GRAPHCACHE: Message Passing as Caching for Sentence-Level Relation Extraction

Anonymous ACL submission

Abstract

Entity types and **textual context** are essential properties for sentence-level relation extraction (RE). Existing work only encodes these properties within individual instances, which limits the performance of RE given the insufficient features in a single sentence. In contrast, we model these properties from the whole dataset and use the dataset-level information to enrich the semantics of every instance. We propose the GRAPHCACHE (**Graph Neural Network as Caching**) module, that propagates the features across sentences to learn better representations for RE. GRAPHCACHE aggregates the features from sentences in the whole dataset to learn **global** representations of properties, and use them to augment the **local** features within individual sentences. The global property features act as dataset-level prior knowledge for RE, and a complement to the sentence-level features. Inspired by the classical caching technique in computer systems, we develop GRAPHCACHE to update the property representations in an online manner. Overall, GRAPHCACHE yields significant effectiveness gains on RE and enables efficient message passing across all sentences in the dataset.

1 Introduction

Sentence-level relation extraction (RE) aims at identifying the relationship between two entities mentioned in a sentence. RE is crucial to the structural perception of human language, and also benefits many NLP applications such as automated knowledge base construction (Distiawan et al., 2019), event understanding (Wang et al., 2020a), discourse understanding (Yu et al., 2020), and question answering (Zhao et al., 2020). The modern tools of choice for RE are the large-scale pre-trained language models (PLMs) that are used to encode individual sentences, therefore obtaining the sentence-level representations (Liu et al., 2019; Joshi et al., 2020; Yamada et al., 2020).

Existing work considers **entity types** and **textual context** as essential **properties** for RE (Peng et al., 2020; Peters et al., 2019; Zhou and Chen, 2021). Nonetheless, most existing RE models only capture these properties *locally* within individual instances, while not *globally* modeling them from the whole dataset. Given the insufficient features of a single sentence, it is beneficial to model these properties from the whole dataset and use them to enrich the semantics of individual instances.

To overcome the aforementioned limitation, we propose to mine the entity and contextual information beyond individual instances so as to further improve the relation representations. Particularly, we first construct a heterogeneous graph to connect the instances sharing common properties for RE. This graph includes the sentences and *property caches*. Each cache represents a property of entity types or contextual topics. We connect every sentence to the corresponding property caches (see Fig. 1), and perform message passing over edges based on a graph neural network (GNN). In this way, the property caches aggregate the features from connected sentences, which will act as a complement to the sentence-level features and provide prior knowledge when identifying relations.

The constructed graph connecting sentences has the same scale as the whole dataset, which leads to high computational complexity of the GNN. To address this issue, our idea is to view the message passing of GNNs as data loading in computer systems, adapting the classical caching techniques to efficiently mining the property information from all sentences. We encapsulate this computational idea in a new GNN module, called GRAPHCACHE (**Graph Neural Network as Caching**), that uses an online updating strategy to refresh the property caches' representations. In addition, we design an attention-based global-local fusion module to augment the sentence-level representations using the property caches with adaptive weights.

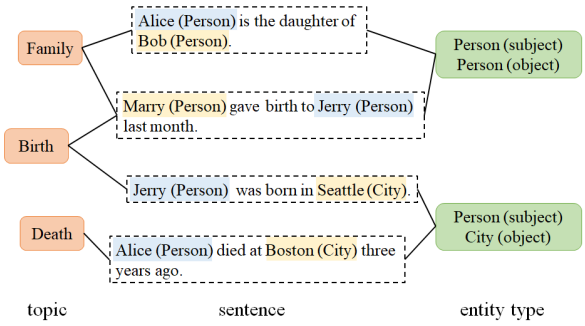

Figure 1: We construct a heterogeneous graph to connect the sentences sharing common properties for RE. We consider two kinds of properties: contextual topics and entity types.

GRAPHCACHE can be incorporated into popular RE models to improve their effectiveness without increasing their time complexity, as analyzed in theory (§3.2). As far as we know, ours is the first work to propagate the features across instances to enrich the semantics for sentence-level RE. We evaluate GRAPHCACHE on three public RE benchmarks including TACRED (Zhang et al., 2017), SemEval-2010 task 8 (Hendrickx et al., 2019), and TACREV (Alt et al., 2020a). Empirical results show that GRAPHCACHE consistently improves the effectiveness of popular RE models by a significant margin and propagates features between all sentences in an efficient manner.

2 Related Work

Sentence-Level Relation Extraction. Early research efforts (Zeng et al., 2014; Wang et al., 2016; Zhang et al., 2017) train RE models from scratch based on lexicon-level features. Recent work has shifted to fine-tuning pretrained language models (PLMs; Devlin et al. 2019; Liu et al. 2019) resulting in better performance. For example, BERT-MTB (Baldini Soares et al., 2019) continually fine-tunes the PLM with a matching-the-blanks objective that decides whether two sentences share the same entity. SpanBERT (Joshi et al., 2020) pre-trains a masked language model on random contiguous spans to learn span-boundaries and predict the entire masked span. LUKE (Yamada et al., 2020) extends the PLM's vocabulary with entities from Wikipedia and proposes an entity-aware self-attention mechanism. K-Adapter (Wang et al., 2020b) fixes the parameters of the PLM and uses feature adapters to infuse factual and linguistic knowledge. Despite their effectiveness, most existing work on sentence-level RE exploits the entity information and context within only an individual instance, while we propose to globally capture the semantic information from the whole dataset to augment the relation representations. Our model can be flexibly plugged into existing RE models and improve their effectiveness without increasing the time complexity.

