RSGT: Relational Structure Guided Temporal Relation Extraction

Jie Zhou* **Shenpo Dong*** **Hongkui Tu✉** **Xiaodong Wang** **Yong Dou**
College of Computer
National University of Defense Technology
{jiezhou,dsp,tuhongkui,xdwang,yongdou}@nudt.edu.cn

Abstract

Temporal relation extraction aims to extract temporal relations between event pairs, which is crucial for natural language understanding. Few efforts have been devoted to capturing the global features. In this paper, we propose **RSGT**: **R**elational **S**tructure **G**uided **T**emporal Relation Extraction to extract the relational structure features that can fit for both inter-sentence and intra-sentence relations. Specifically, we construct a syntactic-and-semantic-based graph to extract relational structures. Then we present a graph neural network based model to learn the representation of this graph. After that, an auxiliary temporal neighbor prediction task is used to fine-tune the encoder to get more comprehensive node representations. Finally, we apply a conflict detection and correction algorithm to adjust the wrongly predicted labels. Experiments on two well-known datasets, MATRES and TB-Dense, demonstrate the superiority of our method (**2.3%** F1 improvement on MATRES, **3.5%** F1 improvement on TB-Dense).

1 Introduction

Temporal relation extraction (TRE) is crucial for natural language understanding and can facilitate various downstream applications such as summarization (Zhou et al., 2010), question answering (Yu et al., 2017), and clinical diagnosis (Zhou et al., 2021). As shown in Figure 1, the goal of TRE is to determine the temporal order between an event pair (*BEFORE, AFTER*, etc.).

Most early methods were based on statistical machine learning (Mani et al., 2006; Chambers, 2013). In recent years, neural network based methods and large-scale pre-trained language models such as BERT (Devlin et al., 2018) have contributed to a substantial increase in the performance of TRE task (Ning et al., 2019; Wang et al., 2020).

S1: Former President Nicolas Sarkozy was **(e1,informed)** Thursday that he would face a formal investigation into whether he **(e2,abused)** the frailty of Liliane Bettencourt, to get funds for his 2007 presidential campaign.
S2: Mr. Sarkozy has **(e3,denied)** accepting illegal campaign funds from Ms. Bettencourt, either personally or through his party treasurer at the time, Eric Woerth, as **(e4,alleged)** by her former butler.

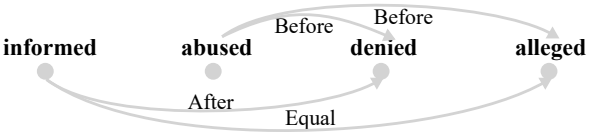

Figure 1: An example of temporal relation extraction. There are four events in these sentences. The graph below shows the pair-wise event temporal relationships.

However, these methods may ignore the global structure features which carry non-consecutive and long-distance semantics (Peng et al., 2018). This shortcoming is obvious in dealing with an event pair that the two events belong to different sentences (inter-sentences event pair), such as $< e_1, e_4 >$ in Figure 1. Few previous works differentiate inter-sentence event pairs from intra-sentence ones (where the two events appear in the same sentence). Thus, the performance may be impacted. For example, we test that there is a 5 accuracy points gap between intra-sentences and inter-sentence event pairs with the recent state-of-the-art method (Wen and Ji, 2021a).

To fill this gap, we aim to develop a structural features method that captures temporal semantic relations for both the inter-sentence and intra-sentence event pairs. Specifically, we adopt graph neural networks (GNNs), which have been proved to be effective in preserving global structure information of a graph in graph embeddings (Yao et al., 2019), to bridge the temporal relations.

Based on the above analysis, we present **RSGT**: **R**elational **S**tructure **G**uided **T**emporal Relation Extraction. To enable our model to learn more ef-

*These authors contributed equally to this work

Proceedings of the 29th International Conference on Computational Linguistics, pages 2001–2010
October 12–17, 2022.

fective representation for relational structures, we take the following strategies: First, to obtain more relational information, we create different types of connections for the graph nodes based on their syntactic and semantic information. Such connections are combined together to generate a rich relational graph. In particular, the node embeddings are obtained with the GGNN algorithm (Li et al., 2016). To avoid graph over-smoothing, RoBERTa (Liu et al., 2019) embeddings are concatenated with GGNN embeddings to make the final prediction.

Second, unlike most previous graph-based models which directly use the pre-trained language model as the node encoder, we present a task called temporal event neighbor prediction to fine-tune the encoder. This task aims to predict the neighbor node of event mentions from the relational graph. The fine-tuned encoder can help RSGT better understand the correlation between the relational structure and raw text. Ablation studies demonstrate that it can significantly boost efficiency.

Finally, we present a conflict detection and correction algorithm based on the transitivity rule of temporal relations to promote performance.

