Reviewing Labels: Label Graph Network with *Top-k* Prediction Set for Relation Extraction

Bo Li[1,2], **Wei Ye**[1*], **Jinglei Zhang**[1,2], **Shikun Zhang**[1*]
[1]National Engineering Research Center for Software Engineering, Peking University
[2]School of Software and Microelectronics, Peking University
deepblue.lb@stu.pku.edu.cn, wye@pku.edu.cn,
jinglei.zhang@stu.pku.edu.cn, zhangsk@pku.edu.cn

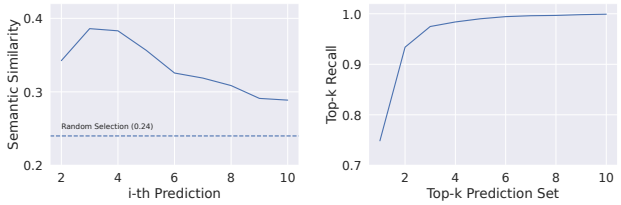
Abstract

The typical way for relation extraction is fine-tuning large pre-trained language models on task-specific datasets, then selecting the label with the highest probability of the output distribution as the final prediction. However, the usage of the *Top-k* prediction set for a given sample is commonly overlooked. In this paper, we first reveal that the *Top-k* prediction set of a given sample contains useful information for predicting the correct label. To effectively utilizes the *Top-k* prediction set, we propose **L**abel **G**raph Network with ***Top-k*** Prediction Set, termed as **KLG**. Specifically, for a given sample, we build a label graph to review candidate labels in the *Top-k* prediction set and learn the connections between them. We also design a dynamic k-selection mechanism to learn more powerful and discriminative relation representation. Our experiments show that **KLG** achieves the best performances on three relation extraction datasets. Moreover, we observe that **KLG** is more effective in dealing with long-tailed classes.

1 Introduction

Relation extraction (RE) is a fundamental natural language processing task with various downstream applications. RE aims to classify the relation between two target entities under a given input. By incorporating powerful pre-trained language models (PLMs) (Devlin et al. 2019; Liu et al. 2019), researchers achieve impressive performances on most of relation extraction datasets (Ye et al. 2019; Yamada et al. 2020; Zhou and Chen 2021; Lyu and Chen 2021; Roy and Pan 2021).

Generally, these approaches first fine-tune a PLM on downstream datasets, then get the prediction based on the output probability distribution of the classification layer. Selecting the label with the highest probability as the final prediction is a default solution, however, the *Top-k* prediction set—*Top-k* predictions with the highest probability—is largely overlooked. We argue that the *Top-k* prediction set contains some information that may be useful for the prediction. Specifically, given a well-trained PLM and a sample, if the prediction is wrong, the *Top-k* prediction set may contain the ground truth label. We can review labels in it and correct the prediction. Even if the prediction is correct, we can still establish the label connections between the ground truth and the other labels in the *Top-k* prediction set, and further refine the relation representation.



(a) (b)

Figure 1: The statistical information of the *Top-k* prediction sets from a well-trained relation extraction model. (a) shows the average semantic similarity between the ground truth label and its i-th prediction. (b) shows the recall of different *Top-k* prediction set. We can observe that labels existing in the *Top-k* prediction set has strong connections with the corresponding ground truth label. Besides, the *Top-6* prediction set achieves a recall score of more than 0.99, meaning that most ground truth labels exist in *Top-6* prediction sets.

We first conduct a pilot experiment to explore whether *Top-k* prediction sets contain meaningful information. We first train a relation extraction model using RoBERTa-large (Liu et al. 2019) on the TACRED (Zhang et al. 2017) dataset, and then use this well-trained model to obtain the *Top-k* prediction set of each test sample. The statistical information is visualized in Figure 1. We report the average semantic similarity between the ground truth label and the i-th label in the corresponding *Top-k* prediction set.[1] For each example, we also compute the semantic similarity between its ground truth and a randomly selected relation as the lower bound, denoted as random selection. We find that labels in the *Top-k* prediction set have strong connections with its ground truth label, while random selection shows much lower similarity. In Figure 1 (b), we show the recall of the *Top-k* prediction set with different k. We can easily observe that the recall grows fast as the k increases, e.g., *Top-1* recall is only around

[*] Corresponding author.

[1]We use SBERT(Reimers and Gurevych 2019) to compute the cosine similarity between relation names, since relation names in TACRED are meaningful phases.

0.75, and *Top-6* recall is already larger than 0.99. The above results verify our assumption that the *Top-k* prediction set contains available information, and we may benefit relation extraction if it can be used properly.

Based on the above observations, we propose **L**abel **G**raph Network with ***Top-K*** Prediction Set (**KLG**), which could utilize the *Top-k* prediction set to improve the performance on relation extraction task. We first fine-tune a PLM on downstream datasets with a standard supervised learning paradigm. This PLM then automatically generates the *Top-k* prediction set for each sample. These *Top-k* prediction sets will be treated as supplementary information to train **KLG**. Specifically, **KLG** first builds a label graph, which sets all pre-defined labels as nodes, and extracts the adjacent information from the *Top-k* prediction set. In this way, our model could review candidate labels existing in the *Top-k* prediction set and learn the connections between them.

