, is obtained by feeding the sentence S_i into the BERT-Large model (Devlin et al., 2019) and retrieving the predicate's corresponding contextualized embedding. The second feature v^{mwp} is a rank list of 10 alternative words that can be used to replace v in sentence S_i. Specifically, we replace the original word v in S_i with a special `[MASK]` token and feed the masked sentence S_i^{mask} into BERT-Large for masked word prediction. From the prediction results, we select the top 10 most likely words and sort them into v^{mwp}.

After obtaining the predicate representation, we compute the representations of its candidate senses in the dictionary. Suppose the lemma of this predicate v has N_v candidate senses in the dictionary and each sense E_j, $j \in [1, \ldots, N_v]$ has N_j example sentences $\{S_{j,k}\}_{k=1}^{N_j}$ in the dictionary. Then, within each example sentence $S_{j,k}$, we locate where the predicate lemma v occurs and compute its corresponding feature $\mathbf{v}_{j,k}^{emb}$ and $\mathsf{v}_{j,k}^{mwp}$ similarly as discussed before. After that, we obtain two types of features for each sense E_j as follows:

$$\mathbf{E}_j^{emb} = \frac{1}{N_j} \sum_{k=1}^{N_j} \mathbf{v}_{j,k}^{emb}, \quad \mathsf{E}_j^{mwp} = RA(\{\mathsf{v}_{j,k}^{mwp}\}|_{k=1}^{N_j}), \quad (2)$$

where $RA(\cdot)$ stands for the rank aggregation operation based on mean reciprocal rank. This method is widely used in previous literature (Shen et al., 2017, 2020; Zhang et al., 2020; Huang et al., 2020) for fusing ranked lists. Finally, we choose the sense that is most similar to the predicate v as follows:

$$j^* = \underset{j \in [1,\ldots,N_v]}{\arg\max} \cos(\mathbf{v}^{emb}, \mathbf{E}_j^{emb}) \cdot \mathrm{rbo}(\mathsf{v}^{mwp}, \mathsf{E}_j^{mwp}), \quad (3)$$

where $\cos(\mathbf{x}, \mathbf{y})$ is the cosine similarity between two vectors $\mathbf{x}$ and $\mathbf{y}$, and $\mathrm{rbo}(\mathsf{a}, \mathsf{b})$ is the rank-biased overlap similarity (Webber et al., 2010) between two ranked lists.

We evaluate our method on the verb subset of standard word sense disambiguation benchmarks (Navigli et al., 2017). Our method achieves 55.7% F1 score. In comparison, the supervised IMS method in (Huang et al., 2016) gets a 56.9% F1 score. Thus, we think our method is comparable to supervised IMS but being more lightweight and requires no training data.

ACE		ERE		Pandemic	
PredL	ObjH	PredL	ObjH	PredL	ObjH
kill	weapon	pay	money	infect	virus
pay	iraqis	kill	people	suspect	outbreak
guess	nations	rape	kid	sicken	vaccine
convict	states	send	weapon	test	case
fire	marines	attack	cadre	circulate	infection

Table 2: Top 5 salient predicate lemmas (PredL) and object heads (ObjH) in three datasets.

3.4 Latent Space Joint Predicate Sense and Object Head Clustering

After obtaining salient ⟨predicate sense, object head⟩ pairs (P-O pairs for short), we aim to cluster them into event types. Below, we first discuss how to obtain the initial features for predicate senses and object heads (Section 3.4.1). As those predicate senses and object heads are living in two separate spaces, we aim to fuse them into one joint feature space wherein the event cluster structures are better preserved. We achieve this goal by proposing a latent space generative method that jointly embeds

P-O pairs into a unified spherical space and performs clustering in this space. Finally, we discuss how to train this generative model in Section 3.4.3.

3.4.1 Initial Feature Acquisition

We obtain two types of features for each term w (either a predicate sense w_p or an object head w_o) by first locating its mentions in the corpus and then aggregating mention-level representations into term-level features. Suppose term w appears M_w times, for each of its mentions $m_{w,l}, l \in [1, \ldots, M_w]$, we extract this mention's content feature $\mathbf{m}^{emb}_{w,l}$ and context feature $\mathsf{m}^{mwp}_{w,l}$, following the same process discussed in Section 3.3. Then, we average all mentions' content features into this term's content feature $\mathbf{m}^{emb}_{w} = \frac{1}{M_w}\sum_{l=1}^{M_w} \mathbf{m}^{emb}_{w,l}$.

The aggregation of mention context features is more difficult as each $\mathsf{m}^{mwp}_{w,l}$ is not a numerical vector but instead a set of words predicted by BERT to replace $m_{w,l}$. In this work, we propose the following aggregation scheme. For each term w, we first construct a pseudo document D_w using the bag union operation[10]. Then, we obtain the vector representations of pseudo documents based on TF-IDF transformation and apply Principal Component Analysis (PCA) to reduce the dimensionality of document vectors. A similar idea is discussed before in (Amrami and Goldberg, 2018). The resulting vector will be considered as the term's context feature vector $\mathbf{m}^{mwp}_{w}$. Finally, we concatenate $\mathbf{m}^{emb}_{w}$ with $\mathbf{m}^{mwp}_{w}$ to obtain the initial feature vector of predicate senses (denoted as $\mathbf{h}_p$) and object heads (denoted as $\mathbf{h}_o$).

3.4.2 Latent Space Generative Model

To cluster P-O pairs into K event types based on two separate feature spaces ($\mathbf{H}_p$ for predicate sense and $\mathbf{H}_o$ for object head), one straightforward approach is to represent each P-O pair $x = (p, o)$ as $\mathbf{x} = [\mathbf{h}_p, \mathbf{h}_o]$ and directly applying clustering algorithms to all pairs. However, this approach cannot guarantee the concatenated space $\mathbf{H} = [\mathbf{H}_p, \mathbf{H}_o]$ will be naturally suited for clustering. Therefore, we propose to jointly embed and cluster P-O pairs in latent space $\mathbf{Z}$. By doing so, we can unify two feature spaces $\mathbf{H}_p$ and $\mathbf{H}_o$. More importantly, the latent space learning is guided by the clustering objective, and the clustering process can benefit from the well-separated structure of the latent space, which achieves a mutually-enhanced effect.

