From the One, Judge of the Whole: Typed Entailment Graph Construction with Predicate Generation

Zhibin Chen[123] **Yansong Feng**[13*] **Dongyan Zhao**[123]
[1]Wangxuan Institute of Computer Technology, Peking University, China
[2]Center for Data Science, Peking University, China
[3]The MOE Key Laboratory of Computational Linguistics, Peking University, China
{czb-peking, fengyansong, zhaody}@pku.edu.cn

Abstract

Entailment Graphs (EGs) have been constructed based on extracted corpora as a strong and explainable form to indicate context-independent entailment relations in natural languages. However, EGs built by previous methods often suffer from the severe sparsity issues, due to limited corpora available and the long-tail phenomenon of predicate distributions. In this paper, we propose a multi-stage method, Typed Predicate-Entailment Graph Generator (TP-EGG), to tackle this problem. Given several seed predicates, TP-EGG builds the graphs by generating new predicates and detecting entailment relations among them. The generative nature of TP-EGG helps us leverage the recent advances from large pretrained language models (PLMs), while avoiding the reliance on carefully prepared corpora. Experiments on benchmark datasets show that TP-EGG can generate high-quality and scale-controllable entailment graphs, achieving significant in-domain improvement over state-of-the-art EGs and boosting the performance of down-stream inference tasks[1].

1 Introduction

The entailment relation between textual predicates plays a critical role in natural language inference and natural language understanding tasks, including question answering (Pathak et al., 2021; McKenna et al., 2021) and knowledge graph completion (Yoshikawa et al., 2019; Hosseini et al., 2019, 2021). To detect entailment relations, previous works pay attention to the Recognizing Textual Entailment (RTE) task, which takes a pair of sentences as input and predicts whether one sentence entails the other (Bowman et al., 2015; He et al., 2021b; Pilault et al., 2020). Current RTE models perform well on RTE benchmarks, but most of them are lacking in explainability, as they make use of the black-box Language Models (LM) without providing any explainable clues.

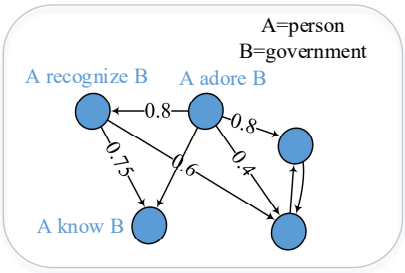


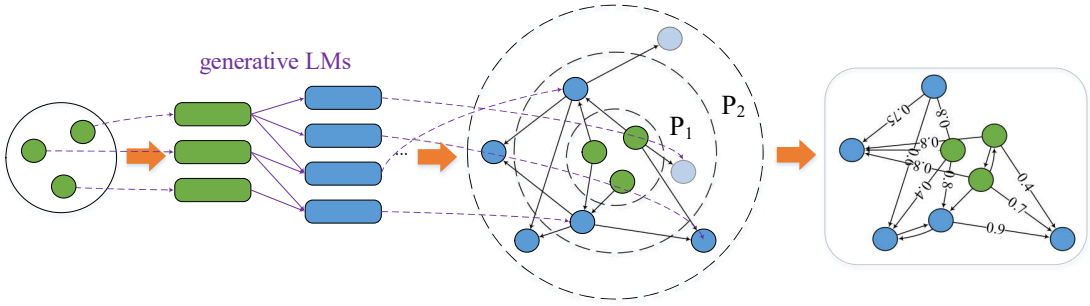
Figure 1: An example of typed entailment graph including several predicates, with argument types *person* and *government*.

Recent works focus on learning the Entailment Graph (EG) structure, which organizes typed predicates in directional graphs with entailment relations as the edges (Hosseini et al., 2018, 2019; McKenna et al., 2021), as shown in Figure 1. With the explicit graph structure containing predicates and their entailment relations, similar to Knowledge Graphs (KGs), using EGs becomes an explainable and context-independent way to represent the knowledge required in natural language inference and other NLP tasks.

Most existing EGs are constructed with the Distributional Inclusion Hypothesis (DIH), which suggests that all typical context features of a predicate v can also occur with another predicate w if v entails w (Geffet and Dagan, 2005). Constructing EGs with DIH requires distributional co-occurrences of contextual features from large corpora to calculate the semantic similarity between predicates (Szpektor and Dagan, 2008; Schoenmackers et al., 2010). However, the EGs constructed from large corpora often suffer from two different kinds of sparsity issues: *the predicate sparsity* and *the edge sparsity*. Existing corpora used for EG construction are mainly collected from

*Corresponding author.
[1]Our code is available at https://github.com/ZacharyChenpk/TP-EGG

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 3534–3551



(a) generate predicates with LMs (b) complete the predicate set of EGs step by step (c) selecting potential entailment relations and calculate edge weights

Figure 2: An illustration of our TP-EGG. Given three seed predicates, TP-EGG generates a graph with 8 predicates and 15 entailment relations. The circles represents different predicates, while the rounded rectangles is sentences in natural language. Seed predicates is in green, and newly generated predicates is in blue.

specific resources (Zhang and Weld, 2013), such as news articles. As a result, entailment relations could not be learned between those predicates that do not appear in the corpora, which leads to the *predicate sparsity* issue. Meanwhile, if two predicates scarcely appear around similar contexts in the given corpora, the DIH could not indicate the potential entailment relationship between them. It leads to the *edge sparsity* of EGs as the corresponding edges may be missing due the limited coverage of the corpora.

To tackle the sparsity issues, previous works pay attention to learning global graph structures to mine latent entailment relations and alleviate the edge sparsity (Berant et al., 2011, 2015; Hosseini et al., 2018; Chen et al., 2022), but predicate sparsity is still holding back the improvement of EGs. Solving predicate sparsity by simply scaling up the distributional feature extraction is impracticable, due to the long-tail phenomenon of predicate distribution (McKenna and Steedman, 2022).

The shortcomings of extractive methods come in quest for non-extraction way to overcome. Recent progress in deep generative LMs, including GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2022), makes it possible to produce predicates and entailment relations by generative methods. Inspired by the Commonsense Transformer (Bosselut et al., 2019), we propose a novel generative multi-stage EG construction method, called Typed Predicate-Entailment Graph Generator (TP-EGG). As shown in Figure 2, TP-EGG takes several seed predicates as input of the LM-based predicate generator to depict the domain of predicates and generate more in-domain predicates. With generated predicates, TP-EGG uses a novel transitivity-ensured edge selector by representing predicates as spheres in the vector space, to pick out the potential entailment relations among generated predicates. Then TP-EGG calculates the corresponding edge weights by the LM-based edge calculator. Our key insight is that by re-modeling the predicate extraction process as a generation process, we can leverage the underlying knowledge about natural language inference inside the LMs to avoid the data sparsity issues of extractive methods. By choosing appropriate seed predicates and setting the parameters of TP-EGG, one can generate EGs containing knowledge from a specific domain in arbitrary scales to fit the downstream requirement, without limitations from the uncontrollable distribution in domain-independent corpora. Since almost all the EG construction modules in TP-EGG is controlled by pre-trained LMs, the output EGs can be seen as explicit representations of the knowledge in LMs and used in downstream tasks, such as RTE in our experiments.

In a word, our contributions can be summarized as follows: (1) We propose a novel generative EG construction method to alleviate the data sparsity issues on generated EGs and avoid the reliance on corpora preparation in traditional EG methods; (2) We propose a new method to evaluate the quality of EGs in downstream tasks such as RTE; (3) Our TP-EGG outperforms strong baselines with significant improvement on benchmark datasets, and we show that generation-based EGs methods can alleviate the predicate sparsity by leveraging pre-trained LMs as predicate generators.

