Paraphrase Generation by Learning How to Edit from Samples

Amirhossein Kazemnejad[†], **Mohammadreza Salehi**[‡], **Mahdieh Soleymani Baghshah**[‡]
[†]Iran University of Science and Technology, [‡]Sharif University of Technology
a_kazemnejad@comp.iust.ac.ir, mrezasalehi@ce.sharif.edu,
soleymani@sharif.edu

Abstract

Neural sequence to sequence text generation has been proved to be a viable approach to paraphrase generation. Despite promising results, paraphrases generated by these models mostly suffer from lack of quality and diversity. To address these problems, we propose a novel retrieval-based method for paraphrase generation. Our model first retrieves a paraphrase pair similar to the input sentence from a pre-defined index. With its novel editor module, the model then paraphrases the input sequence by editing it using the extracted relations between the retrieved pair of sentences. In order to have fine-grained control over the editing process, our model uses the newly introduced concept of Micro Edit Vectors. It both extracts and exploits these vectors using the attention mechanism in the Transformer architecture. Experimental results show the superiority of our paraphrase generation method in terms of both automatic metrics, and human evaluation of relevance, grammaticality, and diversity of generated paraphrases.

1 Introduction

Paraphrases are texts conveying the same meaning while using different words (Bhagat and Hovy, 2013). Paraphrase generation is an important task in Natural Language Processing (NLP) that has many applications in other down-stream tasks, such as text summarization, question answering, semantic parsing, and information retrieval (Cao et al., 2017; Fader et al., 2014; Berant and Liang, 2014).

Early works on paraphrasing mostly investigated rule-based or statistical machine translation approaches to this task (Bannard and Callison-Burch, 2005). With the recent advances of neural sequence-to-sequence (Seq2Seq) framework in different NLP tasks, especially in machine translation, an increasing amount of literature have also applied

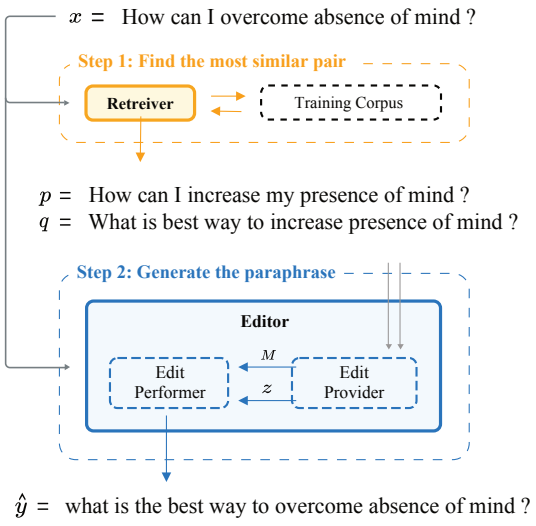

Figure 1: An overview of the proposed model. This model retrieves the most similar paraphrase pair to the input x from the training corpus (Retriever), computes a set of edit vectors $[M, z]$ based on the retrieved pair (Edit Provider), and applies these edits to the input sequence x to generate its paraphrase (Edit Performer).

Seq2Seq models to the task of paraphrase generation (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2018).

Although the proposed Seq2Seq methods for paraphrase generation have shown promising results, they are not yet as dominant as their counterparts used in neural machine translation. The main reason is that the available training data for paraphrasing is scarce and domain-specific (Wang et al., 2019). In fact, the necessity to generate sequences from scratch, which is a major drawback of traditional Seq2Seq models (Guu et al., 2018), magnifies itself when dealing with scarce training data. Thus, one can expect that the model would not be trained well and consequently, would not be able to generate diverse outputs.

Although retrieval-based text generation has

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 6010–6021

been evaluated recently in Guu et al. (2018); Hashimoto et al. (2018); Wu et al. (2019) as a remedy for this problem, to the best of our knowledge, there is no previous study exploring the usage of this approach in paraphrase generation. Moreover, none of the existing works in the realm of retrieval text generation, such as Guu et al. (2018); Wu et al. (2019); Hashimoto et al. (2018), focuses on learning how to extract edits from the retrieved sentences. Indeed, Guu et al. (2018); Wu et al. (2019) computes a single edit vector heuristically through concatenating the weighted sum of the inserted word embeddings and the weighted sum of deleted word embeddings. Moreover, Hashimoto et al. (2018) only focuses on improving the retrieving stage and uses a standard Seq2Seq model to edit the retrieved sentence.

In this paper, we present an effective retrieval-based approach to paraphrase generation by proposing a novel editor module. Our method can be summarized as follows: Given an input sentence x, the model first retrieves a similar sentence p and its associated paraphrase q from the training data. Then, by getting x and (p, q), the editor both learns how to extract the fine-grained relations between p and q as a set of edits, and also when and how to use these extracted edits to paraphrase x. By incorporating the retrieved pairs into the editing process, we invigorate our model with a non-parametric memory, which enables it to produce non-generic and more diverse outputs. Both the retriever and editor components of our method are modeled by deep neural networks. We employ the Transformer architecture (Vaswani et al., 2017) as the backbone of our model, and use its attention mechanism as an effective tool to apply edits in a selective manner.

Our main contributions are:

- We propose the Fine-grained Sample-based Editing Transformer (FSET) model. It contains a novel editor that can be used in a retrieval-based framework for paraphrase generation. This editor learns how to discover the relationship between a pair of paraphrase sentences as a set of edits, and transforms the input sentence according to these edits. It is worth noting that the set of edits is learned in an end-to-end manner as opposed to Guu et al. (2018); Wu et al. (2019) that compute the edit vector heuristically.