[10]Namely, D_w contains a word T times if this word appears in T different $\mathsf{m}^{mwp}_{w,l}, l \in [1, \ldots, M_w]$.

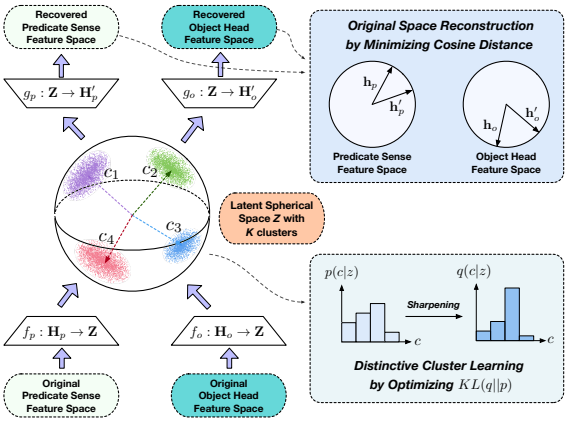

Figure 4: Overview of joint predicate sense and object head latent spherical space clustering. Detailed descriptions in Section 3.4.

We design the latent space to have a spherical topology because cosine similarity more naturally captures word/event semantic similarities than Euclidean/L2 distance. Previous studies (Meng et al., 2019a, 2020) also show that learning spherical embeddings directly is better than first learning Euclidean embeddings and normalizing them later. Thus, we assume there is a spherical latent space $\mathbf{Z}$ with K clusters[11]. Each cluster in this space corresponds to one event type and is associated with a von Mises-Fisher (vMF) distribution (Banerjee et al., 2005) from which event type representative P-O pairs are generated. The vMF distribution of an event type c is parameterized by a mean vector $\mathbf{c}$ and a concentration parameter κ. A unit-norm vector $\mathbf{z}$ is generated from $\text{vMF}_d(\mathbf{c}, \kappa)$ with the probability as follows:

$$p(\mathbf{z}|\mathbf{c}, \kappa) = n_d(\kappa)\exp(\kappa \cdot \cos(\mathbf{z}, \mathbf{c})), \tag{4}$$

where d is the dimensionality of latent space $\mathbf{Z}$ and $n_d(\kappa)$ is a normalization constant.

Each P-O pair (p_i, o_i) with the initial feature $[\mathbf{h}_{p_i}, \mathbf{h}_{o_i}] \in \mathbf{H}_p \times \mathbf{H}_o$ is assumed to be generated as follows: (1) An event type c_k is sampled from a uniform distribution over K types; (2) a latent embedding $\mathbf{z}_i$ is generated from the vMF distribution associated with c_k; and (3) a function g_p (g_o) maps the latent embedding $\mathbf{z}_i$ to the original embedding $\mathbf{h}_{p_i}$ ($\mathbf{h}_{o_i}$) corresponding to the predicate sense p_i (object head o_i). Namely, we have:

$$\begin{aligned} c_k \sim \text{Uniform}(K), \quad & \mathbf{z}_i \sim \text{vMF}_d(\mathbf{c}_k, \kappa), \\ \mathbf{h}_{p_i} = g_p(\mathbf{z}_i), \quad & \mathbf{h}_{o_i} = g_o(\mathbf{z}_i). \end{aligned} \tag{5}$$

[11]K is a hyper-parameter. We can either set K to the true number of event types (if it is known) or directly set K based on application-specific knowledge or adopt statistical methods to estimate K. In practice, we can set it to a relatively high number and the resulting event types are still useful.

We parameterize g_p and g_o as two deep neural networks and jointly learn the mapping function $f_p : \mathbf{H}_p \rightarrow \mathbf{Z}$ as well as $f_o : \mathbf{H}_o \rightarrow \mathbf{Z}$ from the original space to the latent space. Such a setup closely follows the autoencoder architecture (Hinton and Zemel, 1993) which is shown to be effective for preserving input information.

3.4.3 Model Training

We learn our generative model by jointly optimizing two objectives. The first one is a *reconstruction objective* defined as follows:

$$\mathcal{O}_{\text{rec}} = \sum_{i=1}^{N} \Big(\cos(\mathbf{h}_{p_i}, g_p(f_p(\mathbf{h}_{p_i}))) + \cos(\mathbf{h}_{o_i}, g_o(f_o(\mathbf{h}_{o_i})))\Big) \tag{6}$$

This objective encourages our model to preserve input space semantics and generate the original data faithfully.

The second *clustering-promoting objective* enforces our model to learn a latent space with K well-separated cluster structures. Specifically, we use an expectation-maximization (EM) algorithm to sharpen the posterior event type distribution of each input P-O pair. In the expectation step, we first compute the posterior distribution based on current model parameters as follows:

$$\begin{aligned} p(c_k|\mathbf{z}_i) &= \frac{p(\mathbf{z}_i|c_k)p(c_k)}{\sum_{k'=1}^{K} p(\mathbf{z}_i|c_{k'})p(c_{k'})} \\ &= \frac{\exp(\kappa \cdot \cos(\mathbf{z}_i, \mathbf{c}_k))}{\sum_{k'=1}^{K} \exp(\kappa \cdot \cos(\mathbf{z}_i, \mathbf{c}_{k'}))}. \end{aligned} \tag{7}$$

We then compute a new estimate of each P-O pair's cluster assignment $q(c_k|\mathbf{z}_i)$ and use it to update the model in the maximization step. Instead of making hard cluster assignments like K-means which directly assigns each $\mathbf{z}_i$ to its closest cluster, we compute a soft assignment $q(c_k|\mathbf{z}_i)$ as follows:

$$q(c_k|\mathbf{z}_i) = \frac{p(c_k|\mathbf{z}_i)^2/s_k}{\sum_{k'=1}^{K} p(c_{k'}|\mathbf{z}_i)^2/s_{k'}}, \tag{8}$$

where $s_k = \sum_{i=1}^{N} p(c_k|\mathbf{z}_i)$. This squaring-then-normalizing formulation has a sharpening effect that skews the distribution towards its most confident cluster assignment, as shown in (Xie et al., 2016; Meng et al., 2018, 2019b). The formulation encourages unambiguous assignment of P-O pairs to event types so that the learned latent space will have gradually well-separated cluster structures. Finally, in the maximization step, we update the model parameters to maximize the expected log-probability of the current cluster assignments under the new cluster assignment estimates as follows:

$$\mathcal{O}_{\text{clus}} = \sum_{i=1}^{N} \sum_{k=1}^{K} q(c_k|\mathbf{z}_i) \log p(c_k|\mathbf{z}_i), \tag{9}$$

where p is updated to approximate fixed target q.

