Integrating Linguistic Knowledge to Sentence Paraphrase Generation

Zibo Lin,[1,2*] **Ziran Li,**[1,2*] **Ning Ding,**[1,2] **Hai-Tao Zheng**[1,2†]
Ying Shen,[3] **Wei Wang,**[1,2] **Cong-Zhi Zhao**[4]
[1]Department of Computer Science and Technology, Tsinghua University
[2]Tsinghua Shenzhen International Graduate School, Tsinghua University
[3]School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School
[4]Giiso Information Technology Co., Ltd
{lzb18, lizr18}@mails.tsinghua.edu.cn

Abstract

Paraphrase generation aims to rewrite a text with different words while keeping the same meaning. Previous work performs the task based solely on the given dataset while ignoring the availability of external linguistic knowledge. However, it is intuitive that a model can generate more expressive and diverse paraphrase with the help of such knowledge. To fill this gap, we propose Knowledge-Enhanced Paraphrase Network (KEPN), a transformer-based framework that can leverage external linguistic knowledge to facilitate paraphrase generation. (1) The model integrates synonym information from the external linguistic knowledge into the paraphrase generator, which is used to guide the decision on whether to generate a new word or replace it with a synonym. (2) To locate the synonym pairs more accurately, we adopt an incremental encoding scheme to incorporate position information of each synonym. Besides, a multi-task architecture is designed to help the framework jointly learn the selection of synonym pairs and the generation of expressive paraphrase. Experimental results on both English and Chinese datasets show that our method significantly outperforms the state-of-the-art approaches in terms of both automatic and human evaluation.

Introduction

Paraphrase generation is a fundamental task in natural language processing, which aims to restate a text with different words while keeping the meaning approximately the same as the original. Automatic paraphrase generation can be applied to many scenarios to promote the study of natural language processing. For example, questions answering systems are often sensitive to the way questions are asked, rephrasing questions can help people get better answers in many real-world question answering applications (Fader, Zettlemoyer, and Etzioni 2014). Additionally, paraphrases can also help diversify responses of dialogue assistants (Shah et al. 2018), augment training data (Yang et al. 2019b) and extend coverage of semantic parsers (Berant and Liang 2014).

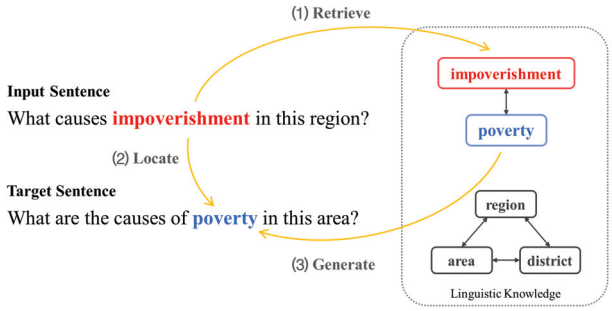

Figure 1: An example of paraphrase generation guided by linguistic knowledge.

Recently, various neural models have been put forward for automatic paraphrase generation, which modeled the task as a Seq2Seq learning problem from the original sentence to the target paraphrase (Prakash et al. 2016; Gupta et al. 2018; Li et al. 2019a). Although these methods generate fluent and grammatically correct restatements, the performances are far from perfect. Because this kind of data-driven methods can only make limited modifications to the original text such as changing word order or part of speech, which are lack of lexical and phrasal diversity.

If a model can be guided by external linguistic knowledge like thesauri, it can replace a word (especially a rare word) in the sentence with a corresponding synonym and thus generate a more complete and expressive paraphrase. For instance, as shown in Figure 1, to rewrite the input sentence "*What causes impoverishment in this region?*" to the target sentence "*What are the causes of poverty in this area?*", we need synonym pairs provided by thesauri such as (*impoverishment*, *poverty*) and (*region*, *area*). And it is worth noting that the word "*impoverishment*" in the original sentence is a low-frequent word. Without the guidance of external linguistic knowledge, this word is likely to be masked as an unknown tag in restatement.

Some prior works have been conducted to introduce external linguistic knowledge in paraphrase generation. Specifically, Cao et al. (2017) propose a Seq2Seq model with a copying decoder and equips the decoder with a paraphrase

*indicates equal contribution.
†Corresponding author: zheng.haitao@sz.tsinghua.edu.cn.

collocation table to regulate the generated words. Huang et al. (2019) adopts an extra synonym dictionary to guide the paraphrase decoder, in which a soft attention mechanism is used to learn the semantic vectors of both the original words and the synonyms. However, such models face with the following two issues. **First**, they pay little attention to the location of each paraphrase pair, which makes it difficult for decoder to make accurate use of the introduced knowledge. **Second**, they tend to copy the low-frequent words in the original sentence to alleviate the issue of out-of-vocabulary (OOV) words. This kind of copy mechanism is not in line with the intention of paraphrase generation, because the task prefers to use different words to rewrite the original text.