Experiments on two popular benchmarks, MATRES (Ning et al., 2018) and TB-Dense (Cassidy et al., 2014), show that RSGT outperforms the state-of-the-art methods (**2.3%** F1 points improvement on MATRES, **3.5%** F1 points improvement on TB-Dense). Meanwhile, we improve the accuracy of inter-sentence relations to the same level as intra-sentence relations.

2 Method

We formulate the TRE problem as a multi-class classification task. For a document D with n sentences $(S_1, S_2, ..., S_n)$, it can have multiple event mentions $E = (e_1, e_2, ..., e_m)$. The goal of TRE is to predict the temporal relation type between event pairs. For an event pair $< e_i, e_j >$, the input of our model is the sentence they belong to. In particular, if two events belong to different sentences (we call it inter-sentence event pair), two sentences $< S_i, S_j >$ are concatenated together as the input.

Our work RSGT involves five major parts: (i) Structure Generation to generate a relational-guided graph based on syntactic-and-semantic information, (ii) Temporal Event Neighbor Prediction to transform words into embedding vectors, (iii) Relational-guided Graph Model to predict temporal relations, (iv) Conflict Detect and Correct algorithm to revise temporal errors.

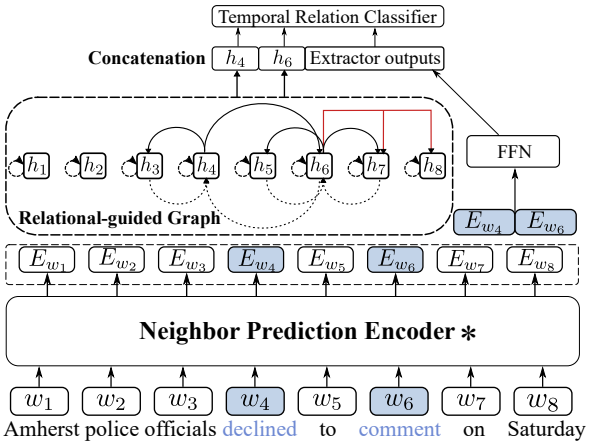

Figure 2: The illustrative architecture of the proposed Relational-guided Graph Model. Our goal is to extract the temporal relation of $< w_4, w_6 >$. In the relational-guided graph, black arcs mean syntactic-guided edges $\mathcal{E}_d$ and red arcs mean semantic-guided edges $\mathcal{E}_t$. * indicates a RoBERTa model fine-tuned on the Temporal Event Neighbor Prediction task.

2.1 Structure Generation

Building graphs is a feature selection process that can facilitate representation learning for the TRE problem. Given an input sentence S, our goal is to generate a relational graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ as the input of our graph neural network. Our relational graph is based on syntactic and semantic information extracted from S. The node set $\mathcal{N}$ and edge set $\mathcal{E}$ in $\mathcal{G}$ are constructed as follows strategies.

2.1.1 Nodes

The node set $\mathcal{N}$ should capture all objects related to temporal events. We take two types of nodes to make up the node set. The first type is from the original words $w_i \in S$. The second type is the event arguments extracted by the Semantic Role Labeling (SRL) model, which we will introduce in the semantic-guided edges section. Formally, let $W = \{w_1, w_2, ..., w_{|W|}\}$, $Arg = \{a_1, a_2, ..., a_{|Arg|}\}$ be the set of words and event arguments, respectively. Then the node set of the input sentence should consist of two parts: $\mathcal{N} = \{W \cup Arg\}$. After the generation of $\mathcal{G}$, nodes with no edges pointing to other nodes are removed from $\mathcal{N}$.

2.1.2 Syntactic-guided Edges

Dependency Parsing (DP) can examine the dependencies between the phrases of a sentence to determine its syntactic structure. As such, we apply the dependency parsing tree of the input sentence

to build syntactic-guided edges $\mathcal{E}_d$. For the dependency tree consisting of multiple head-dependent arcs, the arcs whose head is event mention are converted to edges $\mathcal{E}_{d\_along}$ as the solid black arcs in Figure 2. In addition, we assume that information flows not only along the syntactic dependency arcs, so we create edges $\mathcal{E}_{d\_rev}$ in the opposite direction as well (i.e., from dependents to heads). Following Kipf and Welling (2016), we also add self-edges for each nodes as $\mathcal{E}_{d\_loop}$. Therefore, syntactic-guided edges $\mathcal{E}_d$ contains three kinds of edges $\mathcal{E}_{d\_along}$, $\mathcal{E}_{d\_rev}$ and $\mathcal{E}_{d\_loop}$.

2.1.3 Semantic-guided Edges

We design semantic-guided edges $\mathcal{E}_t$ to fetch semantic information related to a temporal event. Specifically, we want to import an event extraction model that can extract event arguments based on event mentions. SRL-BERT (Shi and Lin, 2019) becomes our final choice because it not only meets our above requirements but also marks out the argument's types. As shown in the Figure 2's red arcs, arguments are connected to the event mentions as $\mathcal{E}_t$. SRL task assumes event mentions trigger the arguments, so we only consider unidirectional edges from event nodes. Some particular argument types, such as Temporal and Discourse, which can probably provide extra information to understand the temporal relation, are assigned to different edge types with higher weight.