We summarize our training procedure in Algorithm 1. We first pretrain the model using only the reconstruction objective, which provides a stable initialization of all parameterized mapping functions. Then, we apply the EM algorithm to iteratively update all mapping functions and event type parameters $\mathbf{C}$ with a joint objective $\mathcal{O}_{\text{rec}} + \lambda\mathcal{O}_{\text{clus}}$ where the hyper-parameter λ balances two objectives. The algorithm is considered converged if less than $\delta = 5\%$ of the P-O pairs change cluster assignment between two iterations or a maximum iteration number is reached. Finally, we output each P-O pair's distribution over K event types.

Algorithm 1: Latent Space Generative Model Training.

Input: A set of P-O pairs $\{x_i\}_{i=1}^{N}$; Initial feature spaces $\mathbf{H}_p$ and $\mathbf{H}_o$; # of event types K.
Output: Event-pair distributions $p(x_i|c_k)$.

```
f_o, f_p, g_o, g_p ← max O_rec in Eq. (6)  // Pretraining;
Initialize C = {c_k}_{k=1}^K;
while not converaged do
    // Update cluster assignment estimation;
    q(c_k|z_i) ← Eq. (8);
    // Update model parameteres;
    f_o, f_p, g_o, g_p, C ← max O_rec + λO_clus;
Return p(x_i|c_k) = p(z_i|c_k);
```

4 Evaluation on ACE/ERE Datasets

We first evaluate ETYPECLUS on two widely used event extraction datasets: ACE (Automatic Content Extraction) 2005[12] and ERE (Entity Relation Event) (Song et al., 2015). For both datasets, we follow the same preprocessing steps from (Lin et al., 2020; Li et al., 2021) and use sentences in the training split as our input corpus. The ACE dataset contains 17,172 sentences with 33 event types and the ERE dataset has 14,695 sentences with 38 types. We test the performance of ETYPECLUS on event type discovery and event mention clustering.

4.1 Event Type Discovery

We apply ETYPECLUS on each input corpus to discover 100 candidate event clusters and follow (Huang et al., 2016) to manually check whether

[12] https://www.ldc.upenn.edu/collaborations/past-projects/ace

Event Type	Top Ranked P-O Pairs	Example Sentences in Corpus
Arrest-Jail	⟨arrest_0, protester⟩ ⟨arrest_0, militant⟩ ⟨arrest_0, suspect⟩	• For the most part the marches went off peacefully, but in New York a small group of *protesters* were **arrested** after they refused to go home at the end of their rally, police sources said. • On Tuesday, Saudi security officials said three suspected al-Qaida *militants* were **arrested** in Jiddah, Saudi Arabia.
Build$^{\nabla}$	⟨build_0, facility⟩ ⟨build_0, center⟩ ⟨build_0, housing⟩	• Plans were underway to **build** destruction *facilities* at all other locations but now the Bush junta has removed from its proposed defense budget for fiscal year 2006 all but the minimum funding. • Virginia is apparently going to be **build** a data *center* in Richmond, a back-up data center, and a help desk/call center as a follow-on to the creation of VITA, the Virginia Information Technology Agency.
Transfer-Money	⟨fund_0, activity⟩ ⟨fund_0, operation⟩ ⟨fund_0, people⟩	• The grants will **fund** advisory *activities*, including local capacity building, infrastructure development and product development. • The White House had hoped to hold off asking for more money to **fund** military *operations* in Iraq and Afghanistan until after the election, but with costs rising faster than expected, it sent a request for an early installment of $25 billion to Congress this week.
Bombing$^{\nabla}$	⟨bomb_0, factory⟩ ⟨bomb_0, checkpoint⟩ ⟨bomb_0, base⟩	• He **bombed** the Aspirin *factory* in 1998 (which turned out to have nothing to do with Bin Laden) the week he revealed he had been lying to us for eight months about Lewinsky. • Prosecutors then also pointed to the men's suicide bomber training in 2011 in Somalia and association with Beledi, who prosecutors said **bombed** a government *checkpoint* in Mogadishu that year.

Table 3: Example outputs of ETYPECLUS discovered event types with their associated sentences in ACE and ERE datasets. The first two types come from ACE and the remaining two are from ERE. The event types with superscript "$^{\nabla}$" originally do not exist in human-labeled schemas and are discovered by ETYPECLUS framework. **Predicates** are in bold and *object heads* are underlined and in italics.

Methods	ACE				ERE			
	ARI (std)	NMI (std)	ACC (std)	BCubed-F1 (std)	ARI (std)	NMI (std)	ACC (std)	BCubed-F1 (std)
Kmeans	26.27 (1.60)	48.02 (1.55)	41.57 (3.07)	41.33 (1.75)	11.17 (1.83)	35.10 (2.36)	31.65 (1.82)	29.97 (1.79)
sp-Kmeans	26.06 (2.12)	47.30 (1.65)	40.41 (2.46)	39.52 (1.42)	13.62 (2.14)	37.33 (2.25)	33.28 (3.12)	30.73 (2.03)
AggClus	24.45 (0.00)	45.71 (0.00)	41.00 (0.00)	40.20 (0.00)	6.07 (0.00)	29.62 (0.00)	30.84 (0.00)	29.90 (0.00)
Triframes (Ustalov et al., 2018)	19.35 (6.60)	36.38 (4.91)	—	38.91 (2.36)	10.89 (2.51)	34.94 (2.54)	—	33.53 (4.47)
JCSC (Huang et al., 2016)	36.10 (4.96)	49.50 (2.70)	46.17 (3.64)	43.83 (3.17)	17.07 (4.40)	39.50 (3.97)	33.76 (2.43)	34.04 (2.23)
ETYPECLUS	**40.78 (3.20)**	**57.57 (2.40)**	**48.35 (2.55)**	**51.58 (2.50)**	**24.09 (1.93)**	**49.40 (1.37)**	**41.19 (1.87)**	**39.78 (1.45)**