Graph Neural Networks for Natural Language Processing. Due to the large body of work on applying GNNs to NLP, we refer readers to a recent survey (Wu et al., 2021) for a general review. GNNs have been explored in several NLP tasks such as semantic role labeling (Marcheggiani and Titov, 2017), machine translation (Bastings et al., 2017), and text classification (Henaff et al., 2015; Defferrard et al., 2016; Kipf and Welling, 2016; Peng et al., 2018; Yao et al., 2019). GNNs have also been widely adopted in various variants of relation extraction on the sentence level, (Zhang et al., 2018; Zhu et al., 2019; Guo et al., 2019a), the document level (Sahu et al., 2019; Christopoulou et al., 2019; Nan et al., 2020; Zeng et al., 2020), and the dialogue level (Xue et al., 2021). However, on the sentence-level relation extraction, most existing work (Zhang et al., 2018; Guo et al., 2019b; Wu et al., 2019) uses the graph neural networks to encode the relation representations from individual instances instead of operating the message passing between instances. In contrast, we build a heterogeneous graph to connect the instances that share the properties for RE, and design the caching updater to efficiently perform the message passing between instances.

3 Methodology

Task Definition. Sentence-level relation extraction (RE) aims to identify the relation between a pair of entities in a sentence. In this task, each instance is composed of a sentence, the subject and object entities, and entity types. For example, in the sentence *'Mary gave birth to Jerry at the age of 21.'*[1], *'Mary'* and *'Jerry'* are the entities, the entity types are both *person*, and the ground-truth relation between *'Jerry'* and *'Mary'* is *parent*.

We propose GRAPHCACHE (Graph Neural Networks as Caching) as a message passing methodology to model the dataset-level property repre-

[1]We use underline and wavy line to denote subject and object respectively by default.

sentations and use them to enrich every instance's semantics. GRAPHCACHE creates a graph representation where sentences with shared property information are connected with property caches. GRAPHCACHE first models the global semantic information by aggregating the features from the whole dataset, and then fuses the global and local features to augment the relational representations for every sentence.

We analogize the message passing in GNNs to caching in computer systems. Caching is about loading data from high volume disks to low volume caches, so as to accelerate data loading. Analogously, when GNNs perform the message passing between sentences through a smaller number of bridge nodes, we can think of the massive sentences in the dataset as the disk data, and the properties, which aggregates the features from sentences, as caches. GRAPHCACHE can be flexibly plugged into existing RE models. As far as we know, ours is the first work to propagate the features between instances to enrich the semantics for RE. GRAPHCACHE takes an existing RE model as the backbone, e.g., BERT, and takes the sentence-level representations given by the backbone as the inputs of message passing.

A GRAPHCACHE module consists of three key components: (i) *A graph construction technique* builds a few property caches. Each cache represents a property for RE: entity type or contextual topic. We connect each sentence to its corresponding properties, so that every property aggregates the features from its neighbor sentences. (ii) *Caching message passing* aggregates the sentence-level representations to model the properties' representations in an online manner. (iii) *Global-local fusion* fuses the global property representations and local sentence-level ones to augment the relation representations. Next, we will discuss the three main components in more detail.

3.1 Graph Construction for Sentence-level Relation Extraction

We build a large and heterogeneous graph to connect the sentences sharing the properties: **entity types** and **textual context**, which are essential for RE (Peng et al., 2020; Peters et al., 2019; Zhou and Chen, 2021). The heterogeneous graph is defined as $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges. $\mathcal{V} = \mathcal{V}_S \cup \mathcal{V}_P$, where $\mathcal{V}_S$ is the set of sentences, and $\mathcal{V}_P = \mathcal{V}_C \cup \mathcal{V}_E$ is the property caches. Here $\mathcal{V}_C$ is the set of latent topics (Zeng et al., 2018) mined from the latent topics from the text corpus using LDA (Blei et al., 2003), which has been found effective in modeling useful contextual patterns (Jelodar et al., 2019). Each topic is represented by a probability distribution over the words, and we assign each sentence to the top P topics with the largest probabilities. $\mathcal{V}_E$ is the set of entity types, where every cache represents the types of an entity pair. The entity types are also crucial for predicting relations (Peng et al., 2020; Zhou and Chen, 2021). An edge $(p, s) \in \mathcal{E}$ exists if the sentence $s \in \mathcal{V}_S$ has the property $p \in \mathcal{V}_P$.

We will implement a GNN on this graph. Specifically, to incorporate the global property information into relation extraction, the property caches aggregates the features from the connected neighboring sentences. This step enables property caches to globally model the properties from the whole dataset. We then use the global property representations from the caches to enrich every sentence's semantics. In this way, the property caches act as prior knowledge when identifying relations and provide each sentence with more representative features.