2 Related Work

Previous EG construction methods construct feature representations for typed predicates, weighted

by counts or Pointwise Mutual Information (Berant et al., 2015), and compute the distribution similarity guided by DIH. For a predicate pair, different similarities are calculated, such as cosine similarity, Lin (Lin, 1998), Weed (Weeds and Weir, 2003), and Balanced Inclusion (Szpektor and Dagan, 2008). Markov chain of predicate-argument transition (Hosseini et al., 2019) and temporal information from extracted corpora (Guillou et al., 2020) are also used in EGs construction. These methods independently calculate the entailment relations for each pair, called **local** methods. Besides, **global** constraints are used to detect new entailment relations beyond local relations. The transitivity in EGs, which means a entails b and b entails c indicate a entails c for three predicates a, b and c, is the most widely used in previous works as hard constraints (Berant et al., 2011, 2015) or soft loss functions (Hosseini et al., 2018; Chen et al., 2022). The weight similarity constraints between different typed EGs and similar predicates are also taken into consideration (Hosseini et al., 2018).

As one of the most important areas of NLP, text generation, or Natural Language Generation (NLG), has also been advanced by the surgent development of pre-trained LMs. BART (Lewis et al., 2020) uses encoder-decoder transformer architecture to re-correct the corrupted data in pre-training phase; GPT-3 (Brown et al., 2020) uses transformer decoder to achieve in-context learning with massive multi-task unsupervised data. T5 (Raffel et al., 2022) unifies different tasks into natural language prefixes and solves them by text generation.

Pre-trained LMs are also applied in recent EG methods. CNCE (Hosseini et al., 2021) initializes the contextualized embeddings of entity-relation triplets by BERT (Devlin et al., 2019) and uses random walk to get the entailment probability; EGT2 (Chen et al., 2022) fine-tunes a pattern-adapted LM on the predicate sentences and recalculates high-quality edge weights for global constraints; McKenna and Steedman (2022) applies RoBERTa (Liu et al., 2019) as predicate encoder and matches missing predicates in EGs with K-Nearest Neighbor algorithm to alleviate the predicate sparsity. As far as we are concerned, our method is the first attempt to use generative LM in EG construction and directly generate EGs without the distributional features from large corpora.

3 Our Approach

EGs store predicates as nodes and entailment relations between them as edges in graph structures. Following previous EG methods (Hosseini et al., 2018, 2019; Chen et al., 2022), we use the neo-Davisonian semantic form of binary relation (Parsons, 1990) to indicate typed predicates, whose types are defined by the combination of argument types. Predicate p connecting two arguments a_1, a_2 with types t_1, t_2 can be represented as $p = (w_1.i_1, w_2.i_2, t_1, t_2)$, where w_j is the center relation tokens (and perhaps prepositions) about a_j, and i_j is corresponding argument order of a_j in w_j. For example, the event "*The government is elected in 1910 and adored by natives*" contains two predicates $(elect.2, elect.in.2, government, time)$ and $(adore.1, adore.2, person, government)$. We denote P as the collection of all typed predicates, T as the collection of all argument types, and $\tau_1, \tau_2 : P \rightarrow T$ as type indicator functions, where $\tau_1(p) = t_1$ and $\tau_2(p) = t_2$ for any predicate $p = (w_1.i_1, w_2.i_2, t_1, t_2)$.

We formally define that a typed entailment graph $G(t_1, t_2)$=<$P(t_1, t_2), E(t_1, t_2)$> includes the collection of typed predicates $P(t_1, t_2) = \{p|(\tau_1(p), \tau_2(p)) \in \{(t_1, t_2), (t_2, t_1)\}\}$, and the directional weighted edge set $E(t_1, t_2)$, which can be represented as an adjacent matrix $W(t_1, t_2) \in [0, 1]^{|P(t_1,t_2)|\times|P(t_1,t_2)|}$. For those $G(t_1, t_2)$ whose $t_1 \neq t_2$, the order of types $t1, t2$ is naturally determined. When $t_1 = t_2 = t$, argument types are ordered such that $G(t, t)$ can determine the order of types like "Thing A" and "Thing B" to distinguish predicates like "*Thing A eat Thing B*" and "*Thing B eat Thing A*". This order obviously affect the meaning of predicates, as “Thing A eats Thing B” entails “Thing B is eaten by Thing A”, but “Thing eats Thing” is doubtful to entail “Thing is eaten by Thing”.

3.1 Predicate Generation

In order to avoid the predicate sparsity issue in a given corpus, TP-EGG uses a predicate generator $\mathcal{G}$ to generate novel in-domain predicates. $\mathcal{G}$ takes a set of seed predicates $P_{seed} \subset P(t_1, t_2)$ as input and outputs a set of generated predicates $P_{\mathcal{G}}$, where P_{seed} are expected to contain the domain knowledge of required EGs and $P_{\mathcal{G}}$ should be semantically related to P_{seed} in varying degrees.

Our $\mathcal{G}$ is designed to be based on generative LMs, thus the input predicates $p \in P_{seed}$ should

be converted into natural language forms to fit in the LMs. We use Chen et al. (2022)'s sentence generator S to convert predicate p into its corresponding sentence $S(p)$. For example, $p = (elect.2, elect.in.2, government, time)$ will be converted into *Government A is elected in Time B*. With converted sentences, generator $\mathcal{G}$ uses a generative LM, T5-large (Raffel et al., 2022) in our experiments, to generate new sentences and then re-converts them into generated predicates by a sentence-predicate mapping function S^{-1} (details in Appendix C). Starting from the seed sentences $S_0 = \{S(p)|p \in P_{seed}\}$, the generative LM outputs sentences S_1 for the next step, and S_1 is used to generate S_2 and so on, while S^{-1} is used to re-convert S_i to $P_i = S^{-1}(S_i)$ for every step. The generation process continues until the union of seed predicates and generated predicates $P'_i = P_{seed} \cup P_1 ... \cup P_i$ is equal to P'_{i-1} or its size $|P'_i|$ exceeds a pre-defined scale parameter K_p.

To use T5-large as the generation component, we need to design an input template to generate new sentences. For sentence $s \in S_i$, the input template will be constructed like:

s, which entails that t_1 A <*extra_id_0*> t_2 B.

s, which entails that t_2 B <*extra_id_0*> t_1 A.

where <*extra_id_0*> is the special token representing the generating location of the T5-large output. The max length of stripped output sequence s' is limited to 5, and the new predicate p' is produced by S^{-1}("t_1 A s' t_2 B.") or S^{-1}("t_2 B s' t_1 A.") correspondingly. For each s, T5-large uses beam-search algorithm with beam size K_{beam} to find top-K_{sent} output sequences s' with highest probabilities.

To ensure the quality of generated predicates and filter noisy ones, only those predicates which are generated by T5-large from at least two different predicates in P'_{i-1} could be included in P_i. Algorithm 1 depicts how predicate generator $\mathcal{G}$ works (more details and exmaples in Appendix D).

Algorithm 1 The predicate generator $\mathcal{G}$.

Require: A set of seed predicates P_{seed}, sentence generator S, parameter K_{beam}, K_{sent}, K_p
Ensure: A set of generated predicates $P_{\mathcal{G}}$

$P_{once} = \{\}$
$i = 0, P_0 = P'_0 = P_{seed}$
while $|P'_i| \leq K_p$ **do**
  $S_i = \{S(p)|p \in P_i\}$
  $P_{i+1} = \{\}$
  for $s \in S_i$ **do**
    $S^g = T5(s, K_{beam}, K_{sent})$
    $P^g = Set(S^{-1}(s^g)|s^g \in S^g)$
    $P^g = P^g - P'_i$
    P_{i+1}.update($P^g \cap P_{once}$)
    $P_{once} = P_{once}$ XOR P_g
  end for
  $P_{i+1} = P_{i+1} - P'_i$
  $P'_{i+1} = P'_i \cup P_{i+1}$
  if $P'_{i+1} = P'_i$ **then**
    return $P_{\mathcal{G}} = P'_i$
  end if
  $i = i + 1$
end while
return $P_{\mathcal{G}} = P'_i$

3.2 Edge Selection

After generating new predicates $P(t_1, t_2) = P_{\mathcal{G}}$, TP-EGG constructs $G(t_1, t_2)$ by generating weighted edge set $E(t_1, t_2)$. As TP-EGG does not use large corpora to calculate distributional features regarding context coherence, we need to determine which predicate pairs could be potential entailment relations for later calculation. Regarding ALL pairs as candidates is a simple solution, but when $P(t_1, t_2)$ scales up, calculating all $|P|^2$ pairs will be unacceptably expensive as we intend to adopt an LM-based edge weight calculator, which only takes one pair as input at a time. Therefore, we require an effective edge selector $\mathcal{M}$ to select potential pairs $E' \subset P(t_1, t_2) \times P(t_1, t_2)$ with acceptable computational overhead, where $|E'|$ should be equal to a given parameter K_{edge}.