However, the performance maybe sensitive to a fixed k. Performing a grid search is time-expensive and hinders model generalization. Therefore, we considered bypassing the determination of k and letting the model learn by itself from multiple possible *Top-k* prediction sets (or multiple label graphs). To make the model more robust and learn diverse information from the *Top-k* prediction set, we design a dynamic k-selection mechanism. For a given sample and its *Top-k* prediction set, we randomly select different *Top-*$\hat{k}$ predictions to form various label graphs. Then a graph contrastive learning loss is used to guide the representations of the same sample from different graphs to be close (positive examples), while the representations of different samples to be apart from each other (negative examples). Our dynamic k-selection mechanism makes **KLG** more robust and further achieves better performances.

To verify the effectiveness of **KLG**, we conduct extensive experiments on three different relation extraction datasets. We find that **KLG** brings significant improvements over strong baseline models and achieves better results than previous state-of-the-art methods (§3.3). Besides, we also explore why does **KLG** work, and the experiments show that **KLG** has a strong ability to improve the performance on long-tailed classes (§4.1).

To sum up, our contributions are as follows:

1. We show that labels existing in the *Top-k* prediction set has strong connections with the corresponding ground truth label, which is largely ignored by previous works.

2. We propose **L**abel **G**raph Network with **Top-K** Prediction Set (**KLG**), which contains a label graph and the dynamic k-selection mechanism. **KLG** could effectively leverage useful information in the *Top-k* prediction set. We conduct extensive experiments and achieve new state-of-the-art performances on three relation extraction datasets.

3. We also explore why does **KLG** work and we find that **KLG** is particularly good at handling long-tailed classes.

2 Our Approach

2.1 *Top-k* Prediction Set Generation

In the first step, we need to generate the *Top-k* prediction set for each sample. In this research, we use a pre-trained PLM to train a baseline model M and obtain the *Top-k* prediction sets. Specifically, for a given input sentence s and two target entities e_1 and e_2, we first use an entity marker to inclose the entities, and add the entity type information before each target entity. The above two markers highlight the entities and are demonstrated as crucial for relation extraction.[2] Then we use the max-pooling of the entity spans as the entity representations $\mathbf{e}_1$ and $\mathbf{e}_2$, where $\mathbf{e}_1, \mathbf{e}_2 \in \mathbb{R}^d$. d is the hidden size of the PLM's output, e.g., 1024 for the RoBERTa-large model. After concatenating the representations of the head entity and the tail entity, we use a single layer to obtain the relation representation $\mathbf{r} \in \mathbb{R}^d$.

$$\mathbf{r} = MLP([\mathbf{e}_1||\mathbf{e}_2]), \tag{1}$$

where $||$ denotes vector concatenation. Finally, a classification layer with a softmax activation function is used to output the probability distribution of all pre-defined labels.

After we finish the training of baseline model M, we use M to predict all samples in the training, dev, and test sets.[3] For each sample, we generate its *Top-k* prediction set depending on the probability distribution of the classification layer's output, denoted as *s(Top-k)*.

2.2 Label Graph Fusion

In this subsection, we will fine-tune a new PLM by introducing a label graph to leverage the *s(Top-k)* effectively. Typically, a graph $\mathcal{G}$ should have a node set $\mathcal{V}$ and an edge set $\mathcal{E}$, denoted as $\mathcal{G}(\mathcal{V}, \mathcal{E})$. The edge links a pair of nodes, denoted as the $(u, v) \in \mathcal{E}$, where $u \in \mathcal{V}$ and $v \in \mathcal{V}$, and all the graphs are undirected, i.e., $(u, v) \in \mathcal{E} \leftrightarrow (v, u) \in \mathcal{E}$. The number of nodes is $N = |V|$. For the node representation, we have a feature matrix $\mathbf{X} \in \mathbb{R}^{N*d}$, where d is the feature dimension. For convenience, we set this dimension the same as the hidden size of the PLM's output in §2.1. To represent the edge set $\mathcal{E}$, researchers usually introduce an adjacency matrix $A \in \mathbb{R}^{N*N}$, where $A_{i,j} = 1$ if $(u, v) \in \mathcal{E}$ and $A_{i,j} = 0$ otherwise.

Suppose the current downstream task has N pre-defined labels, denoted as $c_1, c_2, ..., c_N$. We first initialize the label graph with a feature matrix $\mathbf{X} \in \mathbb{R}^{N*d}$, and each node representation $\mathbf{h}_i$ corresponds to a pre-defined label c_i. Where $\mathbf{X}$ is random initialized and will be optimized during training. Then for each sample, we construct the edge set $\mathcal{E}$ based on its *s(Top-k)*. We set labels in the *s(Top-k)* are connected, including the self-loop. Specifically, for the node u and node v, the edge (u, v) is computed as follows:

$$(u, v) = \begin{cases} 1, c_u, c_v \in \textit{s(Top-k)} \\ 0, otherwise, \end{cases} \tag{2}$$

[2] We only use the entity marker if the entity type information is unavailable, such as the SemEval2010 dataset.