3.2 Caching Message Passing

We take an existing RE model as the backbone, e.g., BERT (Devlin et al., 2019), which produces the sentence-level representation as $\mathbf{h}_s$. Next, we deploy a two-layer GNN on our heterogeneous graph for message passing across sentences. Specifically, the first GNN layer aggregates the sentence-level representations to property caches at the tth training step:

$$\begin{aligned} \bar{\mathbf{h}}_p(t) &= \text{MEAN}\left(\{\mathbf{h}_s(t), s \in \mathcal{N}(p)\}\right), \\ \mathbf{h}_p(t) &= \text{FFN}\left(\bar{\mathbf{h}}_p(t)\right), \end{aligned} \tag{1}$$

where $p \in \mathcal{V}_P$ is a property, $s \in \mathcal{N}(p)$ is a sentence having property p, $\text{MEAN}(\cdot)$ is the mean aggregator (Hamilton et al., 2017), and $\text{FFN}(\cdot)$ is the feed-forward network. $\text{FFN}(\cdot)$ can be a linear layer in SGC (Wu et al., 2019), a linear layer followed by a nonlinear activation function in GraphSAGE (Hamilton et al., 2017), or a multi-layer perception in GIN (Xu et al., 2018), etc. We follow SGC (Wu et al., 2019) to implement $\text{FFN}(\cdot)$ by default. For each property p, this layer aggregates the sentence-level representations $\mathbf{h}_s(t)$ from $s \in \mathcal{N}(p)$ to obtain a global property embedding $\mathbf{h}_p(t)$. In this way, the generalized context of each

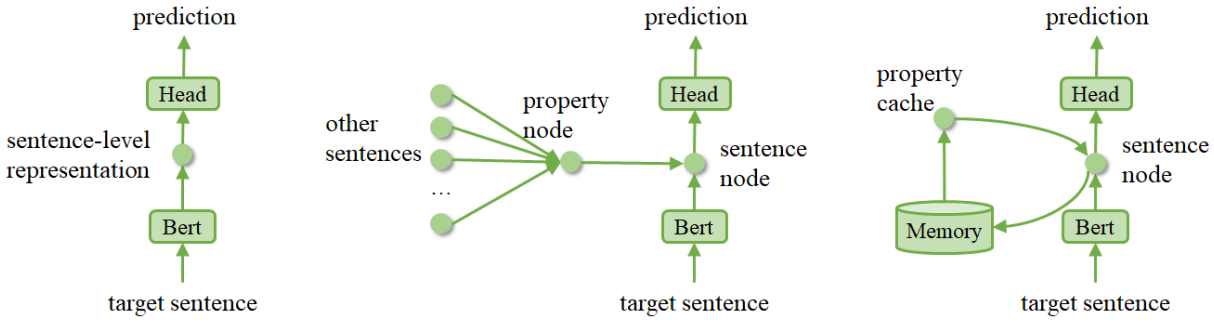

Figure 2: (left) Existing models encode individual instances for RE. (middle) In standard GNNs (Kipf and Welling, 2016), we predict for an instance by aggregating the features from many other sentences in the dataset, leading to high time complexity. (right) Our GRAPHCACHE implements a caching updater (see Eq. 2) to update the properties' representations in an online manner, which significantly reduces the time complexity.

Algorithm 1 GRAPHCACHE for Relation Extraction

Input: The number of training steps T, the dataset $\mathcal{D} = \{se_s, r_s | s = 1, 2, \ldots, N\}$, where se_s, r_s are the sentence and relation of the sth instance, our graph G defined in §3.1, and the batch size B.
Output: The model's trained parameters.

Initialize the model's parameters as random values, and initialize the values of memory $\mathcal{M}$ and property caches $\hat{\mathbf{h}}_p(t)$ as zero.
for $t \leftarrow 1$ to T **do**
  Sample a batch $\mathcal{B}(t)$ from $\mathcal{D}$.
  for s in $\mathcal{B}(t)$ **do**
    $\mathbf{h}_s(t) \leftarrow \text{Backbone}(se_s)$
  end for
  for p in $\mathcal{V}_p$ **do**
    Update $\hat{\mathbf{h}}_p(t)$ as Eq. 2.
    $\mathbf{h}_p(t) \leftarrow \text{FFN}(\hat{\mathbf{h}}_p(t))$ as Eq. 1.
  end for
  for s in $\mathcal{B}(t)$ **do**
    Update $\hat{r}_s(t)$ as Eq. 3.
    $\mathcal{M}[s] \leftarrow \mathbf{h}_s(t)$.
  end for
  Back-propagate to update the parameters by minimizing the cross entropy loss between $\hat{r}_s(t)$ and r_i of instances in $\mathcal{B}$.
end for

property is captured from the whole dataset, which is further used to enhance the relation representations for each sentence in the second GNN layer. We describe the details of the second GNN layer in §3.3.

Recall our heterogeneous graph for RE defined in §3.1. At each training step, classical GNNs perform message passing across edges between the sentences and properties. In this case, the time complexity of the first GNN layer at each training step is $|\mathcal{E}|$. Note that $|\mathcal{E}|$ is larger than $|V_s|$, which is the number of sentences in the dataset. This leads to poor scalability of GNN, since $|V_s|$ is large in practice.