Calculating embeddings for each predicate and quickly getting similarities between all pairs in $P(t_1, t_2)$ perform worse than pair-wise LMs with cross attention in general, but are good enough as the edge selector to maintain high-quality pairs in high ranking. Inspired by Ristoski et al. (2017), we represent predicate p as a sphere in the vector space. TP-EGG uses BERT-base (Devlin et al., 2019) to calculate embedding vector v_p for every predicate p based on $S(p)$, and represents p as a sphere $\odot_p$ in a vector space with center c_p and radius r_p:

$$\begin{aligned} v_p &= BERT(S(p)) \in R^{d_v}, \\ c_p &= f_c(v_p) \in R^{d_c}, \\ r_p &= f^+(f_r(v_p)) \in R_+. \end{aligned} \tag{1}$$

where f_c, f_r are two-layer trainable neural networks, d_v, d_r are corresponding vector dimensions, $f^+(x) \in \{\exp(x), x^2\}$ ensures the positive radius. By representing p as a sphere, we expect that when p entails q, $\odot_q$ should enclose $\odot_p$, as all points in $\odot_p$ are also included in $\odot_q$. Under such assumption, the transitivity referred in Section 2 is naturally satisfied as $\odot_a \subset \odot_b \subset \odot_c$. The overlapping ratio between spheres can be seen as the entail-

ment probability $Pr(p \rightarrow q)$, and we simplify the calculation of sphere overlapping to diameter overlapping along the straight line between two centers:

$$d_{pq} = ||c_p - c_q||_2$$
$$Pr(p \rightarrow q) = \begin{cases} 0 & , r_q \leq d_{pq} - r_p \\ 1 & , r_q \geq d_{pq} + r_p \\ \frac{r_p + r_q - d_{pq}}{2r_p} & , otherwise \end{cases} \quad (2)$$

Chen et al. (2022) defines soft transitivity as $Pr(a \rightarrow b)Pr(b \rightarrow c) \leq Pr(a \rightarrow c)$ for all predicate pairs above a threshold. Similar in spirit, our simplified sphere-based probability holds transitivity in part:

Theorem 1 *Given a threshold $\epsilon \in (0, 1)$, $\forall a, b, c$ where $Pr(a \rightarrow b) > \epsilon$ and $Pr(b \rightarrow c) > \epsilon$, we have $Pr(a \rightarrow c) > \epsilon - (1 - \epsilon)\frac{r_b}{r_a}$.*

We give its proof in Appendix A. Noted that while ϵ is close to 1, the right part $\epsilon - (1 - \epsilon)\frac{r_b}{r_a}$ will be nearly equal to ϵ. As we use this probability in edge selection, higher $Pr(a \rightarrow b)$ and $Pr(b \rightarrow c)$ will naturally ensure the appearance of (a, c) in final entailment relations, without the disturbance from low-confident edges. As $Pr(p \rightarrow q)$ is constant when $r_q \leq d_{pq} - r_p$ or $r_q \geq d_{pq} + r_p$, its gradient becomes zero which makes it untrainable. Therefore, we smooth it with order-preserving Sigmoid function and interpolation, and finally get the selected edge set for $G(t_1, t_2)$:

$$\mathcal{M}(p, q) = \sigma(\frac{2r_q - 2d_{pq}}{r_p}),$$
$$E(t_1, t_2) = \{topK_{edge}(\mathcal{M}(p, q))|p, q \in V(t_1, t_2)\} \quad (3)$$

where σ is Sigmoid function $\sigma(x) = 1/(1+e^x)$. A geometrical illustration presenting how the selector $\mathcal{M}$ works can be found in Appendix B.

3.3 Edge Weight Calculation

With the selected edge set $E(t_1, t_2) \subset P(t_1, t_2) \times P(t_1, t_2)$, TP-EGG calculates the edge weight $W_{p,q}$ for each predicate pairs (p, q) individually in the adjacent matrix $W(t_1, t_2)$. Inspired by Chen et al. (2022), as the distributional features of generated predicates are unavailable for TP-EGG, we re-implement their local entailment calculator $\mathcal{W}$ to obtain the entailment edge weight $W_{p,q}$. $\mathcal{W}$ is based on DeBERTa (He et al., 2020, 2021a) and fine-tuned to adapt to the sentence patterns generated by S. The entailment-oriented LM will produce three scores, corresponding to entailment (E), neutral (N) and contradiction (C) respectively, for each sentence pair. The score of entailment class is used as the entailment edge weight in our EGs:

$$W_{p,q} = \mathcal{W}(p, q) = \frac{exp(LM(\text{E}|p, q))}{\sum_{r \in \{\text{E,N,C}\}} exp(LM(r|p, q))} \quad (4)$$

where $LM(r|p, q)$ is the score of class r. After calculating all predicate pairs $(p, q) \in E(t_1, t_2)$ by the LM-based calculator $\mathcal{W}$, TP-EGG completes the adjacent matrix $W(t_1, t_2)$, and consequently constructs $G(t_1, t_2)$, as shown in Figure 2.

Name	Valid	Test	Total	#Pos/#Neg
Levy/Holt	5,486	12,921	18,407	0.270
Levy/Holt-r	5,450	12,817	18,267	0.261
Berant	-	39,012	39,012	0.096
SherLIiC	996	2,989	3,985	0.498

Table 1: The dataset statistics.

4 Experimental Setup

Datasets. Following previous works (Hosseini et al., 2018, 2019, 2021; Chen et al., 2022), we include Levy/Holt Dataset (Levy and Dagan, 2016; Holt, 2018) and Berant Dataset (Berant et al., 2011) into EG evaluation datasets. Besides, we reorganize the SherLIiC Dataset (Schmitt and Schütze, 2019), a dataset for Lexical Inference in Context (LIiC), into an EG benchmark. We further re-annotate conflicting pairs in Levy/Holt, referred as Levy/Holt-r Dataset. Dataset statistics are shown in Table 1. More details can be found in Appendix F.

Metrics. Following previous works, we evaluate TP-EGG on the test datasets by calculating the area under the curves (AUC) of Precision-Recall Curve (PRC) for precision>0.5 and traditional ROC curve.[2] The evaluated EGs are used to match the predicate pairs in datasets and return the entailment scores. Noted that our generated predicates might be semantically same with required ones but have different forms, like *(use.2,use.in.2,thing,event)* and *(be.1,be.used.in.2,thing,event)* are both reasonable for *"Thing A is used in Event B"* while our S^{-1} generates the first one. Hence we relax the predicate matching standard in evaluation from exactly matching to sentence matching, i.e., $S(p) = S(p')$

[2]We have found that the evaluation scripts written by Hosseini et al. (2018) do not connect the curve with (1,0) and (0,1) point correctly, which wrongly decreases the performance. We fix and use the scripts to generate results in this paper.

rather than $p = p'$. This modification has nearly no effect on previous extraction-based EGs, but can better evaluate generative methods..

Implementation Details. In experiments, TP-EGG uses BERT-base in $\mathcal{M}$ and T5-large in $\mathcal{G}$ implemented by the Hugging Face transformer library (Wolf et al., 2020)[3], and DeBERTa re-implementation from Chen et al. (2022) to fine-tune on MNLI and adapt to sentence pattern in $\mathcal{W}$. Taking both EG performance and computational overhead into account, we set $K_p = 5 \times 10^3$, $K_{edge} = 2 \times 10^7$, $K_{beam} = 50$, $K_{sent} = 50$, $d_r = 16$, $d_v = 768$. Discussion about K_p and K_{edge} can be found in Appendix E.

For EG generation, TP-EGG uses the predicates in validation set of Levy/Holt-r and SherLIiC Dataset respectively as the seed predicate P_{seed}. With different P_{seed}, we also only use corresponding validation set as the training data for all later modules to keep the EGs *in-domain*, called TP-EGG$_{L/H-r}$ and TP-EGG$_{SherLIiC}$ respectively.