- For the first time, we utilize the Transformer as an efficient fully-attentional architecture for the task of retrieval-based text generation.

- Experimentally, we compare our method with the recent paraphrase generation methods, and also with the retrieval-based text generation methods that have been introduced recently. Both of the quantitative and qualitative results show the superiority of our model.

2 Related Work

2.1 Neural paraphrase generation

Prakash et al. (2016) was the first work that adapted a neural approach to paraphrase generation with a residual stacked LSTM network. Gupta et al. (2018) combined a variational auto-encoder with a Seq2Seq LSTM model to generate multiple paraphrases for a given sentence. Li et al. (2018) proposed a model in which a generator is first trained on the paraphrasing dataset, and then is fine-tuned by using reinforcement learning techniques. Cao et al. (2017) utilized separate decoders for copying and rewriting as the two main writing modes in paraphrasing. Mallinson et al. (2017) addressed paraphrasing with bilingual pivoting on multiple languages in order to better capture different aspects of the source sentence. Iyyer et al. (2018) proposed a method to generate syntactically controlled paraphrases and use them as adversarial examples. Chen et al. (2019) addressed the same problem, but the syntax is controlled by a sentence exemplar. Kajiwara (2019) proposed a model that first identifies a set of words to be paraphrased, and then generates the output by using a pre-trained paraphrase generation model. Wang et al. (2019) proposed a Transformer-based model that utilizes structured semantic knowledge to improve the quality of paraphrases. Kumar et al. (2019) modified the beam search algorithm with a sub-modular objective function to make the generated set of paraphrases syntactically diverse. Li et al. (2019) decomposed paraphrasing into sentential and phrasal levels and employed separate Transformer-based models for each of these levels. Fu et al. (2019) decomposes paraphrasing into two steps: content planning and surface realization, and improves the interpretability of the first step by incorporating a latent bag of words model.

2.2 Retrieval-based text generation

Retrieval-based text generation has received much attention in the last few years. Song et al. (2016);

Wu et al. (2019) augmented Seq2Seq generation-based models with retrieval frameworks to make the dialog responses more meaningful and non-generic. Gu et al. (2017) utilized a search engine to retrieve a set of source-translation pairs from the training corpus, both at train and test time, and use them as a guide to translate an input query. Guu et al. (2018) proposed the neural editor model for unconditional text generation, which produces a new sentence by editing a retrieved prototype using an edit vector. Hashimoto et al. (2018) proposed a task-specific retriever using the variational framework to generate complex structured outputs, such as Python code. This work, however, does not have any novelty in the editor's architecture and uses a standard Seq2Seq model with attention and copy mechanism (Hashimoto et al., 2018).

3 Proposed Approach

Let $\mathcal{D} = \{x_n, y_n\}_{n=1}^{N}$ denotes a dataset where x_n is a sequence of words, and y_n is its target paraphrase. In the paraphrasing task, our goal is to find the set of parameters of the model that maximizes $\prod_{n=1}^{N} p_{\text{model}}(y_n|x_n)$. Figure 1 illustrates the overview of our proposed model which is composed of a *Retriever* and an *Editor*. Given an input sequence x, the retriever first finds a paraphrase pair (p, q) from the training corpus based on similarity of x and p. Then, the editor utilizes the retrieved pair (p, q) to paraphrase x. We discuss the details in the following subsections.

3.1 Retriever

The goal of the retriever module is to select the paraphrase pairs (from the training corpus) that are similar to the input sequence x. To do that, the retriever finds a neighborhood set $\mathcal{N}(x)$ consisting of the K most similar source sentences $\{p_k\}_{k=1}^{K}$ to x and their associated paraphrases $\{q_k\}_{k=1}^{K}$ (K is a hyper-parameter of the model). To measure similarity of sentences, we first embed them employing the pre-trained transformer-based sentence encoder proposed by Cer et al. (2018). The similarity is then calculated using cosine similarity measure in the resulted embedding space. We call this retriever as *General Retriever* throughout the paper. Note that using a pre-trained retriever can help us to alleviate the scarcity problem of the training data available for paraphrasing[1].

[1]Pre-trained model is available at https://tfhub.dev/google/universal-sentence-encoder-large/3

In order to search for the similar sentences to an input sequence efficiently, we use the FAISS software package (Johnson et al., 2019) to create a fast search index from the sentences in the training corpus. We would also pre-compute the neighborhood set of each source sentence in the training set, so at the training time, our model just needs to sample one of the pairs in the neighborhood set uniformly and feed it as an input to the editor module. The probability of retrieving a pair can thus be stated as

$$p((p, q)|x) = \frac{1}{K}\mathbf{1}[(p, q) \in \mathcal{N}(x)]. \quad (1)$$

Note that the same procedure also holds for the test time, and the retriever computes $\mathcal{N}(x)$ so the model can sample any one of the pairs in $\mathcal{N}(x)$ to generate the output based on that pair.

3.2 Editor

To edit a sentence according to a retrieved pair, we propose an editor module consisting of two components: 1) Edit Provider and 2) Edit Performer. The Edit Provider computes a set of edit vectors based on the retrieved pair of sentences (p, q). After that, the Edit Performer rephrases the input sequence x by utilizing this prepared set of edits.