To address this efficiency issue, we propose Caching GNN for RE in Alg. 1. Our GRAPHCACHE implements a memory dictionary $\mathcal{M}$ to store the sentence-level representations from the backbone. To keep consistency with the updating parameters during training, we deploy a caching updater to refresh the properties' representations at each training step:

$$\begin{aligned}&\hat{\mathbf{h}}_p(t)\\ =&\text{Updater}(\hat{\mathbf{h}}_p(t-1), \{\mathbf{h}_s(t), s \in \mathcal{B}(t)\})\\ =&\hat{\mathbf{h}}_p(t-1) + \sum_{s \in \mathcal{N}(p) \cap \mathcal{B}(t)} \frac{\mathbf{h}_s(t) - \mathcal{M}[s]}{|\mathcal{N}(p)|}, \end{aligned} \quad (2)$$

where $\mathcal{B}(t)$ denotes the batch at the tth training step. By doing so, GRAPHCACHE greatly reduces the time complexity from $|\mathcal{E}|$ to $|\mathcal{B}(t)|$ at each training step by using Updater to obtain the property caches' representations $\hat{\mathbf{h}}_p(t)$.

Our caching updater is much more efficient than the classical message passing of GNNs, since $|\mathcal{B}(t)| \ll |V_s| < |\mathcal{E}|$ generally holds in practice. When we aggregate the sentence-level representations from $\mathcal{M}$, we provide the following proposition to show that our cache updater is as effective as the first GNN layer in Eq. 1.

Proposition 1. *At the tth training step, denote the property caches' representations in the first GNN layer (see Eq. 1) as $\bar{\mathbf{h}}_p(t)$, and those returned by*

our updater in Eq. 2 as $\hat{\mathbf{h}}_p(t)$. *There is* $\hat{\mathbf{h}}_p(t) = \bar{\mathbf{h}}_p(t)$ *for* $\forall p \in \mathcal{V}_P, t > 0$.

We provide the proof of this proposition in the appendix.

3.3 Global-Local Fusion

In the second GNN layer, we propagate the properties' representations from the property cache to their neighboring sentences in the batch. Since a sentence s may have more than one latent topic $|\mathcal{V}_C \cap \mathcal{N}(i)| > 1$, we utilize the attention mechanism to enable the target sentence to attend to different topics with adaptive weights.

$$\mathbf{h}_s^{topic}(t) = \text{Attention}(\mathbf{h}_s(t), \{\mathbf{h}_p(t), p \in \mathcal{V}_C\}),$$

where we follow (Vaswani et al., 2017) to implement Attention. The output $\mathbf{h}_s^{topic}(t)$ is the topic embedding fused for sentence s. In this way, a sentence can be trained to attend to more relevant topics with higher weights.

Next, we have the entity type embedding of sentence s as $\mathbf{h}_s^{entity}(t) = \mathbf{h}_p(t), p \in \mathcal{V}_E \cap \mathcal{N}(s)$, where $p \in \mathcal{V}_E \cap \mathcal{N}(s)$ is the entity type node connected to sentence s. $\mathbf{h}_s^{topic}(t)$ and $\mathbf{h}_s^{entity}(t)$ are the global representations of the properties related to sentence s, while $\mathbf{h}_s$ is the local representation of sentence s. We fuse the global and local representations to enrich the semantics of sentence s through a sentence-wise head:

$$\hat{r}_i(t) = \text{Head}\left(\mathbf{h}_s(t) \| \mathbf{h}_s^{topic}(t) \| \mathbf{h}_s^{entity}(t)\right), \quad (3)$$

where $\|$ denotes concatenation. GRAPHCACHE makes sentence-wise relation predictions $\hat{r}_i(t)$ using a sentence-wise Head, implemented as a multi-layer perception (MLP), analogous to a PointNet (Qi et al., 2017). Since GRAPHCACHE predicts a relation label for each sentence, it can be trained by standard task-specific classification losses, e.g., cross-entropy (Mannor et al., 2005). During inference, we take $\hat{r}_i(t)$ after convergence as the output for RE.

4 Experiments

In this section, we evaluate the effectiveness of our GRAPHCACHE method when incorporated into various RE models. We compare our methods against a variety of strong baselines on the task of sentence-level RE. We closely follow the experimental setting of the previous work (Zhang et al., 2017; Zhou and Chen, 2021; Zhang et al., 2018) to ensure a fair comparison, as detailed below.

Dataset	#Train	#Dev	#Test	#Classes
TACRED	68,124	22,631	15,509	42
SemEval	6,507	1,493	2,717	19
TACREV	68,124	22,631	15,509	42

Table 1: Statistics of datasets.

Method	TACRED	SemEval	TACREV
PA-LSTM (Zhang et al., 2017)	65.1	82.1	73.3
GCN (Zhang et al., 2018)	64.0	80.7	71.9
C-GCN (Zhang et al., 2018)	66.4	84.2	74.6
C-SGC (Wu et al., 2019)	67.0	84.8	75.1
SpanBERT (Joshi et al., 2020)	70.8	86.1	78.0
RECENT (Lyu and Chen, 2021)	75.2	85.8	83.0
IRE_{BERT} (Zhou and Chen, 2021)	72.9	86.4	81.3
LUKE (Yamada et al., 2020)	72.7	87.8	80.6
LUKE + GRAPHCACHE (ours)	74.8	**89.1**	81.5
$\text{IRE}_{\text{RoBERTa}}$ (Zhou and Chen, 2021)	74.6	87.5	83.2
$\text{IRE}_{\text{RoBERTa}}$ + GRAPHCACHE (ours)	**75.5**	88.2	**84.2**

Table 2: F1 scores (%) of Relation Extraction on the test set of TACRED, SemEval, and TACREV. The best results in each column are highlighted in **bold** font.