Only positive pairs are used to generate the training inputs and outputs to fine-tune T5-large in the predicate generator $\mathcal{G}$ with learning rate $\alpha_{\mathcal{G}} = 10^{-3}$. We use $f^+(x) = \exp(x)$ for TP-EGG$_{L/H-r}$ and $f^+(x) = x^2$ for TP-EGG$_{SherLIiC}$. The edge selector $\mathcal{M}$ is also trained by the validation predicate pairs, but the positive examples are repeat 5 times (for Levy/Holt-r) or 2 times (for SherLIiC) to alleviate the label imbalance in training. BERT-base parameters are trained with learning rate $\alpha_{\mathcal{M},1} = 10^{-5}$, while other parameters, including f_c and f_r, are trained with learning rate $\alpha_{\mathcal{M},2} = 5 \times 10^{-4}$. The edge weight calculator $\mathcal{W}$ is trained by the same method in Chen et al. (2022).

All modules are trained by AdamW optimizer (Loshchilov and Hutter, 2018) with cross entropy loss function, and controlled by early-stop mechanism, which stops the training when performances (loss for $\mathcal{G}$ and F_1 for others) on validation set do not reach the highest in the last 10 epoches. It takes about 5-6 hours to train all modules in TP-EGG, and about 2-3 hours to generate a typed EG on GeForce RTX 3090. The three modules, $\mathcal{G}$, $\mathcal{M}$ and $\mathcal{W}$, contain 738M, 109M and 139M parameters respectively.

To be comparable with previous works (Hosseini et al., 2018), we apply their lemma-based heuristic on all datasets except SherLIiC, and their average backup strategy on all datasets.

Compared Methods We compare TP-EGG with the best local distributional feature, Balanced Inclusion or called BInc (Szpektor and Dagan, 2008), and existing state-of-the-art local and global EG construction methods, including Hosseini et al. (2018, 2019), CNCE (Hosseini et al., 2021) and EGT2 (Chen et al., 2022).

Downstream Task. Despite of evaluating on EG construction benchmarks, we adapt an LM-based three-way RTE framework into the EG evaluation testbed. For premise pm and hypothesis h, RTE models take their concatenation $[pm; h]$ as inputs, and return three probability scores of three classes. In order to incorporate the knowledge in EGs into RTE models, we design the following architecture available to any LM-based RTE model: given pm and h, we extract binary predicates from them, and try to match the predicates in our EGs. Each matched predicates a in premise pm will be replaced by its K_{nbr} neighbors b with highest weight W_{ab}. For h, the neighbors b are with highest weight W_{ba}. Replaced sentences $pm_1, ..., pm_j$ and $h_1, ..., h_k$ for pm and h will be concatenated to represent the information from EGs in calculation:

$$\begin{aligned}(s_{\mathrm{E1}}, s_{\mathrm{N1}}, s_{\mathrm{C1}}) &= \mathrm{Softmax}(LM_1([pm; h])),\\ (s_{\mathrm{E2}}, s_{\mathrm{N2}}, s_{\mathrm{C2}}) &= \mathrm{Softmax}(LM_2([pm;\\ &\quad pm_1; ...; pm_j; h; h_1; ...; h_k])),\\ s_i &= (s_{i1} + s_{i2})/2, \quad i \in \{\mathrm{E}, \mathrm{N}, \mathrm{C}\}.\end{aligned} \tag{5}$$

where LM_1 and LM_2 represent two different LMs followed by a linear layer respectively. As the additional calculation unfairly requires more parameters, we also consider the models with equal parameters but do not use the EGs, referred as *NO-EG* setting, by inputting $[pm; h]$ into LM_2 directly. We use SNLI (Bowman et al., 2015) and SciTail (Khot et al., 2018) as our RTE benchmark datasets. We use BERT-base and DeBERTa-base as the backbone, learning rate $\alpha_{RTE} = 10^{-5}$, $K_{nbr} = 5$ for SNLI and $K_{nbr} = 3$ for SciTail.

5 Results and Analysis

5.1 Main Results

The performance of different EGs on benchmark datasets are shown in Table 2, and the Precision-Recall Curves of EGs on Levy/Holt-r and Berant datasets are presented in Figure 3. Without

[3]https://github.com/huggingface/transformers

Methods	L/H		L/H-r		Berant		SherLIiC	
	PRC	ROC	PRC	ROC	PRC	ROC	PRC	ROC
BInc (Szpektor and Dagan, 2008)	.262	.632	.254	.632	.242	.676	.170	.605
Hosseini et al. (2018)	.271	.638	.254	.637	.268	.682	.184	.611
Hosseini et al. (2019)	.275	.640	.270	.640	.213	.678	.148	.566
CNCE (Hosseini et al., 2021)	.301	.643	.300	.645	.269	.705	.233	.602
EGT2-Local (Chen et al., 2022)	.453	.733	.447	.732	.562	.779	.385	.665
- w/ L_3 global	.477	.755	.478	.756	.583	.780	.391	**.705**
TP-EGG$_{L/H-r}$	.543	.755	.527	.748	.633	.780	.175	.606
- w/ EGT2-L_1 global	**.549**	**.778**	**.532**	**.773**	**.637**	**.822**	.184	.615
TP-EGG$_{SherLIiC}$	.263	.589	.261	.588	.171	.642	**.394**	.669
- w/ EGT2-L_1 global	.264	.616	.261	.616	.173	.658	**.394**	.680

Table 2: The main results for TP-EGG$_{L/H-r}$, TP-EGG$_{SherLIiC}$ and baselines on EG benchmark datasets. The best performances of each metric are **boldfaced**, and the out-domain results are with gray ground color.

using extracted features from large corpora, TP-EGG achieves significant improvement or at least reaches comparable performance with baselines for in-domain evaluations (L/H and L/H-r for TP-EGG$_{L/H-r}$ and SherLIiC for TP-EGG$_{SherLIiC}$). Interestingly, TP-EGG always performs better on the AUC of PRC, which indicates the strong ability of our generative methods to maintain impressive recall with high precision as shown in the curves. On Levy/Holt-r, TP-EGG$_{L/H-r}$ significantly outperforms all other extraction-based methods on precision>0.5, showing that with higher classification threshold, extraction-based methods fail to detect the entailment relations between rare predicates due to the sparsity issues, while generation-based TP-EGG successfully finds these relations by generating more predicates and correctly assigns high probabilities between them.

Noted that our TP-EGG is a local method, although certain global properties are ensured by our edge selector $\mathcal{M}$. We try to apply a state-of-the-art global method, EGT2-L_1 (Chen et al., 2022) on our local EGs[4]. As shown in the bottom of Table 2, the global method further improves the performance of TP-EGG, demonstrating the potential of our local EGs to continuously reducing the data sparsity with global EG learning methods.

Although we have observed the significant improvement of evaluation metrics by TP-EGG, it is not clear enough to determine TP-EGG can alleviate the predicate sparsity to what extent. Therefore, we count the predicate pairs in Levy/Holt testset that exactly appeared as edges in EGs. We find

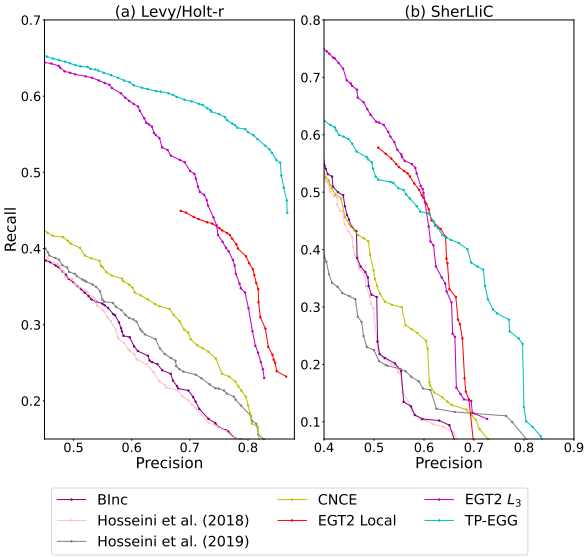


Figure 3: The Precision-Recall Curves of EGs on (a) Levy/Holt-r Dataset and (b) SherLIiC Dataset. For TP-EGG, the EGs are constructed with in-domain data.

that 6,873 pairs appear in TP-EGG$_{L/H-r}$, meanwhile 875 in EGT2-L_3. The far more appearance of in-domain predicates indicates the alleviation of predicate sparsity.