[3] We use the checkpoint that achieves the best results on the dev set as the final baseline model.

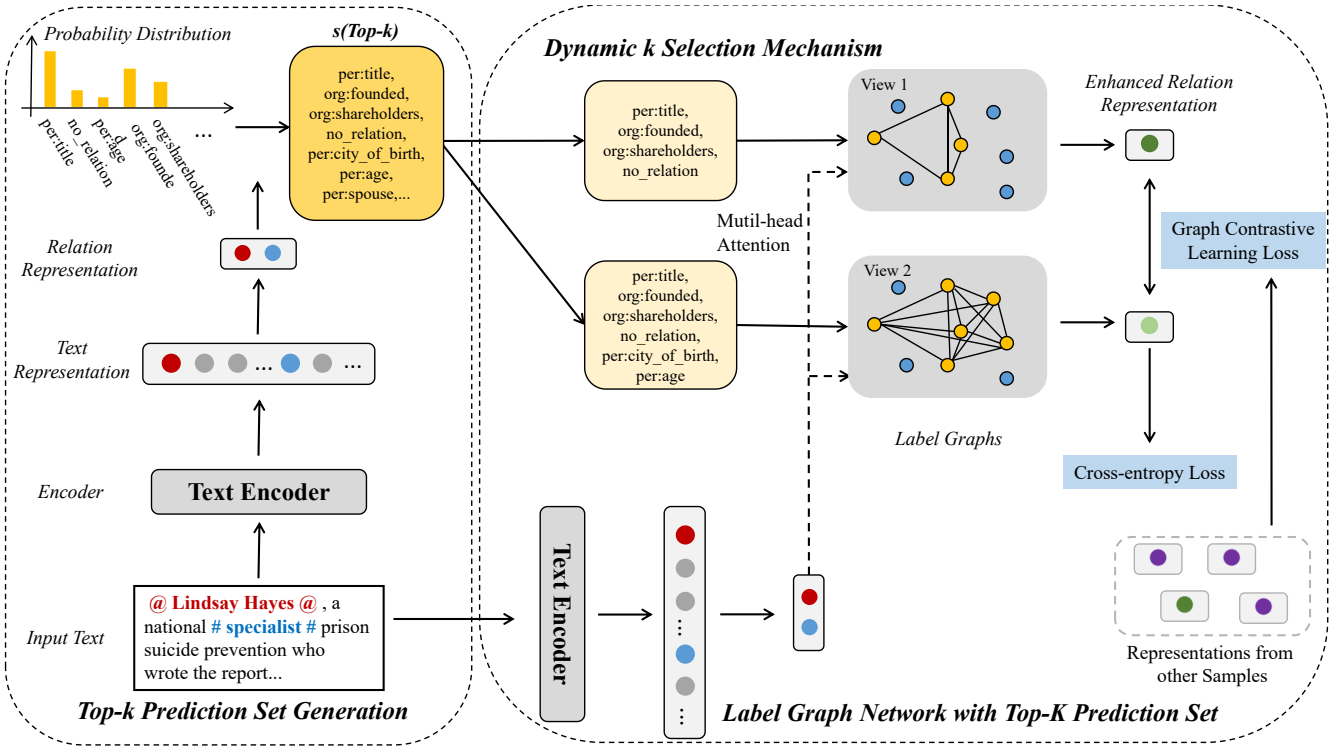


Figure 2: Illustration of our proposed method KLG, the example is taken from TACRED training set. Left: the procedure of *Top-k* prediction set generation. This is a classical supervised learning paradigm. For each sample, instead of only outputting the label with the highest probability, we extract the *Top-k* prediction set depending on the probability distribution. Right: the overall architecture of KLG. We first use pre-defined labels and *s(Top-k)* to build a label graph. Then for each sample, we design a dynamic k-selection mechanism to create diverse *Top-k* prediction sets and label graphs, together with graph contrastive learning. KLG could achieve more powerful relation representations. Better viewed in color.

For a given *s(Top-k)*, the label graph has $\frac{k(k+1)}{2}$ edges. It is worth noticing that if the ground truth label is not in the *s(Top-k)*, then these edges will not link with the ground truth label in the graph. As we can see in Figure 1, there are still very few samples that are not recalled under *s(Top-10)*.

Then we use graph attention network (GAT) (Velickovic et al. 2018) to process the label graph $\mathcal{G}$. A scoring function $e : \mathbb{R}^d * \mathbb{R}^d \Rightarrow \mathbb{R}$ computes a score for every edge (u, v), where $A_{i,j} = 1$. $e(\mathbf{h}_u, \mathbf{h}_v)$ indicates the importance of the features of the neighbor v to the node u:

$$e(\mathbf{h}_u, \mathbf{h}_v) = LeakyReLU(a^T \cdot [\boldsymbol{W}\mathbf{h}_u || \boldsymbol{W}\mathbf{h}_v]), \quad (3)$$

where $a \in \mathbb{R}^{2d'}$m $\boldsymbol{W} \in \mathbb{R}^{d \times d'}$ are learned parameters, we set $d^{'} = d$ in this research. After we obtain attention scores across all neighbors of node u, a softmax function is used to normalize these scores:

$$\alpha(u, v) = \frac{exp(e(\mathbf{h}_u, \mathbf{h}_v))}{\sum_{c_{v'} \in s(Top\text{-}k)} exp(e(\mathbf{h}_u, \mathbf{h}_{v'}))} \quad (4)$$