Table 4: Event mention clustering results. All values are in percentage. We run each method 10 times and report its averaged result for each metric with the standard deviation. Note that ACC is not applicable for Triframes because it assumes the equal number of clusters in ground truth and generated results.

discovered clusters can reconstruct ground truth event types. On ACE, we recover 24 out of 33 event types (19 out of 20 most frequent types) and 7 out of 9 missing types have a frequency less than 10. On ERE, we recover 28 out of 38 event types (18 out of 20 most frequent types). We show some example clusters in Table 3 which includes top ranked P-O pairs and their occurring sentences. We observe that ETYPECLUS successfully identifies human defined event types (e.g., `Arrest-Jail` in ACE and `Transfer-Money` in ERE). It can also identify finer-grained types compared with the original ground truth types (e.g., the 4th row of Table 3 shows one discovered event type `Bombing` in ERE which is in finer scale than "Conflict:Attack", the closest human-annotated type in ERE). Further, ETYPECLUS is able to identify new salient event types (e.g., finding new event type `Build` in ACE). Finally, ETYPECLUS not only induces event types but also provides their example sentences, which serve as the *corpus-specific* annotation guidance.

4.2 Event Mention Clustering

We evaluate the effectiveness of our latent space generative model via the event mention clustering task. We first match each event mention with one extracted P-O pair if possible, and select 15 event types with the most matched results[13]. Then, for each selected type, we collect its associated mentions and add them into a candidate pool. We represent each mention using the feature of its corresponding P-O pair. Finally, we cluster all mentions in the candidate pool into 15 groups and evaluate whether they align well with the original 15 types.

The event mention clustering quality also serves as a good proxy of the event type quality. This is because if a method can discover good event types from a corpus, it should also be able to generate good event mention clusters when the ground truth number of clusters is given.

Compared Methods. We compare the following methods: (1) **Kmeans**: A standard clustering algorithm that works in the Euclidean feature space. We run this algorithm with the ground truth number of clusters. (2) **sp-Kmeans**: A variant of Kmeans that clusters event mentions in a spherical space based on the cosine similarity. (3) **AggClus**: A hierarchical agglomerative clustering algorithm with

[13]More details are discussed in Appendix Section D.

Event Type	Top Ranked P-O Pairs	Example Sentences in Corpus
Spread Virus	⟨spread_2, virus⟩ ⟨spread_2, disease⟩ ⟨spread_2, coronavirus⟩	• What is the best way to keep from **spreading** the *virus* through coughing or sneezing? • Farmers quickly mobilized to fight the misperceptions that pigs could **spread** the *disease*. • In the UK, Asians have been punched in the face, accused of **spreading** *coronavirus*.
Prevent Spread	⟨prevent_1, spread⟩ ⟨mitigate_1, spread⟩ ⟨mitigate_1, transmission⟩	• Infection prevention and control measures are critical to **prevent** the possible *spread* of MERS-CoV. • A vaccine can **mitigate** *spread*, but not fully prevent the virus circulating. • Asymptomatic infection could also potentially be directly harnessed to **mitigate** *transmission*.
Vaccinate People	⟨vaccinate_0, person⟩ ⟨immunize_0, people⟩ ⟨vaccinate_0, family⟩	• All *persons* in a recommended vaccination target group should be **vaccinated** with the 2009 H1N1 monovalent vaccine and the seasonal influenza vaccine. • U.K. Will Start **Immunizing** *People* Against COVID-19 On Tuesday, Officials Say. • "..." says Henrietta Aviga, a nurse travelling around villages to **vaccinate** and educate *families*.

Table 5: Example outputs of ETYPECLUS discovered event types with their associated sentences in the corpus. **Predicates** are in bold and *object heads* are underlined and in italics.

Methods	K-Menas	AggClus	JCSC	ETYPECLUS
Accuracy	86.7	64.4	54.4	**91.1**

Table 6: Intrusion test results in percentage.

Euclidean distance function and Ward linkage. A stop criterion is set to be reaching the target number of clusters. (4) **Triframes** (Ustalov et al., 2018): A graph-based clustering algorithm that constructs a k-NN event mention graph and uses a fuzzy graph clustering algorithm WATSET to generate the clusters. (5) **JCSC** (Huang et al., 2016): A joint constrained spectral clustering method that iteratively refines the clustering result with a constraint function to enforce inter-dependent predicates and objects to have coherent clusters. (6) **ETYPECLUS**: Our proposed latent space joint embedding and clustering algorithm. For fair comparison, all methods start with the same $[\mathbf{h}_p, \mathbf{h}_o]$ embeddings as described in Section 3.4.2. More implementation details and hyper-parameter choices are discussed in Appendix Sections A and B.

Evaluation Metrics. We evaluate clustering results with several standard metrics. (1) **ARI** (Hubert and Arabie, 1985) measures the similarity between two cluster assignments based on the number of pairs in the same/different clusters. (2) **NMI** denotes the normalized mutual information between two cluster assignments. (3) **BCubed-F1** (Bagga and Baldwin, 1998) estimates the quality of the generated cluster assignment by aggregating the precision and recall of each element. (4) **ACC** measures the clustering quality by finding the permutation function from predicted cluster IDs to ground truth IDs that gives the highest accuracy. The math formulas of these metrics are in Appendix Section E. For all four metrics, the higher the values, the better the model performance.

Experiment Results. Table 4 shows ETYPECLUS outperforms all the baselines on both datasets in terms of all metrics. The major advantage of ETYPECLUS is the latent event space: different types of information can be projected into the same space for effective clustering. We also observe that JCSC is the strongest among all baselines. We think the reason is that it uses a joint clustering strategy where event types are defined as predicate clusters and the constraint function enables objects to refine predicate clusters. Thus, a predicate-centric clustering algorithm can outperform all other baselines, which supports our verb-centric view of events.