To address the issues mentioned above, we propose Knowledge-Enhanced Paraphrase Network (KEPN), a transformer-based framework that can leverage synonym information provided by external linguistic knowledge to facilitate paraphrase generation. (1) The model retrieves a set of synonyms of words in source sentence from external thesauri and uses a soft attention mechanism to compute the weighted sum of the synset embeddings. Then the weighted synonym representation is combined with the hidden vector of the decoder to guide the decision on whether to generate a new word or replace it with a synonym. (2) To locate synonym pairs more accurately, we adopt an incremental encoding scheme to incorporate position information of each synonym into the decoder. What's more, we design a multi-task architecture with synonym labeling as an auxiliary task. The synonym labeling task aims to identify the position of each synonym in the input sentence, which can help the model jointly learn the selection of synonym pairs and the generation of expressive paraphrase.

We conduct sets of experiments on both English and Chinese benchmark datasets for paraphrase generation. In addition, because most of the existing paraphrase datasets are derived from question matching corpus, in which sentences are all short questions, we construct a new Chinese paraphrase dataset named TCNP (Translation-based Chinese News Paraphrase) for more diversity in test domains. Experimental results on all datasets show that our model significantly outperforms state-of-the-art methods on automatic evaluation metrics with improvements of 1.0-1.9 points in BLEU. We also perform the qualitative human evaluation to show the quality of paraphrase sentence. The result indicates that the generated paraphrases are well-formed, diverse, and relevant to the input sentence. [1]

Related Work

Paraphrase generation is approached as using different words to rewrite a semantically equivalent sentence (Madnani and Dorr 2010). Feature-based methods (McKeown 1983; Bolshakov and Gelbukh 2004; Carl, Schmidt, and Schütz 2005) are widely used in paraphrase generation, but they heavily rely on the hand-crafted rules and are hard to scale up. Recent efforts involve neural methods have achieved great success, which model the task as a Seq2Seq (Sutskever, Vinyals, and Le 2014) learning problem from the original sentence to the target paraphrase. As a pioneer, Prakash et al. (2016) first explore deep learning models for paraphrase generation through a stacked LSTM network. Following by Prakash's work, Gupta et al. (2018) combine LSTM and Variational AutoEncoder (VAE) to generate multiple paraphrases. Further, Li et al. (2019a) propose a multiple encoders and decoders network to generate paraphrases at different granularity levels. However, these methods perform the task based solely on the given dataset, which ignore the availability of external linguistic knowledge.

Recently, introducing external structured knowledge into designed models has achieved great success in many studies of natural language processing (Zhou et al. 2018; Yang et al. 2019a; Li et al. 2019b). Inspired by this, various methods are proposed to use extra knowledge to improve paraphrase generation. Specifically, Cao et al. (2017) introduce a Seq2Seq model that fuses two decoders, in which the generated words are restricted in the paraphrase table of current sentence. Huang et al. (2019) propose a method to generate paraphrase in the guide of an extra dictionary and use a soft attention to learn synonym semantic vector. Moreover, Wang et al. (2019) first exploit the multi-head attention mechanism (Vaswani et al. 2017) for paraphrase generation and utilize external resources (PropBank labels) for further improvement. However, these studies pay little attention to the location of each paraphrase pair, which make it difficult to make accurate use of the introduced knowledge. By contrast, our model applies external thesauri to facilitate paraphrase generation, with an incremental encoding scheme and a multi-task architecture for better locating of synonym pairs.

Methodology

The overall architecture of the proposed KEPN is shown in Figure 2, which will be introduced from three parts: (1) **Sentence Encoder**, which captures contextual features of each word in the input sentence. (2) **Paraphrase Decoder**, which generates paraphrase with the guidance of linguistic knowledge by a soft attention mechanism. (3) **Synonym Labeling**, which plays as an auxiliary task in our designed multi-task architecture to help the decoder make better use of synonym information.

In the following subsections, we first give the definition of the task of paraphrase generation, and then introduce the three parts mentioned above in detail.

Paraphrase Definition

Given a sentence $\boldsymbol{x} = \{x_1, ..., x_n\}$, sentence paraphrase aims to generate another sentence $\boldsymbol{y} = \{y_1, ..., y_m\}$ from $\boldsymbol{x}$. Here, the lengths of $\boldsymbol{x}$ and $\boldsymbol{y}$ may not be equal. But the sentence $\boldsymbol{x}$ and $\boldsymbol{y}$ are required to have the same semantic meanings. In our work, we assume that there is an access to a corpus of linguistic knowledge $D = \{(w_i, s_i)\}_{i=1}^{N}$, which is specifically referring to the synonym table. In table D, w_i is treated as a raw word and s_i is the synonym of w_i. Our goal is to learn a paraphrase generator with the use of D to generate a paraphrase $\boldsymbol{y}$ for a sentence $\boldsymbol{x}$.

