Joint Extraction of Entities, Relations, and Events via Modeling Inter-Instance and Inter-Label Dependencies

Minh Van Nguyen[1], **Bonan Min**[2], **Franck Dernoncourt**[3], **and Thien Huu Nguyen**[1,4]
[1] Dept. of Computer and Information Science, University of Oregon, Eugene, OR, USA
[2] Raytheon BBN Technologies, USA
[3] Adobe Research, San Jose, CA, USA
[4] VinAI Research, Vietnam
{minhnv,thien}@cs.uoregon.edu,
bonan.min@raytheon.com, dernonco@adobe.com

Abstract

Event trigger detection, entity mention recognition, event argument extraction, and relation extraction are the four important tasks in information extraction that have been performed jointly (Joint Information Extraction - JointIE) to avoid error propagation and leverage dependencies between the task instances (i.e., event triggers, entity mentions, relations, and event arguments). However, previous JointIE models often assume heuristic manually-designed dependency between the task instances and mean-field factorization for the joint distribution of instance labels, thus unable to capture optimal dependencies among instances and labels to improve representation learning and IE performance. To overcome these limitations, we propose to induce a dependency graph among task instances from data to boost representation learning. To better capture dependencies between instance labels, we propose to directly estimate their joint distribution via Conditional Random Fields. Noise Contrastive Estimation is introduced to address the maximization of the intractable joint likelihood for model training. Finally, to improve the decoding with greedy or beam search in prior work, we present Simulated Annealing to better find the globally optimal assignment for instance labels at decoding time. Experimental results show that our proposed model outperforms previous models on multiple IE tasks across 5 datasets and 2 languages.

1 Introduction

To extract structured information from unstructured text, a typical information extraction (IE) pipeline involves four major tasks: event trigger detection (ETD), event argument extraction (EAE), entity mention recognition (EMR), and relation extraction (RE). Previous work has performed such IE tasks via pipelined approaches (Li et al., 2013; Chen et al., 2015; Du and Cardie, 2020; Li et al., 2020), where a model for one task uses output predictions from other models performing other tasks. Consequently, errors from the predictions can be propagated between the models in the pipeline.

Recently, ETD, EMR, EAE, and RE have been solved jointly in a single model, i.e., Joint Information Extraction - JointIE (Wadden et al., 2019; Lin et al., 2020; Nguyen et al., 2021a; Zhang and Ji, 2021), to avoid error propagation and leverage dependency between prediction instances of the four IE tasks (i.e., event trigger, entity mention, relation, and event argument candidates in a sentence). For example, if a *Person* entity mention is a *Victim* argument for a *Die* event, it is likely that the same entity mention is also a *Target* argument for an *Attack* event in the same sentence. To implicitly exploit instance dependency for representation learning, Wadden et al. (2019) and Lin et al. (2020) employ a shared encoder to obtain representation vectors to classify instances of different IE tasks. Later work heuristically captures dependency between IE task instances via explicitly connecting the task instances that share an entity mention or event trigger (Nguyen et al., 2021a) or aligning the task instances that share text spans with some nodes on a semantic graph (Zhang and Ji, 2021) to aid representation learning. While natural, these manual designs for dependency between task instances might not be optimal for representation learning of JointIE.

In addition to representation learning, at the prediction level, previous work tends to factorize the joint distribution of labels for all the task instances in JointIE into the product of label distributions for each individual instance (i.e., performing local normalization), thus hindering the ability to fully exploit the interactions of instance labels across IE tasks. (Lin et al., 2020) and (Zhang and Ji, 2021) mitigate this problem by decoding instance labels with handcrafted global features while (Nguyen et al., 2021a) focuses on encoding label interactions via consistency regularization over global type dependency graphs. However, these approaches still

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4363 - 4374

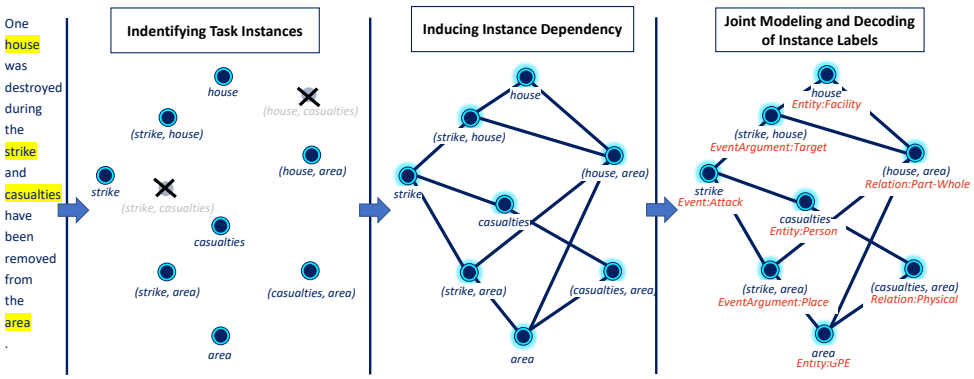

Figure 1: Overview of the three stages in our proposed model: i) identifying task instances, ii) inducing instance dependency, and iii) joint modeling and decoding of instance labels. Each node represents an instance for one of the four IE tasks, and edges (with weights > 0.3) between nodes represent induced instance dependency.

assume a factorization of the joint label distribution for prediction instances, thus unable to fundamentally address the label dependency encoding issue. Recently, some works have attempted to directly model the joint distribution of instance labels by reformulating JointIE tasks as text generation problems using state-of-the-art pre-trained seq2seq models, e.g., BART or T5 (Lewis et al., 2020; Raffel et al., 2020). In such generative models, text spans and labels for task instances are generated by the decoder in an autoregressive fashion to encode label dependency for joint distribution computation (Lu et al., 2021; Hsu et al., 2021). Unfortunately, this approach needs to assume an order of the task instances to be decoded (e.g., from left to right) that disallows later instances in the order to interfere/correct predictions for earlier instances, causing suboptimal performance for JointIE.