3.2.1 Edit Provider

This part of the editor extracts the edits from the retrieved pair as a set of vectors which we call **M**icro **E**dit **V**ectors (**MEVs**). MEVs are responsible for encoding the information about fine-grained edits that transform p into q. Each one of the MEVs represents the most plausible soft alignment between a token in p and the semantically relevant parts in q:

$$M = \{m_i := \text{small edit applied on } p_i | 1 \leq i \leq l\}$$

where l is the length of p.

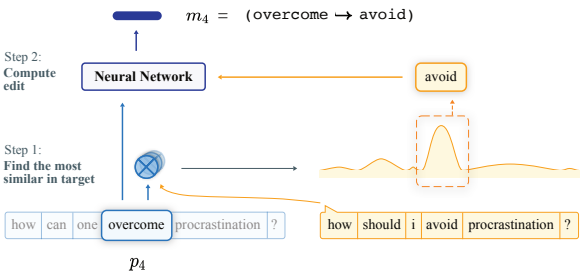

Figure 2: The general scheme of computing a MEV corresponding to a token of p.

Figure 2 presents, in schematic form, the procedure of computing one MEV. For each arbitrary

token of p, such as p_i, we intend to compute a MEV that encodes the edit corresponding to p_i using attention over q. Then, given p_i as the source of the edit, and the attention's result as the target, we concatenate their representations and feed it as the input to a neural network, which calculates m_i as the corresponding edit vector. To make this process differentiable and parallelizable, we use a fully-attentional architecture consisting of two main sub-modules: 1) Edit Encoder and 2) Target Encoder. Figure 3 shows the overview of the Edit Provider.

In this model, at first, a context-aware representation $R_q = [r_q^1, ..., r_q^k]$ of the sequence q is computed using the Target Encoder which is the encoder sub-graph of the Transformer architecture (Vaswani et al., 2017). The Edit Encoder is also the encoder of the Transformer model, but, with an extra multi-head attention over R_q. This module outputs a vector that encodes the most semantically relevant parts of q to p_i. After that, the MEVs, i.e. m_is, are computed by feeding these vectors one by one into a single dense layer (with the $tanh(.)$ activation function). By setting the output dimension of the dense layer to be smaller than the dimension of the word embeddings, we introduce a bottleneck, which hinders the Edit Encoder from copying q directly.

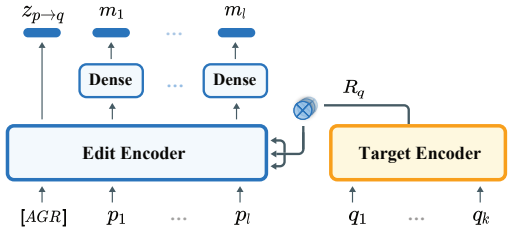

Figure 3: Architecture of Edit Provider. The Edit Encoder uses multi-head attention on R_q to select the target of edit for each token of p. Note that by prepending $[AGR]$ to p, we can encode all of the MEVS into a single edit vector $z_{p\to q}$.

Finally, all of the MEVs are aggregated into a single vector z by leveraging a technique inspired by Devlin et al. (2019); we prepend a special token $[AGR]$ to p in order to encode all the edits into a single vector $z_{p\to q}$. The intuition behind encoding into a single vector $z_{p\to q}$ is to allow the model learn a global edit that can be applied to the whole sentence, in addition to the MEVs as local edits. We run the Edit Performer with the same parameters in the reverse direction, i.e. from q to p, to

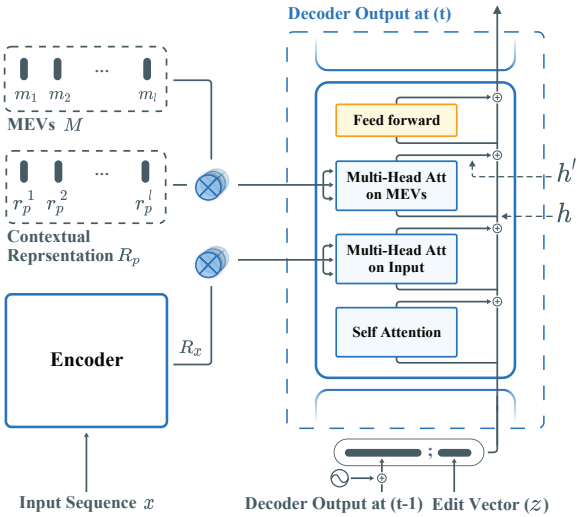

Figure 4: Illustration of the Edit Performer generating the output token at t-th time step. Note that only one layer of the decoder is depicted and the layernorms are not shown for simplicity.

compute R_p and $z_{q\to p}$. The final edit vector z is then computed as

$$z = \texttt{Linear}(z_{p\to q} \oplus z_{q\to p}),$$

where `Linear` denotes a dense layer without activation and bias.

3.2.2 Edit Performer

The Edit Performer transforms the input sequence $x = [x_1, ..., x_s]$ to the final output $\hat{y}$ using the edit vectors. We employ a fully-attentional Seq2Seq architecture composed of an encoder and a decoder for this part of the model.

The encoder of the Edit Performer has exactly the same architecture as the original encoder of the Transformer model and outputs a context-aware representation $R_x = \{r_x^i\}_{i=1}^s$ of the input sequence. For the decoder, we use a slightly modified version of the original Transformer's decoder. Indeed, the Transformer learns to model $p(y|x)$, while we would like to model a conditional setting $p(y|x, (p, q))$. Moreover, as mentioned in the description of the Edit Provider, the relation between p and q is encoded in MEVs M and the vector z. Therefore, in order to edit x, instead of using (p, q) directly, we only need M and z to specify the edits, and the sentence p to identify the locations in x to which the edits should be applied. Thus, we aim to model $p(y|x, p, M, z)$ with the Edit Performer.