5 Evaluation on Pandemic Dataset

To evaluate the portability of ETYPECLUS to a new open domain, we collect a new dataset that includes 98,000 sentences about disease outbreak events[14]. We run the top-3 performing baselines and ETYPECLUS to generate 30 candidate event types and evaluate their quality using intrusion test. Specifically, we inject a negative sample from other clusters into each cluster's top-5 results and ask three annotators to identify the outlier. More details on how we construct the intrusions are in Appendix. The intuition behind this test is that the annotators will be easier to identify the intruders if the clustering results are clean and tuples are semantically coherent. As shown in Table 6, ETYPECLUS achieves the highest accuracy among all the baseline methods, indicating that it generates semantically coherent types in each cluster.

Table 5 shows some discovered event types of

[14]The detailed creation process is in Appendix Section F.

ETypeClus.[15] Interesting examples include tuples with the same predicate sense but object heads with different granularities (e.g., ⟨spread_2, virus⟩ and ⟨spread_2, coronavirus⟩ for `Spread-Virus` type), tuples with same object head but different predicate senses (e.g., ⟨prevent_1, spread⟩, and ⟨mitigate_1, spread⟩ for `Prevent-Spread` type), and event types with predicate verb lemmas that are not directly linkable to OntoNotes Senses grouping (e.g., "*immunize*" and "*vaccinate*" for `Vaccinate` type).

6 Related Work

Event Schema Induction. Early studies on event schema induction adopt rule-based approaches (Lehnert et al., 1992; Chinchor et al., 1993) and classification-based methods (Chieu et al., 2003; Bunescu and Mooney, 2004) to induce templates from labeled corpus. Later, unsupervised methods are proposed to leverage relation patterns (Sekine, 2006; Qiu et al., 2008) and coreference chains (Chambers and Jurafsky, 2011) for event schema induction. Typical approaches use probabilistic generative models (Chambers, 2013; Cheung et al., 2013; Nguyen et al., 2015; Li et al., 2020, 2021) or ad-hoc clustering algorithms (Huang et al., 2016; Sha et al., 2016) to induce predicate and argument clusters. In particular, (Liu et al., 2019) takes an entity-centric view toward event schema induction. It clusters entities into semantic slots and finds predicates for entity clusters in a post-processing step. (Yuan et al., 2018) studies the event profiling task and includes one module that leverages a Bayesian generative model to cluster ⟨predicate:role:label⟩ triplets into event types. These methods typically rely on discrete hand-crafted features derived from bag-of-word text representations and impose strong statistics assumptions; whereas our method uses pre-trained language models to reduce the feature generation complexity and relaxes stringent statistics assumptions via latent space clustering.

Weakly-Supervised Event Extraction. Some studies on event extraction (Bronstein et al., 2015; Ferguson et al., 2018; Chan et al., 2019) propose to leverage annotations for a few seen event types to help extract mentions of new event types specified by just a few keywords. These methods reduce the annotation efforts but still require all target new types to be given. Recently, some studies (Huang et al., 2018; Lai and Nguyen, 2019; Huang and Ji, 2020) use transfer learning techniques to extend traditional event extraction models to new types without explicitly deriving schemas of new event types. Compared to our study, these methods still require many annotations for a set of seen types and their resulting vector-based event type representations are less human interpretable. Another related work by (Wang et al., 2019) uses GAN to extract events from an open domain corpus. It clusters ⟨entity:location:keyword:date⟩ quadruples related to the same event rather than finds event types.

7 Conclusions and Future Work

In this paper, we study the event type induction problem that aims to automatically generate salient event types for a given corpus. We define a novel event type representation as a ⟨predicate sense, object head⟩ cluster, and propose ETypeClus that can extract and select salient predicates and object heads, disambiguate predicate senses, and jointly embed and cluster P-O pairs in a latent space. Experiments on three datasets show that ETypeClus can recover human curated types and identify new salient event types. In the future, we propose to explore the following directions: (1) improve predicate and object extraction quality with tools of higher semantic richness (e.g., a SRL labeler or an AMR parser); (2) leverage more information from lexical resources to enhance event representation; and (3) cluster objects into argument roles for each discovered event type.

Acknowledgements

Research was supported in part by US DARPA KAIROS Program No. FA8750-19-2-1004, SocialSim Program No. W911NF-17-C-0099, and INCAS Program No. HR001121C0165, NSF IIS-19-56151, IIS-17-41317, and IIS 17-04532, and the Molecule Maker Lab Institute: An AI Research Institutes program supported by NSF under Award No. 2019897. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and do not necessarily represent the views, either expressed or implied, of DARPA or the U.S. Government. We want to thank Martha Palmer and Ghazaleh Kazeminejad for the help on VerbNet and OntoNotes Sense Groupings. We also would like to thank Sha Li, Yu Meng, Lifu Huang for insightful discussions and anonymous reviewers for valuable feedback.

[15] More example outputs are in Appendix Section H.

Impact Statement

Both event extraction and event type induction are standard tasks in NLP. We do not see any significant ethical concerns. The expected usage of our work is to identify interesting event types from user input corpus such as a set of news articles or a collection of scientific papers.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*.

David J. Allerton. 1979. Essentials of grammatical theory: A consensus view of syntax and morphology.

Asaf Amrami and Yoav Goldberg. 2018. Word sense induction with neural bilm and symmetric patterns. In *EMNLP*.

Amit Bagga and Breck Baldwin. 1998. Entity-based cross-document coreferencing using the vector space model. In *ACL/COLING*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *COLING-ACL*.

Arindam Banerjee, Inderjit S. Dhillon, Joydeep Ghosh, and Suvrit Sra. 2005. Clustering on the unit hypersphere using von mises-fisher distributions. In *JMLR*.

Ofer Bronstein, Ido Dagan, Qi Li, Heng Ji, and Anette Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *ACL*.

Razvan C. Bunescu and Raymond J. Mooney. 2004. Collective information extraction with relational markov networks. In *ACL*.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *ACL*.