In this work, we aim to overcome these issues by inducing dependency between the task instances for JointIE from data to boost representation learning, and directly modeling the joint distribution of the labels for all the task instances to fully enable label interactions. To this end, we consider each task instance as a node in a fully connected dependency graph; the weight for each edge is then learned to capture the dependency level between two corresponding instances. Note that this is different from prior work (Nguyen et al., 2021a; Zhang and Ji, 2021) that heuristically designs sparser dependency graphs with disconnected task instance pairs, thus failing to explore all possible interactions between instance pairs for optimal representations. In our method, the induced dependency graph for instance nodes is then employed by Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017; Nguyen and Grishman, 2018) to enhance the representation for each instance node with information from all the other nodes according to their dependency levels. Afterwards, the enhanced instance representations and the induced dependency graph are utilized to estimate the joint distribution of instance labels via Conditional Random Fields (CRFs) (Lafferty et al., 2001). This formulation enables us to approximately maximize the intractable joint likelihood of the ground-truth instance labels via Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012), which converts the maximization problem into the nonlinear logistic regression discriminating between the true labels and the noise labels.

Finally, previous work for JointIE has employed a greedy or beam search for decoding instance labels, which is not optimal due to their greedy nature. In this work, we propose a novel decoding algorithm for JointIE via Simulated Annealing (SA) (Kirkpatrick et al., 1983), which has been shown to be able to approximate the global optimum of a function (Kirkpatrick et al., 1983; Van Laarhoven and Aarts, 1987). Experimental results show that our proposed model for JointIE significantly outperforms previous models on multiple tasks with large margins across 5 datasets and 2 languages.

2 Problem Statement

Given an input sentence, ETD aims to predict text spans and event types for event triggers based on a predefined set of event types, e.g., *"Attack"* and *"Transport"* (Lai et al., 2020). Similarly, EMR seeks to determine text spans and entity types (e.g., *"Person"*, *"Organization"*) for entity mentions in the sentence (Nguyen et al., 2016b). Different from the

first two tasks, EAE and RE involves predictions for a pair of objects at a time. Given an event trigger and an entity mention, EAE aims to predict the argument role (e.g, *"Victim"*) of the entity mention for the event trigger (Veyseh et al., 2020c). An argument role can be *"Not-an-argument"* indicating that the entity mention is not an argument for the trigger. For RE (Veyseh et al., 2020a,b), the task focuses on the classification of relation (e.g, *"Work for"*) for a given pair of entity mentions. There is also a special type *"No-relation"* to specify no relation between two entity mentions. As such, we call the union set C of the predefined event types, entity types, argument roles, and relation types as the information types (excluding *"Not-an-argument"* and *"No-relation"*).

3 Model

To capture dependency among task instances for JointIE, an approach is to obtain all text spans for entity/event mention candidates along with their possible pairs to form the nodes for a dependency graph to improve representation learning. However, this approach will retain many text spans for non-entity/event mentions to introduce noise into the modeling. It will also entail a large dependency graph that can hinder the efficiency of the model. To this end, our model for JointIE first identifies text spans for entity mentions and event triggers. Afterwards, all possible pairs of event-entity and entity-entity mentions are considered to identify positive pairs for event arguments and relations respectively. The detected entity mentions, event triggers, event arguments, and relations are called task instances that should be classified to obtain corresponding information types in C. In our model, a dependency graph among the detected task instances will be learned to provide inputs for GCNs to compute dependency-enhanced representations for the task instances. Finally, the enhanced representations will be used to compute a joint distribution over labels for all the task instances to train our model. We will also employ Simulated Annealing to achieve the global optimum for label assignment of the task instances in the decoding phase.

3.1 Identifying event and entity mentions

Given an input sentence $\mathbf{w} = [w_1, \ldots, w_N]$ with N words, we identify its event triggers and entity mentions by solving two corresponding sequence tagging problems for event and entity mentions. In particular, we use the BIO tagging schema to assign two labels to each word in $\mathbf{w}$ to mark the text spans of event triggers and entity mentions, i.e., {*"B-TRIGGER"*, *"I-TRIGGER"*, *"O"*} labels for event triggers, and {*"B-ENTITY"*, *"I-ENTITY"*, *"O"*} labels for entity mentions. The pre-trained transformer-based language model BERT (Devlin et al., 2019) is first utilized to obtain the contextualized embeddings for the words in the sentence: $\mathbf{X} = \mathbf{x}_1, \ldots, \mathbf{x}_N = \text{BERT}([w_1, \ldots, w_N])$.

Next, the vector sequence $\mathbf{X}$ is sent to two different CRF layers (Lafferty et al., 2001; Chiu and Nichols, 2016) to compute two distributions for the tag sequences of $\mathbf{w}$ for event triggers and event mentions. The negative log-likelihoods L_t and L_e for golden trigger and entity tag sequences are then obtained to be included in the overall training loss. At test time, the Viterbi algorithm (Forney, 1973) is employed to determine the best tag sequences for event triggers and event mentions in $\mathbf{w}$.

Let V^t and V^e be the sets of text spans for event triggers and entity mentions respectively in $\mathbf{w}$ (i.e., golden spans in the training time and predicted spans in the test time). To prepare for the next components, we compute the representations vectors $\mathbf{z}^t_i$ and $\mathbf{z}^e_j$ for each event trigger/instance $t_i \in V^t$ and entity mention/instance $e_j \in V^e$ respectively by averaging over the contextualized embeddings of the words inside the spans.

3.2 Identifying event arguments and relations

Given the detected event triggers and entity mentions, we obtain a representation vector $\mathbf{z}^a_{ij}$ for each pair of event-entity mentions $a_{ij} = (t_i, e_j)$ (i.e., $t_i \in V^t$, $e_j \in V^e$), and a representation vector $\mathbf{z}^r_{ij}$ for each pair of entity-entity mentions $r_{ij} = (e_i, e_j)$ (i.e., $e_i, e_j \in V^e$) via:

$\mathbf{z}^a_{ij} = FFN^{down}_a(\text{concat}(\mathbf{z}^t_i, \mathbf{z}^e_j))$ and $\mathbf{z}^r_{ij} = FFN^{down}_r(\text{concat}(\mathbf{z}^e_i, \mathbf{z}^e_j))$.