Figure 4 depicts the architecture of the Edit Performer. To condition the generation process on

the edit vector z, we append it to each token of the decoder's input. To apply the edits in a fine-grained manner, we would like the model to attend to the most similar token of p and select the corresponding edit in MEVs M to be applied to the input sentence. Therefore, in addition to the input sequence representation R_x, the model also attends to MEVs M using an extra multi-head attention sub-layer which computes the representation

$$h' = \texttt{MultiHeadAtt}(\textbf{Q:}\ h, \textbf{K:}\ R_p, \textbf{V:}\ M),$$

where h comes from the previous sub-layer and R_p is the context-aware representation of the retrieved sequence p, which is calculated by the Edit Provider. Hence, this sub-layer allows the model to apply edits only when the current context matches somewhere in p. Finally, we project h' (after applying the residual connection and the layernorm) using a fully-connected sub-layer and feed it to the above layer. For the last layer, a softmax activation is employed to predict the next token of the output.

3.3 Training

During the training phase, our aim is to maximize the log likelihood objective

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \log p(y|x). \tag{2}$$

As we decompose the training procedure to two stages of retrieving and editing, we can rewrite $p(y|x)$ as

$$p(y|x) = \sum_{(p,q)\in\mathcal{D}} p(y|x,(p,q))p((p,q)|x). \tag{3}$$

Substituting Eq. 1 into Eq. 3 and then inserting the resulted $p(y|x)$ into Eq. 2 yields the following formulation for the log likelihood:

$$\mathcal{L} = \sum_{(x,y)\in\mathcal{D}} \log(\frac{1}{K} \sum_{(p,q)\in\mathcal{N}(x)} p(y|x,(p,q))).$$

We train our model by maximizing the following lower bound of the log likelihood (obtained by Jensen's inequality):

$$\mathcal{L} \geq \mathcal{L}' = \frac{1}{K} \sum_{(x,y)\in\mathcal{D}} \sum_{(p,q)\in\mathcal{N}(x)} \log p(y|x,(p,q)).$$

Note that $p(y|x,(p,q)) = p_\theta(y|x,p,m_\phi(p,q),z_\phi(p,q))$, where θ denotes the parameters of the Edit Performer and ϕ shows the parameters of the Edit Provider. Thus, we solve the following optimization problem:

$$\theta^*, \phi^* = \underset{\theta,\phi}{\operatorname{argmax}}\, \mathcal{L}'(\theta,\phi).$$

Except for the retriever which is a pre-trained component of our model, other components are fully coupled and trained together. To prevent the model from ignoring the information coming from the retrieval pathway during the training procedure (i.e. ignoring the edit vectors extracted from the retrieved pair), we use a simple yet effective trick; we manually add extra (x,y) pairs to $\mathcal{N}(x)$ proportionate to the number of retrieved pairs K so the presence of y as the exact ground-truth paraphrase encourages the model to use the retrieved pairs more. Please refer to A.1 for further details.

4 Experiments

In this section, we empirically evaluate the performance of our proposed method in the task of paraphrase generation, and compare it with various other methods, including previous state-of-the-art paraphrasing models.

4.1 Datasets

We conduct experiments on two of the most frequently used datasets for paraphrase generation: the Quora question pair dataset and the Twitter URL paraphrasing corpus. For the Quora dataset, we only consider the paraphrase pairs. Similar to Li et al. (2018), we sample 100k, 30k, 3k instances for train, test, and validation sets, respectively. Twitter URL paraphrasing dataset consists of two subsets, one is labeled by human annotators, and the other is labeled automatically, thus, it is noisier compared to the Quora dataset. Similar to Li et al. (2018), we sample 110k instances from automatically labeled part as our training set and two non-overlapping subsets of 5k and 1k instances from the part annotated by humans for the test and validation sets, respectively. As in Li et al. (2018, 2019), we truncate sentences in both of the datasets to 20 tokens.

Hyperparameter	Edit Performer	Edit Provider
Hidden dimension	64	64
# Layers	6	4
# Heads	8	4
MEV dimension m_i	-	40
Edit vector z dimension	-	64

Table 1: Settings of the Model

4.2 Baselines

We compare our method with both the existing paraphrasing methods that are not retrieval-based, and also with the existing or newly created retrieval-based text generation methods which we adapt for paraphrasing:

- **Non-retrieval** paraphrasing methods:
  - **Residual LSTM** (Prakash et al., 2016) which is the first Seq2Seq model proposed for paraphrase generation,
  - **RbM** (Li et al., 2018) that fine-tunes a paraphrase generation model using reinforcement learning,
  - **Transformer** (Vaswani et al., 2017) which is a Seq2Seq model relying entirely on attention mechanism,
  - **DNPG** (Li et al., 2019) that decomposes paraphrasing to sentential and phrasal levels and utilizes separate Transformers for each level,
  - **DiPS** (Kumar et al., 2019) which aims to generate diverse paraphrases by adopting a novel approach in the decoding stage instead of beam search.

  The latter two of the above list have been reported as the state-of-the-art models in paraphrase generation (Kumar et al., 2019; Li et al., 2019).