Then we compute a weighted average of transformed features of the neighbor node using the normalized attention coefficient as the new node representation:

$$\mathbf{h}_u^{'} = \sigma(\sum_{c_v \in s(Top\text{-}k)} \alpha(u, v) \cdot \boldsymbol{W}\mathbf{h}_v) \quad (5)$$

Then we use a mutil-head attention layer (Vaswani et al. 2017) to aggregate the relation representation $\mathbf{r}$ and those k node representations $\mathbf{X}_k$, and $\mathbf{r}$ is the query vector:

$$\hat{\mathbf{r}} = MultiHeadAtt(\mathbf{r}, \mathbf{X}_k, \mathbf{X}_k). \quad (6)$$

Where $\mathbf{X}_k$ contains k node representations corresponding to labels in *s(Top-k)*. $\hat{\mathbf{r}}$ is the enhanced relation representation that learns label connections from the label graph. Finally, we concatenate the two relation representations and use an MLP layer with a softmax activation function to obtain the final prediction:

$$p = MLP([\mathbf{r}||\hat{\mathbf{r}}]) \quad (7)$$

We use a standard cross-entropy loss to optimize our model, and the loss is denoted as $\mathcal{L}^{CE}$.

2.3 Dynamic k-Selection Mechanism

While leveraging the label graph could achieve consistent improvements, one crucial problem is determining a proper k in *s(Top-k)*. In §2.2, we select the k where the *Top-k* recall

achieves 0.99 on the dev set. However, the performance may sensitive to a fixed k and result in an unstable output. In this subsection, we propose a **D**ynamic k **S**election Mechanism (DS), which could create diverse *Top-k* prediction sets for each sample, and further leverages graph contrastive learning for better relation representations. To be specific, we first choose the value of k that the *Top-k* recall achieves 0.99 on the dev set. Then we randomly choose another $k^{'}$ from the following uniform distribution:

$$k^{'} \in [\lceil \frac{1}{2}k \rceil, \lceil \frac{3}{2}k \rceil]. \tag{8}$$

Where $\lceil * \rceil$ means round up, and equation 8 ensures that we could select diverse k at each training step.[4]

After applying the dynamic k-selection mechanism, we can build multiple label graphs using different *Top-k* prediction sets. In our research, we use *s(Top-k)* and *s(Top-k$^{'}$)* to construct two different label graphs for a given sample, denoted as $\mathcal{G}$ and $\mathcal{G}^{'}$. To fully use these label graphs and achieve more powerful relation representations, we design a graph contrastive learning loss. Specifically, for a given sample x_i, we first compute two relation representations $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{r}}_i^{'}$ through Equation 6. These two relation representations are generated from the specific sample x_i, thus $\hat{\mathbf{r}}_i^{'}$ can be seen as the augmentation version of $\hat{\mathbf{r}}_i$. We treat $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{r}}_i^{'}$ as a positive pair. Additionally, in a mini-batch, samples that hold the same label with x_i also can be seen as positives for $\hat{\mathbf{r}}_i$, and samples that hold different labels are treated as negative samples. Then for a mini-batch with N training examples, denoted as $(x_i, y_i)_{i=1}^{N}$, we have $2N$ relation representations, where the last half examples of the batch are the augmented views of the first half, and they share the same labels. And $i \in I = [2N]$ is the index of an arbitrary relation representations. Our graph contrastive learning loss is defined as follows:

$$\mathcal{L}^{GCL} = \sum_{i \in I} \frac{-1}{P(i)} \sum_{j \in P(i)} \mathcal{L}_{i,j}^{GCL} \tag{9}$$

$$\mathcal{L}_{i,j}^{GCL} = log \frac{exp(\hat{\mathbf{r}}_i^{T} \cdot \hat{\mathbf{r}}_j / \tau)}{exp(\hat{\mathbf{r}}_i^{T} \cdot \hat{\mathbf{r}}_j / \tau) + \sum_{q \in N(i)} exp(\hat{\mathbf{r}}_i^{T} \cdot \hat{\mathbf{r}}_q / \tau)} \tag{10}$$

Here $P(i)$ is the positive example set for $\hat{\mathbf{r}}_i$, and $N(i)$ is the negative example set for $\hat{\mathbf{r}}_i$. τ is a scalar temperature parameter. Through graph contrastive learning, **KLG** could effectively utilize diverse label graphs and learn more powerful relation representations.

Finally, the overall training loss for **KLG** is shown as follows:

$$\mathcal{L} = \alpha \mathcal{L}^{CE} + (1 - \alpha)\mathcal{L}^{GCL}, \tag{11}$$

where α is a loss weighting factor.