Previous works have claimed that LMs for entailments might be strong in undirectional paraphrasing, but weak in directional entailment recognizing (Cabezudo et al., 2020; Chen et al., 2022). To check out the directional entailment ability of TP-EGG and other methods, we evaluate them on the directional portion[5] of Levy/Holt Dataset as shown in Table 3. The directional portion contains entailment pairs (p, q) where $(p \rightarrow q) XOR (q \rightarrow p)$ is

[4] Chen et al. (2022) reports that L_3 variant performs better on their local graphs, but we find L_1 is better on TP-EGG.

[5] https://github.com/mjhosseini/entgraph_eval/tree/master/LevyHoltDS

Methods	PRC	ROC
BInc	.538	.528
Hosseini et al. (2018)	.535	.529
Hosseini et al. (2019)	.554	.556
CNCE	.557	.561
EGT2-Local	.597	.604
- w/ EGT2-L_3 global	.626	.644
TP-EGG$_{L/H-r}$	.609	.596
- w/ EGT2-L_1 global	.636	.633

Table 3: Performance on the directional portion of Levy/Holt Dataset.

	P_{seed}	$\mathcal{G}$	$\mathcal{M}, \mathcal{W}$	L/H-r	SLIC
①	L/H-r	L/H-r	L/H-r	.527	.175
②	L/H-r	L/H-r	SLIC	.426	.213
③	L/H-r	SLIC	L/H-r	.411	.323
④	L/H-r	SLIC	SLIC	.312	.384
⑤	SLIC	L/H-r	L/H-r	.452	.261
⑥	SLIC	L/H-r	SLIC	.361	.328
⑦	SLIC	SLIC	L/H-r	.307	.320
⑧	SLIC	SLIC	SLIC	.261	.392

Table 4: Performance (AUC of PRC) on Levy/Holt-r and SherLIiC with different combinations of training data and modules. SLIC represents SherLIiC.

True, and therefore symmetric models will have AUC<0.5. TP-EGG performs better than baselines on the directional portion, and the AUC far higher than 0.5 indicates its directional entailment ability. Global models perform better here, which is reasonable as global constraints are strongly related to the directional reasoning.

5.2 Learning with Multiple Domains

Although TP-EGG performs well on in-domain evaluation, the out-domain scenario is still hard, as the knowledge required for out-domain evaluation is inaccessible in all training and generation steps of TP-EGG. To check the impact of training data domains in different modules of TP-EGG, we use Levy/Holt-r and SherLIiC Dataset to produce seed predicates P_{seed} and train different modules, including predicate generator $\mathcal{G}$, edge selector $\mathcal{M}$ and weight calculator $\mathcal{W}$, with different combinations of two datasets. As shown in Table 4, involving in-domain training data into more modules will lead to higher performance on corresponding dataset in general, which is in accordance with expectation.

Interestingly, by comparing different combinations, we find that fine-tuning $\mathcal{G}$ with data from domains different with P_{seed} will lead to better overall performance on two datasets. For example, row ③ attains improvement about 0.15 on SherLIiC with dropping about 0.11 on Levy/Holt-r by changing training data of $\mathcal{G}$ from Levy/Holt-r (①) to SherLIiC, and when P_{seed} also changes to SherLIiC (⑦), the performance on Levy/Holt-r is severely damaged without benefit to SherLIiC. Similar situation is also observed in row ②,④ and ⑧. We assume that involving knowledge from different domains in predicate generation, i.e. P_{seed} and $\mathcal{G}$, could alleviate the over-fitting by mixing two predicate domains and encouraging $\mathcal{G}$ to find more novel predicates to cover the gap between training and testing. Empirically, involving different data in $\mathcal{G}$ leads to the best performance among the modules.

	P_{seed}	$\mathcal{G}, \mathcal{M}, \mathcal{W}$	L/H-r	SherLIiC
❶	L+S	L+S	.496	.388
❷	L/H-r	L/H-r	.527	.175
❸	L+S	L/H-r	.532	.286
❹	L/H-r	L+S	.518	.321
❺	SherLIiC	SherLIiC	.261	.394
❻	L+S	SherLIiC	.322	.416
❼	SherLIiC	L+S	.405	.367

Table 5: Performances (AUC of PRC) on Levy/Holt-r Dataset and SherLIiC Dataset of TP-EGG trained with merged multi-domain data.

Next, we study the effect of using merged validation sets of Levy/Holt-r and SherLIiC Dataset at different modules. The performance of TP-EGG trained with the merged data, referred as L+S, are shown in Table 5. While using merged data as P_{seed} and also as training data for other modules (❶), TP-EGG reaches impressive performances on both datasets, which is not surprising, as both datasets are in-domain in this situation.

Using merged dataset to train $\mathcal{G}, \mathcal{M}$ and W boosts out-domain performance with in-domain performance loss (comparing ❷ and ❹, ❺ and ❼). However, adding some out-domain predicates into P_{seed} is surprisingly beneficial to the in-domain evaluation while improving out-domain generalization (comparing ❷ and ❸, ❺ and ❻). We attribute it to the diversity of generated predicates led by the newly incorporated seed predicates, which might not be generated with the in-domain seed predicates. The out-domain predicates help TP-EGG to find new predicates related to in-domain predicates as Algorithm 1 might tend to generate predicates

Model	EG	SNLI	SciTail
BERT	Original	90.03±0.04	91.42±0.21
	NO-EG	90.17±0.19	92.64±0.07
	CNCE	90.10±0.19	92.15±0.98
	EGT2-L_3	90.08±0.05	92.35±0.05
	TP-EGG	**90.28±0.22**	**92.94±0.92**
DeBERTa	Original	91.59±0.26	94.20±0.55
	NO-EG	91.69±0.03	94.62±0.23
	CNCE	91.57±0.19	95.06±0.33
	EGT2-L_3	91.35±0.24	94.57±0.46
	TP-EGG	**91.90±0.11**	**95.19±0.20**

Table 6: Performances of RTE models supported with different EGs on RTE datasets (average over 3 runs). The best performances are **boldfaced**.

from at least two predicates across two domains. Therefore, the predicate coverage over evaluation datasets can be increased.

5.3 Results on RTE

In downstream task evaluation, we use EGs generated by different methods to enhance LM-based RTE models, and report the results in Table 6. Compared with CNCE and EGT2, our TP-EGG achieves better performance on two RTE datasets with both BERT$_{base}$ and DeBERTa$_{base}$ backbones. The performances of TP-EGG on DeBERTa$_{base}$ are significantly better than NO-EG (p<0.05). Noted that TP-EGG offers pm_j, h_k for 4,600 sentences in SNLI testset, which is 5,596 for EGT2-L_3. Even with lower coverage over predicates in the dataset, TP-EGG supports RTE models with more high-quality entailment relations to generate pm_j, h_k and improve the performance. On the other hand, the noisy entailment relations in CNCE and EGT2 perhaps misguide RTE models, thus lead to even worse results than *NO-EG* in some cases.

5.4 Ablation Study

We run the ablation experiments which directly use the original version of LMs in $\mathcal{G}$, $\mathcal{M}$ and $\mathcal{W}$ without fine-tuning on EG benchmark datasets. For $\mathcal{M}$, as non-LM parameters are involved, we replace it with *randomly* selecting K_{edge} edges. As shown in Table 7, without fine-tuning $\mathcal{G}$ or $\mathcal{W}$, the performance on Levy/Holt-r suffers a significant drop (about 0.1), indicating the importance of fine-tuned modules for EG generation. The performance on SherLIiC also decreases severely without fine-tuning $\mathcal{G}$, as the fine-tuning step can improve the quality of generated predicates and cover more out-domain predicates. Fine-tuning $\mathcal{W}$ critically affects the result on Berant Dataset, which is compatible with the results in Chen et al. (2022), showing the importance of fine-tuning and pattern adaptation in weight calculation on this dataset. Fine-tuning $\mathcal{M}$ is mainly beneficial to SherLIiC by comparison. From the results, we can see that high quality predicate pair construction from $\mathcal{G}$ and $\mathcal{M}$ is more beneficial to out-domain evaluation, while the weight calculation from $\mathcal{W}$ plays a more important role for in-domain cases.