Here, we use the feed-forward networks FFN^{down}_a and FFN^{down}_r to make sure that $\mathbf{z}^t_i$, $\mathbf{z}^e_j$, $\mathbf{z}^a_{ij}$, and $\mathbf{z}^r_{ij}$ have the same dimensionality. Next, the pair representation vectors $\mathbf{z}^a_{ij}$ and $\mathbf{z}^r_{ij}$ are sent into two different feed-forward networks followed by sigmoid activations to compute the possibilities for being positive examples for event arguments and relations of a_{ij} and r_{ij} respectively: $p^a_{ij} = \sigma(\text{FFN}^a(\mathbf{z}^a_{ij}))$, and $p^r_{ij} = \sigma(\text{FFN}^r(\mathbf{z}^r_{ij}))$. Here, $p^a_{ij} \in (0, 1)$ is the probability for the entity mention e_j being an actual argument for the

event trigger t_i while $p^r_{ij} \in (0, 1)$ is the likelihood that there exists a relation of interest between the entity mentions e_i and e_j. At training time, we obtain the the negative log-likelihoods L_a and L_r for the golden event argument and relation identification to be included in the overall loss function for minimization. At test time, the event-entity pair a_{ij} and entity-entity pair r_{ij} are retained as positive examples for event arguments and relations if their likelihooods p^a_{ij} and p^r_{ij} are greater than 0.5.

For convenience, let V^a and V^r be the sets of positive event-entity pairs a_{ij} (called argument instances) and entity-entity pairs r_{ij} (called relation instances) respectively. Also, let $V = V^t \cup V^e \cup V^a \cup V^r$ be the set of all detected event, entity, argument, and relation instances. For each instance $v_i \in V$, we will use $\mathbf{v}_i$ for its corresponding instance representation (i.e., from $\mathbf{z}^t_i$, $\mathbf{z}^e_j$, $\mathbf{z}^a_{ij}$, or $\mathbf{z}^r_{ij}$).

3.3 Inducing Instance Dependency

Given the detected event, entity, argument, and relation instances in V, it remains to predict the information types in C for the instances to solve JointIE. While it is possible to directly employ the instance representations $\mathbf{v}_i$ for label prediction, our goal is to exploit instance dependency in IE to enhance the representation vector for one instance with the information from other instances to facilitate type prediction. In particular, using the instances v_i in V as the nodes in a dependency graph G, we aim to enrich instance representations by feeding them into a GCN model. As such, instead of assuming a heuristic manually-designed dependency graph among the instances as in previous work (Zhang and Ji, 2021; Nguyen et al., 2021a), we propose to automatically learn the dependency graph G for the instances in V. To this end, our dependency graph G is a fully connected graph among the nodes in V where a weight $\alpha_{ij} \in (0, 1)$ is learned for each edge to quantify the dependency between the instances v_i and v_j in V. In this work, we present two sources of information that can be used for determining the dependency between the task instances: (i) semantic and (ii) syntactic information.

Semantic Information: The semantic-based weight α^{sem}_{ij} for the edge between v_i and v_j quantifies their relatedness/dependency based on semantic information, i.e., via the representation vectors $\mathbf{v}_i$ and $\mathbf{v}_j$: $\alpha^{sem}_{ij} = FFN^{sem}(\text{concat}(\mathbf{v}_i, \mathbf{v}_i))$. Here, FFN^{sem} is a feed-forward network with the sigmoid function in the end.

Syntactic Information: The syntax-based weight α^{syn}_{ij} for the edge between v_i and v_j is computed in a similar way as α^{sem}_{ij}. In particular, for each word $w_k \in \mathbf{w}$, we retrieve the dependency relation d_k between w_k and its governor in the dependency tree of $\mathbf{w}$, which is generated by the Trankit's dependency parser (Nguyen et al., 2021b). We then obtain the embedding $\mathbf{m}_k$ of d_k for w_k by looking up the learnable dependency embedding matrix $\mathbf{M}$. Afterwards, the syntax-based representation vector u_i for the instance $v_i \in V$ is computed via: $u_i = \text{max-pool}_{w_k \in SPAN_{v_i}}(\mathbf{m}_k)$. Here, $SPAN_{v_i}$ involves the words in the corresponding text span of v_i in $\mathbf{w}$ if v_i is an event trigger or entity mention instance. Otherwise, $SPAN_{v_i}$ contains the words inside the text spans of the involving event triggers and entity mentions in the pair for v_i. As such, we compute the syntax-based dependency weight α^{syn}_{ij} for v_i and v_j via: $\alpha^{syn}_{ij} = FFN^{syn}(\text{concat}(\mathbf{u}_i, \mathbf{u}_i))$ where FFN^{syn} is also a feed-forward network with the sigmoid function in the end. Finally, we combine the semantic- and syntax-based weights to obtain the overall dependency weight α_{ij} for v_i and v_j in V: $\alpha_{ij} = (\alpha^{sem}_{ij} + \alpha^{syn}_{ij})/2$.

3.4 Enhancing Representations with GCNs

To enhance the representation vectors for the instances $v_i \in V$, a GCN model with K layers is applied over the induced dependency graph G to compute richer representations for the instances:

$$\mathbf{h}^k_i = \text{ReLU}(\frac{\sum_{v_j \in V} \alpha_{ij} \mathbf{W}^k \mathbf{h}^{k-1}_j + \mathbf{b}^k}{\sum_{v_j \in V} \alpha_{ij}}), 1 \leq k \leq K \quad (1)$$

Here, h^k_i is the representation for the instance v_i at the k-th layer of the GCN ($\mathbf{h}^0_i \equiv \mathbf{v}_i$), and $\mathbf{W}^k, \mathbf{b}^k$ are trainable weight and bias for the layer.

In this way, representation information from all the other instances v_j ($j \neq i$) will be incorporated into the enhanced representation vector for v_i according to their learned dependency weights. Finally, the last layer's representation $\mathbf{h}^K_i \equiv \mathbf{h}_i$ (we omit K for simplicity) is used to compute the score vector $\mathbf{s}_i \in \mathbb{R}^{|C|}$ for v_i, where $\mathbf{s}_i[c]$ measure the possibility for v_i to have the c-th label in the label set C: $\mathbf{s}_i = FFN^{score}(\mathbf{h}_i)$ (FFN^{score} is a scoring feed-forward network). The score vectors $\mathbf{s}_i$ will later be used for modeling the joint distribution of the labels for all the instances in V.