2.2 Temporal Event Neighbor Prediction

In the graph model, we need to apply an encoder to transform each word $w_i \in S$ into a contextual represented vector for nodes. Most previous studies directly use pre-trained language models as the encoder. However, Chien et al. (2021) argues that these pre-trained language models ignore the correlations between graph topology and raw text features. Inspired by this work, we propose a task called Temporal Event Neighbor Prediction. Given a syntactic-guided graph $\mathcal{G}_d$ we construct, this task aims to distinguish whether a node is the neighbor of the event mention's node or not. We pick k words before and after per event mention respectively in the sentence, and they can form node pairs with its event mention.

Take the sentence in 2 as an example. Suppose we are using $k = 2$, so for the first event mentions w_4, we pick 4 words before and after w_4, which are $\{w_2, w_3, w_5, w_6\}$. Node pairs $< w_4, w_3 >, < w_4, w_6 >$ are neighbors, so their labels are 1. $< w_4, w_2 >, < w_4, w_5 >$ are not neighbors and their labels are 0. The second event mentions w_6 can be treated in the same way. The input of this task is each event-neighbor pair $< w_e, w_{nbr} >$ and its raw sentence.

To handle this task, we first apply RoBERTa to encode the sentences and extract the nodes' embeddings of $< w_e, w_{nbr} >$. The represented vector of two nodes then passes through a Feed-Forward Network (FFN) layer with a $tanh$ activation function, respectively. For the output of FFN layer h_e and h_{nbr}, we concatenate them together and apply Batch Normalization as the representation of node pair. Then a FFN layer with softmax is added for prediction. The model can be formalized as:

$$\begin{aligned} h_e &= \tanh(\mathbf{FFN}_1(\phi(w_e))) \\ h_{nbr} &= \tanh(\mathbf{FFN}_2(\phi(w_{nbr}))) \\ \hat{y_{nbr}} &= \mathrm{softmax}(\mathbf{FFN}_3(\mathbf{BN}[h_{nbr}; h_e])) \end{aligned} \tag{1}$$

where BN denotes Batch Normalization, and ϕ is the encoder that maps w to feature vectors. We adjust k to ensure that the distribution of labels is balanced. To make sure RoBERTa can maintain more topology information from the relational graph, the learning rate of RoBERTa is larger than other layers.

This task allows the encoder to understand not only the contextual information from the raw text but also the topology information from our relational graph $\mathcal{G}$. We select the model with the best accuracy in the validation set as the encoder. Then we apply this fine-tuned encoder to represent the node set $\mathcal{N}$. This task can be further extended to other graph-related models as an efficient way for the encoder's fine-tuning.

2.3 Relational-guided Graph Model

We have already generated a relational graph $\mathcal{G}$ and the represented vector x for each node. We apply Gated Graph Sequence Neural Networks (GGNN) to handle our relational graph. GGNN employs a gated recurrent unit (GRU) as a recurrent function, reducing the recurrence to a fixed number of steps. The advantage is that it no longer needs to constrain parameters to ensure convergence. We parse each sentence into the relational graph and use GGNN to digest this structural information. The forward process of GGNN is:

$$
\begin{aligned}
x_u &= \phi(w_u) \\
h_u^0 &= [x_u \| \mathbf{0}] \\
a_u^t &= \sum_{v \in \mathcal{N}(u)} W_{e_{uv}} h_v^t \\
h_u^{t+1} &= \mathrm{GRU}(a_u^t, h_u^t)
\end{aligned} \tag{2}
$$

where u denotes the current node and v denotes the neighbor node of u. ϕ is the fine-tuned encoder, and h_u^t denotes the t step hidden states of u.

As discussed in Chen et al. (2020), over-smoothing is a common issue faced by GNNs, which means that the representations of the graph nodes of different classes would become indistinguishable when stacking multiple layers. To avoid over-smoothing problems, the embeddings $< x_i, x_j >$ from fine-tuned RoBERTa are passed through a fully connected layer parallel with GGNN. For event pair $< x_i, x_j >$, the representation H_F from the fully connected layer is then concatenated with GGNN's final hidden states H_G. Concatenation may help us maintain some contextual information from RoBERTa encoder and increase the differentiation of event representations. In the end, we apply a two-layer FFN as classifier f and a BatchNorm layer for the final temporal prediction. The final output of event pair $< e_i, e_j >$ is:

$$
\hat{y}_{ij} = f(\mathbf{BN}[H_{G_i}; H_{G_j}; H_{F_i}; H_{F_j}]) \tag{3}
$$

The overall loss function to train our model is:

$$
\mathcal{L} = -\sum_{i,j} y_{ij}^* log(\hat{y}_{ij}) + \gamma \mathcal{L}_{reg} \tag{4}
$$

where y_{ij}^* is the gold labels of temporal relations and γ is a trade-off parameter for regularization techniques.

