Neural Snowball for Few-Shot Relation Learning

Tianyu Gao,[1] **Xu Han,**[1] **Ruobing Xie,**[2] **Zhiyuan Liu,**[1*] **Fen Lin,**[2] **Leyu Lin,**[2] **Maosong Sun**[1]

[1]Department of Computer Science and Technology, Tsinghua University, Beijing, China
Institute for Artificial Intelligence, Tsinghua University, Beijing, China
Beijing National Research Center for Information Science and Technology
[2]Search Product Center, WeChat Search Application Department, Tencent, China
{gty16, hanxu17}@mails.tsinghua.edu.cn

Abstract

Knowledge graphs typically undergo open-ended growth of new relations. This cannot be well handled by relation extraction that focuses on pre-defined relations with sufficient training data. To address new relations with few-shot instances, we propose a novel bootstrapping approach, Neural Snowball, to learn new relations by transferring semantic knowledge about existing relations. More specifically, we use Relational Siamese Networks (RSN) to learn the metric of relational similarities between instances based on existing relations and their labeled data. Afterwards, given a new relation and its few-shot instances, we use RSN to accumulate reliable instances from unlabeled corpora; these instances are used to train a relation classifier, which can further identify new facts of the new relation. The process is conducted iteratively like a snowball. Experiments show that our model can gather high-quality instances for better few-shot relation learning and achieves significant improvement compared to baselines. Codes and datasets are released on https://github.com/thunlp/Neural-Snowball.

Introduction

Knowledge graphs (KGs) such as WordNet (Miller 1995), Freebase (Bollacker et al. 2008) and Wikidata (Vrandečić and Krötzsch 2014) have multiple applications in information retrieval, question answering and recommender systems. Such KGs consist of relation facts with triplet format (e_h, r, e_t) representing a relation r between entities e_h and e_t. Though existing KGs have acquired large amounts of facts, they still have huge growth space compared to real-world data. To enrich KGs, relation extraction (RE) is investigated to extract relation facts from plain text.

One challenge of RE is that novel relations emerge rapidly in KGs, yet most RE models cannot handle those new relations well since they rely on RE datasets with only a limited number of predefined relations. One of the largest RE dataset, FewRel (Han et al. 2018), only has 100 relations, yet there were already 920 relations in Wikidata in 2014 (Vrandečić and Krötzsch 2014), let alone it contains nearly 6,000 relations now.
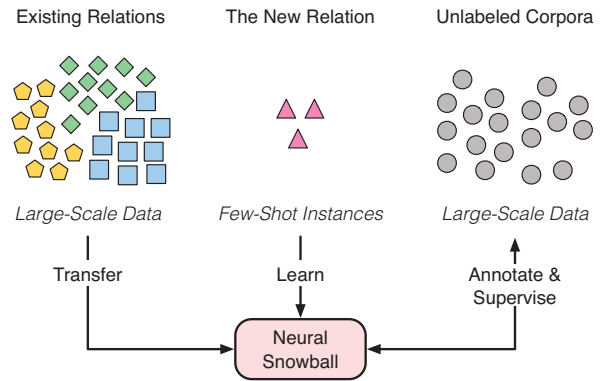
Figure 1: An illustration of how Neural Snowball utilizes three different kinds of data to learn new relations.

To extract relation facts of novel relations, many existing approaches have studied bootstrapping RE, which extracts triplets for a new relation with few seed relation facts. Brin (1998) proposes to extract author-book facts with a small set of (author, book) pairs as input. It iteratively finds mentions of seed pairs from the web, and then extracts sentence patterns from those mentions and finds new pairs by pattern matching. Agichtein and Gravano (2000) further improve this method and name it as Snowball, for that relation facts and their mentions accumulate like a snowball.

However, most existing bootstrapping models confine themselves to only utilize seed relation facts and fail to take advantage of available large-scale labeled datasets, which have been proved to be a valuable resource. Though data of existing relations might have a very different distribution with new relations, it still can be used to train a deep learning model that extracts abstract features at the higher levels of the representation, suiting both historical and unseen relations (Bengio 2012). This technique, named as transfer learning, has been widely adopted in image few-shot tasks. Previous work has investigated transferring metrics (Koch, Zemel, and Salakhutdinov 2015) to measure similarities between objects and meta-information (Ravi and Larochelle 2017) to fast adapt to new tasks.

Based on bootstrapping and transfer learning, we present **Neural Snowball** for learning to classify new relations with insufficient training data. Given seed instances with relation facts of a new relation, Neural Snowball finds reliable mentions of these facts. Then they are used to train a relation classifier, which aims at discovering reliable instances with new relation facts. These instances then serve as the inputs of the new iteration.

We also apply Relational Siamese Networks (RSN) to select high-confidence new instances. Siamese networks (Bromley et al. 1994) usually contain dual encoders and measure similarities between two objects by learning a metric. Wu et al. (2019) designed RSN, utilizing neural siamese networks to determine whether two sentences express the same relation. In conventional bootstrapping systems, patterns are used to select new instances. Since neural networks bring better generalization than patterns, we use RSN to select high-confidence new instances by comparing candidates with existing ones.