- **Retrieval-based** models: We compare our method with one existing retrieval-based text generation model and two other combinational methods that we create by ourselves:
  - **Seq2Seq+Ret** which is an extended version of Seq2Seq Residual LSTM. This model conditions the generation process at each time step on an edit vector encoding the differences between the retrieved sentences p and q. To make the comparison fair, we use the General Retriever (introduced in the Retriever subsection of the Proposed Approach Section) to find (p, q). The edit vector for this pair is also computed by concatenating the sum of inserted word embeddings with the sum of deleted word embeddings as it is stated by Guu et al. (2018).
  - **RaE** that is proposed by Hashimoto et al. (2018) as a method with an in-domain retriever. The editor of this model is a Seq2Seq LSTM equipped with attention mechanism over the input x, and copy mechanism over the retrieved pair p and q.
  - **CopyEditor+Ret** which is composed of the editor of Hashimoto et al. (2018), and the General Retriever. We compare FSET with this baseline model to further evaluate the role of our proposed editor.

4.3 Experimental settings

Table 1 shows the settings of our model. We select the hyperparameters suggested by Li et al. (2018) for the LSTM-based Seq2Seq baselines, and the hyperparameters mentioned by Li et al. (2019) for the Transformer-based baselines. It is worth noting that our model's size w.r.t. the number of parameters is approximately $\frac{1}{2}$ of the baseline LSTM's size and $\frac{1}{5}$ of the baseline Transformer's size. The newly created retrieval-based baselines have the same hidden size and the same number of layers as the non-retrieval models. For the Seq2Seq+Ret model, we keep the ratio of hidden size to the edit vector dimension same as the reported ratio in Guu et al. (2018). We train all of the models for 100k iterations, and choose the best version based on their validation loss after training. We set the batch size to 128 and the vocabulary size to 8k in all of the experiments. The embeddings are also trained from scratch. In all of the experiments on the retrieval-based methods, the hyper-parameter K is set to 1. However, results for different values of K are also reported in A.2. During the decoding stage, we use beam search to generate a set of outputs. In order to select the final output, an approach similar to Gupta et al. (2018) is used which chooses the most lexically similar sentence to the input where the similarity is calculated based on the Jaccard measure.

4.4 Results and analysis

We compare different methods using BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) as the most common metrics for automatic evaluation of paraphrase generation methods. Table 2 summarizes the results of different methods. These results indicate that our model outperforms the previous state-of-the-art models in terms of all of the metrics.

It is worth noting that the models which have utilized copy mechanism, such as DNPG, RbM, RaE, and CopyEditor+Ret, generally outperform the other baselines. The Seq2Seq+Ret, i.e. the

Models	Quora					Twitter URL Paraphrasing				
	ROUGE-2	ROUGE-1	BLEU-4	BLEU-2	METEOR	ROUGE-2	ROUGE-1	BLEU-4	BLEU-2	METEOR
Residual LSTM (Prakash et al., 2016)	32.71	59.69	24.56	38.52	29.39	27.94	41.77	25.92	32.13	24.88
Seq2Seq+Ret (Ours)	32.71	60.83	25.23	42.71	32.51	21.56	40.18	20.11	31.58	22.38
DiPS (Kumar et al., 2019)	31.77	59.79	25.37	40.35	29.28	23.67	43.64	27.66	37.92	25.69
Transformer (Vaswani et al., 2017)	34.23	61.25	30.38	42.91	34.65	29.55	44.53	32.14	40.34	28.26
DNPG (Li et al., 2019) [2]	37.75	63.73	25.03	-	-	-	-	-	-	-
RbM (Li et al., 2018) [2]	38.11	64.39	-	43.54	32.84	24.23	41.87	-	44.67	19.97
RaE (Hashimoto et al., 2018)	35.07	62.71	29.22	46.21	29.92	31.53	47.55	34.16	44.33	30.09
CopyEditor+Ret (Ours)	35.59	62.93	29.78	46.55	35.56	27.35	45.54	28.06	40.30	26.93
FSET (Ours)	**39.55**	**66.17**	**33.46**	**51.03**	**38.57**	**32.04**	**49.53**	**34.62**	**46.35**	**31.67**

Table 2: Results of the different models on two paraphrasing datasets.

retrieval-based Residual LSTM, shows an improvement over Residual LSTM on Quora dataset. However, this is not the case on the Twitter dataset and we hypothesize that it is due to uncommon texts in this corpus (i.e. informal text with hashtags and abbreviated words), on which the General Retriever has not been trained. Therefore, a pre-trained retriever cannot help in this case. The CopyEditor+Ret model which incorporates a more powerful editor than Seq2Seq+Ret shows better results than both of the Residual LSTM and Seq2Seq+Ret. However, a phenomenon similar to what was stated for Seq2Seq+Ret is also observed for this model on the Twitter dataset. The RaE model with the same editor as CopyEditor but with a supervised (task-specific) retriever leads to near state-of-the-art results. This indicates the role of the supervised task-specific retriever used in RaE, especially in the results on Twitter dataset. The superiority of our method over RaE in all of the metrics could be a sign of the effectiveness of our proposed editor module. Although our model uses the General Retriever, it still outperforms all other methods even on the Twitter dataset. It is worth mentioning that we can replace the General Retriever in our method with other retrievers like supervised task-specific ones to improve the results even more. Moreover, it is worth noting that our model that is only based on the Transformer architecture and the General Retriever (that is not required to be trained in each domain) needs much less training time than RaE.