2.4 Conflict Detect and Correct

There exists a transitivity rule in temporal relationships. Take the events depicted in Figure 1 as an example. We consider the intra-sentence and inter-sentence event pairs relationships together and build the temporal diagram on the left side of the Figure 3. A transitivity rule could be explained as "e_2 happens before e_1, e_1 and e_4 occur simultaneously, then e_4 should happen after e_2". On the right side of Figure 3 is a counterexample. The red arrows can form a cycle, which indicates that at least one temporal relation edge violates the transitivity rule.

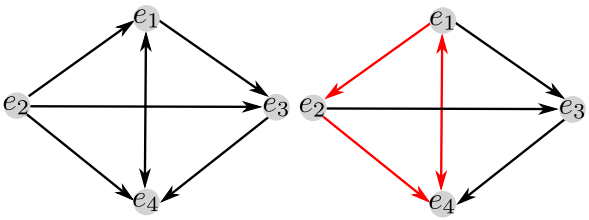

Figure 3: The example of transitivity rule in temporal relationship. Unidirectional arrows represent *BEFORE*, like $e_2 \rightarrow e_1$ refers to e_2 happens before e_1. Bidirectional arrows represent two events occurring simultaneously.

To take full advantage of this rule, we design an algorithm to find potential conflicts. From the output of the classifier f, we obtain a temporal relationship prediction $\hat{y}_{ij}$ for the event pair $< e_i, e_j >$. We can build a document-level temporal relational graph by collecting temporal relation predictions as edges and events as nodes from document D. For *BEFORE* relation of $< e_i, e_j >$, we add an edge from e_i to e_j. On the contrary, we add an edge from e_j to e_i for *AFTER*. We treat *EQUAL* as a bidirectional edge. Other temporal relations are ignored (e.g. *VAGUE*). Obviously, this graph should be a Directed Acyclic Graph (DAG). So our goal is to find the conflict cycles and correct them.

We re-implement the Johnson cycle algorithm (Johnson, 1975) as our temporal event conflict detection algorithm. It was presented to find all the elementary cycles of a directed graph, which time bounded by $O((n+e)(c+1))$ for n nodes, e edges and c elementary cycles.

Then we use algorithm 1 to detect and correct conflict. Basically, we:

1. Apply $conflict\_detect$ algorithm to find elementary cycles in the $edges$.
2. Pick the longest cycle from step 1 and initialize variables $cycle\_n$ as cycle's length, m_logit, m_edge as the smallest logit and its edge (lines 5-7).
3. Traverse all nodes in the $cycle$ and find the smallest logit (lowest probability of confidence $edge\_logit$). Store the start and end node of m_edge(lines 8-20).
4. Reverse the edge found in step 3 to solve the conflicts. Remove m_edge from the graph if it has been corrected twice. Go back to step 1 and repeat until the graph is a directed acyclic graph (lines 21-26).

Algorithm 1: Correct Algorithm

Input : $edges$
Output : Corrected edges

$revised$ = [] ;
while *True* **do**
  $cycles$ = conflict_detect($edges$);
  if *no cycles* **then** break;
  $cycle \leftarrow$ longest($cycles$);
  $cycle\_n \leftarrow$ length($cycle$);
  $m\_logit, m\_edge \leftarrow -1, (-1, -1)$;
  for *i in range*$(1, cycle\_n)$ **do**
    if $i! = cycle\_n$ **then**
      j = i+1 ;
    else
      j = 1;
    end
    $fr \leftarrow cycle[i]$;
    $to \leftarrow cycle[j]$;
    $edge\_logit = edges[now][to]$;
    if $m\_logit \leq edge\_logit$ **then**
      $m\_logit = edge\_logit$;
      $m\_edge = (fr, to)$
  end
  $fr, to \leftarrow m\_edge$;
  if fr, to *in* $revised$ **then**
    remove $edge_{fr,to}$
  $revised$. add(m_edge) ;
  reverse $edge_{fr,to}$ to $edge_{to,fr}$;
  $cycles \leftarrow$ collision detection (adj_{d_i})
end

This algorithm is concise and efficient, and it can be well adapted to the correction work of various datasets without training.

3 Experiments

3.1 Datasets

We conduct our experiments on two well-known benchmarks for the TRE task, MATRES(Ning et al., 2018) and TB-Dense(Cassidy et al., 2014). MATRES contains refined annotations on TimeBank(Pustejovsky et al., 2003) , AQUAINT and Platinum documents. It contains four types of temporal labels: *BEFORE, AFTER, EQUAL, VAGUE*. TB-Dense is a densely annotated dataset from TimeBank and TempEval(UzZaman et al., 2013). This dataset contains six label types. In addition to the four label types from MATRES, it has two more label types: *INCLUDES* and *IS_INCLUDED*. For compatible comparison, we apply the same data splits as in prior work for the considered datasets. The detailed statistics can be found in Table 1.

