 doi:10.20944/preprints202103.0754.v1

Article

Augmenting Paraphrase Generation with Syntax Information using Graph Convolutional Networks

Xiaoqiang Chi [1] and Yang Xiang [2,*]

[1] College of Electronics and Information Engineering, Tongji University; chixiaoqiang@foxmail.com
[2] College of Electronics and Information Engineering, Tongji University; tjdxxiangyang@gmail.com
* Correspondence: tjdxxiangyang@gmail.com

Abstract: Paraphrase generation is an important yet challenging task in NLP. Neural network-based approaches have achieved remarkable success in sequence-to-sequence(seq2seq) learning. Previous paraphrase generation work generally ignores syntactic information regardless of its availability, with the assumption that neural nets could learn such linguistic knowledge implicitly. In this work we make an endeavor to probe into the efficacy of explicit syntactic information for the task of paraphrase generation. Syntactic information can appear in the form of dependency trees which could be easily acquired from off-the-shelf syntactic parsers. Such tree structures could be conveniently encoded via graph convolutional networks(GCNs) to obtain more meaningful sentence representations, which could improve generated paraphrases. Through extensive experiments on four paraphrase datasets with different sizes and genres, we demonstrate the utility of syntactic information in neural paraphrase generation under the framework of seq2seq modeling. Specifically, our GCN-enhanced models consistently outperform their syntax-agnostic counterparts in multiple evaluation metrics.

Keywords: paraphrase generation; syntax information; Graph Convolutional Network; sequence-to-sequence

1. Introduction

Paraphrase generation is the task of restate a sentence with different wording while keeping the same semantic meaning. Since variation is a characteristic of language, paraphrase generation systems could be a key component in numerous natural language processing tasks[1,2]. For instance, paraphrase generation could be used to expand patterns in Information Extraction systems[3], reformulate queries for Information Retrieval[4,5], generate diverse text in question answering[6] and dialog systems. It also finds applications in semantic parsing[7], summarization[8] and sentence simplification[9,10]. More generally, it could be employed as a data augmentation technique to improve robustness of NLP models against language variation. Recent years have witnessed great successes in NMT and related generation tasks that could be formulated under the sequence-to-sequence learning framework. As a result, the field of paraphrase generation is also gaining attention.

Previous work employs the seq2seq model for the task of paraphrase generation[11], where the authors stack multiple LSTM layers with residual connections. Some adopt more complex architures: [12] use multiple decoders. Another line of research resorts to NMT, where a source sentence is first translated into a pivot language, then the translated sentence is back-translated into the same language as the source sentence. All the above approaches do not take linguistic knowledge into consideration. [13,14] utilize lexical resources in the form of synonym dictionaries. [15] make use of semantic information which is represented as PropBank[16] style frame-semantic labels. [17] exploit linguistic knowledge at the syntactic level, where constituency parses are fed into their models. [18,19] do not use explicit syntactic parses, but adopt an example sentence

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

whose underlying syntax is regarded as a syntactic exemplar. Inspired by previous work employing GCN to encode dependency trees in learning sentence representations, we make an attempt to integrate linguistic knowledge in the form of dependency parses via GCNs into the process of paraphrase generation.

2. Background

In this section, we will provide a brief description of two component models that are necessary to understand the method proposed in this work.

2.1. Sequence-to-Sequence Models and Attention

Neural machine translation is typically conducted by building a neural network which takes a source sentence as input and generates its corresponding sentence in the target language. The network is made up with two component networks, namely the encoder and the decoder. An encoder-decoder network is also called a sequence-to-sequence model[20].

Formally, a source sentence $X = (x_1, ..., x_J)$ is fed into the encoder to give a fixed-length hidden vector, presumably encoding the "meaning" of the sentence. This hidden vector is a continuous representation often referred to as the context vector, and it is the last hidden state in the case of the encoder being a RNN, which is the most widely used network architecture. Then the target sentence $Y = (y_1, ..., y_I)$ is produced one symbol at a time, conditioning on the context vector provided by the encoder, where I is not necessarily equal to J. Essentially, the decoder is a conditional language model: given the context vector c and previously predicted tokens $y_{1:i-1}$, the decoder is trained to generate the next token y_i. In other words, the decoder models a distribution over possible translation sentences that can decompose into a series of individual conditionals:

$$p(Y) = \prod_{i=1}^{I} p(y_i | y_{1:i-1}, c)$$

Each conditional probability can be defined as

$$p(y_i | y_{1:i-1}, c) = g(y_{i-1}, s_i, c)$$

where g is a nonlinear function and s_i is the hidden state for step i in RNN.

It is hard for the encoder to learn a fixed dimensional vector that contains the entire "meaning" of the input sentence, whose length may vary wildly. This is where the *attention*[21] mechanism comes into play. In their model, the context vector varies with time i, hence the probability of y_i is conditioned on a different context vector c_i for each prediction.

$$p(y_i | y_{1:i-1}, c) = g(y_{i-1}, s_i, c_i)$$

where

$$s_i = f(s_{i-1}, y_{i-1}, c_i)$$

The context vector c_i is a weighted sum of all encoder hidden states, where each weight α_{ij} is the amount of "attention" paid to the corresponding encoder state h_j

$$c_i = \sum_{j=1}^{J} \alpha_{ij} h_j$$

where each weight α_{ij} is a normalized (over all steps) attention "energy" e_{ij}

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{J} \exp(e_{ik})}$$

where each attention energy is calculated with some function a (such as another linear layer) using the last hidden state s_{i-1} and that particular encoder output h_j:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

$$e_{ij} = a(s_{i-1}, h_j)$$

2.2. Graph Convolutional Networks

For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes (vertices) and $\mathcal{E}$ is a set of edges (connections). Assume there are n nodes in the graph, each node is associated with a feature vector of dimensionality d. As we are concerned with modeling sentences, the feature vectors will simply be word embeddings. Then all node features could be represented by a matrix $X \in \mathbb{R}^{d \times n}$. For the ith node u, the corresponding feature vector $\mathbf{x}_u \in \mathbb{R}^d$ will be column i of matrix X. A GCN network takes two representations as input:

- node representation: feature descriptions for all nodes summarized as the feature matrix X;
- graph structure representation: the topology of the graph which is typically represented in the form of an adjacency matrix A.