Method	L/H-r	Berant	SherLIiC
TP-EGG$_{L/H-r}$	.527	.633	.175
- w/o fine-tuning $\mathcal{G}$	.422	.508	.132
- w/o training $\mathcal{M}$	.518	.615	.152
- w/o fine-tuning $\mathcal{W}$	.429	.305	.166

Table 7: Experiment results of ablation study with different modules in TP-EGG.

6 Conclusions

In this work, we propose a novel generative typed entailment graph construction method, called TP-EGG, with predicate generation, edge selection and calculation modules. TP-EGG takes several seed predicates as input to the predicate generator to find novel predicates, selects potential entailment predicate pairs as edges, and calculates the edge weights without distributional features. TP-EGG can construct high-quality EGs with flexible scales and avoid the data sparsity issues to some extent. Experiments on EG benchmacks and RTE task show the significant improvement of TP-EGG over the state-of-the-art EG learning methods. We find that mixing data from different domains in different ways can improve the generalization of TP-EGG in varying degrees, and using out-domain data in predicate generation modules brings the most significant improvement.

Limitations

First, as we do not rely on specific corpora and avoid the shortcomings of extractive methods, we also lose their advantages. The typed EGs generated by our TP-EGG is strongly related to the seed predicates and training data of generation modules, while extractive EGs can generate domain-independent EGs from large corpora and do not require supervised training data to a considerable degree. Second, the edge calculator $\mathcal{W}$ is time-

consuming even we can control the scales of output EGs, as the edge num $|E(t_1, t_2)|$ will be relatively large for TP-EGG to generate powerful EGs. Furthermore, how to effectively select seed predicates still remains a difficult problem which has not been discussed thoroughly in this work by using the validation datasets. We assume that this problem could be solved by carefully confirming how the seed predicates represent corresponding domain knowledge and we leave it to future work.

Ethics Statement

We re-annotate the Levy/Holt Dataset which is a publicly available dataset for entailment graph evaluation. Annotators receive a competitive pay of about 100 yuan per hour under the agreement of the institute, which is more than 4 times the local minimum wage. The annotation complies with the ACL Code of Ethics. The sentences used in annotation are generated from the original dataset and we do not incorporate external content into the sentences. However, there may still be sentences containing potentially improper content, which do not reflect the views or stances of the authors. The re-annotation results are confirmed by the majority voting of annotators, and may still contain natural errors. Further usage of the re-annotated dataset should be aware of the limitation and the authors are not responsible for any issues in further usage of this dataset.

Acknowledgements

This work is supported in part by NSFC (62161160339). We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

Jonathan Berant, Noga Alon, Ido Dagan, and Jacob Goldberger. 2015. Efficient global learning of entailment graphs. *Computational Linguistics*, 41(2):221–263.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 610–619, Portland, Oregon, USA. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Marco Antonio Sobrevilla Cabezudo, Marcio Inácio, Ana Carolina Rodrigues, Edresson Casanova, and Rogério Figueredo de Sousa. 2020. Natural language inference for portuguese using bert and multilingual information. In *International Conference on Computational Processing of the Portuguese Language*, pages 346–356. Springer.

Zhibin Chen, Yansong Feng, and Dongyan Zhao. 2022. Entailment graph learning with textual entailment and soft transitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5899–5910, Dublin, Ireland. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of

clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0).

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan. Association for Computational Linguistics.

Liane Guillou, Sander Bijl de Vroe, Mohammad Javad Hosseini, Mark Johnson, and Mark Steedman. 2020. Incorporating temporal information in entailment graph mining. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 60–71, Barcelona, Spain (Online). Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021a. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021b. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Xavier Holt. 2018. Probabilistic models of relational implication. *arXiv preprint arXiv:1907.12048*.

Mohammad Javad Hosseini, Nathanael Chambers, Siva Reddy, Xavier R. Holt, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2018. Learning typed entailment graphs with global soft constraints. *Transactions of the Association for Computational Linguistics*, 6:703–717.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2019. Duality of link prediction and entailment graph induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4736–4746, Florence, Italy. Association for Computational Linguistics.

Mohammad Javad Hosseini, Shay B. Cohen, Mark Johnson, and Mark Steedman. 2021. Open-domain contextual link prediction and its complementarity with entailment graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2790–2802, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI Conference on Artificial Intelligence*.

Omer Levy and Ido Dagan. 2016. Annotating relation inference in context via question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 249–255, Berlin, Germany. Association for Computational Linguistics.

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.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 2*, pages 768–774, Montreal, Quebec, Canada. 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. *CoRR*, abs/1907.11692.

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

Nick McKenna, Liane Guillou, Mohammad Javad Hosseini, Sander Bijl de Vroe, Mark Johnson, and Mark Steedman. 2021. Multivalent entailment graphs for question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10758–10768, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nick McKenna and Mark Steedman. 2022. Smoothing entailment graphs with language models. *ArXiv*, abs/2208.00318.

Terence Parsons. 1990. Events in the semantics of english: A study in subatomic semantics.

Amarnath Pathak, Riyanka Manna, Partha Pakray, Dipankar Das, Alexander Gelbukh, and Sivaji Bandyopadhyay. 2021. Scientific text entailment and a textual-entailment-based framework for cooking domain question answering. *Sādhanā*, 46(1):1–19.

Jonathan Pilault, Amine Elhattami, and Christopher J. Pal. 2020. Conditionally adaptive multi-task learning: Improving transfer learning in NLP using fewer parameters & less data. *CoRR*, abs/2009.09139.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Petar Ristoski, Stefano Faralli, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Large-scale taxonomy induction using entity and word embeddings. In *Proceedings of the International Conference on Web Intelligence*, WI '17, page 81–87, New York, NY, USA. Association for Computing Machinery.

Martin Schmitt and Hinrich Schütze. 2019. SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 902–914, Florence, Italy. Association for Computational Linguistics.

Stefan Schoenmackers, Jesse Davis, Oren Etzioni, and Daniel Weld. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098, Cambridge, MA. Association for Computational Linguistics.

Idan Szpektor and Ido Dagan. 2008. Learning entailment rules for unary templates. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 849–856, Manchester, UK. Coling 2008 Organizing Committee.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 81–88.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 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.

Masashi Yoshikawa, Koji Mineshima, Hiroshi Noji, and Daisuke Bekki. 2019. Combining axiom injection and knowledge base completion for efficient natural language inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7410–7417.

Congle Zhang and Daniel S. Weld. 2013. Harvesting parallel news streams to generate paraphrases of event relations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1776–1786, Seattle, Washington, USA. Association for Computational Linguistics.

A The Proof of Theorem 1

Theorem 1 *Given a threshold* $\epsilon \in (0,1)$, $\forall a,b,c$ *where* $Pr(a \to b) > \epsilon$ *and* $Pr(b \to c) > \epsilon$, *we have* $Pr(a \to c) > \epsilon - (1-\epsilon)\frac{r_b}{r_a}$.

As $Pr(p \to q) = \frac{r_p + r_q - d_{pq}}{2r_p}$ holds when $d_{pq} - r_p < r_q < d_{pq} + r_p$, and $Pr(p \to q) = 1 \leq \frac{r_p + r_q - d_{pq}}{2r_p}$ holds when $r_q \geq d_{pq} + r_p$, we have:

$$\begin{aligned} &\frac{r_a + r_b - d_{ab}}{2r_a} \geq Pr(a \to b) > \epsilon \\ &\quad \to d_{ab} < r_b + (1-2\epsilon)r_a. \end{aligned} \tag{6}$$

Similarly, for b, c:

$$d_{bc} < r_c + (1-2\epsilon)r_b. \tag{7}$$

For the case $Pr(a \to c) = 1$, obviously the theorem holds for $\epsilon \in (0,1)$;

For the case $Pr(a \to c) = 0$ or $Pr(a \to c) = \frac{r_p + r_q - d_{pq}}{2r_p}$, we have $Pr(a \to c) \geq \frac{r_p + r_q - d_{pq}}{2r_p}$ as $r_p + r_q - d_{pq} < 0$ under $Pr(a \to c) = 0$, and therefore:

$$\begin{aligned} &Pr(a \to b) \\ \geq &\frac{r_a + r_c - d_{ac}}{2r_a} \\ \geq &\frac{r_a + r_c - (d_{ab} + d_{bc})}{2r_a} \quad (d_{ac} \leq d_{ab} + d_{bc}) \\ > &\frac{r_a + r_c - (r_b + (1-2\epsilon)r_a + r_c + (1-2\epsilon)r_b)}{2r_a} \\ = &\frac{\epsilon r_a + (\epsilon - 1)r_b}{r_a} \\ = &\epsilon + (\epsilon - 1)\frac{r_b}{r_a}. \end{aligned} \tag{8}$$

Q.E.D.