Experiment results show that Neural Snowball achieves significant improvements on learning novel relations in few-shot scenarios. Further experiments demonstrate the efficiency of Relational Siamese Networks and the snowball process, proving that they have the ability to select high-quality instances and extract new relation facts.

To conclude, our main contributions are threefold:

- We propose Neural Snowball, a novel approach to better train neural relation classifiers with only a handful of instances for new relations, by iteratively accumulating novel instances and facts from unlabeled data with prior knowledge of existing relations.

- For better selecting new supporting instances for new relations, we investigate Relational Siamese Networks (RSN) to measure relational similarities between candidate instances and existing ones.

- Experiment results and further analysis show the efficiency and robustness of our models.

Related Work

Supervised RE Early work for fully-supervised RE uses kernel methods (Zelenko, Aone, and Richardella 2003) and embedding methods (Gormley, Yu, and Dredze 2015) to leverage syntactic information to predict relations. Recently, neural models like RNN and CNN have been proposed to extract better features from word sequences (Socher et al. 2012; Zeng et al. 2014). Besides, dependency parsing trees have also been proved to be efficient in RE (Xu et al. 2015; Liu et al. 2015).

Distant Supervision Supervised RE methods rely on hand-labeled corpora, which usually cover only a limited number of relations and instances. Mintz et al. (2009) propose distant supervision to automatically generate relation labels by aligning entities between corpora and KGs. To alleviate wrong labeling, Riedel, Yao, and McCallum (2010) and Hoffmann et al. (2011) model distant supervision as a multi-instance multi-label task.

RE for New Relations Bootstrapping RE can fast adapt to new relations with a small set of seed facts or sentences. Brin (1998) first proposes to extract relation facts by iterative pattern expansion from web. Agichtein and Gravano (2000) propose Snowball to improve such iterative mechanism with better pattern extraction and evaluation methods. Based on that, Zhu et al. (2009) adopt statistical methods for better pattern selection. Batista, Martins, and Silva (2015) use word embeddings to further improve Snowball. Many similar bootstrapping ideas have been widely explored for RE (Pantel and Pennacchiotti 2006; Rozenfeld and Feldman 2008; Nakashole, Theobald, and Weikum 2011).

Compared to distant supervision, bootstrapping expands relation facts iteratively, leading to higher precision. Moreover, distant supervision is still limited to predefined relations, yet bootstrapping is scalable for open-ended relation growth. Many other semi-supervised methods can also be adopted for RE (Rosenberg, Hebert, and Schneiderman 2005; French, Mackiewicz, and Fisher 2017; Lin et al. 2019), yet they still require sufficient annotations and mainly aim at classifying predefined relations rather than discovering new ones. Thus, we do not further discuss these methods.

Inspired by the fact that people can grasp new knowledge with few samples, few-shot learning to solve data deficiency appeals to researchers. The key point of few-shot learning is to transfer task-agnostic information from existing data to new tasks (Bengio 2012). Vinyals et al. (2016), Snell, Swersky, and Zemel (2017) and Zhang et al. (2018) explore learning a distance distribution to classify new classes in a nearest-neighbour-style strategy. Ravi and Larochelle (2017), Munkhdalai and Yu (2017) and Finn, Abbeel, and Levine (2017) propose meta-learning to understand how to fast optimize models with few samples. Qiao et al. (2018) propose learning to predict parameters for classifiers of new tasks. Existing few-shot learning models mainly focus on vision tasks. For exploiting it on text, Han et al. (2018) release FewRel, a large-scale few-shot RE dataset.

OpenRE Both bootstrapping and few-shot learning handle new tasks with minimal human participation. Open relation extraction (OpenRE), on the other hand, aims at extracting relations from text without predefined types. One kind of OpenRE systems focuses on finding relation mentions (Banko et al. 2007), while others exploit to form relation types automatically by clustering semantic patterns (Shinyama and Sekine 2006; Yao et al. 2011; ElSahar et al. 2017). It is a different and challengeable view on RE compared to conventional methods and remains to be explored.