4.1 Experimental Settings

Datasets. We use the standard sentence-level RE datasets: TACRED (Zhang et al., 2017), SemEval-2010 Task 8 (Hendrickx et al., 2019), and TACREV (Alt et al., 2020b) for evaluation. TACRED contains over 106k mention pairs drawn from the yearly TAC KBP challenge. SemEval does not provide entity type annotations, for which we only construct the topic caches for message passing. Alt et al. (2020b) relabeled the development and test sets of TACRED to build TACREV. The statistics of these datasets are shown in Tab. 1. We follow (Zhang et al., 2017) to use F1-micro as the evaluation metric.

Compared Methods. We compare GRAPHCACHE with the following state-of-the-art RE models: (1) **PA-LSTM** (Zhang et al., 2017) extends the bi-directional LSTM by incorporating positional information to the attention mechanism. (2) **GCN** (Zhang et al., 2018) uses a graph convolutional network to gather relevant contextual information along syntactic dependency paths. (3) **C-GCN** (Zhang et al., 2018) combines GCN and LSTM, leading to improved performance than each method alone. (4) **C-SGC** (Wu et al., 2019) simplifies GCN by removing the nonlinear layers and achieves higher effectiveness. (5) **SpanBERT** (Joshi et al., 2020) extends BERT by intro-

ducing a new pretraining objective of continuous span prediction. (6) **RECENT** (Lyu and Chen, 2021) restricts the candidate relations based on the entity types. (7) **LUKE** (Yamada et al., 2020) pretrains the language model on both large text corpora and knowledge graphs and further proposes an entity-aware self-attention mechanism. (8) **IRE** (Zhou and Chen, 2021) proposes an improved entity representation technique in data preprocessing, which enables RoBERTa to achieve state-of-the-art performance on RE.

Model Configuration. For the hyper-parameters of the considered baseline methods, e.g., the batch size, the number of hidden units, the optimizer, and the learning rate, we set them as those in the original papers. For LDA used in GRAPHCACHE, we set the number of topics K as 50, and the number of top relevant topics for every sentence P as 2. For all experiments, we report the median F-1 scores of five runs of training using different random seeds.

4.2 Overall Performance

We incorporate the GRAPHCACHE framework with LUKE and $\text{IRE}_{\text{RoBERTa}}$, and report the results in Tab. 2. Our GRAPHCACHE method improves LUKE by 2.9% on TACREV, 1.5% on SemEval, and 1.1% on TACREV in the F1 score. For $\text{IRE}_{\text{RoBERTa}}$, GRAPHCACHE leads to the improvement of 1.2% on TACRED, 0.8% on SemEval, 1.2% on Re-TACRED. As a result, our GRAPHCACHE achieves substantial improvements for LUKE and $\text{IRE}_{\text{RoBERTa}}$ and enables them to outperform the baseline methods.

Note that LUKE and $\text{IRE}_{\text{RoBERTa}}$ are both based on large pre-trained models, which have sufficiently large learning capacity to encode the individual instances. In this case, our GRAPHCACHE still improves their effectiveness by a large margin, which validates the benefits of modeling the properties: entity types and contextual topics, globally from the whole dataset. This is due to the use of the global property representations that enrich the semantics of each instance, which effectively act as prior knowledge that helps identify the relations and complements the sentence-level features.

4.3 Efficiency and Effectiveness of GRAPHCACHE

As analyzed in §3.2, GRAPHCACHE enhances the backbone RE models without increasing their time complexity. In the experiments, we analyze the efficiency and effectiveness of GRAPHCACHE on the TACRED dataset, following the experimental setting of RE in §4.2.

Method	Complexity	Time	F1 (%)
$\text{IRE}_{\text{RoBERTa}}$ (Zhou and Chen, 2021)	$\mathcal{O}(B)$	7492s	74.6
$\text{IRE}_{\text{RoBERTa}}$ + GNN	$\mathcal{O}(N)$	N.A.	N.A.
$\text{IRE}_{\text{RoBERTa}}$ + GRAPHCACHE (ours)	$\mathcal{O}(B)$	7681s	**75.5**

Table 3: Training time, the time complexity per training step, and F1 scores of $\text{IRE}_{\text{RoBERTa}}$ with our proposed message passing implemented as GNN and GRAPHCACHE on TACRED. The training time of $\text{IRE}_{\text{RoBERTa}}$ with the classical GNN is unavailable due to the our-of-memory error. B and N are the batch and dataset sizes respectively.