3.2 Evaluation Metrics

We adopt micro averaged precision, recall, and F1 scores as evaluation metrics following the previous works(Ning et al., 2018; Wen and Ji, 2021a; Cao et al., 2021). For the MATRES, *VAGUE* is considered to be non-temporal information and is excluded from the F1 calculation. For the TB-Dense, *VAGUE* is taken into consideration (i.e., all label types are seen as positive classes) so the metric should share the same precision, recall, and F1 value. We follow these different settings for our experiments to ensure fair comparisons.

Dataset	Train	Validation	Test	Labels
MATRES	10888	1852	837	a,b,e,v
TB-Dense	4032	629	1427	a,b,s,v,i,ii

Table 1: Data splits and relation pairs statistics. a: *AFTER*, b: *BEFORE*, e: *EQUAL*, s: *SIMULTANEOUS*, v: *VAGUE*, i: *INCLUDES*, ii: *IS_INCLUDED*.

3.3 Implement Details

The hyperparameters used in the experiment are listed. **Neighbor Prediction:** RoBERTa-large is adopted to encode the sentence. The learning rate for RoBERTa and FFN are 1e-5, 1e-4, respectively. **Syntactic Information:** We apply SpaCy [*] toolkit to build dependency trees based on input sentences. **Semantic Information:** The event arguments corresponding to each event mention are extracted from SRL-BERT. **Graphs Training:** AdamW with learning rate of 5e-6, β_1 = 0.9, β_2 = 0.999 and weight decay of 0.01 is used for optimization. We set the training epochs and batch size to 40 and 32, respectively. Besides, we exploit a dropout with a rate of 0.5 on the concatenated feature representations.

3.4 Baselines

We conduct experiments to compare our approach RSGT with the state-of-the-art models for TRE in each benchmark dataset as follows. Note that MATRES is a relatively new dataset, so we can hardly find more baselines that perform well on both MATRES and TB-Dense.

[*] https://spacy.io/

Dataset	Models	P	R	F1
MATRES	Siamese	66.6	60.8	63.0
	Constrained	72.1	80.8	76.2
	UAST	76.6	84.9	80.5
	SMTL	-	-	81.6
	Stack-Propagation	78.4	85.2	81.7
	RSGT	**82.2**	**85.8**	**84.0**
TB-Dense	Timelines	56.6	56.6	56.6
	UAST	64.3	64.3	64.3
	CTRL-PG	65.2	65.2	65.2
	RSGT	**68.7**	**68.7**	**68.7**

Table 2: Model performance on MATRES and TB-Dense. The performance improvement of RSGT over the baselines is significant with $p < 0.01$

MATRES For this dataset, the following baselines are chosen for comparison. (i) **Siamese** (Ning et al., 2019): A Siamese encoder of a temporal commonsense knowledge base, and global inference via integer linear programming (ILP). (ii) **Constrained** (Wang et al., 2020): A framework bridges temporal and subevent relation extraction tasks with a comprehensive set of logical constraints. (iii) **SMTL** (Ballesteros et al., 2020): A model relies on multi-task learning and self-training techniques. (iv) **Stack-Propagation** (Wen and Ji, 2021a): A Stack-Propagation framework to further incorporate predicted timestamp explicit for relation classification.

TB-Dense We use the following baselines for comparison. (i) **Timelines** (Vashishtha et al., 2019) A semantic framework for modeling fine-grained temporal relations and event duration that maps pairs of events to real-valued scales and constructs document-level event timelines. (ii) **UAST** (Cao et al., 2021) An uncertainty-aware self-training framework to quantify the model uncertainty. (iii) **CTRL-PG** (Zhou et al., 2021) A method with probabilistic soft logic Regularization and global inference at the document-level.

3.5 Overall Performance

The most important observation from Table 2 is that model **RSGT** has significantly outperformed all the baseline systems on both MATRES and TB-Dense. Thus evidently indicating the effectiveness of the proposed RSGT model for the TRE task. Compared with the previous SOTA method Stack-Propagation, which also uses RoBERTa, our RSGT has 2.3 % F1 improvement on the MATRES dataset.

	Intra-sentences			Inter-sentences		
	P	R	F1	P	R	F1
RoBERTa-F	81.6	81.9	81.7	77.0	78.7	78.0
Stack-Propagation	77.9	84.5	81.1	73.6	85.7	79.2
RSGT	**83.1**	**84.5**	**83.8**	**81.8**	**86.4**	**84.1**

Table 3: Performance of different models on **MATRES**.

For the more complex dataset TB-sense with six temporal relation types, RSGT also has a 2.6% F1 improvement over the previous SOTA method CTRL-PG. Overall, our method RSGT establishes a new state-of-the-art on two popular datasets of the TRE task.