3.5 Computing Joint Distribution of Labels

Let Y be the set of labels y_i for the instances v_i in V. To infer the labels for the instances in V, we

need to estimate the joint distribution $P(Y|\mathbf{w}, V)$. In previous work (Wadden et al., 2019; Lin et al., 2020; Nguyen et al., 2021a; Zhang and Ji, 2021), JointIE methods mostly focus on learning representations for the task instances to compute a label distribution for each instance v_i for prediction: $P(y_i|\mathbf{w}, V) := \text{softmax}(\mathbf{s}_i)$. This practice essentially implies the following factorization for $P(Y|\mathbf{w}, V)$: $P(Y|\mathbf{w}, V) = \prod_{y_i \in Y} P(y_i|\mathbf{w}, V)$. As a result, this factorization assumes the independence of the instance labels, thus unable to fully capture beneficial label dependency for IE tasks.

To address this issue, we directly estimate the joint distribution $P(Y|\mathbf{w}, V)$ so that the dependency between instance labels can be facilitated to improve prediction performance. To this end, we formulate the joint distribution $P(Y|\mathbf{w}, V)$ with Conditional Random Fields (Lafferty et al., 2001):

$$P(Y|\mathbf{w}, V) = \frac{1}{Z(V)} \prod_{(v_i, v_j)} \psi_{ij}(y_i, y_j, V) \quad (2)$$

where $\psi_{ij}(y_i, y_j, V)$ is a positive potential function defined on the edge (v_i, v_j) of the dependency graph G, and $Z(V) = \sum_{Y' \in C_V} \prod_{(v_i, v_j)} \psi_{ij}(y'_i, y'_j, V)$ is the normalization term to make sure that $P(Y|\mathbf{w}, V)$ is a valid probability distribution (C_V is the set of all possible label assignments Y for the instances in V). Considering the instance information, the instance dependency, and the label dependency, we propose the potential function as:

$$\psi_{ij}(y_i, y_j, V) := \exp(\mathbf{s}_i[y_i] + \mathbf{s}_j[y_j] + \alpha_{ij} \pi_{y_i \leftrightarrow y_j}) \quad (3)$$

where $\mathbf{s}_i[y_i]$ is the local score for instance v_i being assigned with the label y_i, α_{ij} is the induced dependency weight for the edge (v_i, v_j) in G, and $\pi_{y_i \leftrightarrow y_j}$ is a learnable transition score indicating the dependency between the labels y_i and y_j. With this formulation, we can derive the joint distribution $P(Y|\mathbf{w}, V)$:

$$P(Y|\mathbf{w}, V) = \frac{\exp(s(Y))}{\sum_{Y' \in C_V} \exp(s(Y'))} \quad (4)$$

where:

$$s(Y) = \gamma \sum_{v_i \in V} \mathbf{s}_i[y_i] + \sum_{(v_i, v_j)} \alpha_{ij} \pi_{y_i \leftrightarrow y_j} \quad (5)$$

is the global score for the label assignment/configuration Y of the instances. γ is a hyper-parameter to balance the local and transition scores.

To train the model, we need to maximize the joint likelihood in Equation (4) for the golden label configuration Y^*. However, this requires the computation of the normalization term $\sum_{Y' \in C_V} \exp(s(Y'))$, which is intractable. To overcome this issue, we employ Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012; Mikolov et al., 2013). NCE converts the maximization problem into the nonlinear logistic regression that discriminates between the golden label configurations and the noise label configurations. In particular, the maximization of $P(Y^*|\mathbf{w}, V)$ is done with NCE via minimizing the contrastive loss:

$$L_{NC} = -\log\sigma(s(Y^*)) - \sum_{n=1}^{N_{noi}} \mathbb{E}_{Y'_n \sim P_{noi}} \left[\log\sigma(-s(Y'_n))\right] \quad (6)$$

where σ is the sigmoid function and N_{noi} is the number of noise configurations Y'_n drawn from P_{noi}, assumed to be a uniform distribution. Intuitively, the minimization of L_{NC} increases the global score $s(Y^*)$ for the true label configuration Y^* while decreasing the global scores $s(Y')$ for the noise label configurations Y' to appropriately train the model. To the end, the overall loss function to train our model is: $L = L_t + L_e + L_a + L_r + L_{NC}$.

Algorithm 1: Simulated Annealing Search

Input : $\hat{Y}_0$ where $\hat{y}_{i,0} = \text{argmax}_{c \in C} \mathbf{s}_i[c]$.

$\hat{Y}_{cur} \leftarrow \hat{Y}_0$; $n \leftarrow 1$;
while $n \leq N_{iter}$ **do**
  $t \leftarrow T/n$;
  if $t < \epsilon$ **then**
    return $\hat{Y}_{cur}$;
  else
    $\hat{Y}_{new} = random\_successor(\hat{Y}_{cur})$;
    $\delta_n = s(\hat{Y}_{new}) - s(\hat{Y}_{cur})$;
    if $\delta_n > 0$ **then**
      $\hat{Y}_{cur} \leftarrow \hat{Y}_{new}$;
    else
      $\hat{Y}_{cur} \leftarrow \hat{Y}_{new}$ with $p = \exp(\frac{\delta_n}{t})$;
    end
  end
  $n \leftarrow n + 1$;
end
return $\hat{Y}_{cur}$.

3.6 Joint Decoding via Simulated Annealing

At inference time, we need to search for the configuration $\hat{Y}$ that has the highest global score $s(\hat{Y})$ in C_V: $\hat{Y} = \text{argmax}_{Y' \in C_V} s(Y')$. A brute-force search for $\hat{Y}$ cannot be done as the search space C_V is exponentially large ($|C_V| = |C|^{|V|}$). Previous work has made several attempts to deal with this issue. (Wadden et al., 2019) and (Nguyen et al., 2021a) simply perform greedy decoding for each

instance label independently, thus unable to exploit the label dependency. (Lin et al., 2020) and (Zhang and Ji, 2021) resort to beam search that step by step constructs a complete decoding assignment Y for the instances in V by expanding an initially empty assignment. Each step corresponds to an instance in V where only top candidate labels for the instance are considered for assignment expansion and only top partial assignments produced so far are kept for the next step. Unfortunately, the selection of top candidate labels for expansion at each step is based only on the local scores $\mathbf{s}_i$, which might discard the candidates that can eventually provide greater global scores. To overcome this issue, we propose to apply Simulated Annealing (SA) (Kirkpatrick et al., 1983) to search for the optimal assignment $\hat{Y}$ for V. SA is a probabilistic algorithm that is able to approximately find the global optimum of a function (Kirkpatrick et al., 1983; Van Laarhoven and Aarts, 1987). Algorithm 1 presents our implementation for SA to find $\hat{Y}$.