[1] Code of our model is publicly available at https://github.com/LINMouMouZiBo/KEPN

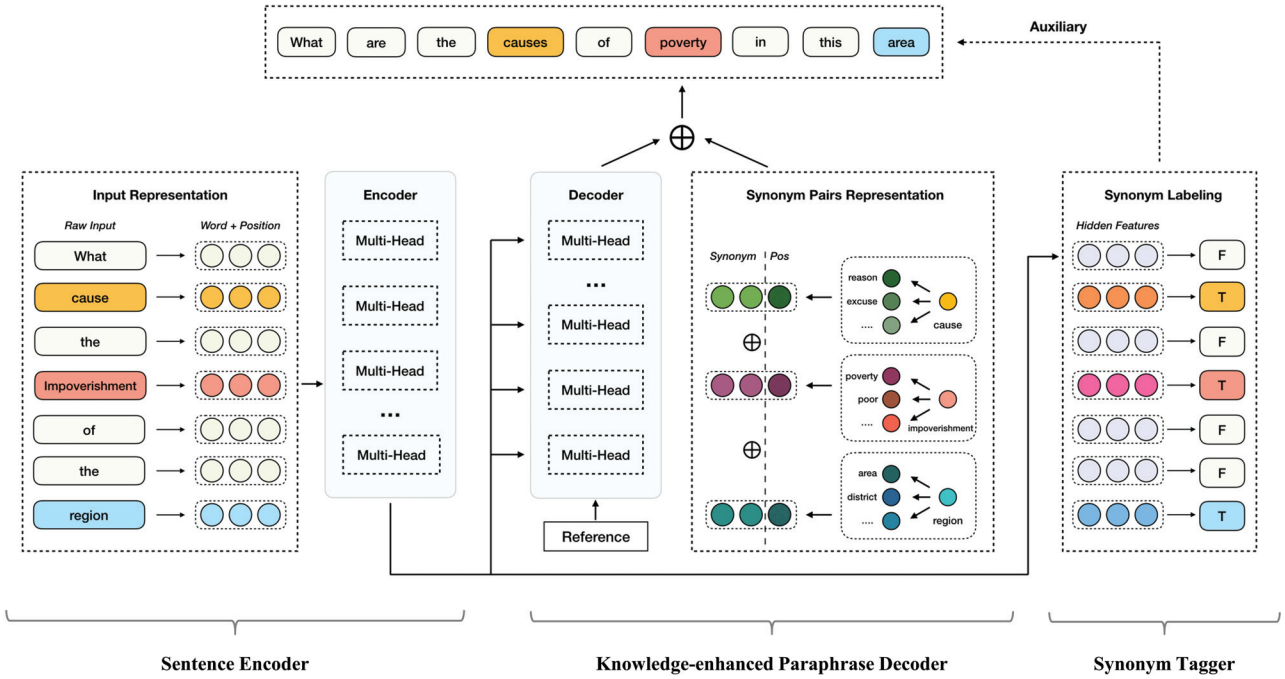

Figure 2: The overall framework of Knowledge-Enhanced Paraphrase Network (KEPN). The source sequence is fed into a Sentence Encoder, then read by a raw Transformer decoder. Finally, the output of basic decoder is combined with the context representation of synonyms to generate the target sequence.

Sentence Encoder

The Sentence Encoder first converts the input word sequence into embedding vectors and then encodes the input via multi-head attention.

Input Representation At the begin of the Sentence Encoder, the input sentence $\boldsymbol{x} = \{x_1, ..., x_n\}$ is represented as a sequence of embedding vectors $e = \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$ by looking up a word embedding matrix. The matrix is initialized with pretrained embedding and optimized as parameters during training. Apart from the word embedding, a position embedding vector $\boldsymbol{v}_i$ is introduced to encode position information of the i-th token in the sentence (Vaswani et al. 2017). The position embedding has the same dimension as the word embedding, and is formulated as:

$$\boldsymbol{v}_i[j] = sin(i/10000^{2j/d\_model}), \quad (1)$$

$$\boldsymbol{v}_i[2j+1] = cos(i/10000^{2j/d\_model}), \quad (2)$$

where i is the position where the word is indexed in sentence $\boldsymbol{x}$, j is the index of dimension and d_model is the number of dimensions.

The input vector $\boldsymbol{k}$ of our Sentence Encoder is the sum of the word embedding $\boldsymbol{e}$ and the position embedding $\boldsymbol{v}$:

$$\boldsymbol{k} = \boldsymbol{e} + \boldsymbol{v}. \quad (3)$$

Encoder With sequence embedding as input, the Sentence Encoder circumvents token-by-token encoding with a parallel encoding step that uses token position information. The encoder is composed of a stack of 6 identical blocks which are formulated as:

$$Block(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = LNorm(FFNN(\boldsymbol{m})) + \boldsymbol{m}, \quad (4)$$

$$\boldsymbol{m} = LNorm(MultiAttn(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V})) + \boldsymbol{Q}, \quad (5)$$

where $FFNN$ means a fully connected feed-forward network, and $LNorm$ stands for layer normalization. $MultiAttn$ is the crucial building part of the encoder, which allows the model to jointly attend to information from different representation subspaces at different positions. It operates on queries $\boldsymbol{Q}$, keys $\boldsymbol{K}$, and values $\boldsymbol{V}$, as follows:

$$MultiAttn(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = (\boldsymbol{h}_1 \oplus, ..., \oplus \boldsymbol{h}_i)\boldsymbol{W}, \quad (6)$$

$$\boldsymbol{h}_i = Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q; \boldsymbol{K}\boldsymbol{W}_i^K; \boldsymbol{V}\boldsymbol{W}_i^V), \quad (7)$$

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{\boldsymbol{d}_k}})\boldsymbol{V}, \quad (8)$$

where W, W_i^Q, W_i^K and W_i^V are trainable parameters.