Siamese Networks Siamese networks measure similarities between two objects with dual encoders and trainable distance functions (Bromley et al. 1994). They are exploited for one/few-shot learning (Koch, Zemel, and Salakhutdinov 2015) and measuring text similarities (Mueller and Thyagarajan 2016). Wu et al. (2019) propose Relational Siamese Networks (RSN) to learn a relational metric between given instances. Here we use RSN to select high-confidence instances by comparing candidates with existing ones.
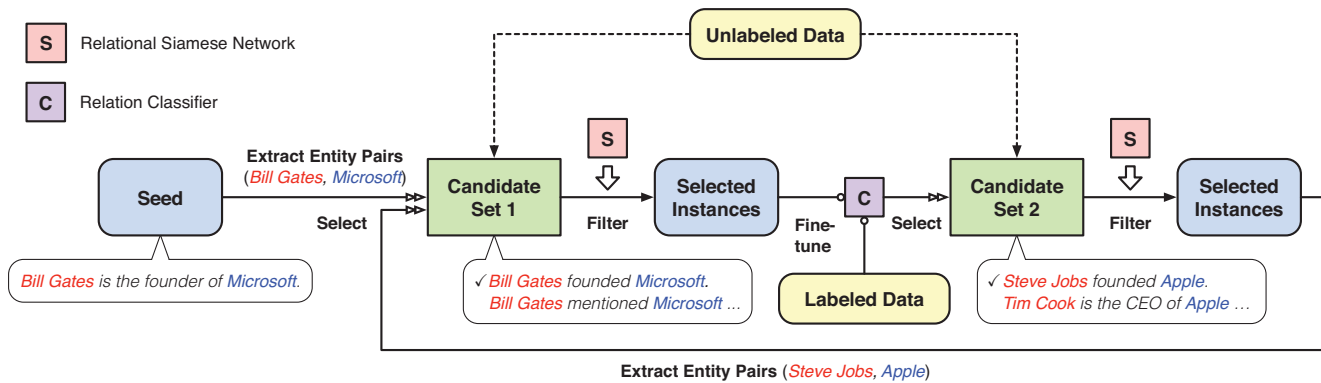
Figure 2: The framework of Neural Snowball with examples of the relation *founder*. Candidate set 1 ($\mathcal{C}_1$) contains all instances that have the same entity pairs as extracted. Candidate set 2 ($\mathcal{C}_2$) consists of high-confidence instances selected by the relation classifier. Instances in both candidate sets are filtered by RSN and then added to the selected instance set $\mathcal{S}_r$ of the relation r.

Methodology

In this section, we will introduce Neural Snowball, starting with notations and definitions.

Terminology and Problem Definition

Given an instance x containing a word sequence $\{w_1, w_2, ..., w_l\}$ with tagged entities e_h and e_t, RE aims at predicting the relation label r between e_h and e_t. **Relation mentions** are instances expressing given relations. **Entity pair mentions** are instances with given entity pairs. **Relation facts** are triplets (e_h, r, e_t) indicating there is a relation r between e_h and e_t. x^r indicates x is a relation mention of the relation r.

Since we emphasize learning to extract a new relation in a real-world scenario, we adopt a different problem setting from existing supervised RE or few-shot RE. Given a large-scale labeled dataset for existing relations and a small set of instances for the new relation, our goal is to extract instances of the new relation from a query set containing instances of existing relations, the new relation and unseen relations.

Inputs of this task contain a large-scale labeled corpus $\mathcal{S}_N = \{x_j^{r_i} | r_i \in \mathcal{R}_N\}$ where $\mathcal{R}_N$ is a predefined relation set, an unlabeled corpus $\mathcal{T}$ and a seed set $\mathcal{S}_r$ with k instances for the new relation r. We firstly pre-train the neural modules on $\mathcal{S}_N$. Then for the new relation r, we train a binary classifier g. To be more specific, given an instance x, $g(x)$ outputs the probability that x expresses the relation r. During the test phase, the classifier g performs classification on a query set $\mathcal{Q}$ containing instances expressing predefined relations in $\mathcal{R}_N$, instances with the new relation r and some instances of other unseen relations, which is a simulation of the real-world scenario.

Neural Snowball Process

Neural Snowball gathers reliable instances for a new relation r iteratively with a small seed set $\mathcal{S}_r$ as the input. In each iteration, $\mathcal{S}_r$ will be extended with selected unlabeled instances, and the new $\mathcal{S}_r$ becomes the input of the next iteration. Figure 2 illustrates the framework of Neural Snowball. When a new relation arrives with its initial instances, Neural Snowball shall process as follows,

Input The seed instance set $\mathcal{S}_r$ for the relation r.

Phase 1 Structure the entity pair set,

$$\mathcal{E} = \{(e_h, e_t) | \text{Ent}(x) = (e_h, e_t), x \in \mathcal{S}_r\}, \quad (1)$$

where $\text{Ent}(x)$ means the entity pair of the instance x. Then, we get the candidate set $\mathcal{C}_1$ from the corpus $\mathcal{T}$ with

$$\mathcal{C}_1 = \{x | \text{Ent}(x) \in \mathcal{E}, x \in \mathcal{T}\}. \quad (2)$$

Since those instances in $\mathcal{C}_1$ share same entity pairs with those in $\mathcal{S}_r$, we believe that they are likely to express the relation r. Yet to further alleviate false positive instances, for each x in $\mathcal{C}_1$, we pair it with all instances $x' \in \mathcal{S}_r$ that share the same entity pair with x, and use the Relational Siamese Network (RSN) to get similarity scores. Averaging those scores we will get a confidence score of x, noted as $\text{score}_1(x)$.