Yee Seng Chan, Joshua Fasching, Haoling Qiu, and Bonan Min. 2019. Rapid customization for event extraction. In *ACL*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *HLT-NAACL*.

Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. 2003. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *ACL*.

Nancy Chinchor, Lynette Hirschman, and David D. Lewis. 1993. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Comput. Linguistics*, 19:409–449.

Greville G. Corbett, Norman M. Fraser, Scott McGlashan, et al. 1993. *Heads in grammatical theory*. Cambridge University Press.

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

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *EMNLP*.

James Ferguson, Colin Lockard, Daniel S. Weld, and Hannaneh Hajishirzi. 2018. Semi-supervised event extraction with paraphrase clusters. In *NAACL-HLT*.

Geoffrey E. Hinton and Richard S. Zemel. 1993. AutoEncoders, minimum description length and helmholtz free energy. In *NIPS*.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: The 90% solution. In *HLT-NAACL*.

Jiaxin Huang, Yiqing Xie, Yu Meng, Jiaming Shen, Yunyi Zhang, and Jiawei Han. 2020. Guiding corpus-based set expansion by auxiliary sets generation and co-expansion. In *WWW*.

Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han, and Avirup Sil. 2016. Liberal event extraction and event schema induction. In *ACL*.

Lifu Huang and Heng Ji. 2020. Semi-supervised new event type induction and event detection. In *EMNLP*.

Lifu Huang, Heng Ji, Kyunghyun Cho, and Clare R. Voss. 2018. Zero-shot transfer learning for event extraction. In *ACL*.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

Viet Dac Lai and Thien Nguyen. 2019. Extending event detection to new types with learning from keywords. In *W-NUT@EMNLP*.

Wendy Lehnert, Claire Cardie, David Fisher, John McCarthy, Ellen Riloff, and Stephen Soderland. 1992. University of massachusetts: Muc-4 test results and analysis. In *MUC*.

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 *EMNLP*.

Sha Li, Heng Ji, and Jiawei Han. 2021. Document-level event argument extraction by conditional generation. In *NAACL-HLT*.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *ACL*.

Xiao Liu, Heyan Huang, and Yue Zhang. 2019. Open domain event extraction using neural latent variable models. In *ACL*.

Yu Meng, Jiaxin Huang, Guangyuan Wang, Chao Zhang, Honglei Zhuang, Lance M. Kaplan, and Jiawei Han. 2019a. Spherical text embedding. In *NeurIPS*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019b. Weakly-supervised hierarchical text classification. In *AAAI*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Yu Zhang, Chao Zhang, and Jiawei Han. 2020. Hierarchical topic mining via joint spherical tree and text embedding. In *KDD*.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38:39–41.

Roberto Navigli, José Camacho-Collados, and Alessandro Raganato. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *EACL*.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. 2015. Generative event schema induction with entity disambiguation. In *ACL*.

Timothy J. O'Gorman, Kristin Wright-Bettner, and Martha Palmer. 2016. Richer event description: Integrating event coreference with temporal, causal and bridging annotation. In *Proceedings of the 2nd Workshop on Computing News Storylines*.

Martha Palmer, Paul R. Kingsbury, and Daniel Gildea. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31:71–106.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Gilles Louppe, Peter Prettenhofer, Ron Weiss, Ron J. Weiss, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2008. Modeling context in scenario template creation. In *IJCNLP*.

Karin Schuler and Martha Palmer. 2005. VerbNet: A broad-coverage, comprehensive verb lexicon.

Satoshi Sekine. 2006. On-demand information extraction. In *ACL*.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. 2016. Joint learning templates and slots for event schema induction. In *HLT-NAACL*.

Jiaming Shen, Wenda Qiu, Jingbo Shang, Michelle Vanni, Xiang Ren, and Jiawei Han. 2020. SynSetExpan: An iterative framework for joint entity set expansion and synonym discovery. In *EMNLP*.

Jiaming Shen, Zeqiu Wu, Dongming Lei, Jingbo Shang, Xiang Ren, and Jiawei Han. 2017. SetExpan: Corpus-based set expansion via context feature selection and rank ensemble. In *ECML/PKDD*.

Zhiyi Song, Ann Bies, Stehphanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: Annotation of entities, relations, and events. In *EVENTS@HLP-NAACL*.

Dmitry Ustalov, Alexander Panchenko, Andrey Kutuzov, Chris Biemann, and Simone Paolo Ponzetto. 2018. Unsupervised semantic frame induction using triclustering. In *ACL*.

Rui Wang, Deyu Zhou, and Yulan He. 2019. Open event extraction from online text using a generative adversarial network. In *EMNLP*.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28:20:1–20:38.

Thomas Wolf, Lysandre Debut, Victor Sanh, and Others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *ICML*.

Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, and Jiawei Han. 2018. Open-schema event profiling for massive news corpora. In *CIKM*.

Yunyi Zhang, Jiaming Shen, Jingbo Shang, and Jiawei Han. 2020. Empower entity set expansion via language model probing. In *ACL*.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*.

A ETypeClus Implementation Details

We use the OntoNotes sense grouping[16] as our input verb sense dictionary. The background Wikipedia corpus is obtained from (Shen et al., 2020). We implement our latent space clustering model using PyTorch 1.7.0 with the Huggingface Library (Wolf et al., 2020). To obtain each predicate sense's context information $\mathbf{m}_w^{mwp}$ (c.f. Section 3.4.1 in main text), we leverage PCA to reduce the original pseudo-document multi-hot representations into 500-dimension vectors. The hyper-parameters of our latent space generative model are set as follows: the latent space dimension $d = 100$, the DNN hidden dimensions are 500-500-1000 for encoder f_p/f_o and 1000-500-500 for decoder g_p/g_o; the shared concentration parameter of event type clusters $\kappa = 10$, the weight for clustering-promoting object $\lambda = 0.02$, the convergence threshold $\delta = 0.05$, and the maximum iteration number is 100. We learn the generative model using Adam optimizer with learning rate 0.001 and batch size 64.