[4]For example, if k = 5, we will choose k from [3, 8] to build different label graphs

3 Experiments

3.1 Setting

Dataset and Metrics. We evaluate **KLG** on TACRED (Zhang et al. 2017), TACRED-Revisit (Alt, Gabryszak, and Hennig 2020), and SemEval2010 (Hendrickx et al. 2010), three commonly used relation extraction datasets. Due to the space limitation, we present detailed information about these datasets in the Appendix C. Following previous work, we use the Micro-F1 score excluding 'No Relation' as the metric. Besides, we also report the precision, recall, and Macro-F1 score in our detailed analysis.

Model Details. We use Pytorch (Paszke et al. 2019) and RoBERTa-large as the text encoder for both parts of **KLG**. The checkpoint can be downloaded here. [5] The batch size is 16, and the optimizer is AdamW (Loshchilov and Hutter 2019) with a 1e-5 learning rate and a warm-up strategy. The maximum training epoch is 10, and the maximum input length is 256. The scalar temperature parameter τ is 0.05, and the loss weighting factor α is 0.9. We set k as the *Top-k* recall achieves 0.99 on the dev set. The checkpoint that achieves the best result on the dev set is used for testing. [6] We conduct each experiment three times and report the average result to reduce the randomness. Besides, we also report the performances using different backbone networks. Please refer Appendix A for more details.

3.2 Comparison Methods

We compare **KLG** with state-of-the-art RE systems that represent a diverse array of approaches.

Classification-based Methods. These methods fine-tune PLMs on RE datasets with a standard classification loss. **LUKE** (Yamada et al. 2020) is a popular model for various information extraction tasks, it is pre-trained with large number of external weakly supervised data. **IRE** (Zhou and Chen 2021) is a strong model which uses entity type marker and achieves impressive performances. **RECENT-SpanBERT** (Lyu and Chen 2021) designs multi-task learning for better usage of entity type information.

Sequence-to-sequence Methods. These methods use text generation models such as BART (Lewis et al. 2020) and T5 (Raffel et al. 2020) for RE. **REBEL** (Cabot and Navigli 2021) leverages a BART-large model and is pre-trained on a huge external corpus for RE. **TANL**(Paolini et al. 2021) reformulates RE as a translation task by utilizing the powerful T5 model. Since the above two methods do not leverage entity type information in their original settings. To have a fair comparison, we re-implement these methods with official code and add the entity type information. We also directly treat relation names as generation objectives and fine-tune a **BART** model on RE datasets.

Prompt-based Methods. This line of research uses prompt and treats RE as a cloze-style task. **PTR** (Han et al. 2021) and **KnowPrompt** (Chen et al. 2022) use different

[5]https://huggingface.co/roberta-large

[6]Please refer to Appendix B for the detailed information of parameter searching.

prompt and answer word engineering for RE. While **NLI-DeBERTa** (Sainz et al. 2021) reformulates relation extraction as an entailment task.

Besides, we also design several other models for comparison. **Base Model** is the basic network that used for generating the *Top-k* prediction set in §2.1. Based on the **Base Model**, we design two improved methods that utilize the *Top-k* prediction set in different ways. **Base Model(P)** adds a **P**rompt P after the input text, where P is: Choose a relation from *s(Top-k)* for e_1 and e_2. **Base Model(LS)** uses **L**abel **S**moothing (Ioffe and Szegedy 2015) and modifies the one-hot label as the soft label.

3.3 Main Results

Table 1 shows the performances on three RE datasets, and we can have the following observations.

KLG outperforms all previous state-of-the-art methods on three RE datasets, including pushing the TACRED F1-score to 75.6%, TACRED-Revisit F1-score to 84.1%, and SemEval2010 F1-score to 90.5%. Compared with the most popular classification-based methods, **KLG** shows favorable results without any external dataset usage or additional pre-training stages. Although the newly emerging methods offer desirable performances with larger PLMs, such as T5 and DeBERTa, **KLG** still suppresses them by a large margin. The above results verify the effectiveness of **KLG**. By utilizing *s(Top-k)* via a label graph and the dynamic k-selection mechanism, **KLG** could fully use the available information existing in the *Top-k* prediction set and achieve better performance.

Although we apply different technologies to utilize the *Top-k* prediction set, the performances of Base Model(P) and Base Model(LS) are undesirable. Base Model(P) shows that directly injecting the *Top-k* prediction set into the input text can not achieve better performance as we expected, even with the help of a prompt. We think there are two possible reasons for the above phenomenon: 1) the backbone network can not understand the semantic meaning of relation names, and appending the *Top-k* prediction set after the input text may confuse the model's encoder, making the model pays less attention to the original input text. 2) With sufficient training samples, models may learn to select a relation name from the *Top-k* prediction set directly without considering the original input text. As for Base Model(LS), it obtains slight improvements compared with its corresponding baseline model. Meanwhile, **KLG** achieves much better performances than the above methods.

We also report the ablation study on two crucial components of **KLG**: 1) the usage of label graph with *Top-k* prediction set, and 2) the dynamic k-selection mechanism. The results show that both parts bring notable improvements over the Base Model. With the help of the label graph, our model could review candidate labels existing in the *top-k* prediction set and learn useful information from them. In addition, after being equipped with the dynamic k-selection mechanism, **KLG** learns more powerful relation representations and further improves its performance.