Datasets	Split	#sents	#ents	#rels	#events
ACE05-R	Train	10,051	26,473	4,788	-
	Dev	2,424	6,362	1,131	-
	Test	2,050	5,476	1,151	-
ACE05-E	Train	17,172	29,006	4,664	4,202
	Dev	923	2,451	560	450
	Test	832	3,017	636	403
ACE05-E+	Train	19,240	47,525	7,152	4,419
	Dev	902	3,422	728	468
	Test	676	3,673	802	424
ERE-EN	Train	14,219	38,864	5,045	6,419
	Dev	1,162	3,320	424	552
	Test	1,129	3,291	477	559
ERE-ES	Train	7,067	11,839	1,698	3,272
	Dev	556	886	120	210
	Test	546	811	108	269

Table 1: Data statistics. **#sents**, **#ent**, **#rels**, and **#events** indicate the number of sentences, entity mentions, relations, and events respectively.

The input for the algorithm is the initial configuration $\hat{Y}_{cur} = \hat{Y}_0 = \{\hat{y}_{i,0}\}$, which contains the greedily predicted labels for each instance: $\hat{y}_{i,0} = \text{argmax}_{c \in C} \mathbf{s}_i[c]$. The algorithm then runs over N_{iter} iterations to improve the global score $s(\hat{Y}_{cur})$ for the current label configuration $\hat{Y}_{cur}$. This is done via updating the current configuration to a successor configuration $\hat{Y}_{new}$ that gives a higher global score (i.e., $\delta_n > 0$). A successor configuration is obtained via the function $random\_successor()$ by randomly changing some label $\hat{y}_i \in \hat{Y}_{cur}$. Different from beam search decoding with partial assignments, each searching step in SA examines a complete label assignment for the instances in V to provide complete information to measure the global scores/quality of the assignments. Importantly, SA sometimes allows the current configuration to transition to a successor configuration with a lower global score (i.e., $\delta_n \leq 0$) with an acceptance probability of $p = \exp(\frac{\delta_n}{t})$. Here, t is the temperature of the algorithm, gradually decreased via $t \leftarrow T/n$ (T is a hyper-parameter). This exploration property enables SA to escape from local optimum configurations, thus increasing the chance to find the globally optimal configuration $\hat{Y}$.

4 Experiments

Datasets: Following previous work (Wadden et al., 2019; Lin et al., 2020; Zhang and Ji, 2021; Nguyen et al., 2021a; Lu et al., 2021; Hsu et al., 2021), we conduct experiments on 5 different datasets created by the 2005 Automatic Content Extraction (ACE05) (Walker et al., 2006) and Entity Relation Event (ERE) (Song et al., 2015) programs. The three ACE05 datasets feature **ACE05-R**, **ACE05-E**, and **ACE-E+**, all in English, involving 33 event types, 7 entity types, 6 relation types, and 22 argument roles. The two ERE datasets are **ERE-EN** (English portion) and **ERE-ES** (Spanish portion), introducing 38 event types, 7 entity types, 5 relation types, and 20 argument roles. We use the same data processing and train/dev/test splits as the prior work for a fair comparison. Detailed statistics for the datasets are shown in Table 1.

Baselines: We compare our method, called *GraphIE*, with the following baselines for JointIE:

Generative baselines: *Text2event* (Lu et al., 2021) and *DEGREE* (Hsu et al., 2021). The generative baselines perform ETD and EAE via formulating the tasks as text generation. The models receive an input sentence and generate an output text containing text spans and labels for event triggers and event arguments, structured in a way that a post-processing step can be used to extract ETD and EAE predictions for the models.

Classification baselines: *OneIE* (Lin et al., 2020), *AMRIE* (Zhang and Ji, 2021), and *FourIE* (Nguyen et al., 2021a). The classification baselines represent the instances for ETD, EMR, EAE, and RE via a shared encoder and perform classification for the instances based on task-specific label distributions. *AMRIE* and *FourIE* employ a heuristic dependency graph among task instances to improve representation learning. Dependency between instance labels is exploited in *OneIE* and *AMRIE* via a beam search decoding with manually-designed

PLMs	Model	ACE05-R		ACE05-E			ACE05-E+				ERE-EN				ERE-ES			
		Ent	Rel	Ent	Trg	Arg	Ent	Rel	Trg	Arg	Ent	Rel	Trg	Arg	Ent	Rel	Trg	Arg
T5	Text2event	-	-	-	71.9	53.8	-	-	71.8	54.4	-	-	59.4	48.3	-	-	-	-
BART	DEGREE	-	-	-	72.2	56.0	-	-	71.7	58.0	-	-	56.6	51.1	-	-	-	-
BERT	OneIE	88.6	63.4	90.2	74.7	56.8	89.6	58.6	72.8	54.8	87.0	53.2	57.0	46.5	81.3	48.1	56.8	40.3
	AMRIE*	88.7	67.2	90.8	75.3	58.2	90.4	62.9	72.8	56.3	86.9	55.5	58.3	44.2	-	-	-	-
	FourIE	88.9	68.9	**91.3**	75.4	58.0	**91.1**	63.6	73.3	57.5	**87.4**	56.1	57.9	48.6	**82.2**	57.9	57.1	42.3
	GraphIE	**88.9**	**69.5**	90.6	**75.7**	**58.8**	91.0	**65.4**	**74.8**	**59.9**	87.2	**57.8**	**61.4**	**52.2**	81.4	**58.9**	**61.3**	**45.7**
RoBERTa	OneIE*	89.0	65.2	90.2	74.7	55.6	90.8	60.4	72.5	56.3	86.3	52.8	57.1	47.1	83.7	57.5	58.3	42.5
	AMRIE	89.2*	66.8*	**92.1**	75.0	58.6	91.0*	62.8*	72.7*	57.7*	87.9	55.2	61.4	45.0	-	-	-	-
	FourIE*	89.1	67.5	91.6	74.9	58.7	91.1	63.1	72.8	58.3	**88.0**	56.2	61.5	49.1	83.9	61.0	62.3	44.2
	GraphIE	**89.3**	**68.5**	91.4	**75.1**	**59.4**	**91.6**	**66.0**	**73.3**	**60.2**	87.7	**57.0**	**62.0**	**54.7**	**84.3**	**62.3**	**65.7**	**46.9**