Following the above calculating procedures, the output of the Sentence Encoder $\boldsymbol{z}$ inferring from the embedding $\boldsymbol{k}$ can be shorted as:

$$\boldsymbol{z} = \begin{cases} \boldsymbol{h}_i = Block_i(\boldsymbol{h}_{i-1}, \boldsymbol{h}_{i-1}, \boldsymbol{h}_{i-1}) & i \geq 1 \\ \boldsymbol{h}_0 = \boldsymbol{k} & i = 0 \end{cases}. \quad (9)$$

The output of encoder $\boldsymbol{z}$ is the semantic representation and fed into the decoder to drive word generation step-by-step. Furthermore, we treat the encoder as a shared module in our multi-task architecture. The output $\boldsymbol{z}$ is trained in not only the decoder but also the Synonym Labeling (More details are listed in following subsections).

Paraphrase Decoder

The input of Paraphrase Decoder is the representation vector from the Sentence Encoder, along with synonym-position pairs provided by external linguistic knowledge. Paraphrase Decoder first acts as a basic decoder of Transformer to generate a draft vector. Then a set of synonyms are retrieved and represented through a soft attention mechanism. Finally the synonym information is used to revise the draft by replacing some words with synonyms, which adds more diversity in lexical and phrasal level.

The basic decoder of Transformer is almost identical to the encoding block, with the addition of one more multi-attention layer before the feed-forward layer. The final output of the decoder $\boldsymbol{y}_t^*$ is formulated as:

$$\boldsymbol{y}_t^* = \begin{cases} \boldsymbol{h}_i = Block_i(\boldsymbol{m}, \boldsymbol{z}, \boldsymbol{z}) & i \geq 1 \\ \boldsymbol{h}_0 = \boldsymbol{y}_{t-1} & i = 0 \end{cases}, \tag{10}$$

where $\boldsymbol{m}$ is calculated by Eq. 5, in which $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ are all replaced by $\boldsymbol{h}_{i-1}$.

Synonym Retrieval In our work, synonym-position pairs are retrieved from a thesaurus and act as the linguistic knowledge to improve the diversity of paraphrase results. Given a sentence x, we first retrieve a set of synonyms $P = \{s_i\}_{i=1}^M$ from the synonym set D. To locate each synonym accurately, we also add p_i, the index of s_i which numbers the position of s_i in the sentence x, to each synonym in P. Finally, we obtain a set of synonym-position pairs $P = \{(s_i, p_i)\}_{i=1}^M$.

Two public thesauri are used in experiment: Tongyici Cilin (Extended) and WordNet. The Extended Tongyici Cilin is a Chinese synonym table that collected by HIT-SCIR, and contains 9,995 different pairs of synonyms. The WordNet, released by (Miller 1995), is a well-known lexical database of English, in which nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms.

Synonym Pairs Representation The input synonyms-position pairs $P = \{(s_i, p_i)\}_{i=1}^M$ need to be converted into vector representation before feeding into the Paraphrase Decoder. Specifically, the synonym is represented by looking up the word embedding matrix shared with the Sentence Encoder. If the synonym is a phrase, we sum the embedding of each word to get a phrase vector. For the position, we calculate the position vector $\boldsymbol{p}_i$ by Eq. 1 and Eq. 2, following the positional encoding layer of Transformer. The position vector makes a link between the synonyms and words in the input sentence, which guides the decoder to pay more attention to the location of each paraphrase pairs.

The initial output of the basic decoder is a draft vector, which is not able to generate a expressive paraphrase. Thus, we further use a soft attention mechanism to integrate the synonym-position pairs representation into the decoder. The synonym information $\boldsymbol{c}_t$ is calculated as follows:

$$\boldsymbol{c}_t = \sum_{i=1}^M \boldsymbol{a}_{i,t} \cdot \boldsymbol{s}_i \oplus \sum_{i=1}^M \boldsymbol{a}_{i,t}^* \cdot \boldsymbol{p}_i, \tag{11}$$

$$\boldsymbol{a}_{i,t} = \frac{exp(g(\boldsymbol{y}_t^*, \boldsymbol{s}_i))}{\sum_{i=1}^M exp(g(\boldsymbol{y}_t^*, \boldsymbol{s}_i))}, \tag{12}$$

$$g(\boldsymbol{y}_t^*, \boldsymbol{s}_i) = \boldsymbol{V}^\top tanh(\boldsymbol{W}[\boldsymbol{y}_t^* \oplus \boldsymbol{s}_i]), \tag{13}$$

where $\boldsymbol{V}^\top$ and $\boldsymbol{W}$ are parameters and $\oplus$ denotes concatenation. $\boldsymbol{y}_t^*$ is the output of the basic decoder from Eq. 10. $\boldsymbol{a}_{i,t}^*$ is calculated in the same way as $\boldsymbol{a}_{i,t}$ but replacing the synonym $\boldsymbol{s}_i$ by the position $\boldsymbol{p}_i$.