	TACRED	TACRED-Rev	SemEval
Classification-based Methods			
LUKE	72.7	80.8	90.1
IRE	74.6	83.2	89.8
RECENT	75.2	83.4	-
Sequence-to-Sequence Methods			
BART	72.7	81.5	89.5
REBEL	73.7	-	-
TANL	74.8	-	-
Prompt-based Methods			
PTR	72.4	81.4	89.9
KnowPrompt	72.4	82.4	90.2
NLI-DeBERTa	73.9	-	-
Ours			
Base Model	74.3	83.1	89.6
Base Model(P)	74.1	82.7	89.1
Base Model(LS)	74.6	83.5	89.9
KLG	**75.6(+0.4)**	**84.1(+0.7)**	**90.5(+0.3)**
KLG w/o DS	75.0	83.6	90.1

Table 1: Micro-F1 score of test sets on three relation extraction datasets. Results are all cited from published papers or re-implemented using official code. For each dataset, underlined indicates previous state-of-the-art, and bold indicates the best model performance. Results of our methods are averaged over three random seeds, and the results are statistically significant with $p < 0.05$.

3.4 The Sensitivity of k

In this section, we explore the sensitivity of k. These results also verify the robustness of our dynamic k-selection mechanism (DS). We present the performance on TACRED test set with DS and without DS in Figure 3. We find that without DS, the performances are very sensitive to the k-selection, especially when k is too small or too large. With the help of DS, **KLG** could achieve more stable and much better performances across a wide range of k.

4 Analysis

4.1 Why Does KLG Work?

As **KLG** could learn label connections from the *Top-k* prediction set and reconsider these labels, which may particularly benefit long-tailed classes. We want to explore why does **KLG** work from the perspective of long-tailed classes. We report fine-grained experimental results on the test set of TACRED. After filtering out the 'No Relation' class, we choose 10 classes with the highest frequency as head classes, and 10 other classes with the lowest frequency are long-tailed classes. The proportion of head classes samples is 69.6%, and only 3.6% for long-tailed classes. We report the precision, recall, and Micro-F1 score on both head and long-tailed classes. As the Macro-F1 score is more suitable for the imbalanced scenario, we also report the Macro-F1 score on the whole dataset. We can draw following conclusions from Table 3.

Input Sentence	Ground Truth	Base Model	Top-6 Prediction Set	KLG
PDA's advisory board includes seven members of congress and ..., Thom Hartmann rev. lennox yearwood	org:top_members/ employees	org:shareholders	org:shareholders, no_relation, **org:top_members/employees**, org:founded_by, org:parents, org:subsidiaries	org:top_members/ employees
She testified that she had been forced to sign ..., FBI price and wood, under threats of ...	per:employee_of	no_relation	no_relation, **per:employee_of**, per:schools_attended, per:cities_of_residence, per:other_family, per:countries_of_residence	per:employee_of
He was later flown back to Manila, where he reunited with family and friends .	per:cities_of_ residence	per:city_of_birth	per:city_of_birth, no_relation, per:countries_of_residence, **per:cities_of_residence**, per:city_of_death, per:religion	per:cities_of_ residence

Table 2: Case Study. All examples are extracted from the test set of TACRED. Underline denotes the target entity, and Bold denotes the ground truth label existing in the *Top-6* prediction set.

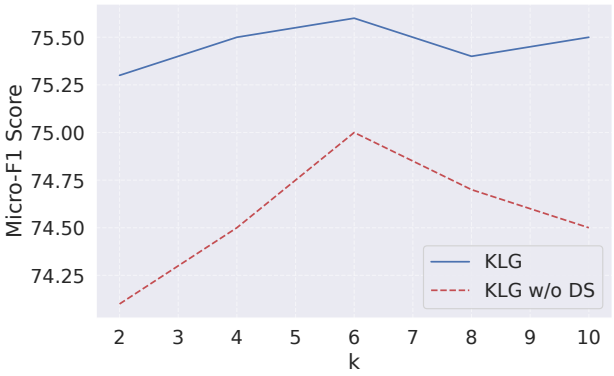


Figure 3: The effectiveness of k. We find that the fixed k results in unstable performances. With the help of the dynamic k-selection mechanism, KLG could achieve desirable results across a wide range of k.

		Base Model	KLG
Head Classes (69.6%)	**Precision**	76.0	78.8(+2.8)
	Recall	79.7	77.3(-2.4)
	Micro-F1	77.8	78.1(+0.3)
Tail Classes (3.6%)	**Precision**	74.4	75.3(+0.9)
	Recall	69.6	74.7(+5.1)
	Micro-F1	71.9	75.0(+3.1)
All Classes	**Micro-F1**	74.3	75.6(+1.3)
	Macro-F1	61.2	65.2(+4.0)

Table 3: Performances of TACRED test set on head classes, tail classes, and all classes. To have a clear picture, we report the precision, recall, Micro-F1 score, and Macro-F1 score. After filtering out the 'No Relation' class, we choose 10 classes with the highest frequency as head classes, and 10 other classes with the lowest frequency are long-tailed classes.