Method	TACRED	TACREV
LUKE (Yamada et al., 2020)	76.5	82.9
LUKE + GRAPHCACHE (ours)	78.9	85.6
$\text{IRE}_{\text{RoBERTa}}$ (Zhou and Chen, 2021)	78.7	86.9
$\text{IRE}_{\text{RoBERTa}}$ + GRAPHCACHE (ours)	**80.1**	**88.2**

Table 4: Test F1 scores (%) of Relation Extraction on the filtered test sets (see §4.4), i.e., the instances containing unseen entities.

The methods we evaluate include $\text{IRE}_{\text{RoBERTa}}$, $\text{IRE}_{\text{RoBERTa}}$ implemented with classical GNN for message passing, and $\text{IRE}_{\text{RoBERTa}}$ with our GRAPHCACHE. Tab. 3 reports the performance, where 'Time' is the training time until convergence using a Linux Server with an Intel(R) Xeon(R) E5-1650 v4 @ 3.60GHz CPU and a GeForce GTX 2080 GPU.

We notice that, compared with the classical message passing of GNN, our GRAPHCACHE method significantly reduces the time complexity per training step. As a result, our GRAPHCACHE method takes significantly less training time than the classical GNN method, and exhibits similar efficiency to the original $\text{IRE}_{\text{RoBERTa}}$ without message passing between sentences. The running time and F1 of $\text{IRE}_{\text{RoBERTa}}$ with GNN is unavailable due to the out-of-memory error. This agrees with the theoretical analysis in §3.2. N and B denote the data and batch sizes respectively. $\text{IRE}_{\text{RoBERTa}}$'s time complexity is $\mathcal{O}(B)$, which is the same as the original RoBERTa, while the time complexity of RoBERTa with GNN is $\mathcal{O}(N)$, being significantly higher than our GRAPHCACHE. In practice, N is generally large, and $N \gg B$, e.g., $|\mathcal{E}| > 1 \times 10^5$ and $B < 100$ holds for TACRED and state-of-the-art models.

Technique	F1 (%)	Δ	Cumu Δ
LUKE (Yamada et al., 2020)	72.7	0	0
+ Entity Types	73.4	+0.7	+0.7
+ Contextual Topics	74.8	**+1.4**	**+2.1**

Table 5: Effects of different properties in our heterogeneous graph on the RE of TACRED.

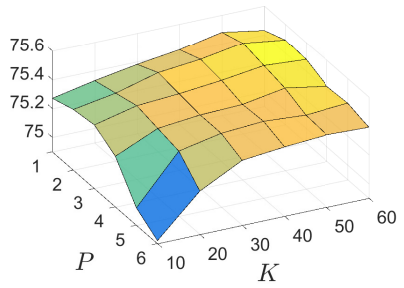

Figure 3: The F1 scores (% in z-axis) of $\text{IRE}_{\text{RoBERTa}}$ with GRAPHCACHE on relation extraction on TACRED with different hyper-parameters P and K.

In terms of effectiveness, our GRAPHCACHE leads to substantial improvements for RoBERTa. Our GRAPHCACHE enriches the input features for RE on every sentence by utilizing the dataset-level information beyond the individual sentences. GRAPHCACHE implements the attention module to incorporate the global property features from different topic caches with adaptive weights, which capture the most relevant information for the target relation. The improvements in effectiveness are rooted in the message passing mechanism between sentences, which mines the property information beyond individual instances and acts as a complementary to the sentence-level semantics. Our GRAPHCACHE method resolves the efficiency issues of message passing based on the caching mechanism, which updates the properties' representations in an online manner.

4.4 Analysis on Unseen Entities

Some previous work (Zhang et al., 2018; Joshi et al., 2020) suggests that RE models may not generalize well to unseen entities. To evaluate whether the RE models can generalize to unseen entities, existing work designs a filtered evaluation setting (Zhou and Chen, 2021). This setting removes all testing instances containing entities from the training set of TACRED and TACREV, which results in filtered test sets of 4,599 instances on TACRED and TACREV. These filtered test sets only contain instances with unseen entities during training.

We present the experimental results on the filtered test sets in Tab. 4. Our GRAPHCACHE still achieves consistently substantial improvements for LUKE and $\text{IRE}_{\text{RoBERTa}}$ on the TACRED and TACREV datasets. Specifically, our GRAPHCACHE improves the F1 scores of LUKE by 3.1% on TACRED, 3.3% on TACREV, and improves $\text{IRE}_{\text{RoBERTa}}$ by 1.8% on TACRED, 1.5% on TACREV. Taking a closer look, we observe that the improvements given by GRAPHCACHE on the filtered test sets are generally larger than those on the original test sets. The reason is that our GRAPHCACHE mines global information from the whole dataset and uses it as the prior knowledge for RE, which is not influenced by the entity names in individual sentences. When the entity names are new to the RE models, the semantic information is relatively scarce and our mined global information plays a more important role to augment the sentence-level representations.

4.5 Ablation Study

We investigate the contributions of properties that we consider for constructing the heterogeneous graph. We apply different kinds of properties sequentially with our GRAPHCACHE on the LUKE model. The results are presented in Tab. 5. Our entity type nodes improve the effectiveness of LUKE by modeling the entity information globally on the dataset level to enrich the semantics of every sentence. This finding is consistent with Peng et al. (2020), suggesting that the entity information can provide richer information to improve RE. Furthermore, the contextual topics lead to more significant improvements than the entity types, since the contextual information is fundamental for identifying the relations.