Then, we sort instances in $\mathcal{C}_1$ in decreasing order of confidence scores and pick the top-K_1 instances as new ones added to $\mathcal{S}_r$. Since there exists the circumstance that less than K_1 instances really belong to the relation, we add an external condition that instances with confidence scores less than a threshold α will be excluded.

After all these steps, we have acquired new instances for the relation r with high confidence. With the expanded instance set $\mathcal{S}_r$, we can fine-tune the relation classifier g as described later, for the classifier is needed in the next step.

Phase 2 In the last phase, we expand $\mathcal{S}_r$, yet the entity pair set remains the same. So in this phase, our goal is to discover instances with new entity pairs for the relation r.

We construct the candidate set for this phase by using the relation classifier g,

$$\mathcal{C}_2 = \{x | g(x) > \theta, x \in \mathcal{T}\}, \tag{3}$$

where θ is a confidence threshold. Then each candidate instance x is paired with each x' in $\mathcal{S}_r$ as input of RSN, and the confidence score $\texttt{score}_2(x)$ is the mean of all the similairy scores of those pairs. Instances having top-K_2 scores and with $\texttt{score}_2$ larger than threshold β are added to $\mathcal{S}_r$.

After one iteration of the process, we go back to phase 1, and another round starts. As the system runs, the instance set $\mathcal{S}_r$ grows bigger and the performance of the classifier increases until it reaches the peak. Best choices of the number of iterations and parameters mentioned above are discussed in the experiment section.

Neural Modules

Neural Snowball contains two key components: (1) the **Relational Siamese Network (RSN)**, which aims at selecting high-quality instances from unlabeled data by measuring similarities between candidate instances and existing ones, and (2) the **Relation Classifier**, which classifies whether an instance belongs to the new relation.

Relational Siamese Network (RSN) $s(x, y)$ It takes two instances as input and outputs a value between 0 and 1 indicating the probability that those two instances share the same relation type. Figure 3 shows the structure of our proposed Relational Siamese Network, which consists of two encoders f_s sharing parameters and a distant function. With instances as input, those encoders output the representation vectors for them. Then we compute the similarity score between the two instances with the following formula,

$$s(x, y) = \sigma\big(\mathbf{w}_s^T (f_s(x) - f_s(y))^2 + b_s\big), \tag{4}$$

where the square notation refers to squaring each dimension of the vector instead of the dot production of the vector, and $\sigma(\cdot)$ refers to sigmoid function. This distance function can be considered as a weighted L2 distance with trainable weights $\mathbf{w}_s$ and bias b_s. A higher score indicates a higher possibility that the two sentences express the same relation ($\mathbf{w}_s$ will be negative to make this possible).

Relation Classifier $g(x)$ The classifier is composed of a neural encoder f, which transfers the raw instance x into a real-valued vector, and a linear layer with parameters $\mathbf{w}$ and b to get the probability that the input instance belongs to a relation r. It can be described by the following expression,

$$g(x) = \sigma\big(\mathbf{w}^T f(x) + b\big), \tag{5}$$

where $g(x)$ is the output probability and $\sigma(\cdot)$ is sigmoid function to constrain the output between 0 and 1. Note that it is a binary classifier so $g(x)$ is just one real value, instead of a vector in the N-way classification scenario.

The reason to set it as a binary classifier instead of training an N-way classifier and utilizing softmax to constrain the outputs is that real-world relation extraction systems need to
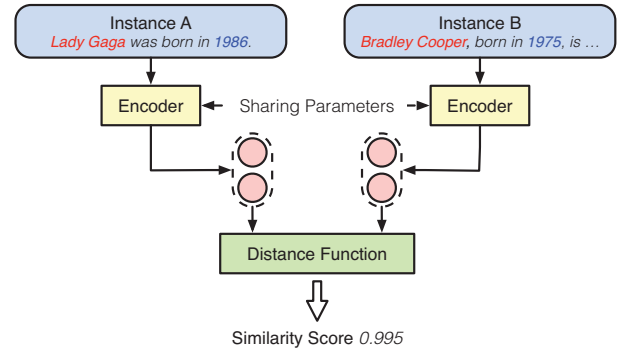

Figure 3: The architecture of Relational Siamese Network (RSN). The encoders produce the representations of instances, and then RSN measures the similarity between them with certain distance function.

deal with negative samples, which express unknown relations and occupy a large proportion in corpora. These negative representations are not clusterable and considering them as "one class" is inappropriate. Another reason is that by using binary classifiers, we can handle the emergence of new relations by adding new classifiers, while the N-way classifier has to be retrained and data unbalance may lead to worse results for both new and existing relations.

With N binary classifiers, we can do N-way classification by comparing the output of each classifier, and the one with the highest probability wins. When no output exceeds a certain threshold, the sentence will be regarded as "negative", which means it does not express any of the existing relations.