B Geometrical Illustration of Edge Selector $\mathcal{M}$

To understand how the edge selector $\mathcal{M}$ works more intuitively, we pick four predicate sentences from Levy/Holt Dataset and visualize their corresponding spheres $\odot_p$ in Figure 4:

p_0: Living Thing A is imported from Location B.

p_1: Living Thing A is native to Location B.

p_2: Living Thing A is found in Location B.

p_3: Living Thing A is concentrated in Location B.

The centers c_p and radius r_p are generated by $\mathcal{M}$ from our final TP-EGG model, while the dimension of c_p are reduced to maintain the distances between them. Three entailment relations, $p_0 \to p_1$, $p_1 \to p_2$ and $p_3 \to p_2$, are annotated in the dataset, and $p_0 \to p_3$ is also plausible. In Figure 4, the hypothesis spheres obviously enclose premise spheres, and the more generic a predicate

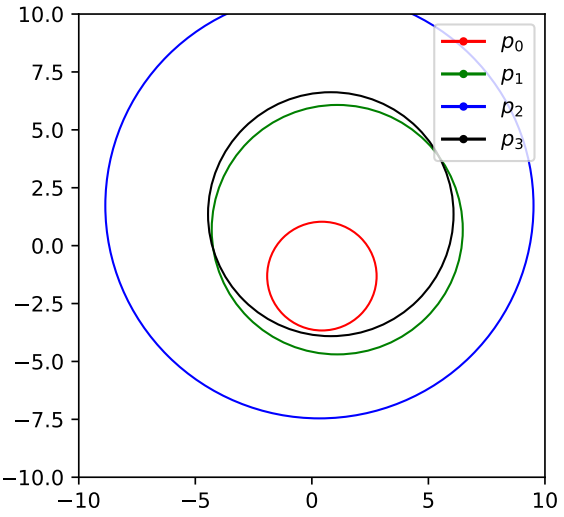


Figure 4: The visualized predicate spheres of four predicates.

is, the bigger its sphere becomes, which is consistent with our expectation about $\mathcal{M}$. With high directional overlapping, all of the four entailment relations will correctly appear in later weight calculation while low-confident inverse edges will be filtered out.

C The Sentence-Predicate Mapping Function S^{-1}

The sentence-predicate mapping function S^{-1} used in predicate generation is described in Algorithm 2. Noted that S^{-1} is a simplified approximation of the reverse function of sentence generator S while different predicates might generate the same sentence by S. Therefore, S^{-1} does not cover all possible predicates and sentences.

Algorithm 2 The mapping function S^{-1}.

Require: A generated sentence s.
Ensure: A predicate p, or $NULL$ indicating that s is not a valid predicate sentence.

Split the sentences into tokens l and strip t_1 A, t_2 B
prefix=""
if $|l| = 0$ **then**
    return NULL
end if
if *not* or *n't* in l[0] **then**
    prefix=NEG // representing the negation
end if
Remove the modal verbs in l
if l begins with *have been* or *has been* **then**
    l=l[1:]
end if
if $|l| > 1$ and $l[:2]$ is *have+P.P.* **then**
    l=l[1:]
end if
if $|l| > 2$ and the present tense of $l[:2]$ is *have to* **then**
    l=l[2:]
end if
if $|l| = 0$ **then**
    return NULL
end if
$i_{head} = 0, i_{tail} = |l| - 1$
while $i_{head} \leq i_{tail}$ and $l[i_{head}]$ is not a verb **do**
    $i_{head} = i_{head} + 1$
end while
while $i_{head} \leq i_{tail}$ and $l[i_{tail}]$ is not a verb or a preposition **do**
    $i_{tail} = i_{tail} - 1$
end while
if $i_{head} > i_{tail}$ **then**
    return NULL
end if
$l' = l[i_{head} : i_{tail} + 1]$ // cut the token between i_{head} and i_{tail}
if $l'[0:2]$ is a verb like *be doing* **then**
    $l' = l'[1:]$
end if
$t = lemmatize(l'[0])$
if t is *be* **then**
    if $|l'| = 1$ **then**
        return prefix+$(be.1, be.2, t_1, t_2)$
    end if
    if $l'[1]$ is not a preposition **then**
        if $l'[1]$ is an adverb **then**
            $l' = l'[0:1] + l'[2:]$
        end if
        if $l'[1]$ is an adjective or a noun, and $l'[-1]$ is a preposition **then**
            $l'[1] = lemmatize(l'[1])$
            return prefix+$(l'[1].1, l'[1:].2, t_1, t_2)$
        end if
        if $l'[1]$ is P.P. verb **then**
            $l'[1] = lemmatize(l'[1])$
            if $l'[-1]$ is a preposition **then**
                return prefix+$(l'[1].2, l'[1:].2, t_1, t_2)$
            else
                return prefix+$(l'[1].2, l'[1:].3, t_1, t_2)$
            end if
        end if
    end if
    return NULL
end if
$l'[0] = lemmatize(l'[0])$
if $|l'| = 1$ **then**
    return prefix+$(l'[0].1, l'[0].2, t_1, t_2)$
end if
if $l'[-1]$ is a preposition **then**
    return prefix+$(l'[0].1, l'.2, t_1, t_2)$
end if
return NULL

D An Example of Generating Predicates from Seed Predicates

We show an example process of generating new predicates by the generator $\mathcal{G}$ of TP-EGG in Table 8. We set $P_{seed} = \{p_1, p_2, p_3\}$, $K_{beam} = K_{sent} = 8$, $K_p = 15$. The predicates repeating in current generation or appearing in previous stages, and sentences that cannot be resolved by S^{-1} are omitted. Predicates generated from at least two different s are in red, and predicates appeared in generation of previous steps are in blue. According to Algorithm 1, only seed predicates and colored predicates will

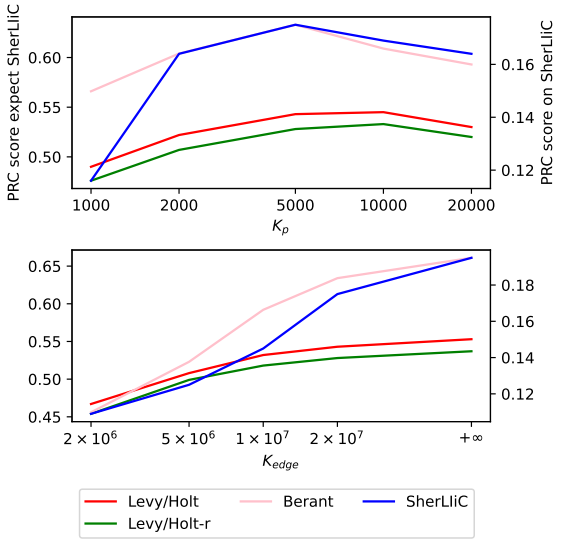


Figure 5: The performances on the evaluation datasets with different K_p and K_{edge} of TP-EGG$_{L/H-r}$. The y-axis of the curves on SherLIiC dataset is on the right side.

appear in final predicate set P_2'.