3.6 Intra- and Inter-sentence

Inter-sentence event pairs make up a considerable proportion of the MATRES dataset (69.53% in the train set and 69.77% in the test set). Consequently, the performance on inter-sentence event pairs can significantly influence the overall performance. To explicitly demonstrate the effect of RSGT on the extraction of intra- and inter-sentence event pairs, we conduct a contrast experiment on the MATRES dataset. We attach a learnable fully-connected layer after RoBERTa as the baseline **RoBERTa-F**. The performance on the intra- and inter-sentences is shown in Figure 3. The previous SOTA method, Stack-Propagation, has a clear 4.3% gap in precision value between intra- and inter-sentences. As a comparison, we can observe an absolute F1 gain from RSGT, 2.7% and 4.9% on the intra-sentences and inter-sentences, respectively. Importantly, we successfully fill the performance gap between intra- and inter-sentence event pairs and improve their F1 result to the same level. These experiments show that the introduction of relational structure is of great help for inter-sentence temporal relations extraction.

3.7 Ablation Study

To illustrate the impact of each component in RSGT, we further conduct ablation studies with different configurations. Note that MATRES is a relatively new dataset, so we can hardly find more baselines that perform well both on MATRES and TB-Dense.

3.7.1 Effect of Neighbor Prediction

We propose the Neighbor Prediction task so that the encoder can learn the correlation between the relational graph's topology and raw text. In the

Model	P	R	F1
RSGT -w/o neighbor prediction	79.7	82.7	81.2
RSGT -w event prediction	69.7	79.2	74.1
RoBERTa	78.4	80.0	79.1
RSGT -w/o $\mathcal{E}_d$	80.5	84.7	82.5
RSGT -w/o $\mathcal{E}_t$	81.7	85.5	83.6
RSGT independent	81.0	84.8	82.8
RSGT	**82.2**	**85.8**	**84.0**

Table 4: Performance of different models on **MATRES**

MATRES dataset, the Neighbor Prediction task can reach 88.6% accuracy. In Table 4, **RSGT -w/o neighbor prediction** is RSGT excluding the Neighbor Prediction task, that is, using the original pre-trained RoBERTa model as encoder. As for **RSGT -w event prediction**, we replace the Neighbor Prediction task with a simple event extraction task to fine-tune the node encoder. Event extraction aims to extract event mentions from an input sentence. The result shows that: (1) The topological knowledge about relational graphs learned by neighbor prediction task can greatly improve the subsequent models. (2) Other types of knowledge, such as knowledge implied by the event extraction model, may not positively affect the TRE task.

3.7.2 Effect of Relational Structure Features

We examine the following ablated models to evaluate the effectiveness of different relational structure features in RSGT on the TRE task. (i) **RoBERTa** is a baseline with RoBERTa model and a fully-connected layer. (ii) **RSGT -w/o** $\mathcal{E}_d$ excludes the syntactic-guided edges. (iii) **RSGT -w/o** $\mathcal{E}_t$ excludes the semantic-guided edges. (iv) **RSGT independent** apply syntactic and semantics information to construct two independent graphs, respectively. At last, we average the embeddings of the two graphs.

The bottom half of Table 4 shows the performance of the above ablated models. We can observe that all the components can contribute to the proposed model RSGT as eliminating any of them degrades the performance in the F1 score. Apparently, the worse performance of **RSGT -** $\mathcal{E}_d$ model illustrates that syntactic information contributes a major improvement on TRE. And the **RSGT -** $\mathcal{E}_t$ model that removes semantic information slightly loses the performance of 0.4% F1. This is because the syntactic information contains more knowledge about the current event pair, and syntactic information may contain semantic information (event arguments) in some cases. Compared with **RSGT independent**, the independent graphs lack the interaction of all relational structure information. Instead, syntactic and semantic guided information should work together to form an interactive graph to enrich the relational structure obtained from RSGT.

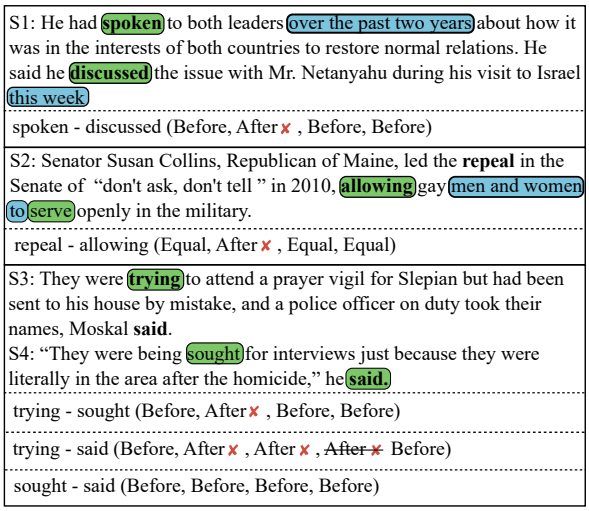

Figure 4: Case study. Event mentions and important relational structure are highlighted by green and blue respectively. Each line after sentence S has a structure like $< e_1, e_2 >$ (G,P_1,P_2,P_3), where e_1, e_2 is an event pair and G is the gold temporal label. P_1, P_2 and P_3 denotes prediction from Stack-Propagation (Wen and Ji, 2021b), RSGT-w/o conflict algorithm and RSGT respectively. Incorrect predictions are denoted by a red mark. Strikethrough means the prediction is corrected by our conflict detect and correct algorithm.