4.5 Human evaluation

As there is no appropriate automatic metric for evaluating the diversity and novelty of generated sentences, we use human evaluation to assess the performance of our model qualitatively. We compare our method with two other methods: 1) RaE (Hashimoto et al., 2018) as a retrieval-based method adapted for paraphrasing, and 2) DiPS (Kumar et al., 2019) as a paraphrasing model which generates semantically diverse outputs by adopting a novel approach instead of beam search during the decoding stage. We choose these models as we would like to compare our method both with a state-of-the-art retrieval-based method and with a method that can generate diverse outputs. It must be noted that many of the recent methods in Table 2 are not able to generate diverse outputs.

We first select 100 sentences randomly from the test set of Quora dataset. Then, for each model, three paraphrases are generated for each one of the sentences, and these three outputs are considered as a paraphrase group. We aggregate and shuffle these paraphrase groups and ask six human annotators to

[2]Results are directly reported from Li et al. (2018, 2019) on the same dataset and settings.

Models	Grammar		Coherency	
	Score	κ	Score	κ
DiPS (Kumar et al., 2019)	3.97	0.253	2.55	0.476
RaE (Hashimoto et al., 2018)	4.70	0.286	3.90	0.483
FSET (Ours)	**4.70**	**0.394**	**4.22**	**0.528**

Table 3: Human evaluation on Quora dataset.

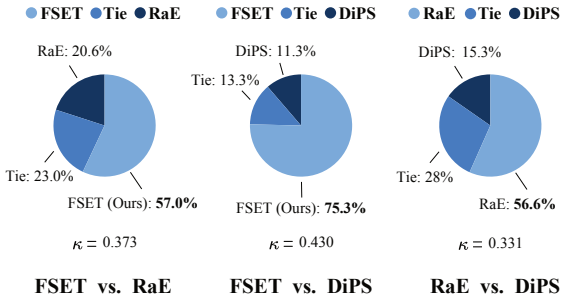

Figure 5: Results of the one-on-one human evaluation (second experiment). Annotators decide "Tie" when the outputs of the two models have the same quality in their opinion.

evaluate them in two scenarios.

In the first scenario, we ask the human annotators to score the outputs individually based on the following two criteria: 1) Grammar and fluency, 2) Consistency and coherency. Similar to Li et al. (2018), we use a 5-scale rating for each criterion. Table 3 presents the results. As can be seen, our model generally outperforms the other methods. Although RaE and our model can both produce grammatically correct outputs, the consistency and coherency for the outputs of our method is much better. Moreover, the inter-annotator agreement measured by Cohen's kappa κ shows fair or intermediate agreement between raters assessing the models.

Since directly scoring diversity and novelty of one paraphrase group is not simple even for humans, in the second scenario, we ask the annotators to make one-on-one comparisons on the groups of generated paraphrases. In other words, for each pair of the models, they have to decide which model produces better outputs for each one of the sentences (Ties are also allowed). Figure 5 depicts the one-on-one diversity evaluation. Our method and RaE both outperform DiPS, probably due to their retrieval-based nature. Moreover, this figure reveals that our method can generate significantly better outputs compared to RaE. We believe the reason is that RaE's editor is not as properly designed as our editor module. We explicitly inject the paraphrasing patterns found in the neighboring paraphrases into the Edit Performer which helps it to generate more diverse paraphrases. Please refer to the A.3 for some further details on the experiments.

4.6 Case study

Table 4 shows some examples of the paraphrases generated by our model. A common pattern among the output paraphrases is that the model has combined different parts of the input sentence and the retrieved target sentence to create a grammatically correct paraphrase. Furthermore, the model is able to change the syntactic form of words correctly, or replace them with their synonyms, as it is the case for "via" and "by using", and "is" and "are". The fact that the model can apply these changes in different levels of granularity can indicate that the idea of editing based on MEVs works properly.

(1)	x	Why do people ask such questions here on Quora which could be easily found on the internet ?
	p	Why do people ask questions on Quora that could simply be googled?
	q	*why do people ask questions on quora that* are easily ***answerable*** **via** *a quick internet search ?*
	y'	*why do people ask questions on quora that* could be easily ***answered*** **by using** *a quick internet search ?*
(2)	x	Who can the Democrats effectively run against Trump in 2020 ?
	p	Who will lead Trump 's movement after the election ?
	q	*Who do you think is most likely* to lead the Trump movement after 2016 ?
	y'	*who do you think is* the *most likely* democrat to run against trump in 2020 ?
(3)	x	What should I do when someone marks my question as needing improvement?
	p	Even though most of my asked questions are in well written form, they are usually marked as needing improvement. What should I do ?
	q	Most of my questions **are** *marked as needing improvement* . Where can I seek professional help ?
	y'	what should I do if my question **is** *marked as needing improvement ?*
(4)	x	What is the best way to make money as a 15 year old ?
	p	What are the best ways to make money as a teenager ?
	q	*I 'm a* teen , *how can I* make some money online ?
	y'	*I 'm a* 15 year old , *how can I* make money ?

Table 4: Example outputs of the model. Red phrases are extracted from the retrieved target sentence q, and blue parts are extracted from the source sentence x. Phrases that appear with slight changes in semantics or syntax in the output are made bold. The sentences are annotated manually for better readability.