Finally, a softmax layer is introduced to compute probability distribution of the t-th time word:

$$\boldsymbol{y}_t = softmax(\boldsymbol{W}_y[\boldsymbol{y}_t^* \oplus \boldsymbol{c}_t]). \tag{14}$$

W_y is a parameter matrix projecting the vector to match the dimension of output vocabulary. For each step of decoder, the generation probability $\boldsymbol{y}_t$ is calculated until meeting an end symbol or reaching the maximum length of generated sentence.

The loss function of paraphrase generation is chosen to minimize the negative log-likelihood of the output generative words as:

$$loss_1 = -\sum\nolimits_t ln(p(\boldsymbol{y}_t | \boldsymbol{y}_{<t}, x, \theta)). \tag{15}$$

Synonym Labeling

It's a clear intuition that paraphrase generation benefits from locating the position where the synonyms come from in the input sentences. To utilize this feature, we propose a multi-task architecture to jointly train our network to perform synonym labeling task and sequence generation task.

The synonym labeling is a token-wise binary classification task, which aims to identify whether there are corresponding synonyms for each token in the sentence. In our work, synonym labeling and paraphrase generation share the same encoder. An incremental feed-forword network is added after the sentence encoder, which converts the output contextual vector $\boldsymbol{z}$ in Eq. 9 to the synonym labeling output $\boldsymbol{f}$:

$$\boldsymbol{f} = \sigma(\boldsymbol{W}_f \boldsymbol{z} + \boldsymbol{b}_f), \tag{16}$$

where $\boldsymbol{W}_f$ and $\boldsymbol{b}_f$ are learnable parameters and σ is the sigmoid function.

With the help of the external synonym knowledge, we automatically tag each token in the input sentence with a binary label as the ground truth, which make it possible to conduct a supervised learning for synonym labeling. The cost function of sequence labeling is chosen to minimize the negative log-likelihood of the as:

$$loss_2 = -\sum_{i=1}^{N}\sum_{j=1}^{n} \widehat{\boldsymbol{y}}_{i,j} \log \boldsymbol{f}_{i,j} + (1 - \widehat{\boldsymbol{y}}_{i,j}) \log (1 - \boldsymbol{f}_{i,j}), \tag{17}$$

where N is the size of dataset, n is the length of sentences and $\widehat{\boldsymbol{y}}$ is the ground truth labels.

Multi-Task Learning Synonym labeling is an auxiliary task that helps to better locate the position of synonyms and bind the linguistic relationship between phrase and synonyms in the original sentence. Therefore, we first independently train the encoder of our network as a synonym labeling task. Then we fine-tune the encoder with the parameters from the labeling task and jointly train the entire network

Dataset		WikiAnswers	Quora	LCQMC	TCNP
Train	Size	1.8M	140K	138K	743K
	Avg Len	7.27	9.85	6.01	22.92
Test	Size	23.6K	4K	6K	15K
	Avg Len	6.73	9.85	5.76	23.32

Table 1: Statistics of datasets.

with both labeling and generation loss. The total loss function we minimize is a linear combination of two task-specific loss functions:

$$L = \alpha * loss_1 + (1 - \alpha) * loss_2, \tag{18}$$

where the weighting coefficient α is a hyper-parameter for trade-off between $loss_1$ and $loss_2$. Given the cost function L, we use the Adam (Kingma and Ba 2015) optimizer with mini-batches to train the model parameters with warmup mechanism (Vaswani et al. 2017).

Experiment

Datasets

We carry out our experiments on three benchmark datasets, including English datasets WikiAnswers (Fader, Zettlemoyer, and Etzioni 2013) and Quora, as well as Chinese dataset LCQMC (Liu et al. 2018). These three datasets are collected from question matching corpus, in which sentences are all short questions.

For more diversity in test domains, we construct a new Chinese paraphrase dataset named TCNP (Translation-based Chinese News Paraphrase). TCNP is built by translating the English sentences in WMT2018 dataset into Chinese via Baidu and Google translation API. Compared with other three datasets, sentences in TCNP are longer and more diverse. Statistics of the datasets is shown in Table 1.

Benchmark

We compare **KEPN** with several approaches:

S2S-A (Bahdanau, Cho, and Bengio 2015) is a Seq2Seq baseline with the attention mechanism.

P-GEN (See, Liu, and Manning 2017) uses a hybrid Pointer-generator which alleviates the issue of OOV words by a copy mechanism.

VAE-SVG (Gupta et al. 2018) uses the Variational AutoEncoder(VAE) to generate paraphrases.

Transformer (Vaswani et al. 2017) has obtained the state-of-the-art performance on machine translation, which generates sentences by multi-head attention mechanism.

DGEN (Huang et al. 2019) is a RNN-based network which retrieves knowledge from PPDB (Pavlick et al. 2015) datasets to generate paraphrase.

Transformer-PB (Wang et al. 2019) is a Transformer-based network with a multi-encoder to integrate PropBank labels.