Table 2: Model performance on the test data of 5 datasets. *"Ent"*, *"Rel"*, *"Trg"*, and *"Arg"* are the F1 scores for identification and classification of entity mentions, event triggers, relations, and event arguments respectively. * indicates results that are not reported in the original papers but produced by their official code. Underlined numbers designate the tasks where *GraphIE* is significantly better ($p < 0.01$) than the baselines.

global features, and in *FourIE* via global type dependency regularization. *FourIE* and *AMRIE* are the current state-of-the-art models for JointIE.

Hyper-parameters: Prior work for JointIE employs two different versions of pre-trained language models (PLM), i.e., *BERT* (Devlin et al., 2019; Lin et al., 2020; Nguyen et al., 2021a) and *RoBERTa* (Liu et al., 2019; Zhang and Ji, 2021), which might cause incompatible compassion. To this end, we explore both BERT and RoBERTa to obtain the word representations $\mathbf{x}_i$ for *GraphIE* for a fair comparison. For the Spanish ERE-ES dataset, following prior work (Lin et al., 2020; Nguyen et al., 2021a), we utilize the multilingual versions of BERT and RoBERTa. For each PLM, we fine-tune the hyper-parameter for *GraphIE* on the development data.

In particular, the best values for the hyper-parameters of the proposed model are reported as follows. We employ the learning rate of $1e-5$ for the models with the BERT-based PLM (i.e., using *bert-large-cased* and *bert-multilingual-cased*) and the learning rate of $5e-6$ for the RoBERTa-based PLM (i.e., using *roberta-large* and *xlm-roberta-large*). For other hyper-parameters, our tuning process results in the same values for BERT-based and RoBERT-based models: Adam (Kingma and Ba, 2014) for the optimizer, batch size of 10, 100 for the size of the dependency relation embeddings, 400 for the size of the hidden vector for the feed-forward networks, 200 for the hidden vector size in the GCN model, 2 for the number of layers for the feed-forward networks and GCN model, $\gamma = 1$ for the trade-off hyper-parameter for the global score, $N_{noi} = 5$ for the number of noise examples for the contrastive loss (we re-sample the noise examples every epoch), $T = 5$ for the initial temperature, $N_{iter} = 50$ for the number of iterations of Simulated Annealing (SA), and $\epsilon = 0.1$ for the temperature threshold for the SA decoding.

Comparison with Baselines: We compare the proposed model *GraphIE* with the baselines on test data of the 5 datasets in Table 2. As can be seen, the generative baselines perform worse than the classification models on most of the settings. This might be due the implicit modeling of the label distributions and the assumption of a decoding order for task instances that limit the interactions of instance labels. Comparing *OneIE*, *FourIE* and *AMRIE*, it is clear that the exploitation of instance and label dependency in the training phase in *FourIE* can lead to better performance for JointIE than using such dependency in the decoding phase as done by *OneIE* and *AMRIE* over most tasks and PLMs. Most importantly, the proposed *GraphIE* significantly outperforms all the baselines across a majority of settings for tasks, datasets and PLMs, thus demonstrating the benefits of induced dependency graph, joint label distribution estimation, and simulated annealing for decoding in our method.

Model *(all use Roberta)*	ACE05-E+			
	Ent	Rel	Trg	Arg
GraphIE	**89.8**	**67.2**	**72.6**	**66.3**
- induced dep	89.3	65.8	71.3	65.0
- semantic-based dep	89.0	66.4	71.6	65.9
- syntactic-based dep	89.4	66.3	72.0	65.4
- induced dep + heuristic dep	89.3	66.2	71.7	65.5
- GCN	89.4	65.6	70.9	64.6

Table 3: Performance (F1) on the ACE05-E+ development data.

Ablation Study: To understand the contributions of each proposed component to *GraphIE*, we conduct ablation experiments where we remove each component from the full model and evaluate the performance of the remaining models.

The first three ablated models in Table 3 are *"- induced dep"*, *"- semantic dep"*, and *"- syntactic dep"*, formed by excluding the dependency weight induction of α_{ij} (i.e., setting $\alpha_{ij} = 1$),

the semantic-based dependency α_{ij}^{sem}, and the syntactic-based dependency α_{ij}^{syn} (respectively) from the model computation. In each case, the performance of *GraphIE* decreases significantly; the removal of both semantic- and syntactic-based dependency in *"- induced dep"* leads to the largest performance drop. This shows that the semantic and syntactic weighting captures complementary information for instance dependency induction that is useful for our model. The next ablated model *"- induced dep + heuristic dep"* is obtained by replacing the induced dependency graph represented by α_{ij} with the heuristic dependency graph for instances from the best baseline *FourIE*. The decrease in the performance of this model suggests that the induced dependency graph is better than the heuristic graph for JointIE. The final ablated model *"- GCN"* in Table 3 eliminates the GCN component from our full model. The result shows that GCN is beneficial to exploit the induced dependency graph to improve representation learning.

Model *(all use Roberta)*	**ACE05-E+**			
	Ent	Rel	Trg	Arg
GraphIE	**89.8**	**67.2**	**72.6**	**66.3**
- joint distribution	89.3	65.5	70.9	64.5
- SA + greedy	89.2	65.9	71.2	65.2
- SA + beam	89.5	66.0	71.5	65.4
- SA + hill climbing	89.5	66.8	71.7	65.3
OneIE	88.7	64.2	69.5	63.2
- beam + SA	88.1	63.9	69.1	62.7
AMRIE	89.4	65.4	71.2	64.4
- beam + SA	88.8	65.1	70.5	64.1

Table 4: Performance (F1) on the ACE05-E+ development data.