3.7.3 Effect of Conflict Detect and Correct

This algorithm is training-free and the time complexity is $O(n)$. Limited by the test set size, the improvement is slight (about 0.1%) on both MATRES and TB-dense datasets. Notes that the performance improvement from conflict detection gradually decreases with the training process. For example, it can bring a 4.3% average improvement in the first epoch, which means conflict detection can bring huge performance improvements in the early stage. We believe that it will play a more critical role in larger-scale datasets or real-world cases.

3.8 Case Study and Error Analysis

To promote a better understanding of our RSGT and guide potential research direction, we analyze three concrete examples in Figure 4. Each case has a pair of events, and the study results can be categorized into different types that are described

below:

Case 1. Sentence S1 contains a conversation event mentions "spoken" and a discussion event mentions "discussed". RSGT correctly predicts the temporal relations while Stack-Propagation fails. RSGT successfully extracts two temporal arguments from S1, enhancing the model's inference ability by providing the time of occurrence. Obviously, "over the past two years" has happened "this week". The previous model does not utilize semantic information, which leads to misclassification.

Case 2. The small proportion of *EQUAL* (about 3.6% in MATRES) makes temporal relationship prediction more challenging, as it can easily be confused with more common labels like *BEFORE* and *AFTER*. Sentence S3 contains two events, "allowing" and "serve". It seems like a simple task for a human. However, Stack-Propagation relies only on two event words and fails to recognize their interaction. We highlight some syntactic information extracted by RSGT. "allow someone to do something" is a typical relational structure that happens simultaneously. As a result, this relational structure makes the prediction much easier for RSGT.

Case 3. S3 and S4 show one intra-sentence and two inter-sentence event temporal relations. Our RSGT correctly classifies <trying, sought>, <sought, said> event pairs. For an inter-sentence event pair like <trying, said>, which is so hard that RSGT fails to predict its temporal relation initially, the conflict detect and correct algorithm can utilize the relationships between the other two event pairs to correct the result. In the directed graph built from the predictions, we obtain three edges (trying $\rightarrow$ sought), (trying $\leftarrow$ said), (sought $\rightarrow$ said). Obviously, this graph does not meet the DAG definition, and our algorithm reverses the edge with a minimum confidence score (trying $\leftarrow$ said) to correct it.

4 Related Work

Earlier efforts on TRE (temporal relation extraction) use statistical machine learning techniques (Support Vector Machine, Max entropy) and handcraft features (e.g Verhagen and Pustejovsky (2008) and Chambers (2013)). Recently, neural methods and large-scale pre-training language models have also achieved promising improvement (Nguyen and Grishman, 2015; Nguyen et al., 2016; Wang et al., 2020; Mathur et al., 2021). The early feature-based methods for TRE have explored different features and resources to improve the performance, including syntactic patterns and lexical features (Cheng and Miyao, 2017; Mirza and Tonelli, 2016). Unlike previous works, our approach RSGT takes account of relational structure features to induce more accurate representations.

A wave of research at the intersection of deep learning on graphs has influenced a variety of NLP tasks, including event extraction (Xu et al., 2021; Yan et al., 2019), relation extraction (Tran Phu and Nguyen, 2021; Su et al., 2022) and event argument extraction (Pouran Ben Veyseh et al., 2020). These graph-structured data can encode complicated relations between event pairs to infer temporal order. Our model is different from such related works in that we designed a relational structure guided graphs that are tailored to our TRE task. In addition, we introduce a novel Temporal Event Neighbor Prediction task for the fine-tuning of the node encoder.

5 Conclusion

This paper proposes RSGT to capture relational structure information for the temporal relation extraction task. The experimental results well demonstrate our model's effectiveness and superiority in both the overall datasets and the inter-sentence event pairs. Ablation experiments show that the relational graph model and *Temporal Event Neighbor Prediction* contribute greatly to RSGT's performance.

Our future work will focus on how to apply *Temporal Event Neighbor Prediction*, and *Conflict Detect and Correct Algorithm* to other tasks with rich relations such as Casual Relations (Caselli and Vossen, 2017). We believe these methods are promising in processing relational structure information from other relational extraction tasks.

References

Miguel Ballesteros, Rishita Anubhai, Shuai Wang, Nima Pourdamghani, Yogarshi Vyas, Jie Ma, Parminder Bhatia, Kathleen McKeown, and Yaser Al-Onaizan. 2020. Severing the edge between before and after: Neural architectures for temporal ordering of events. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5412–5417, Online. Association for Computational Linguistics.