Setting

Hyper-parameters are tuned on the validation dataset. We set the trade-off parameter α to 0.9, the dropout rate to 0.3 and the learning rate of the optimizer to 1e-5. All word embeddings are initialized with 300D GloVe (Pennington, Socher, and Manning 2014) vectors. For other parameters, we follow the setting of the basic Transformer defined in (Vaswani et al. 2017).

For better comparison, we follow the data preprocessing method in previous work (Prakash et al. 2016), which truncated all sentences to 15 words for the WikiAnswers and Quora dataset, 30 words for LCQMC and TCNP.

Automatic Evaluation

For quantitative evaluation of our network, we calculate scores on widely used evaluations metrics in paraphrase generation: METEOR (Lavie and Agarwal 2007) and BLEU (Papineni et al. 2002). These scores have been shown to correlate well with human judgment.

Overall Results

Experimental results on all datasets are listed in Table 2, from which we can observe that:

(1) Our network KEPN consistently achieves the best results on all the datasets in terms of all the evaluation measures. Both Transform-PB and DGEN utilize external knowledge to generate paraphrase and have been reported the state-of-the-art results in WikiAnswers and Quora. Our network still gets obvious higher scores comparing with them. This indicates that the position encoding scheme and synonym labeling task we introduce help the network make more efficient use of the linguistic knowledge.

(2) Removing either the position vector of synonym or multi-task training results in worse performance. Specifically, after removing the position vector of synonym and only reserve the synonym embedding (i.e., KEPN$_{sub_pos}$), performance becomes obviously worse. This indicates that integrating the synonyms embedding independently into the network helps nothing because the network can not locate the synonyms in the sentence accurately. Besides, after training the KEPN in the scheme of multi-task with synonym labeling as an auxiliary task (i.e., KEPN$_{add_SL}$), our network makes further improvement and achieves the best result in our experiments.

(3) In most cases, Transformer-based models (KEPN and Transformer-PB) outperform RNN-based models (S2S-A, P-GEN and VAE-SVG). The reason might be that multi-head attention captures semantic information in the whole sentence at a time while RNN models sentence step-by-step.

Ablation study

We perform an ablation study on KEPN for better understanding the contributions of the main parts of our model. Although results in the last four lines of Table 2 illustrate the effect of each component of KEPN, both the BLEU and METEOR scores can only evaluate the similarity between paraphrase sentences and given references. To further evaluate the diversity among the input and paraphrase sentences, two other automatic metrics are added: the average numbers of generated synonyms (denoted as N_s) and OOV words (denoted as N_o) in the output sentences. We split the TCNP

Model	WikiAnswers		Quora		TCNP		LCQMC	
	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR	BLEU	METEOR
S2S-A (2015)	37.0	32.2	25.9	25.8	32.8	31.7	35.6	33.5
P-GEN (2017)	39.4	35.9	26.8	26.5	33.0	31.8	32.5	33.3
VAE-SVG (2018)	39.2	36.1	26.2	25.7	32.1	31.2	36.9	35.7
DGEN (2018)	39.7	36.2	27.6	29.9	–	–	–	–
Transformer-PB (2019)	42.6	36.4	26.4	29.3	–	–	–	–
Transformer	41.9	35.8	25.3	27.0	33.5	32.2	39.5	35.4
KEPN$_{sub_pos}$	42.1	35.6	26.5	28.1	33.7	32.0	39.1	35.4
KEPN	43.1	36.6	28.6	29.5	34.5	32.7	41.4	36.5
KEPN$_{add_SL}$	**44.3**	**37.1**	**29.2**	**30.4**	**35.6**	**34.4**	**43.1**	**38.0**

Table 2: Automatic evaluation with BLEU and METEOR on all datasets.

Methods	$0 < L \leq 5$		$5 < L \leq 15$		$L > 15$	
	N_s	N_o	N_s	N_o	N_s	N_o
Transformer	0.03	0.15	0.28	0.21	0.34	0.29
KEPN$_{sub_pos}$	0.05	**0.14**	0.27	0.19	0.87	0.24
KEPN	**0.06**	0.16	0.31	0.19	1.18	0.21
KEPN$_{add_SL}$	**0.06**	**0.14**	**0.39**	**0.14**	**1.42**	**0.15**
Ground Truth	0.13	0.0	0.43	0.0	1.40	0.0

Table 3: The average numbers of synonyms (N_s) and OOV words (N_o) which appear in the output sentences of different ablated version of KEPN on the TCNP dataset.

dataset according to the sentence length L and report the results of various versions of our model in these two metrics.

From the results in Table 3, we can find that adding either position encoding scheme or synonym labeling task improves the performance of the model in both metrics. This indicates that both components can not only bring more diversity to the generated results, but also alleviate the OOV issue by replacing a rare word with a corresponding synonym. What's more, we can also observe that as the sentence length grows, the advantages of our model become more obvious. One possible reason is that the longer the sentence, the more synonyms can be utilized.