B Baseline Implementation Details

We implement Kmeans and AggClus based on the Scikit-learn codebase (Pedregosa et al., 2011). We use L_2 distance for both methods. For Kmeans, we use k-means++ strategy for model initialization, and each time the result with the best inertia is used within 10 initializations. We use ward linkage for AggClus and set the stop criterion to be reaching the target number of clusters. For spherical Kmeans, we use an open source implementation[17]. Similar to Kmeans, we use k-means++ to initialize the model and select the best results among 10 initializations. For Triframes (Ustalov et al., 2018), we use its authors' original implementation[18], and tune the parameter k in the k-NN graph construction step for different tasks and datasets to get a reasonable number of clusters. Specifically, we use $k = 30$ for the event mention clustering task, which gives us the overall best evaluation results on both ACE and ERE. On the Pandemic corpus, we take $k = 100$, which generates 35 clusters that contain at least 40 tuples. For JCSC, we implement the clustering algorithm based on Algorithm 1 in (Huang et al., 2016). The spectral clustering used in JCSC is based on Scikit-learn's implementation, and the label assigning strategy is K-means with 30 random initializations each time.

[16] Available to view at http://verbs.colorado.edu/html_groupings/

[17] https://github.com/jasonlaska/spherecluster

[18] https://github.com/uhh-lt/triframes

C Running Environment

We run all experiments on a single cluster with 80 CPU cores and a Quadro RTX 8000 GPU. The BERT model is moved to the GPU for initial predicate sense and object head feature extraction and it consumes about 11GB GPU memory. We also train our latent space generative model on GPU and it consumes about 14GB GPU memory. In principles, ETypeClus should be runnable on CPU.

D Event Mention Clustering Dataset

We create the evaluation dataset for event mention clustering as follows. First, we select event mentions whose trigger is a single token verb. Then, for each selected event mention, we construct a P-O pair by choosing its non-pronoun argument that has some overlap with the object of our extracted ⟨predicate, object⟩ with the same verb trigger. After that, we select the top-15 event types with the most matched results for both datasets to avoid types with too few mentions, and their corresponding event mentions are used as ground truth clusters.

E Evaluation Metrics for Event Mention Clustering

We denote the ground truth clusters as C^*, the predicted clusters as C, and the total number of event mentions as N.

- **ARI** (Hubert and Arabie, 1985) measures the similarity between two cluster assignments. Let $TP(TN)$ denote the number of element pairs in the same (different) cluster(s) in both C^* and C. Then, ARI is calculated as follows:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}(\text{RI})}{\max \text{RI} - \mathbb{E}(\text{RI})}, \quad \text{RI} = \frac{TP + TN}{N},$$

  where $\mathbb{E}(\text{RI})$ is the expected RI of random assignments.

- **NMI** denotes the normalized mutual information between two cluster assignments and is widely used in previous studies. Let $\text{MI}(\cdot;\cdot)$ be the Mutual Information between two cluster assignments, and $\text{H}(\cdot)$ denote the Entropy. Then the NMI is formulated as follows:

$$\text{NMI} = \frac{2 \times \text{MI}(C^*; C)}{\text{H}(C^*) + \text{H}(C)}.$$

- **BCubed** (Bagga and Baldwin, 1998) estimates the quality of the generated cluster assignment by aggregating the precision and recall of each element. B-Cubed precision, recall, and F1 are thus calculated as follows:

$$\text{BCubed-P} = \frac{1}{N}\sum_{i=0}^{N}\frac{|C(e_i) \cap C^*(e_i)|}{|C(e_i)|}$$

$$\text{BCubed-R} = \frac{1}{N}\sum_{i=0}^{N}\frac{|C(e_i) \cap C^*(e_i)|}{|C^*(e_i)|}$$

$$\text{BCubed-F1} = \frac{2}{\text{BCubed-P}^{-1} + \text{BCubed-R}^{-1}}$$

  where $C^*(\cdot)$ ($C(\cdot)$) is the mapping function from an element to its ground truth (predicted) cluster.

- **ACC** measures the quality of the clustering results by finding the permutation function from predicted cluster IDs to ground truth IDs that gives the highest accuracy. Let y_i (y_i^*) denote the i-th element's predicted (ground truth) cluster ID, the ACC is formulated as follows:

$$\text{ACC} = \max_{\sigma \in Perm(k)} \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(y_i^* = \sigma(y_i))$$

  where k is the number of clusters for both C^* and C, $Perm(k)$ is the set of all permutation functions on the set $\{1, 2, \ldots, k\}$, and $\mathbb{1}(\cdot)$ is the indicator function.

F Pandemic Dataset Creation

We follow a similar approach in (Li et al., 2021) to construct our Pandemic Dataset. First, we resort to Wikipedia lists to get a set of Wikipedia articles related to disease outbreaks[19]. Then, we extract the news article links from the "references" section of those Wikipedia article pages. Finally, we crawl these news articles based on their above extracted links[20] and construct a corpus related to disease outbreaks.

G Intrusion Test Construction

Given the top-5 tuples of each detected type, we inject a randomly sampled tuple from the top results of other types to serve as a negative sample. For methods that have cluster centers, we rank tuples within each cluster by their distances to the center. Otherwise, we rank tuples according to their frequencies in the corpus. Then, the intrusion questions from all compared methods are randomly shuffled to avoid bias. Three annotators[21] are asked to identify the injected tuples independently, and we take the average of their labeling accuracy to show the quality of the generated event types.

H More ETypeClus Outputs

Table 3 and Table 8 list example outputs of ETypeClus on ACE/ERE and Pandemic datasets, respectively.

[19] Specifically, we use the list https://en.wikipedia.org/wiki/List_of_epidemics

[20] We use the crawler tool at https://github.com/codelucas/newspaper.

[21] All three annotators are not in the author list of this paper and provide independent judgements of the tuple quality.

