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.
We consider two kinds of properties: contextual topics and 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.

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.

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 TACREVD (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.

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.’, ‘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 representation.
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 $G = (V, E)$, where $V$ is the set of nodes, and $E$ is the set of edges. $V = V_S \cup V_P$, where $V_S$ is the set of sentences, and $V_P = V_C \cup V_E$ is the property caches. Here $V_C$ 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. $V_E$ 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 $(p, s) \in E$ exists if the sentence $s \in V_S$ has the property $p \in V_P$.

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 $h_s$. 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 $t$th training step:

$$h_p(t) = \text{MEAN} \left( \{ h_s(t), s \in N(p) \} \right),$$

$$h_p(t) = \text{FFN}(h_p(t)), \quad (1)$$

where $p \in V_P$ is a property, $s \in N(p)$ 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 $h_s(t)$ from $s \in N(p)$ to obtain a global property embedding $h_p(t)$. 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 $D = \{se_s, r_s|s = 1, 2, \ldots, N\}$, where $se_s, r_s$ are the sentence and relation of the $s$th instance, our graph $G$ defined in §3.1, and the batch size $B$.

Output: The model’s trained parameters.

1: Initialize the model’s parameters as random values, and initialize the values of memory $M$ and property caches $h_p(t)$ as zero.
2: for $t$ ← 1 to $T$ do
3: Sample a batch $B(t)$ from $D$.
4: for $s$ in $B(t)$ do
5: $h_s(t) \leftarrow$ Backbone$(se_s)$
6: end for
7: for $p$ in $V_p$ do
8: Update $\hat{h}_p(t)$ as Eq. 2.
9: $h_p(t) \leftarrow$ FFN$(\hat{h}_p(t))$ as Eq. 1.
10: end for
11: for $s$ in $B(t)$ do
12: Update $\hat{r}_s(t)$ as Eq. 3.
13: $M[s] \leftarrow h_s(t)$.
14: end for
15: Back-propagate to update the parameters by minimizing the cross entropy loss between $\hat{r}_s(t)$ and $r_s$ of instances in $B$.
16: 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 $M$ 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:

$$\hat{h}_p(t) = \text{Updater}(\hat{h}_p(t - 1), \{h_s(t), s \in B(t)\})$$

$$\hat{h}_p(t - 1) + \sum_{s \in \mathcal{N}(p) \cap B(t)} \frac{h_s(t) - M[s]}{|\mathcal{N}(p)|}, \quad (2)$$

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

Our caching updater is much more efficient than the classical message passing of GNNs, since $|B(t)| \ll |V_s| < |\mathcal{E}|$ generally holds in practice. When we aggregate the sentence-level representations from $M$, 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 $t$th training step, denote the property caches’ representations in the first GNN layer (see Eq. 1) as $h_p(t)$, and those returned by
our updater in Eq. 2 as \( \hat{h}_p(t) \). There is \( \hat{h}_p(t) = h_p(t) \) 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 \( |\mathcal{V}_C \cap \mathcal{N}(t)| > 1 \), we utilize the attention mechanism to enable the target sentence to attend to different topics with adaptive weights.

\[
h^\text{topic}_s(t) = \text{Attention}(h_s(t), \{h_p(t), p \in \mathcal{V}_C\}),
\]

where we follow (Vaswani et al., 2017) to implement Attention. The output \( h^\text{topic}_s(t) \) 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 \( h^\text{entity}_s(t) = h_p(t), p \in \mathcal{V}_E \cap \mathcal{N}(s) \), where \( p \in \mathcal{V}_E \cap \mathcal{N}(s) \) is the entity type node connected to sentence \( s \). \( h^\text{topic}_s(t) \) and \( h^\text{entity}_s(t) \) are the global representations of the properties related to sentence \( s \), while \( h_s \) 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:

\[
\hat{r}_i(t) = \text{Head}(h_s(t) || h^\text{topic}_s(t) || h^\text{entity}_s(t)), \tag{3}
\]

where \( || \) denotes concatenation. \textsc{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 \textsc{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 \textsc{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.