Pre-training and Fine-tuning   To measure instance similarities on a new relation and to fast adapt the classifier to a new task, we need to pre-train the two neural modules. With the existing labeled dataset $\mathcal{S}_N$, we can perform a supervised N-way classification to pre-train the hidden representations of the classifier. As for RSN, we randomly sample instance pairs with the same or different relations from $\mathcal{S}_N$ and train the model with a cross entropy loss.

When given a new relation r with its $\mathcal{S}_r$, the parameters for the whole RSN and the encoder of the relation classifier are fixed, since they have already learned to extract generic features during pre-training. Further fine-tuning those parts with a small number of data might bring noise and bias to the distribution of the parameters.

Then we optimize the linear layer parameters $\mathbf{w}$ and b in the classifier by sampling minibatches from $\mathcal{S}_r$ as positive samples and from $\mathcal{S}_N$ as negative samples. Denoting the positive batch as $\mathcal{S}_b$ and the negative batch as $\mathcal{T}_b$, the loss is as follows,

$$\mathcal{L}_{\mathcal{S}_b, \mathcal{T}_b}(g_{\mathbf{w}, b}) = \sum_{x \in \mathcal{S}_b} \log g_{\mathbf{w}, b}(x) \\ + \mu \sum_{x \in \mathcal{T}_b} \log(1 - g_{\mathbf{w}, b}(x)) \tag{6}$$

where μ is a coefficient of the negative sampling loss. Though for each batch we can sample positive and negative set with the same size, the actual numbers of positive instances and negative instances for the new relation differ a lot (a few versus thousands). So it is necessary to give the negative part of loss a smaller weight.

With the sampling strategy and loss function, we can do gradient-based optimization on parameters $\mathbf{w}$ and b. Here we choose Adam (Kingma and Ba 2015) as our optimizer. The hyperparameters include the number of training epochs e, batch size bs, learning rate λ and coefficient of negative sampling loss μ. Algorithm 1 describes the process.

The fine-tuning process is used as one of our baselines. We also adopt this algorithm in each step of Neural Snowball after gathering new instances in $\mathcal{S}_r$. Though it is a simple way to acquire $\mathbf{w}$ and b, it is better than metric-based few-shot algorithms for that it is more adaptive to new relations while metric-based models usually fix all the parameters during few-shot, and it is more scalable to a large number of training instances. Negative sampling also enables the model to improve the precision of extracting new relation.

Neural Encoders

As mentioned above, encoders are parts of our RSN and classifiers and aim at extracting abstract and generic features from raw sentences and tagged entities. In this paper, we adopt two encoders: CNN (Nguyen and Grishman 2015) and BERT (Devlin et al. 2019).

CNN We follow the model structure in Nguyen and Grishman (2015) for our CNN encoder. The model takes word embeddings and position embeddings (Zeng et al. 2014) as input. The embedding sequence is then fed into a one-dim convolutional neural network to extract features. Then those features are max-pooled to get one real-valued vector as the instance representation.

BERT Devlin et al. (2019) propose a novel language model named BERT, which stands for Bidirectional Encoder Representations from Transformers, and has obtained new state-of-the-arts on several NLP tasks, far beyond existing CNN or RNN models. BERT takes tokens of the sentence as input and after several attention layers outputs hidden features for each token. To fit the RE task, we add special marks at the beginning of the sequence and before and after the entities. Note that marks at the beginning, around the head entities and tail entities are different. Then, we take the hidden features of the first token as the sentence representation.

Experiments

In this section, we will show that the relation classifiers trained with our Neural Snowball mechanism achieve significant improvements compared to baselines in our few-shot relation learning settings. We also carry out two quantitative evaluations to further prove the effectiveness of Relational Siamese Networks and the snowball process.

Algorithm 1: Fine-tuning the Classifier

Input: New instance set $\mathcal{S}_r$, historical relation dataset $\mathcal{S}_N$

Result: Optimized $\mathbf{w}$ and b

1 Randomly initialize $\mathbf{w}$ and b
2 **for** $i \leftarrow 1$ **to** e **do**
3   // Get a sequence of minibatches from $\mathcal{S}_r$
4   $\mathcal{S}_{batch\_seq} \leftarrow$ batch_seq($\mathcal{S}_r$,bs)
5   **for** $\mathcal{S}_b \in \mathcal{S}_{batch\_seq}$ **do**
6   // Sample the negative batch
7    $\mathcal{T}_b \leftarrow$ sample($\mathcal{S}_N$,bs)
8   Update $\mathbf{w}$ and b w.r.t. $\mathcal{L}_{\mathcal{S}_b,\mathcal{T}_b}(g_{\mathbf{w},b})$
9   with learning rate λ
10   **end**
11 **end**

Datasets and Evaluation Settings

Our experiment setting requires a dataset with precise human annotations, large amount of data and also it needs to be easy to perform distant supervision on. For now the only qualified dataset is FewRel (Han et al. 2018). It contains 100 relations and 70,000 instances from Wikipedia. The dataset is divided into three subsets: training set (64 relations), validation set (16 relations) and test set (20 relations). We also dump an unlabeled corpus from Wikipedia with tagged entities, including 899,996 instances and 464,218 entity pairs, which is used for the snowball process.