In Table 4, we first eliminate the computation of the joint label distribution $P(Y|\mathbf{w}, V)$ from *GraphIE*. As such, the *"- joint distribution"* model employs the local label distributions $P(y_i|\mathbf{w}, V)$ to train models and infer labels (with greedy decoding). Due to the significantly worse performance of *"- joint distribution"*, it is clear that directly estimating the joint label distribution is helpful for JointIE. To evaluate the benefit of the proposed SA, we replace it with other decoding algorithms for *GraphIE*, including greedy search, beam search and hill climbing. The beam search is implemented with our global score function $s(Y)$ and follows those in (Lin et al., 2020; Zhang and Ji, 2021) while hill climbing is implemented by removing the configuration exploration in lines 11-12 of Algorithm 1. As reported in Table 4, SA performs much better than other decoding algorithms for *GraphIE*, thus demonstrating SA's ability to find globally optimal labels. In addition, we also attempt to replace the beam search decoding in *OneIE* and *AMRIE* with SA, which indeed leads to worse performance for such models as shown in the last four rows of Table 4. We attribute this to the learning of the global scores for configurations in *OneIE* and *AMRIE* that involves a limited set of predefined global features. Such features do not exist for many possible assignments Y for V, thus causing poor global score computation and hindering the configuration ranking critically required by SA.

Label pair	**Transition score**
(Argument:Origin, Argument:Place)	10.02
(Event:Transport, Relation:Physical)	4.33
(Relation:Org-Aff, Relation:Part-Whole)	3.58
(Event:Execute, Event:Sentence)	2.58
(Event:Die, Event:Be-Born)	-2.34
(Event:Attack, Argument:Origin)	-87.07
(Relation:Per-Soc, Entity:Facility)	-93.93
(Transport, Attacker)	-99.91

Table 5: Transition scores for some label pairs learned by our model on ACE05-E+.

Analysis: To further understand the advantages of *GraphIE* over baseline models, we manually analyze the instances on the ACE05-E+ development data where *GraphIE* can make correct predictions, but the best baseline model *FourIE* fails. Figure 2 presents some instances along with their edges and weights in the dependency graphs. The most important insight from our analysis is that *GraphIE* is able to connect an instance (e.g., *blew*) with other supporting instances (e.g., *suicide*) in the dependency graph to provide vital information to facilitate correct prediction. Such supporting instances do not share any event trigger or entity mention with the current instance that cannot establish links in *FourIE* and lead to failure predictions.

Finally, Table 5 shows the transition scores $\pi_{y_i \leftrightarrow y_j}$ learned by *GraphIE* for some label pairs in ACE05-E+. The table show that our model is able to learn high scores for correlated label pairs (e.g., the *Execute* and *Sentence* event types) and very low scores for uncorrelated label pairs (e.g., an argument for a *Transport* event cannot play the role *Attacker*).

5 Related Work

Capturing dependency between IE tasks has been a main focus of previous work on Joint IE. Early work employed feature engineering methods (Roth and Yih, 2004; Yu and Lam, 2010; Li et al., 2013;

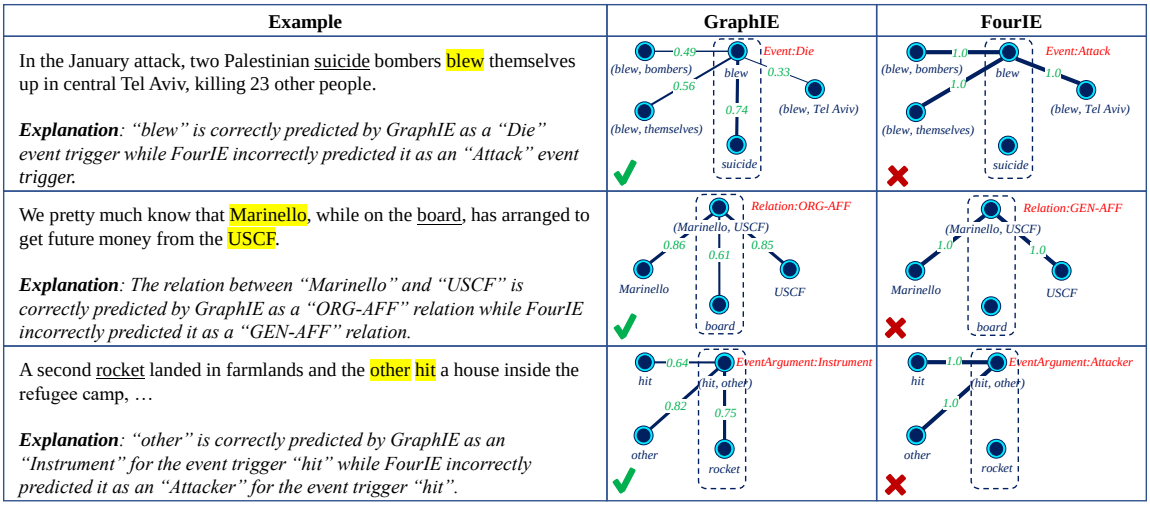

Figure 2: Instances along with their dependency subgraphs in ACE05-E+. Supporting instances are underlined.

Yang and Mitchell, 2016). Later work applied deep learning via shared parameters to facilitate joint modeling for IE, however, for only two or three tasks (Nguyen et al., 2016a; Zheng et al., 2017; Bekoulis et al., 2018; Luan et al., 2019; Zhang et al., 2019; Nguyen and Nguyen, 2019). Recently, the four IE tasks have been solved jointly (Wadden et al., 2019; Lin et al., 2020; Zhang and Ji, 2021; Paolini et al., 2021; Lu et al., 2021; Nguyen et al., 2021a). However, such recent works only employ heuristics to manually design dependency graphs for instances. Mean-field factorization of the joint label distribution for JointIE instances is dominant in prior work.

Our work is also related to prior work that uses CRFs (Lafferty et al., 2001; Chiu and Nichols, 2016) to estimate joint distribution of instance labels. Sequence labeling is a typical problem that has been solved by CRFs, including part of speech tagging and named entity recognition (Lafferty et al., 2001; Ekbal et al., 2007; Shishtla et al., 2008; Sobhana et al., 2010; Zea et al., 2016; Chiu and Nichols, 2016; Xu et al., 2017). However, these prior work only employ CRFs for simple graph structures (i.e., linear chains). A few prior work has considered CRFs for more complicated graph structures (Sun et al., 2017; Gao et al., 2019; Qu et al., 2019; Yuan and Ji, 2020); however, none of such works has applied CRFs for JointIE as we do.