Model Variant	ROUGE-2	ROUGE-1	BLEU-4	BLEU-2
Jaccard Retriever	38.52	65.47	31.72	48.83
No edit vector z	38.31	65.44	30.40	47.77
No Attention on MEVs M	39.36	65.72	29.73	46.66

Table 5: Ablation study

4.7 Model Ablation

In order to further evaluate the role of each module in our model, we train and assess different variants of it where in each variant, a key component has been replaced by an alternative simpler one:

- Jaccard Retriever: The retriever of our model is replaced by a simple retriever that selects neighbor sentences using the Jaccard similarity metric.
- No edit vector z: A variant in which we do not condition the Transformer in the Edit Performer on the aggregated edit vector z, and edit the source sentence merely based on MEVs.
- No Attention on MEVs: In this variant of our model, the Transformer in the Edit Performer is not conditioned on MEVs, and the source sentence is edited based on only z.

We train all of these variants on the Quora paraphrasing dataset. Table 5 shows the results of these models. As it is seen, the model which uses the Jaccard similarity measure performs worse than the original model with the General Retriever. Nonetheless, the results of this version explains that even the combination of our editor module with this simple retriever outperforms previous state-of-the-art methods. This indicates that our proposed editor can distinguish whether the extracted edits are plausible enough to be applied to the input sentence. Moreover, the results show that both eliminating z and M from our editor decrease its performance. In other words, both conditioning on z as the aggregated edit at each step of generation and the attention on MEVs M help the proposed editor.

5 Conclusion

In this paper, we proposed a retrieval-based paraphrase generation model which includes a novel fully-attentional editor. This editor learns how to extract edits from a paraphrase pair and also when and how to apply these edits to a new input sentence. We also introduced the new idea of Micro Edit Vectors, where each one of these vectors represents a small edit that should be applied to the source sentence to get its paraphrase. We incorporated Transformer modules in our editor and augmented them with attention over Micro Edit Vectors. The proposed model outperforms the previous state-of-the-art paraphrase generation models in terms of both automatic metrics and human evaluation. Moreover, the outputs show that our model is able to produce paraphrases by editing sentences in a fine-grained manner using the idea of MEVs. In future work, we intend to adapt our editor module for other learning tasks with both the structured input and structured output.

6 Acknowledgments

We thank anonymous reviewers for their detailed feedback and suggestions.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan. Association for Computational Linguistics.

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

Rahul Bhagat and Eduard Hovy. 2013. Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI17, pages 3152–3158. AAAI Press.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 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*, KDD '14, pages 1156–1165, New York, NY, USA. ACM.

Yao Fu, Yansong Feng, and John P Cunningham. 2019. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems 32*, pages 13645–13656. Curran Associates, Inc.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor O. K. Li. 2017. Search engine guided non-parametric neural machine translation. *CoRR*, abs/1705.07267.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI18, pages 5149–5156. AAAI Press.

Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 10073–10083, USA. Curran Associates Inc.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Herve Jegou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, page 11.

Tomoyuki Kajiwara. 2019. Negative lexically constrained decoding for paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6047–6052, Florence, Italy. Association for Computational Linguistics.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3609–3619, Minneapolis, Minnesota. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. 2016. Two are better than one: An ensemble of retrieval- and generation-based dialog systems. *CoRR*, abs/1610.07149.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: Paraphrase generation with semantic augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7176–7183.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):7281–7288.

A Appendix

A.1 Construction of $\mathcal{N}(x)$ during training

For each pair of sentences, such as (x, y), we augment its neighbourhood set $\mathcal{N}(x)$ with multiple (x, y) pairs to get the new neighbourhood set

$$\mathcal{N}'(x) = [(p_1, q_1), ..., (p_K, q_K), (x, y), ..., (x, y)],$$

where first K pairs are the K-most similar pairs (excluding (x, y) itself), and (x, y) is repeated $K' < K$ times (K' is another hyperparameter of the model). Since the model sees the (x, y) pair $\frac{K'}{K+K'}$ times during training as the retrieved pair, and these particular pairs include the output y themselves, the model is encouraged to use information coming from the retrieved neighboring pairs more often.

A.2 Analysis of Varying K

We conduct an experiment to evaluate the effect of the hyper-parameter K in the proposed method. For each value of $K \in \{1, 3, 5\}$, we train our model once and obtain its results on the Quora dataset. Then, the value of two quality metrics (i.e. BLUE-2 and ROUGE-2) and two diversity metrics (i.e. SelfBLEU-2 And PINC-4) are computed. Figure 6 summarizes the obtained results. According to this figure, increasing the value of K slightly decreases the quality metrics while highly increases the diversity measures (Note that lower values of SELf-BLEU and higher values of PINC indicate more diversity in the outputs). It shows that incorporating wider neighborhood in the editing process results in more diversity in the paraphrasing made by the editor.

A.3 Human Evaluation

The form used for the one-by-one experiment contains the following material:

Which set of the outputs do you prefer? Please opt based on the following criteria:

- Novelty: If any one of the outputs has expressed the semantic content of source sentence in a novel way like what we do as humans (e.g. changing the voice from active to passive or vice versa, using different words, phrases, or sentences that are not present in the source sentence but without changing the meaning considerably).