Our main experiment follows the setting in previous sections. First we further split the training set into training set A and B. We use the training set A as $\mathcal{S}_N$, and for each step of evaluation, we sample one relation as the new relation r and k instances of it as $\mathcal{S}_r$ from val/test set, and sample a query set $\mathcal{Q}$ from both training set B and val/test set. Then the models classify all the query instances in a binary manner, judging whether each instance mentions the new relation r. Note that the sampled query set includes N relations with sufficient training data, one relation r with few instances and many other unseen relations. It is a very challengeable setting and closer to the real-world applications compared to N-way K-shot few-shot (sampling N classes and classifying inside the N classes), since corpora in the real world are not limited to certain relation numbers or types.

Parameter Settings

We tune our hyperparameters on the validation set. For parameters of the encoders, we follow (Han et al. 2018) for CNN and (Devlin et al. 2019) for BERT. For the fine-tuning, after grid searching, we adopt training epochs $e = 50$, batch size $bs = 10$, learning rate $\lambda = 0.05$ and negative loss coefficient $\mu = 0.2$. BERT fine-tuning shares the same parameters except for $\lambda = 0.01$ and $\mu = 0.5$.

For the Neural Snowball process, we also determine our parameters by grid searching. We set K_1 and K_2, the numbers of added instances for each stage, as 5, and the thresholds of RSN for each stage, α and β, as 0.5. We adopt 0.9 for the classifier threshold θ.

Model	5 Seed Instances			10 Seed Instances			15 Seed Instances		
	P	R	F1	P	R	F1	P	R	F1
BREDS	33.71	11.89	17.58	28.29	17.02	21.25	25.24	17.96	20.99
Fine-tuning (CNN)	46.90	9.08	15.22	47.58	38.36	42.48	74.70	48.03	58.46
Relational Siamese Network (CNN)	45.00	31.37	36.96	46.42	30.68	36.94	49.32	30.46	37.66
Distant Supervision (CNN)	44.99	31.06	36.75	42.48	48.64	45.35	43.70	54.76	48.60
Neural Snowball (CNN)	48.07	36.21	41.30	47.28	51.49	49.30	68.25	58.90	63.23
Fine-tuning (BERT)	50.85	16.66	25.10	59.87	55.19	57.43	**81.60**	58.92	68.43
Relational Siamese Network (BERT)	39.07	**51.39**	44.47	42.42	54.93	47.87	44.10	52.73	48.03
Distant Supervision (BERT)	38.06	51.18	43.66	38.45	**76.12**	51.09	35.48	**80.33**	49.22
Neural Snowball (BERT)	**56.87**	40.43	**47.26**	**60.50**	62.20	**61.34**	78.13	66.87	**72.06**

Table 1: Experiment results on our few-shot relation learning settings with different size of seed sets. Here P refers to precision, R refers to recall and F1 refers to F1-measure score.

All the models evaluated in our experiments output a probability of being the mention of the new relation for each query instance, and to get the predicting results we need to set a confidence threshold. For fine-tuning and Neural Snowball we set the threshold as 0.5, and 0.7 for the Relational Siamese Network.

Few-Shot Relation Learning

Table 1 shows the experiment results on our few-shot relation learning tasks. We evaluate five model architectures: **BREDS** (Batista, Martins, and Silva 2015) is an advanced version of the original snowball (Agichtein and Gravano 2000), which uses word embeddings for pattern selection; **Fine-tuning** stands for directly using Algorithm 1 with few-shot instances to train the new classifier; **Relational Siamese Network (RSN)** refers to computing similarity scores between the query instance and each instance in $\mathcal{S}_r$, and averaging them as the probability of the query one expressing the new relation; **Distant Supervision** refers to taking all instances sharing entity pairs with given seeds into the training set and using Algorithm 1; **Neural Snowball** is our proposed method. We do not evaluate other semi-supervised and few-shot RE models for the reason that they do not suit our few-shot new relation learning settings.

From Table 1 we can identify that (1) our Neural Snowball achieves the best results in both settings and with both encoders. (2) While fine-tuning, distant supervision and Neural Snowball improve with the increase of seed numbers, BREDS and RSN have little promotion.

By further comparison between Neural Snowball and other baselines, we notice that our model largely promotes the recall values while maintaining the high precision values. It indicates that Neural Snowball not only gathers new training instances with high quality, but also successfully extracts new relation facts and patterns to widen the coverage of instances for the new relation.

Relation Set	P@5	P@10	P@20	P@50
Train	83.60	80.66	76.03	61.98
Test	82.15	78.64	72.57	55.10

Table 2: Precisions at top-N instances scored by RSN (CNN) in the 5-seed setting. "Train" and "Test" represent results on relations in the training and test sets.

Analysis on Relational Siamese Network

To examine the quality of instances selected by RSN, we randomly sample one relation and 5 instances of it and use the rest data as query instances. We use the method mentioned before to calculate a score for each query instance, then we calculate precisions at top-N instances (P@N).