Event Type	Top Ranked ⟨Predicate Sense, Object Head⟩ Pairs	Example Sentences in Corpus
Arrest-Jail	⟨arrest_0, protester⟩ ⟨arrest_0, militant⟩ ⟨arrest_0, suspect⟩	• For the most part the marches went off peacefully, but in New York a small group of *protesters* were **arrested** after they refused to go home at the end of their rally, police sources said. • On Tuesday, Saudi security officials said three suspected al-Qaida *militants* were **arrested** in Jiddah, Saudi Arabia, in sweeps following the near-simultaneous suicide attacks on three residential compounds on the outskirts of Riyadh on May 12. • can owe tell us exactly the details, the precise details of how you **arrested** the *suspect*?
Build$^{\nabla}$	⟨build_0, facility⟩ ⟨build_0, center⟩ ⟨build_0, housing⟩	• Plans were underway to **build** destruction *facilities* at all other locations but now the Bush junta has removed from its proposed defense budget for fiscal year 2006 all but the minimum funding for these destruction projects. • Virginia is apparently going to be **build** a data *center* in Richmond, a back-up data center , and a help desk/call center as a follow-on to the creation of VITA, the Virginia Information Technology Agency. • The Habitat for Humanity might be a good one to consider, since their expertise is in **building** *housing*, which of course is so beadly needed over there at this time.
Transfer-Money	⟨fund_0, activity⟩ ⟨fund_0, operation⟩ ⟨fund_0, people⟩	• The grants will **fund** advisory *activities*, including local capacity building, infrastructure development, product development, and development of local insurance companies' capacity to provide index-based insurance products. • The White House had hoped to hold off asking for more money to **fund** military *operations* in Iraq and Afghanistan until after the election, but with costs rising faster than expected, it sent a request for an early installment of $25 billion to Congress this week. • Watch 'Secret Pakistan' on the BBC iPlayer , it's an awesome two part documentary about how Pakistan has been supporting and **funding** these *people* for years.
Bombing$^{\nabla}$	⟨bomb_0, factory⟩ ⟨bomb_0, checkpoint⟩ ⟨bomb_0, base⟩	• He **bombed** the Aspirin *factory* in 1998 (which turned out to have nothing to do with Bin Laden) the week he revealed he had been lying to us for eight months about Lewinsky. • Prosecutors then also pointed to the men's suicide bomber training in 2011 in Somalia and association with Beledi, who prosecutors said **bombed** a government *checkpoint* in Mogadishu that year. • Once the war breaks out, Iran will immediately use all kinds of missiles to **bomb** the military *bases* of the United States in the Gulf and Israel to pieces.

Table 7: Example outputs of ETYPECLUS discovered event types with their associated sentences in ACE and ERE datasets. The first two types come from ACE and the remaining two are from ERE. The event types with superscript "$^{\nabla}$" originally do not exist in human-labeled schemas and are discovered by ETYPECLUS framework. **Predicates** are in bold and *object heads* are underlined and in italics.

Event Type	Top Ranked ⟨Predicate Sense, Object Head⟩ Pairs	Example Sentences in Corpus
Spread Virus	⟨spread_2, virus⟩ ⟨spread_2, disease⟩ ⟨spread_2, coronavirus⟩	• What is the best way to keep from **spreading** the *virus* through coughing or sneezing? • Farmers quickly mobilized to fight the misperceptions that pigs could **spread** the *disease*. • In the UK, Asians have been punched in the face, accused of **spreading** *coronavirus*.
Wear Mask	⟨wear_1, mask⟩ ⟨wear_1, facemasks⟩ ⟨wear_1, cover⟩	• Pence chose not to **wear** a face *mask* during the tour despite the facility's policy. • It should not be necessary for workers to **wear** *facemasks* routinely when in contact with the public. • The WHO offers a conditional recommendation that health care providers also **wear** a separate head *cover* that protects the head and neck.
Prevent Spread	⟨prevent_1, spread⟩ ⟨mitigate_1, spread⟩ ⟨mitigate_1, transmission⟩	• Infection prevention and control measures are critical to **prevent** the possible *spread* of MERS-CoV in health care facilities . • A vaccine can **mitigate** *spread*, but not fully prevent the virus circulating. • Asymptomatic infection could also potentially be directly harnessed to **mitigate** *transmission*.
Delay Gathering	⟨delay_1, gathering⟩ ⟨postpone_1, gathering⟩ ⟨suspend_1, gathering⟩	• The 2020 edition of the Cannes Film Festival, was left in limbo following an announcement from the festival's organizers that the *gathering* could be **delayed** until late June or early July. • States with EVD should consider **postponing** mass *gatherings* until EVD transmission is interrupted. • On Thursday, leaders of The Church of Jesus Christ of Latter - day Saints told its 15 million members worldwide all public *gatherings* would be **suspended** until further notice .
Provide Testing	⟨provide_1, testing⟩ ⟨conduct_1, testing⟩ ⟨perform_1, testing⟩	• Governments are racing to buy medical equipment as a debate intensifies over **providing** adequate *testing*, when it 's advisable to wear masks, and whether stricter lockdowns should be imposed. • Additional *testing* is being **conducted** to confirm that the family members had H1N1 and to try to verify that the flu was transmitted from human to cat. • Additional laboratories **perform** antiviral *testing* and report their results to CDC .
Warn Country	⟨warn_1, country⟩ ⟨warn_1, authority⟩ ⟨warn_1, government⟩	• WHO uses six phases of alert to communicate the seriousness of infectious threats and to **warn** *countries* of the need to prepare and respond to outbreaks. • The message showed a photo of a letter, written by the operators of the hospital's oxygen supply plant, **warning** the *authorities* that the supply was running dangerously low . • WHO staff concluded there was a high risk of further spread, and issued a global alert to **warn** all member *governments* of the existence of a new and highly infectious form of "atypical pneumonia" on March 12th .
Vaccinate People	⟨vaccinate_0, person⟩ ⟨immunize_0, people⟩ ⟨vaccinate_0, family⟩	• All *persons* in a recommended vaccination target group should be **vaccinated** with the 2009 H1N1 monovalent vaccine and the seasonal influenza vaccine. • U.K. Will Start **Immunizing** *People* Against COVID-19 On Tuesday, Officials Say. • "In the Samoan language there is no word for bacteria or virus" says Henrietta Aviga, a nurse travelling around villages to **vaccinate** and educate *families*.

Table 8: More example outputs of ETYPECLUS discovered event types with their associated sentences in the corpus. **Predicates** are in bold and *object heads* are underlined and in italics.