E Discussion about Graph Scales

As referred in Section 4, we set the number of predicates $K_p = 5 \times 10^3$ and edges $K_{edge} = 2 \times 10^7$, which determine the final scale of generated EGs. We report the performance of TP-EGG$_{L/H-r}$ on the evaluation datasets with different K_p and K_{edge} in Figure 5. Changing K_p from 1×10^3 to 2×10^4, the overall performance is the best while $K_p = 5 \times 10^3$. We assume that lower K_p might limit the coverage of predicate set, while higher K_p makes the EGs more sparse and miss potential entailment relations. Noted that the computational overhead and space occupation is almost proportional to K_{edge}, setting $K_{edge} = +\infty$ to regard ALL pairs as candidates is impractical (the largest EG in TP-EGG$_{L/H-r}$ will contain 7×10^7 edges). We find that $K_{edge} = 2 \times 10^7$ is able to reach the overall performances comparable with $K_{edge} = +\infty$ under our settings, while further decreasing K_{edge} will significantly cut down the performances. To balance between the overall performance and computational overhead, we finally set $K_p = 5 \times 10^3$ and $K_{edge} = 2 \times 10^7$.

F Details about Datasets

Levy and Dagan (2016) uses questions and candidate answers with textual predicates to collect the entailment relations, and proposes a widely used EG evaluation dataset which is later re-annotated by Holt (2018), called Levy/Holt Dataset. For example, if the annotator figures out that *"The government is adored by natives"* can be used to answer *"Who recognize the government?"*, the dataset will indicate that *"adore"* entails *"recognize"* between type *person* and *government*. Levy/Holt Dataset contains 18,407 predicate pairs (14,491 negative and 3,916 positive). We use the 30%/70% splitting of validation/test set as Hosseini et al. (2018) in our experiments.

However, because the QA annotation form incorporates additional information about entities related to the predicates, some consistent predicates pairs are annotated with different labels, and the transitivity is disobeyed between some predicate pairs. The inconsistent pairs are those (a, b) which $(a, b, True)$ and $(a, b, False)$ both appear in the dataset. The transitivity-disobeying pairs are those (a, b), (b, c) and (a, c) which $(a, b, True)$, $(b, c, True)$ and $(a, c, False)$ all appear. We find that there are 89 inconsistent pairs and 159 transitivity-disobeying pairs in Levy/Holt Dataset, and re-annotate these 248 pairs by five annotators with Fleiss' $\kappa = 0.43$. After re-annotating, we get the final Levy/Holt-r Dataset with 14,490 negative and 3,777 positive pairs.

Berant et al. (2011) proposes an annotated entailment relation dataset, containing 3,427 positive and 35,585 negative examples, called Berant Dataset.

Schmitt and Schütze (2019) extracts verbal relations from ClueWeb09 (Gabrilovich et al., 2013) based on Freebase (Bollacker et al., 2008) entities, and splits the extracted relations into typed one based on their most frequent Freebase types, which is naturally compatible to typed EG settings. We use their manually-labeled 1,325 positive and 2,660 negative examples in our EG benchmark, called SherLIiC Dataset. The dataset is split into 25%(validation) and 75%(test) in our experiments.

Stage	Predicates and Sentences
$P_{seed}(S_0)$	p_1 :(adore.1, adore.2, person,government) (Person A adores Government B), p_2 :(recognize.1, recognize.2, person,government) (Person A recognizes Government B), p_3 :(know.1, know.2, person,government) (Person A knows Government B)
$S^g(P^g)$	p_1 →{Person A is identified with Government B. (identify.2, identify.with.2, person,government), Person A is Government B. (be.1, be.2, person,government), Government B is magnet for Person A. (magnet.1, magnet.for.2, government,person), Government B is worshipped in Person A. (worship.2, worship.in.2, government,person), Government B is drawn to Person A. (draw.2, draw.to.2, government,person), Person A is devoted to Government B. (devote.2, devote.to.2, person,government), Person A is associated with Government B. (associate.2, associate.with.2, person,government), Government B is magnet of Person A. (magnet.1, magnet.of.2, government,person)} p_2 →{Government B is family of Person A. (family.1, family.of.2, government,person), Government B is associated with Person A. (associate.2, associate.with.2, government,person), Person A identifies with Government B. (identify.1, identify.with.2, person,government), Government B is drawn to Person A. (draw.2, draw.to.2, government,person), Person A is associated with Government B. (associate.2, associate.with.2, person,government), Person A identifies with Government B. (identify.1, identify.with.2, person,government), Person A is connected with Government B. (connect.2, connect.with.2, person,government), Government B wants Person A. (want.1, want.2, government,person)} p_3 →{Government B is associated with Person A. (associate.2, associate.with.2, government,person), Person A identifies with Government B. (identify.1, identify.with.2, person,government), Government B awards Person A. (award.1, award.2, government,person), Government B is drawn to Person A. (draw.2, draw.to.2, government,person), Person A embodies Government B. (embody.1, embody.2, person,government), Person A is associated with Government B. (associate.2, associate.with.2, person,government), Person A is connected with Government B. (connect.2, connect.with.2, person,government), Government B is enemy of Person B. (enemy.1, enemy.of.2, government,person)}
P_1	p_4 :(associate.2, associate.with.2, person,government) p_5 :(identify.1, identify.with.2, person,government) p_6 :(connect.2, connect.with.2, person,government) p_7 :(draw.2, draw.to.2, government,person) p_8 :(associate.2, associate.with.2, government,person)
$S^g(P^g)$	p_4 →{Person A is identified with Government B. (identify.2, identify.with.2, person,government), Government B awards Person A. (award.1, award.2, government,person), Person A practices Government B. (practice.1, practice.2, person,government),

	Government B is gravitate towards Person B. (be.1, be.gravitate.towards.2, government,person), Government B is sought after by Person A. (seek.2, seek.after.by.2, government,person),} p_5 →{Government B issues call for Person A. (issue.1, issue.call.for.2, government,person), Person A declares Government B. (declare.1, declare.2, person,government), Person A embodies Government B. (embody.1, embody.2, person,government), Person A declares war on Government B. (declare.1, declare.war.on.2, person,government)} p_6 → {Person A is identified with Government B. (identify.2, identify.with.2, person,government), Government B is after Person A. (be.1, be.after.2, government,person), Government B issues call for Person A. (issue.1, issue.call.for.2, government,person), Government B is identified with Person A. (identify.2, identify.with.2, government,person), Person A practices Government B. (practice.1, practice.2, person,government), Person A embodies Government B. (embody.1, embody.2, person,government)} p_7 →{Person A submits Government B. (submit.1, submit.2, person,government), Government B is attracted to Person A. (attract.2, attract.to.2, government,person), Government B is magnet for Person A. (magnet.1, magnet.for.2, government,person), Person A believes in Government B> (believe.1, believe.in.2, person,government), Government B is magnet of Person A. (magnet.1, magnet.of.2, government,person)} p_8 →{Person A is identified with Government B. (identify.2, identify.with.2, person,government), Person A preaches Government B. (preach.1, preach.2, person,government), Government B issues call for Person A. (issue.1, issue.call.for.2, government,person), Person A practices Government B. (practice.1, practice.2, person,government), Person A demands Government B. (demand.1, demand.2, person,government), Government B is gravitate towards Person B. (be.1, be.gravitate.towards.2, government,person), Government B wants Person A. (want.1, want.2, government,person)}
P_2	p_9 :(identify.2, identify.with.2, person,government) p_{10} :(magnet.1, magnet.for.2, government,person) p_{11} :(issue.1, issue.call.for.2, government,person) p_{12} :(award.1, award.2, government,person) p_{13} :(practice.1, practice.2, person,government) p_{14} :(embody.1, embody.2, person,government) p_{15} :(be.1, be.gravitate.towards.2, government,person) p_{16} :(want.1, want.2, government,person) p_{17} :(magnet.1, magnet.of.2, government,person)
P_2'	Return $p_1, ..., p_{17}$

Table 8: An example of generating predicates P_i' from P_{seed}.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section "Limitations"

☑ A2. Did you discuss any potential risks of your work?
Section "Ethics Statement"

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3 and 4

☑ B1. Did you cite the creators of artifacts you used?
Section 3 and 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 3 and 4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 3 and 4

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Section 3, 4 and "Ethics Statement"

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4

C ☑ Did you run computational experiments?

Section 4 and 5

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and Appendix E

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3 and 4

D ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Section 4

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Section "Ethics Statement"

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Appendix F

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.