We can see that RSN achieves a precision of 82.15% at top-5 instances on the test set. It is relative high considering RSN is only given a small number of instances and it even have not seen the relation before. Also note that though RSN is only trained with relations of the training set, the performance on relations in the test set has only a narrow gap, further proving the effectiveness of RSN.

Analysis on Neural Snowball Process

To further analyze the iterative process of Neural Snowball (NS), we present a quantitative evaluation on the numbers of newly-gathered instances as well as the classifier performance on relation *chairperson* with the 5-seed-instance setting. Note that it is a randomly-picked relation and other relations have shown similar trends. Due to the space limit, we only take the relation *chairperson* as an example.

Figure 4 demonstrates the development of evaluation results as the iteration grows. Here we adopt two settings: **NS setting** refers to fine-tuning the classifier with instances selected by Neural Snowball, and **random setting** refers to fine-tuning on randomly-picked instances of relation *chairperson* with the same amount of NS, under the premise of
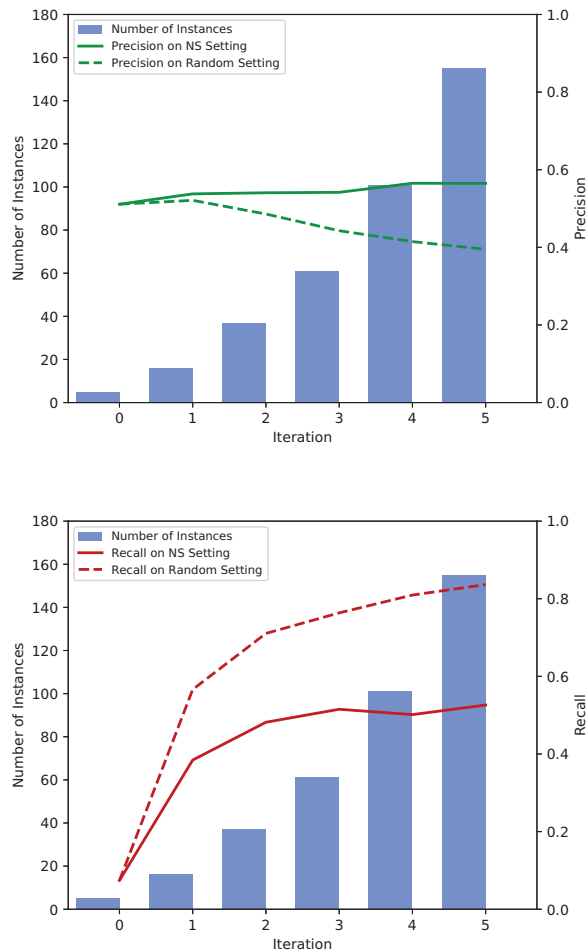
Figure 4: Evaluation results on each iteration of Neural Snowball. Blue bars are numbers of instances added. Solid lines represent performance on the NS setting, and dotted lines represent the random setting.

knowing all the instances of the relation. Note that random setting is an ideal case since it reflects the real distribution of data for the new relation and the overall performance of the random setting serves as an upper bound.

From the results of random setting, we see that the binary classifier obtains higher recall and performs a little lower in precision when trained on larger randomly-distributed data. This can be explained that more data brings more patterns in representations, improving the completeness of extracting while sacrificing a little in quality.

Then by comparing the results between the two settings, we get two observations: (1) As the number of iterations and amount of instances grow, the classifier fine-tuned on NS setting maintains higher precision than the one fine-tuned on random setting, which proves that RSN succeeds in extracting high-confidence instances and brings in high-quality patterns. (2) The recall rate of NS grows less than expected,

indicating that RSN might overfit existing patterns. To maintain high precision of the model, Neural Snowball stucks in the "comfort zone"of existing high-quality patterns and fails to jump out of the zone to discover patterns with more diversity. We plan to further investigate it in future.

Conclusion and Future Work

In this paper, we propose Neural Snowball, a novel approach that learns to classify a new relation with only a small number of instances. We use Relational Siamese Networks (RSN), which are pre-trained on historical relations to iteratively select reliable instances for the new relation from unlabeled corpora. Evaluations on a large-scale relation extraction dataset demonstrate that Neural Snowball brings significant improvement in performance of extracting new relations with few instances. Further analysis proves the effectiveness of RSN and the snowball process. In the future, we will further explore the following directions:

(1) The deficiency of our current model is that it mainly extracts patterns semantically close to the given instances, which limits the increase in recall. In the future, we will explore how to jump out of the "comfort zone" and discover instances with more diversity.

(2) For now, RSN is fixed during new relation learning and shares the same parameters across relations. This can be ameliorated by an adaptive RSN that can be further optimized given new relations and new instances. We will investigate into it and further improve the efficiency of RSN.

Acknowledgments

References

Agichtein, E., and Gravano, L. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of JCDL*, 85–94.

Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Etzioni, O. 2007. Open information extraction from the web. In *Proceedings of IJCAI*, 2670–2676.

Batista, D. S.; Martins, B.; and Silva, M. J. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of EMNLP*, 499–504.

Bengio, Y. 2012. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of the Workshop on Unsupervised and Transfer Learning of ICML*, 17–36.

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of SIGMOD*, 1247–1250.

Brin, S. 1998. Extracting patterns and relations from the world wide web. In *Proceedings of International Workshop on The World Wide Web and Databases*, 172–183.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a" siamese" time delay neural network. In *Proceedings of NIPS*, 737–744.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, 4171–4186.

ElSahar, H.; Demidova, E.; Gottschalk, S.; Gravier, C.; and Laforest, F. 2017. Unsupervised open relation extraction. In *Proceedings of ESWC*, 12–16.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of ICML*, 1126–1135.

French, G.; Mackiewicz, M.; and Fisher, M. 2017. Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*.

Gormley, M. R.; Yu, M.; and Dredze, M. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of EMNLP*, 1774–1784.

Han, X.; Zhu, H.; Yu, P.; Wang, Z.; Yao, Y.; Liu, Z.; and Sun, M. 2018. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *Proceedings of EMNLP*, 4803–4809.

Hoffmann, R.; Zhang, C.; Ling, X.; Zettlemoyer, L.; and Weld, D. S. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, 541–550.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

Koch, G.; Zemel, R.; and Salakhutdinov, R. 2015. Siamese neural networks for one-shot image recognition. In *Proceedings of the Workshop of ICML*.

Lin, H.; Yan, J.; Qu, M.; and Ren, X. 2019. Learning dual retrieval module for semi-supervised relation extraction. In *Proceedings of WWW*, 1073–1083.

Liu, Y.; Wei, F.; Li, S.; Ji, H.; Zhou, M.; and Houfeng, W. 2015. A dependency-based neural network for relation classification. In *Proceedings of ACL-IJCNLP*, 285–290.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-IJCNLP*, 1003–1011.

Mueller, J., and Thyagarajan, A. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of AAAI*.

Munkhdalai, T., and Yu, H. 2017. Meta networks. In *Proceedings of ICML*, 2554–2563.

Nakashole, N.; Theobald, M.; and Weikum, G. 2011. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of WSDM*, 227–236.

Nguyen, T. H., and Grishman, R. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of the Workshop on Vector Space Modeling for NLP*, 39–48.

Pantel, P., and Pennacchiotti, M. 2006. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of COLING/ACL*, 113–120.

Qiao, S.; Liu, C.; Shen, W.; and Yuille, A. L. 2018. Few-shot image recognition by predicting parameters from activations. In *Proceedings of CVPR*, 7229–7238.

Ravi, S., and Larochelle, H. 2017. Optimization as a model for few-shot learning. In *Proceedings of ICLR*.

Riedel, S.; Yao, L.; and McCallum, A. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, 148–163.

Rosenberg, C.; Hebert, M.; and Schneiderman, H. 2005. Semi-supervised self-training of object detection models. In *Proceedings of WACV*, 29–36.

Rozenfeld, B., and Feldman, R. 2008. Self-supervised relation extraction from the web. *KAIS* 17–33.

Shinyama, Y., and Sekine, S. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of NAACL-HLT*, 304–311.

Snell, J.; Swersky, K.; and Zemel, R. 2017. Prototypical networks for few-shot learning. In *Proceedings of NIPS*, 4077–4087.

Socher, R.; Huval, B.; Manning, C. D.; and Ng, A. Y. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, 1201–1211.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Wierstra, D.; et al. 2016. Matching networks for one shot learning. In *Proceedings of NIPS*, 3630–3638.

Vrandečić, D., and Krötzsch, M. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM* 57(10):78–85.

Wu, R.; Yao, Y.; Han, X.; Xie, R.; Liu, Z.; Lin, F.; Lin, L.; and Sun, M. 2019. Open relation extraction: Relational knowledge transfer from supervised data to unsupervised data. In *Proceedings of EMNLP-IJCNLP*, 219–228.

Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of EMNLP*, 1785–1794.

Yao, L.; Haghighi, A.; Riedel, S.; and McCallum, A. 2011. Structured relation discovery using generative models. In *Proceedings of EMNLP*, 1456–1466.

Zelenko, D.; Aone, C.; and Richardella, A. 2003. Kernel methods for relation extraction. *JMLR* 1083–1106.

Zeng, D.; Liu, K.; Lai, S.; Zhou, G.; and Zhao, J. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, 2335–2344.

Zhang, X.; Sung, F.; Qiang, Y.; Yang, Y.; and Hospedales, T. M. 2018. Deep comparison: Relation columns for few-shot learning. *arXiv preprint arXiv:1811.07100*.

Zhu, J.; Nie, Z.; Liu, X.; Zhang, B.; and Wen, J.-R. 2009. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of WWW*, 101–110.