Influence of sentence length

To explore the ability of learning the long-term dependency, we evaluate all model in TCNP dataset which is split into five parts according the length of sentence. The curve of our network (KEPN$_{add_SL}$) is always above others in Figure 3 and descends more smoothly when the sentence length is longer than 16. More interestingly, when the length of sentences ranges from 1 to 5, the RNN-based models outperform the Transformer models. This is caused by the difference of network structures between RNN-based models and Transformer models. RNN-based models is adept at capturing the semantics of short sentences with the help of internal memory units. While, Transformer models can capture dependencies of words without regard to their distance in sequences which makes Transformer more powerful to model the long sentences than RNN.

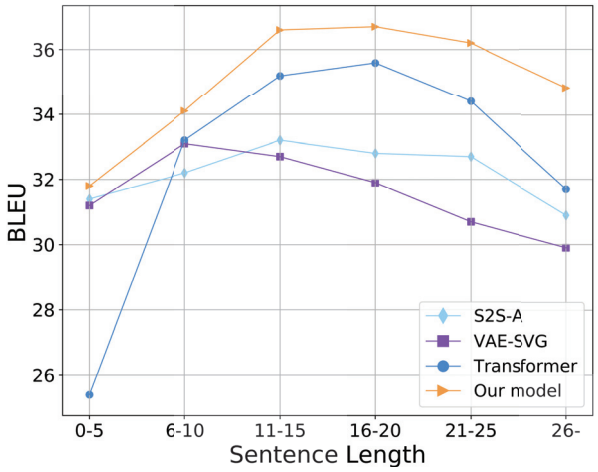

Figure 3: BLEU scores of sentences in different lengths on TCNP. The sentences are split into six groups. The length of sentences in each group falls into the same range. '16-20' means that length ranges from 16 to 20.

Human Evaluation

Though quantitative results show that our network outperforms other approaches, we also conduct a human evaluation to present the real quality of generated paraphrase. We randomly select 200 groups of source sentences from the test set of both English and Chinese datasets. Five well-educated university students are asked to score each sentence according to the following three criteria: 1) Relevance (the paraphrase sentence is semantically close to the source sentence); 2) Fluency (the paraphrase sentence is fluent as a natural language sentence, and the grammar is correct); 3) Diversity (the paraphrase sentence has more expressions compared with the source sentence).

Results in Table 4 show that the generated sentences of our network have the highest relevance to the source sen-

Methods	Quora			LCQMC		
	Rel.	Flu.	Div.	Rel.	Flu.	Div.
VAE-SVG	3.04	3.57	2.87	3.05	3.46	2.72
Transformer	3.76	4.23	3.08	4.28	4.50	2.88
KEPN	**4.03**	**4.36**	**3.38**	**4.49**	**4.66**	**3.01**
Ground Truth	4.26	4.44	3.84	4.73	4.88	3.58

Table 4: Human evaluation results of our network. Each assessor gives three scores (Relevance, Fluency and Diversity, shorted correspondingly as Rel., Flu. and Div.) to each paraphrase, both ranking from 0 to 5, where 0 is the worst and 5 is the best.

Original:	霍乱这个古老的病种，如今已经成为贫穷病。 *Cholera, an ancient species of disease, recently has become a disease of impoverishment.*
Reference:	霍乱是一种古老的疾病，已成为一种贫穷的疾病。 *Cholera is an ancient disease and has become a disease of impoverishment.*
VAE-SVG:	我们目前的<unk><unk>在其基础是一种疾病。 *Our current <unk> <unk> is based on a disease.*
Transformer:	霍乱是一个古老的<unk>，已经成为贫困的疾病。 *Cholera is an ancient <unk> that has become a disease of poverty.*
KEPN (ours):	霍乱是一个古老的疾病，现在已经成为一个贫困的疾病。 *Cholera is an ancient disease and now has become a disease of poverty.*
Synonyms:	(如今 现在) (recently, now) (贫穷 贫困) (impoverishment, poverty) (病种 疾病) (species of disease, disease)
Original:	What are the best ways of falling asleep quickly?
Reference:	What are some ways to fall asleep faster?
VAE-SVG:	What are some ways to stop falling asleep?
Transformer:	How can I stop myself from falling asleep?
KEPN (ours):	How do I fall asleep faster?
Synonyms:	(quickly, fast)

Figure 4: Some cases generated by different model. The texts in same color are synonym pairs.

tence. Besides, the scores of both Transformer and KEPN are high in fluency, indicating that the generated paraphrases are well-formed and grammatically correct. In terms of the Diversity, most of the methods don't perform well while the KEPN has an improvement of 0.2-0.3 points compared with others. This result demonstrates that by introducing synonym information from thesauri, the model can replace words in the original sentence with synonyms and thus generate a more expressive and diverse paraphrase.

Case Study

Some cases are listed in Figure 4. For the first Chinese example from TCNP, we can find that the VAE-SVG generates an unreadable sentence. A possible reason is that the training set contains few sentences about illness and the VAE-SVG fails in a unfamiliar field without the help of external knowledge. The Transformer meets an "unk" word, because the word "*species of disease*" is a low-frequency word in

Figure 5: A visualization of synonym-output attention. Each column is an attention weight distribution over synonym. Darker colors correspond to higher weights.