Both models have undesirable performances on the long-tailed classes than on the head classes. This indicates that dealing with long-tailed classes is more challenging. As for the performance gap between these two types of classes, **KLG** is much smaller than Base Model, i.e., -3.1% v.s. -5.9%. Compared with the Base Model, **KLG** achieves a remarkable improvement in terms of Macro-F1 score, i.e., 4.0 point absolute improvement. The above results show that **KLG** is particularly good at handling imbalanced scenario.

Considering the performances of long-tailed classes, the improvement of **KLG** over the Base Model is more higher on long-tailed classes than on head classes (+3.1% v.s. +0.3%). Furthermore, **KLG** dramatically improves the recall from 69.6% to 74.7%, with around 5.1 points absolute improvement. We attribute this to the usage of the *Top-k* prediction set and the dynamic k-selection mechanism. As the *Top-k* prediction set may contain candidate long-tailed classes which are ignored by previous methods, **KLG** could review these labels and finally benefit the performance of long-tailed classes. These results show that the performance improvements are mainly from the long-tailed classes.

As for the head classes, although the Micro-F1 score has slight improvement, we find that the precision improves over 2.8%, since **KLG** mitigates the trend of false positive on the head classes by paying more attention to the long-tailed classes. Besides, we also observe the recall of head classes has a significant degradation. One possible reason is that **KLG** tends to classify head classes as long-tailed classes, resulting in a relatively low recall of head classes.

We also conduct visualization to verify the above conclusion intuitively. Previous works (Kang et al. 2020) show that the $l2$ norm of the classifier weight vector is positively correlated with the number of training samples of the corresponding label. Having a high weight norm for a class means that the classifier tends to output a high logit score for this class, so a good classifier should have essentially the same vector weights. Figure 4 shows the weight norm of the Base Model and **KLG**. The number of samples is reduced from the left (head classes) to the right (long-tailed classes). We can observe that the weight norms of the Base Model are sig-

Figure 4: The norm of the classifier weight vector for each class. The class indexes are sorted by the number of samples (head to tail). We exclude the 'No Relation' to better view the trend. We can observe that KLG has similar vector norms across all classes.

nificantly different for different classes. On the other hand, **KLG** has a more balanced weight norm compared with the Base Model, thus is more friendly for long-tailed classes and finally achieves remarkable improvements.

4.2 Case Study

To have a clear picture of the effectiveness of **KLG**, we present several cases in Table 2. From the first and third cases, we can see that although the Base Model outputs the wrong prediction, the prediction is related to the ground truth label. For example, in the first case, the ground truth is *org:top_membersemployees*, while the label with the highest probability is *org:shareholders*, which has a similar semantic meaning with *org:top_membersemployees*. We find the *Top-6* prediction set contains the ground truth label, but this information was ignored by previous works. As for the second case, although Base Model gives *per:employee_of* a high probability, it finally outputs *'no_relation'* as the final prediction. Since the majority of examples in the training set are labeled as *'no_relation'*, Base Model tends to output a high logit score for *'no_relation'* class. **KLG** successfully outputs correct predictions in all examples by reviewing candidate labels and learning useful information from the *Top-k* prediction set.

5 Related Work

Relation extraction is a well-studied and popular task in natural language processing. Early works leveraged machine learning to tackle relation extraction, and these methods can be divided into two categories (Kambhatla 2004; Boschee, Weischedel, and Zamanian 2005; Zhou et al. 2007): feature-based methods and kernel-based methods. Feature-based methods heavily rely on handcraft features. These features are domain-specific and data-specific. Kernel-based methods need existing NLP tools to transform an input sentence into a parse tree. However, the NLP tools may make mistakes during processing, which may negatively affect the model's performance.

Then deep learning methods have been the mainstream solutions for RE. Liu et al. (2013) first adopt deep learning for RE via convolutional neural network. Zeng et al. (2015) proposed a model named Piecewise Convolutional Neural Networks(PCNN). Also, there are lots of works modified convolutional neural network and recurrent neural network for RE (Santos, Xiang, and Zhou 2015; Miwa and Bansal 2016). Recently, transformer-based methods which leverage PLMs have shown impressive performances on RE (Yamada et al. 2020; Xue et al. 2021; Li et al. 2020, 2021; Roy and Pan 2021; Zhou and Chen 2021; Lyu and Chen 2021). These methods used the PLMs such as BERT and Roberta as backbone network, and designed novel components or multi-task learning framework for better performance.

Prompt-based methods (Radford et al. 2019) were drawn some attention in recent research as well. PTR (Han et al. 2021) and KnowPrompt (Chen et al. 2022) used a human-designed prompt with different label verbalizers for each relation. While NLI-DeBERTa (Sainz et al. 2021) reformulated relation extraction as an entailment task with relation descriptions. Sequence-to-sequence methods are another interesting research direction for RE. REBEL (Cabot and Navigli 2021) focused on joint entity and relation extraction and pre-trained a BART-large model with external datasets. TANL (Paolini et al. 2021) used T5 and backbone network and achieved impressive performance on various information extraction tasks.