![Table 1: Statistics of datasets.](image)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Train</th>
<th>#Dev</th>
<th>#Test</th>
<th>#Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TACRED</td>
<td>68,124</td>
<td>22,631</td>
<td>15,509</td>
<td>42</td>
</tr>
<tr>
<td>SemEval</td>
<td>6,507</td>
<td>1,493</td>
<td>2,717</td>
<td>19</td>
</tr>
<tr>
<td>TACREV</td>
<td>68,124</td>
<td>22,631</td>
<td>15,509</td>
<td>42</td>
</tr>
</tbody>
</table>

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 \textsc{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) pretrained 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 IRE\textsubscript{RoBERTa}, and report the results in Tab. 2. Our GRAPHCACHE method improves LUKE by 2.9% on TACRED, 1.5% on SemEval, and 1.1% on TACREV in the F1 score. For IRE\textsubscript{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 IRE\textsubscript{RoBERTa} and enables them to outperform the baseline methods.

Note that LUKE and IRE\textsubscript{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.

The methods we evaluate include IRE\textsubscript{RoBERTa}, IRE\textsubscript{RoBERTa} implemented with classical GNN for message passing, and IRE\textsubscript{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 IRE\textsubscript{RoBERTa} without message passing between sentences. The running time and F1 of IRE\textsubscript{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. IRE\textsubscript{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., \( |E| > 1 \times 10^5 \) and \( B < 100 \) holds for TACRED and state-of-the-art models.

<table>
<thead>
<tr>
<th>Method</th>
<th>Complexity</th>
<th>Time</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IRE\textsubscript{RoBERTa} (Zhou and Chen, 2021)</td>
<td>( \mathcal{O}(B) )</td>
<td>7492s</td>
<td>74.6</td>
</tr>
<tr>
<td>IRE\textsubscript{RoBERTa} + GNN</td>
<td>( \mathcal{O}(N) )</td>
<td>N.A.</td>
<td>N.A.</td>
</tr>
<tr>
<td>IRE\textsubscript{RoBERTa} + GRAPHCACHE (ours)</td>
<td>( \mathcal{O}(B) )</td>
<td>7681s</td>
<td>75.5</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Method</th>
<th>TACRED</th>
<th>TACREV</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUKE (Yamada et al., 2020)</td>
<td>76.5</td>
<td>82.9</td>
</tr>
<tr>
<td>LUKE + GRAPHCACHE (ours)</td>
<td>78.9</td>
<td>85.6</td>
</tr>
<tr>
<td>IRE\textsubscript{RoBERTa} (Zhou and Chen, 2021)</td>
<td>78.7</td>
<td>86.9</td>
</tr>
<tr>
<td>IRE\textsubscript{RoBERTa} + GRAPHCACHE (ours)</td>
<td>80.1</td>
<td>88.2</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Technique</th>
<th>F1 (%)</th>
<th>$\Delta$</th>
<th>Cumu $\Delta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>LUKE (Yamada et al., 2020)</td>
<td>72.7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>+ Entity Types</td>
<td>73.4</td>
<td>+0.7</td>
<td>+0.7</td>
</tr>
<tr>
<td>+ Contextual Topics</td>
<td>74.8</td>
<td>+1.4</td>
<td>+2.1</td>
</tr>
</tbody>
</table>

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

[7]

---

**Note:** The table and figure are placeholders as the actual content is not provided in the image. The above text is a natural reading of the provided content.
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 ‘Eugene and Quentin Fabris’ and ‘1947’ (date). Similarly, in the final row, our GRAPHCACHE filters out the incorrect relation ‘schools_attended’, 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’.

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.
References


A Proof of the Proposition

Proposition 2. At the $t$th training step, denote the property caches’ representations in the first GNN layer (see Eq. 1) as $\hat{h}_p(t)$, and those returned by our updater in Eq. 2 as $\tilde{h}_p(t)$. There is $\tilde{h}_p(t) = h_p(t)$ for $\forall p \in V_P, t > 0$.

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

$$\tilde{h}_p(t) = \text{Updater}(\hat{h}_p(t-1), \{h_s(t), s \in B(t)\})$$

$$= \hat{h}_p(t-1) + \sum_{s \in \mathcal{N}(p) \cap B(t)} \frac{h_s(t) - M[s]}{|\mathcal{N}(p)|}$$

$$= \sum_{s \in \mathcal{N}(p) \setminus B(t)} \frac{M[s]}{|\mathcal{N}(p)|} + \sum_{s \in \mathcal{N}(p) \cap B(t)} \frac{h_s(t) - M[s]}{|\mathcal{N}(p)|}$$

$$= \tilde{h}_p(t)$$.  

Besides, because $\hat{h}_p(0) = \tilde{h}_p(0)$ for $\forall p \in V_P$ holds as initialized in Alg. 1, we have $h_p(t) = \hat{h}_p(t)$ for $\forall p \in V_P, t > 0$.  