6 Conclusion

We propose a novel model for jointly solving four IE tasks (EMR, ETD, EAE, and RE). Our proposed model learns a dependency graph among the instances of the tasks via a novel edge weighting mechanism. We also estimate the joint distribution among instance labels to fully enable interactions between instance labels for improved performance. The experimental results show that our model achieves best performance for multiple JointIE tasks across 5 datasets and 2 languages. In the future, we plan to extend our method to cover more IE tasks such as event coreference resolution.

Acknowledgement

This research has been supported by the Army Research Office (ARO) grant W911NF-21-1-0112 and the NSF grant CNS-1747798 to the IUCRC Center for Big Learning. This research is also based upon work supported by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via IARPA Contract No. 2019-19051600006 under the Better Extraction from Text Towards Enhanced Retrieval (BETTER) Program. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ODNI, IARPA, the Department of Defense, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein. This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

References

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, and Chris Develder. 2018. Adversarial training for multi-context joint entity and relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Jason PC Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional lstm-cnns. In *Transactions of the 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*.

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Asif Ekbal, Rejwanul Haque, and Sivaji Bandyopadhyay. 2007. Bengali part of speech tagging using conditional random field. In *Proceedings of the seventh International Symposium on Natural Language Processing, SNLP-2007*.

G David Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

Hongchang Gao, Jian Pei, and Heng Huang. 2019. Conditional random field enhanced graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 276–284.

Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(2).

I Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, Nanyun Peng, et al. 2021. Degree: A data-efficient generative event extraction model. *arXiv preprint arXiv:2108.12724*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations*.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Viet Dac Lai, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Event detection: Gate diversity and syntactic importance scores for graph convolution neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Fayuan Li, Weihua Peng, Yuguang Chen, Quan Wang, Lu Pan, Yajuan Lyu, and Yong Zhu. 2020. Event extraction as multi-turn question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 829–838.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics*.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

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

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2event: Controllable sequence-to-structure generation for end-to-end event extraction. *arXiv preprint arXiv:2106.09232*.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Minh Van Nguyen, Viet Lai, and Thien Huu Nguyen. 2021a. Cross-task instance representation interactions and label dependencies for joint information extraction with graph convolutional networks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 27–38, Online. Association for Computational Linguistics.

Minh Van Nguyen, Viet Dac Lai, Amir Pouran Ben Veyseh, and Thien Huu Nguyen. 2021b. Trankit: A light-weight transformer-based toolkit for multilingual natural language processing. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016a. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*.

Thien Huu Nguyen and Ralph Grishman. 2018. Graph convolutional networks with argument-aware pooling for event detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian. 2016b. Toward mention detection robustness with recurrent neural networks. In *Proceedings of IJCAI Workshop on Deep Learning for Artificial Intelligence (DLAI)*.

Trung Minh Nguyen and Thien Huu Nguyen. 2019. One for all: Neural joint modeling of entities and events. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*.

Meng Qu, Yoshua Bengio, and Jian Tang. 2019. Gmnn: Graph markov neural networks. In *International conference on machine learning*, pages 5241–5250. PMLR.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the Eighth Conference on Computational Natural Language Learning*.

Praneeth M Shishtla, Karthik Gali, Prasad Pingali, and Vasudeva Varma. 2008. Experiments in telugu ner: A conditional random field approach. In *Proceedings of the IJCNLP-08 Workshop on Named Entity Recognition for South and South East Asian Languages*.

N Sobhana, Pabitra Mitra, and SK Ghosh. 2010. Conditional random field based named entity recognition in geological text. *International Journal of Computer Applications*, 1(3):143–147.

Zhiyi Song, Ann Bies, Stephanie 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 *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98, Denver, Colorado. Association for Computational Linguistics.

Xiaofeng Sun, Xiangguo Lin, Shuhan Shen, and Zhanyi Hu. 2017. High-resolution remote sensing data classification over urban areas using random forest ensemble and fully connected conditional random field. *ISPRS International Journal of Geo-Information*, 6(8):245.

Peter JM Van Laarhoven and Emile HL Aarts. 1987. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Dejing Dou, and Thien Huu Nguyen. 2020a. Exploiting the syntax-model consistency for neural relation extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*.

Amir Pouran Ben Veyseh, Franck Dernoncourt, My Thai, Dejing Dou, and Thien Huu Nguyen. 2020b. Multi-view consistency for relation extraction via mutual information and structure prediction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020c. Graph transformer networks with syntactic and semantic structures for event argument extraction. In *Proceedings of the Findings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP Findings)*.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. Ace 2005 multilingual training corpus. In *Technical report, Linguistic Data Consortium*.

Kai Xu, Zhanfan Zhou, Tianyong Hao, and Wenyin Liu. 2017. A bidirectional lstm and conditional random fields approach to medical named entity recognition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 355–365. Springer.

Bishan Yang and Tom M. Mitchell. 2016. Joint extraction of events and entities within a document context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Proceedings of the 23th International Conference on Computational Linguistics*.

Hao Yuan and Shuiwang Ji. 2020. Structpool: Structured graph pooling via conditional random fields. In *Proceedings of the 8th International Conference on Learning Representations*.

Jenny Linet Copara Zea, Jose Eduardo Ochoa Luna, Camilo Thorne, and Goran Glavaš. 2016. Spanish ner with word representations and conditional random fields. In *Proceedings of the sixth named entity workshop*, pages 34–40.

Junchi Zhang, Yanxia Qin, Yue Zhang, Mengchi Liu, and Donghong Ji. 2019. Extracting entities and events as a single task using a transition-based neural model. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*.

Zixuan Zhang and Heng Ji. 2021. Abstract meaning representation guided graph encoding and decoding for joint information extraction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 39–49.

Suncong Zheng, Feng Wang, Hongyun Bao, Yuexing Hao, Peng Zhou, and Bo Xu. 2017. Joint extraction of entities and relations based on a novel tagging scheme. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.