In this research, we focus on leveraging the *top-k* prediction set for RE, which is not explored in past work.

6 Conclusion

In this paper, we first reveal that the *Top-k* prediction set of a given sample contains helpful information when dealing with relation extraction. Then we propose **L**abel **G**raph Network with ***Top-k*** Prediction Set (**KLG**), a new model that fully utilizes the *Top-k* prediction set by building a label graph neural network with the dynamic k-selection mechanism. By digging the potentially useful information from the *Top-k* prediction set and reviewing these labels, **KLG** achieves new state-of-the-art results on three relation extraction datasets. Besides, **KLG** is particularly good at handling long-tailed classes. We hope this research could inspire researchers to further effectively explore the usage of the *Top-k* prediction set.

Acknowledgements

This research is supported by the National Key Research And Development Program of China (No. 2021YFC3340101).

References

Alt, C.; Gabryszak, A.; and Hennig, L. 2020. TACRED Revisited: A Thorough Evaluation of the TACRED Relation Extraction Task. In *ACL*.

Boschee, E.; Weischedel, R.; and Zamanian, A. 2005. Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*.

Cabot, P. H.; and Navigli, R. 2021. REBEL: Relation Extraction By End-to-end Language generation. In *Findings of EMNLP*.

Chen, X.; Zhang, N.; Xie, X.; Deng, S.; Yao, Y.; Tan, C.; Huang, F.; Si, L.; and Chen, H. 2022. KnowPrompt: Knowledge-aware Prompt-tuning with Synergistic Optimization for Relation Extraction. In *WWW*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Han, X.; Zhao, W.; Ding, N.; Liu, Z.; and Sun, M. 2021. PTR: Prompt Tuning with Rules for Text Classification. *CoRR*, abs/2105.11259.

Hendrickx, I.; Kim, S. N.; Kozareva, Z.; Nakov, P.; Séaghdha, D. Ó.; Padó, S.; Pennacchiotti, M.; Romano, L.; and Szpakowicz, S. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *SemEval@ACL*.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*.

Kambhatla, N. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*.

Kang, B.; Xie, S.; Rohrbach, M.; Yan, Z.; Gordo, A.; Feng, J.; and Kalantidis, Y. 2020. Decoupling Representation and Classifier for Long-Tailed Recognition. In *ICLR*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.

Li, B.; Ye, W.; Huang, C.; and Zhang, S. 2021. Multi-view Inference for Relation Extraction with Uncertain Knowledge. In *AAAI*.

Li, B.; Ye, W.; Sheng, Z.; Xie, R.; Xi, X.; and Zhang, S. 2020. Graph Enhanced Dual Attention Network for Document-Level Relation Extraction. In *COLING*.

Liu, C.; Sun, W.; Chao, W.; and Che, W. 2013. Convolution neural network for relation extraction. In *International Conference on Advanced Data Mining and Applications*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR*, abs/1907.11692.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *ICLR*.

Lyu, S.; and Chen, H. 2021. Relation Classification with Entity Type Restriction. In *Findings of ACL/IJCNLP 2021*.

Miwa, M.; and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Paolini, G.; Athiwaratkun, B.; Krone, J.; Ma, J.; Achille, A.; Anubhai, R.; dos Santos, C. N.; Xiang, B.; and Soatto, S. 2021. Structured Prediction as Translation between Augmented Natural Languages. In *ICLR*.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E. Z.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.*

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP-IJCNLP*.

Roy, A.; and Pan, S. 2021. Incorporating medical knowledge in BERT for clinical relation extraction. In *EMNLP*.

Sainz, O.; de Lacalle, O. L.; Labaka, G.; Barrena, A.; and Agirre, E. 2021. Label Verbalization and Entailment for Effective Zero and Few-Shot Relation Extraction. In *EMNLP*.

Santos, C. N. d.; Xiang, B.; and Zhou, B. 2015. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR*.

Xue, F.; Sun, A.; Zhang, H.; and Chng, E. S. 2021. GDPNet: Refining Latent Multi-View Graph for Relation Extraction. In *AAAI*.

Yamada, I.; Asai, A.; Shindo, H.; Takeda, H.; and Matsumoto, Y. 2020. LUKE: Deep Contextualized Entity Representations with Entity-aware Self-attention. In *EMNLP*.

Ye, W.; Li, B.; Xie, R.; Sheng, Z.; Chen, L.; and Zhang, S. 2019. Exploiting Entity BIO Tag Embeddings and Multitask Learning for Relation Extraction with Imbalanced Data. In *ACL*.

Zeng, D.; Liu, K.; Chen, Y.; and Zhao, J. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Zhang, Y.; Zhong, V.; Chen, D.; Angeli, G.; and Manning, C. D. 2017. Position-aware Attention and Supervised Data Improve Slot Filling. In *EMNLP*.

Zhou, G.; Zhang, M.; Ji, D.; and Zhu, Q. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *EMNLP-CoNLL*.

Zhou, W.; and Chen, M. 2021. An Improved Baseline for Sentence-level Relation Extraction. *CoRR*, abs/2102.01373.