Finally, we analyze the sensitivity of GRAPHCACHE to the hyper-parameters K, P, where K is the number of topics and P is the number of relevant topics assigned to an instance. The result is visualized in Fig. 3. We vary K among $\{10, 20, 30, 40, 50, 60\}$ and P among $\{1, 2, 3, 4, 5, 6\}$. The performance of $\text{IRE}_{\text{RoBERTa}}$ with GRAPHCACHE is relatively smooth when parameters are within certain ranges. However, extremely small values of K and large P result in poor performances. Too small K cannot effectively model the complex contextual topics in the large text corpus, while too large P induces irrele-

Input sentence	Method	Prediction	Entity type	Topic keyword
Founded in 1947 by two brothers, Eugene and Quentin Fabris, New Fabris started out making sewing machine parts in the 1990s.	LUKE	founded ✗	subject: Person object: Date	[brother, found, sister, parent, establish, machine, business, organize, instrument, make]
	+ GRAPHCACHE	**no_relation ✓**		
According to the suspect, Gonzalez was strangled and buried the day after the video was made, Rosas said.	LUKE	no_relation ✗	subject: Person object: Date	[strangle, die, after, when, injury, day, hospital, police, murder, later]
	+ GRAPHCACHE	**date_of_death ✓**		
He was forced to close his bar and now works occasionally at the University of Foreigners, which Knox and Kercher attended.	LUKE	no_relation ✗	subject: Person object: Organization	[university, student, attend, opening, work, school, job, professor, exchange, education]
	+ GRAPHCACHE	**schools_attended ✓**		
Margaret Garritsen graduated from the University of Michigan as an American Association of University scholar.	LUKE	schools_attended ✗	subject: Organization object: Organization	[graduate, government, association, degree, university, technology, science, scholar, receive, research]
	+ GRAPHCACHE	**no_relation ✓**		

Table 6: A case study for LUKE and our GRAPHCACHE on the relation extraction dataset TACRED. We report the predicted relations of different methods, the entity types, and the top 10 words with the highest probabilities of the topic that the sentence attends with the highest attention weight.

vant or noisy features for every instance. Moreover, only a poorly set hyper-parameter does not lead to significant performance degradation, which demonstrates that our GRAPHCACHE framework is able to effectively mine the beneficial properties at the dataset level and use them to enhance the relation representations for RE.

4.6 Case Study

We conduct a case study to investigate the effects of our GRAPHCACHE. Tab. 6 gives a qualitative comparison example between LUKE and the LUKE with our GRAPHCACHE on the relation extraction dataset TACRED. The result shows that the global property information that we mine from the whole dataset can guide the RE systems to make correct predictions. For example, in the first row, we model the global entity type information of the subject as the *person* and the object as the *date* from the whole dataset. This type information acts as the prior knowledge that prevents the model from making the wrong relation prediction of 'founded' between the entities '*Quentin Fabris*' and '*1947*' (date). Similarly, in the final row, our GRAPHCACHE filters out the incorrect relation '*schools_attend*', since we model the entity type information from the whole dataset and thus enable the model to be aware that this relation cannot hold for the subject type as '*organization*'.

In addition, in the second row, the sentence '*According to the suspect, Gonzalez was strangled and buried the day after the video was made, Rosas said.*' attends to the topic of keywords '[strangle, die, after, when, injury, day, hospital, police, murder, later]' in our heterogeneous graph, which enriches the semantics of the sentence with the context related to the death and time. This helps the model to make the correct relation prediction '*date_of_death*'.

5 Conclusion

In this paper, we study the efficient message passing to enhance the relation extraction models. We propose a novel method named GRAPHCACHE, which provides efficient message passing between instances in the whole dataset. GRAPHCACHE is a model-agnostic technique that can be incorporated into popular relation extraction models to enhance their effectiveness without increasing their time complexity. In our work, we present a simple yet effective implementation of GRAPHCACHE, which models two universal and essential properties for relation extraction: entity information and textual context. Our experimental results show that GRAPHCACHE, with our heterogeneous graph, yields significant gains for the sentence-level relation extraction in an efficient manner.

References

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020a. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online. Association for Computational Linguistics.

Christoph Alt, Aleksandra Gabryszak, and Leonhard Hennig. 2020b. TACRED revisited: A thorough evaluation of the TACRED relation extraction task. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1558–1569, Online. Association for Computational Linguistics.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

Jasmijn Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. 2017. Graph convolutional encoders for syntax-aware neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1957–1967, Copenhagen, Denmark. Association for Computational Linguistics.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Connecting the dots: Document-level neural relation extraction with edge-oriented graphs. 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 4925–4936, Hong Kong, China. Association for Computational Linguistics.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

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.

Bayu Distiawan, Gerhard Weikum, Jianzhong Qi, and Rui Zhang. 2019. Neural relation extraction for knowledge base enrichment. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 229–240.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019a. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, and Wei Lu. 2019b. Attention guided graph convolutional networks for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 241–251, Florence, Italy. Association for Computational Linguistics.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1025–1035.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2019. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *arXiv preprint arXiv:1911.10422*.

Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. 2019. Latent dirichlet allocation (lda) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, 78(11):15169–15211.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

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

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*.

Shengfei Lyu and Huanhuan Chen. 2021. Relation classification with entity type restriction. *arXiv preprint arXiv:2105.08393*.

Shie Mannor, Dori Peleg, and Reuven Rubinstein. 2005. The cross entropy method for classification. In *Proceedings of the 22nd international conference on Machine learning*, pages 561–568.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1506–1515, Copenhagen, Denmark. Association for Computational Linguistics.

Guoshun Nan, Zhijiang Guo, Ivan Sekulic, and Wei Lu. 2020. Reasoning with latent structure refinement for document-level relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1546–1557, Online. Association for Computational Linguistics.

Hao Peng, Tianyu Gao, Xu Han, Yankai Lin, Peng Li, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2020. Learning from context or names? an empirical study on neural relation extraction. *arXiv preprint arXiv:2010.01923*.

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.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. 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 43–54, Hong Kong, China. Association for Computational Linguistics.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and Sophia Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4309–4316, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Haoyu Wang, Muhao Chen, Hongming Zhang, and Dan Roth. 2020a. 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.

Linlin Wang, Zhu Cao, Gerard de Melo, and Zhiyuan Liu. 2016. Relation classification via multi-level attention CNNs. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1298–1307, Berlin, Germany. Association for Computational Linguistics.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. 2020b. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090*.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Fuzhao Xue, Aixin Sun, Hao Zhang, and Eng Siong Chng. 2021. Gdpnet: Refining latent multi-view graph for relation extraction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14194–14202.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6442–6454, Online. 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*, volume 33, pages 7370–7377.

Dian Yu, Kai Sun, Claire Cardie, and Dong Yu. 2020. Dialogue-based relation extraction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4927–4940.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Jichuan Zeng, Jing Li, Yan Song, Cuiyun Gao, Michael R. Lyu, and Irwin King. 2018. Topic memory networks for short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3120–3131, Brussels, Belgium. Association for Computational Linguistics.

Shuang Zeng, Runxin Xu, Baobao Chang, and Lei Li. 2020. Double graph based reasoning for document-level relation extraction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1630–1640, Online. Association for Computational Linguistics.

Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2205–2215, Brussels, Belgium. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Xinyan Zhao, Feng Xiao, Haoming Zhong, Jun Yao, and Huanhuan Chen. 2020. Condition aware and revise transformer for question answering. In *Proceedings of The Web Conference 2020*, pages 2377–2387.

Wenxuan Zhou and Muhao Chen. 2021. An improved baseline for sentence-level relation extraction. *arXiv preprint arXiv:2102.01373*.

Hao Zhu, Yankai Lin, Zhiyuan Liu, Jie Fu, Tat-Seng Chua, and Maosong Sun. 2019. Graph neural networks with generated parameters for relation extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1331–1339, Florence, Italy. Association for Computational Linguistics.

A Proof of the Proposition

Proposition 2. *At the tth training step, denote the property caches' representations in the first GNN layer (see Eq. 1) as $\bar{\mathbf{h}}_p(t)$, and those returned by our updater in Eq. 2 as $\hat{\mathbf{h}}_p(t)$. There is $\hat{\mathbf{h}}_p(t) = \bar{\mathbf{h}}_p(t)$ for $\forall p \in \mathcal{V}_P, t > 0$.*

Proof. When $t > 1$, if $\hat{\mathbf{h}}_p(t-1) = \bar{\mathbf{h}}_p(t-1)$, we have:

$$
\begin{aligned}
&\hat{\mathbf{h}}_p(t) \\
=&\mathrm{Updater}(\hat{\mathbf{h}}_p(t-1), \{\mathbf{h}_s(t), s \in \mathcal{B}(t)\}) \qquad (4)\\
=&\hat{\mathbf{h}}_p(t-1) + \sum_{s \in \mathcal{N}(p) \cap \mathcal{B}(t)} \frac{\mathbf{h}_s(t) - \mathcal{M}[s]}{|\mathcal{N}(p)|} \\
=& \sum_{s \in \mathcal{N}(p)} \frac{\mathcal{M}[s]}{|\mathcal{N}(p)|} + \sum_{s \in \mathcal{N}(p) \cap \mathcal{B}(t)} \frac{\mathbf{h}_s(t) - \mathcal{M}[s]}{|\mathcal{N}(p)|} \qquad (5)\\
=& \sum_{s \in \mathcal{N}(p) \setminus \mathcal{B}(t)} \frac{\mathcal{M}[s]}{|\mathcal{N}(p)|} + \sum_{s \in \mathcal{N}(p) \cap \mathcal{B}(t)} \frac{\mathbf{h}_s(t)}{|\mathcal{N}(p)|} \qquad (6)\\
=&\bar{\mathbf{h}}_p(t). \qquad (7)
\end{aligned}
$$

Besides, because $\hat{\mathbf{h}}_p(0) = \bar{\mathbf{h}}_p(0)$ for $\forall p \in \mathcal{V}_P$ holds as initialized in Alg. Alg. 1, we have $\hat{\mathbf{h}}_p(t) = \bar{\mathbf{h}}_p(t)$ for $\forall p \in \mathcal{V}_P, t > 0$. □