Chinese and thus replaced by an "unk" tag. By contrast, our network replaces the rare word with corresponding synonym "*disease*" thanks to the synonym pairs provided by thesauri, successfully alleviating the issue of OOV words. Besides, unlike the Transformer, more words are replaced by synonyms in the output of our network, making the paraphrase more diverse and expressive. For the second English case from Quora, we find that both the generations of the Transformer and the VAE-SVG do not convey the same meanings as the original sentence. Instead, our network not only match the input, but also generate a new word "*faster*" rather than copy the word "*quickly*" from the input sentence.

Moreover, we also examine the soft attention mechanism of our network for indirect evidence of the contribution of synonym. For instance, the output sentence in Figure 5 is rewritten from "*The medicine has widespread usage in high-income countries due to less side effects*". When replacing the word "*medicine*" of the input sentence, there are three synonyms (i.e. "*drug*", "*pill*", "*remedy*") to be chosen. Three candidates in Figure 5 receive the certain degrees of attention, and the attention weight of "*drug*" is the highest and wins the vote. This illustrates that the dynamic interaction between the weight attention and the generated text helps our network to choose correct synonyms, improving the diversity of the output.

Conclusion

In this paper, we present a Knowledge-Enhanced Paraphrase Network for paraphrase generation through editing the original sentence with synonym information provided by external linguistic knowledge. To further locate the synonym pairs more accurately, a multi-task architecture with synonym labeling as an auxiliary task is also designed. Experiments on both Chinese and English datasets demonstrate that our network significantly outperforms the existing methods on paraphrase generation task.

Acknowledgments

This research is supported by National Natural Science Foundation of China (Grant No. 61773229 and 61972219), Shenzhen Giiso Information Technology Co. Ltd., the National Natural Science Foundation of Guangdong Province (Grand No. 2018A030313422), and Overseas Cooperation Research Fund of Graduate School at Shenzhen, Tsinghua University (Grant No. HW2018002).

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. *In Proceedings of ICLR, pages 1–15*.

Berant, J., and Liang, P. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1415–1425.

Bolshakov, I. A., and Gelbukh, A. 2004. Synonymous paraphrasing using wordnet and internet. In *International Conference on Application of Natural Language to Information Systems*, 312–323. Springer.

Cao, Z.; Luo, C.; Li, W.; and Li, S. 2017. Joint copying and restricted generation for paraphrase. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Carl, M.; Schmidt, P.; and Schütz, J. 2005. Reversible template-based shake & bake generation. In *Proceedings of MT Summit X, Workshop on EBMT*, 17–25.

Fader, A.; Zettlemoyer, L.; and Etzioni, O. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1608–1618. Sofia, Bulgaria: Association for Computational Linguistics.

Fader, A.; Zettlemoyer, L.; and Etzioni, O. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1156–1165. ACM.

Gupta, A.; Agarwal, A.; Singh, P.; and Rai, P. 2018. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Huang, S.; Wu, Y.; Wei, F.; and Luan, Z. 2019. Dictionary-guided editing networks for paraphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6546–6553.

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

Lavie, A., and Agarwal, A. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, 228–231. Association for Computational Linguistics.

Li, Z.; Jiang, X.; Shang, L.; and Liu, Q. 2019a. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3403–3414. Florence, Italy: Association for Computational Linguistics.

Li, Z.; Ding, N.; Liu, Z.; Zheng, H.; and Shen, Y. 2019b. Chinese relation extraction with multi-grained information and external linguistic knowledge. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4377–4386.

Liu, X.; Chen, Q.; Deng, C.; Zeng, H.; Chen, J.; Li, D.; and Tang, B. 2018. Lcqmc: A large-scale chinese question matching corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, 1952–1962.

Madnani, N., and Dorr, B. J. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics* 36(3):341–387.

McKeown, K. R. 1983. Paraphrasing questions using given and new information. *Computational Linguistics* 9(1):1–10.

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

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, 311–318. Association for Computational Linguistics.

Pavlick, E.; Rastogi, P.; Ganitkevitch, J.; Van Durme, B.; and Callison-Burch, C. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 425–430.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Prakash, A.; Hasan, S. A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; and Farri, O. 2016. Neural paraphrase generation with stacked residual lstm networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2923–2934.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, 1073–1083.

Shah, P.; Hakkani-Tür, D.; Tür, G.; Rastogi, A.; Bapna, A.; Nayak, N.; and Heck, L. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, S.; Gupta, R.; Chang, N.; and Baldridge, J. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7176–7183.

Yang, M.; Chen, L.; Chen, X.; Wu, Q.; Zhou, W.; and Shen, Y. 2019a. Knowledge-enhanced hierarchical attention for community question answering with multi-task and adaptive learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 5349–5355. AAAI Press.

Yang, M.; Yin, W.; Qu, Q.; Tu, W.; Shen, Y.; and Chen, X. 2019b. Neural attentive network for cross-domain aspect-level sentiment classification. *IEEE Transactions on Affective Computing*.

Zhou, H.; Young, T.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, 4623–4629.