Pengfei Cao, Xinyu Zuo, Yubo Chen, Kang Liu, Jun Zhao, and Wei Bi. 2021. Uncertainty-aware self-

training for semi-supervised event temporal relation extraction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2900–2904.

Tommaso Caselli and Piek Vossen. 2017. The event storyline corpus: A new benchmark for causal and temporal relation extraction. In *Proceedings of the Events and Stories in the News Workshop*, pages 77–86.

Taylor Cassidy, Bill McDowell, Nathanel Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. Technical report, Carnegie-Mellon Univ Pittsburgh PA.

Nathanael Chambers. 2013. NavyTime: Event and time ordering from raw text. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 73–77, Atlanta, Georgia, USA. Association for Computational Linguistics.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Fei Cheng and Yusuke Miyao. 2017. Classifying temporal relations by bidirectional LSTM over dependency paths. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1–6, Vancouver, Canada. Association for Computational Linguistics.

Eli Chien, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, Jiong Zhang, Olgica Milenkovic, and Inderjit S Dhillon. 2021. Node feature extraction by self-supervised multi-scale neighborhood prediction. *arXiv preprint arXiv:2111.00064*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Donald B Johnson. 1975. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. 2016. Gated Graph Sequence Neural Networks. In *Proceedings of ICLR'16*. Edition: Proceedings of ICLR'16.

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*.

Inderjeet Mani, Marc Verhagen, Ben Wellner, Chungmin Lee, and James Pustejovsky. 2006. Machine learning of temporal relations. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 753–760.

Puneet Mathur, Rajiv Jain, Franck Dernoncourt, Vlad Morariu, Quan Hung Tran, and Dinesh Manocha. 2021. TIMERS: Document-level temporal relation extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 524–533, Online. Association for Computational Linguistics.

Paramita Mirza and Sara Tonelli. 2016. CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 64–75, Osaka, Japan. The COLING 2016 Organizing Committee.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 300–309, San Diego, California. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with 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 (Volume 2: Short Papers)*, pages 365–371, Beijing, China. Association for Computational Linguistics.

Qiang Ning, Sanjay Subramanian, and Dan Roth. 2019. An improved neural baseline for temporal relation extraction. *arXiv preprint arXiv:1909.00429*.

Qiang Ning, Hao Wu, and Dan Roth. 2018. A multi-axis annotation scheme for event temporal relations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1318–1328, Melbourne, Australia. Association for Computational Linguistics.

Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pages 1063–1072.

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Graph transformer networks with syntactic and semantic structures for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3651–3661, Online. Association for Computational Linguistics.

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

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

B. Y. Su, S. L. Hsu, K. Y. Lai, and A. Gupta. 2022. Temporal relation extraction with a graph-based deep biaffine attention model. *arXiv e-prints*.

Minh Tran Phu and Thien Huu Nguyen. 2021. Graph convolutional networks for event causality identification with rich document-level structures. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3480–3490, Online. Association for Computational Linguistics.

Naushad UzZaman, Hector Llorens, Leon Derczynski, James Allen, Marc Verhagen, and James Pustejovsky. 2013. Semeval-2013 task 1: Tempeval-3: Evaluating time expressions, events, and temporal relations. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9.

Siddharth Vashishtha, Benjamin Van Durme, and Aaron Steven White. 2019. Fine-grained temporal relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2906–2919, Florence, Italy. Association for Computational Linguistics.

Marc Verhagen and James Pustejovsky. 2008. Temporal processing with the TARSQI toolkit. In *Coling 2008: Companion volume: Demonstrations*, pages 189–192, Manchester, UK. Coling 2008 Organizing Committee.

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

Haoyang Wen and Heng Ji. 2021a. Utilizing relative event time to enhance event-event temporal relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10431–10437.

Haoyang Wen and Heng Ji. 2021b. Utilizing relative event time to enhance event-event temporal relation extraction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10431–10437, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mingzhou Xu, Liangyou Li, Derek F. Wong, Qun Liu, and Lidia S. Chao. 2021. Document graph for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8435–8448, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Haoran Yan, Xiaolong Jin, Xiangbin Meng, Jiafeng Guo, and Xueqi Cheng. 2019. Event detection with multi-order graph convolution and aggregated attention. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5766–5770, Hong Kong, China. Association for Computational Linguistics.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*.

Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2017. Improved neural relation detection for knowledge base question answering. *arXiv preprint arXiv:1704.06194*.

Guodong Zhou, Longhua Qian, and Jianxi Fan. 2010. Tree kernel-based semantic relation extraction with rich syntactic and semantic information. *Information Sciences*, 180(8):1313–1325.

Yichao Zhou, Yu Yan, Rujun Han, J. Harry Caufield, Kai-Wei Chang, Yizhou Sun, Peipei Ping, and Wei Wang. 2021. Clinical Temporal Relation Extraction with Probabilistic Soft Logic Regularization and Global Inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(16):14647–14655.