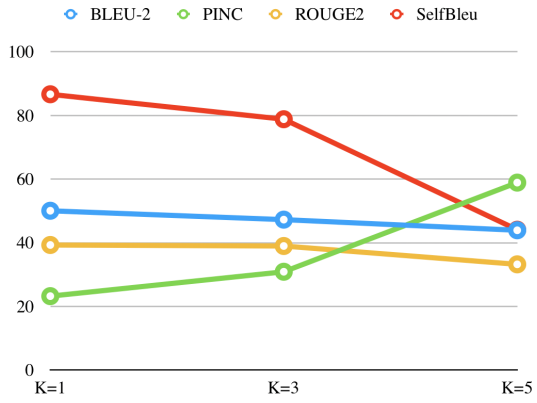

Figure 6: Results of the proposed method with some different values of the hyperparameter K according to four different metrics

- Diversity: if the three outputs are not expressed in the same way using the same words.
- Quality: If the outputs are paraphrases of the input sentence to a good extent.
- Readability: If the outputs are understandable by humans.

Note: Please select the option 'both(#1 and #2)' if you can not decide which one is better.

A.4 Example outputs

Table 6 shows the paraphrases generated for more sample inputs from the Quora dataset.

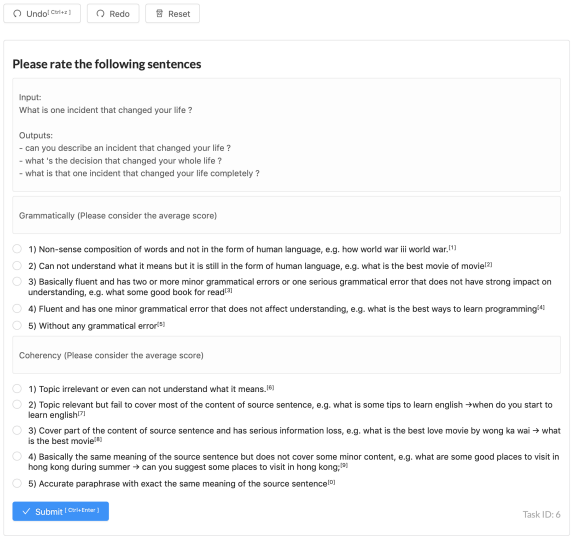

Figure 7: The human evaluation form for the first experiment.

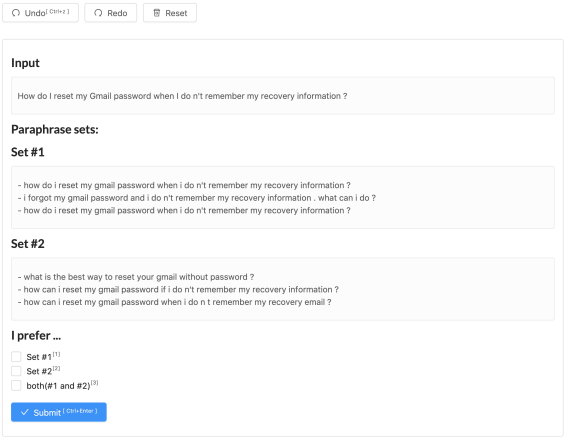

Figure 8: The human evaluation form for the second experiment.

(1)	x	What are great examples of absent mindedness ?
	p	What are the common characteristics of ignorant people ?
	q	What are the most common characteristics of stupid , ignorant or uneducated people ?
	y'	what are the most common examples of absent mindedness ?
(2)	x	What should I know about moving from Pennsylvania to South Carolina ?
	p	What are the pros and cons of living in Richmond , VA ?
	q	What is your review of Richmond , VA ?
	y'	what is your review of moving from pennsylvania to south carolina ?
(3)	x	How does Quora determine how many views an answer gets ?
	p	How does Quora calculate how many views does an answer have ?
	q	How does Quora count the views of my / your answers ?
	y'	how does quora determine the views of your answers ?
(4)	x	Do you think India should revisit Indus Water Treaty with Pakistan ?
	p	Should India hold a referendum in Kashmir ?
	q	Would India ever consider a referendum for Kashmir independence ?
	y'	would india ever consider a indus water treaty with pakistan ?
(5)	x	Which is your favourite time of the day ?
	p	What is your favorite month of the year ?
	q	What is your favorite month of the year and why ?
	y'	what is your favourite time of the day and why ?
(6)	x	How do I homeschool my kids ?
	p	How do I teach my child ?
	q	What are the best ways to teach your child ?
	y'	what are the best ways to homeschool my child ?
(7)	x	How did Andy Warhol paint Marilyn Monroe ?
	p	Who was Mona Lisa and why was she painted ?
	q	What is story behind mona lisa painting ?
	y'	what is the story behind andy warhol paint marilyn monroe ?
(8)	x	Can we start a Sainik School in the private sector ?
	p	Is n't it time for Indian goverment to change its reservation system for jobs and education from caste based to economic based system ?
	q	Is n't it time India change its reservation policies from implemented on caste basis to based on financial backgrounds ?
	y'	is it possible to start a sainik school in the private sector ?
(9)	x	Can we start a Sainik School in the private sector ?
	p	Should the Indian education system be revamped ?
	q	Does the Indian education system need to change and why ?
	y'	does the indian government need to start a sainik school in the private sector ?
(10)	x	What is the best thing any one has ever done for you ?
	p	What is the nicest thing anyone has ever done for you ?
	q	What 's the nicest thing anyone has ever done for you ?
	y'	what 's the best thing that has ever done for you ?
(11)	x	What are some of the things people like or dislike in general ?
	p	What are some things that you hate ?
	q	What are some things that irritate you and that you hate ?
	y'	what are some things that most people dislike in general ?

Table 6: More examples of our model's outputs