In the first layer of a multi-layer GCN network, the node representations are computed as:

$$\mathbf{h}_v = f\left(\sum_{u \in \mathcal{N}(v)} W\mathbf{x}_u + \mathbf{b}\right) \tag{1}$$

where $W \in \mathbb{R}^{d \times d}$ is a weight matrix and $\mathbf{b} \in \mathbb{R}^d$ is a bias vector. W is called filter or kernel in convolution networks. Essentially, W and $\mathbf{b}$ together constitute a linear neural network. f denotes an activation function such as `ReLU`. $\mathcal{N}(v)$ stands for the set of neighbor nodes of v, which could be obtained via the adjacency matrix A.

Through one GCN layer node representations could be updated by aggregating information from immediate neighbors. By stacking GCN layers we could allow information to propagate through multiple hops. Computation of node representations becomes recursive, for the kth layer we have:

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v)} W^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{2}$$

for the first layer $k = 1$ and $\mathbf{h}_u^{(k-1)} = \mathbf{h}_u^{(0)} = \mathbf{x}_u$.

2.2.1. Syntactic GCNs

Dependency trees could be formulated as directed and labeled graphs. Marcheggiani and Titov[22] generalize GCNs to take directionality and labels into account. For each node v there are three possible edge directions: incoming edges(e.g., $u \to v$), outgoing edges(e.g., $v \to u$) and self-loop($v \to v$). By making weight matrices and bias vectors in a GCN network label-specific we have:

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u \in \mathcal{N}(v)} W_{\text{lab}(u,v)}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{b}_{\text{lab}(u,v)}^{(k)}\right) \tag{3}$$

where $\text{lab}(u, v)$ determines the weight matrix corresponding to each direction and label combination.

Syntactic GCNs exploit the concept of gating for the graph edges. By modulating the contribution of individual edges, it allows the model to differentiate between information flowing along each edge: some edges could contain information more pertinent to

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

the task while others could possibly represent erroneous predicted syntactic structure. Formally, a scalar gate is calculated for each edge as follows:

$$g_{u,v}^{(k)} = \sigma\left(\mathbf{h}_u^{(k-1)}\hat{\mathbf{w}}_{\text{lab}(u,v)}^{(k)} + \hat{b}_{\text{lab}(u,v)}^{(k)}\right) \tag{4}$$

where σ is the logistic sigmoid function, $\hat{\mathbf{w}}_{\text{lab}(u,v)}^{(k)} \in \mathbb{R}^d$ and $\hat{b}_{\text{lab}(u,v)}^{(k)} \in \mathbb{R}$ are training parameters for the gate. Putting it all together, the computation for Syntactic GCNs becomes:

$$\mathbf{h}_v^{(k)} = f\left(\sum_{u\in\mathcal{N}(v)} g_{u,v}^{(k)}\left(W_{\text{lab}(u,v)}^{(k)}\mathbf{h}_u^{(k-1)} + \mathbf{b}_{\text{lab}(u,v)}^{(k)}\right)\right) \tag{5}$$

3. Methods

We focus on the case where dependency information on the source side is available and hypothesize that by integrating syntax information with GCNs, the encoder would learn more meaningful sentence representations. We hope such enriched representation, when fed into the decoder, would lead to improved quality in generated paraphrases. For RNN networks we employ GRU units. Now we describe the encoder and the decoder respectively.

3.1. Encoder

The encoder is composed by three modules: an embedding module, a BiRNN module and a GCN module. The embedding module is essentially a lookup table which turns word indices into corresponding dense vectors. word vectors are then processed by BiRNN networks to obtain a sentence represention. This sentence vector is then refined by the GCN module that follows. The GCN network takes syntax information in the form of adjacency matrix as input. The adjacency matrix for each sentence remains constant in the process of model training. Let $\mathbf{x}_j$ denote the embedding vector for word j. hidden vectors output by the forward GRU and backward GRU are computed as:

$$\overrightarrow{\mathbf{h}}_j = \text{GRU}_f(\mathbf{x}_j, \overrightarrow{\mathbf{h}}_{j-1})$$

$$\overleftarrow{\mathbf{h}}_j = \text{GRU}_b(\mathbf{x}_j, \overleftarrow{\mathbf{h}}_{j+1})$$

where GRU_f and GRU_b denote the forward GRU and the backward GRU respectively. Then the two hidden vectors are concatenated as a single vector:

$$\mathbf{h}_j = \overrightarrow{\mathbf{h}}_j \bigoplus \overleftarrow{\mathbf{h}}_j$$

which is taken as input for the GCN module:

$$\tilde{\mathbf{h}}_j = \text{GCN}(\{\mathbf{h}_j\}_{j=1}^J, A)$$

where A represents the adjacency matrix determining the neighborhood of node j. Here we assume the length of source sentence is J.

3.2. Decoder

For the decoder part we also adpot a GRU network. Let s_i denote the hidden state vector at time step i. A context vector is computed by an attention module, taking the set of all encoder hidden states and the previous decoder hidden state:

$$c_i = \text{Attention}(\{\tilde{\mathbf{h}}_j\}_{j=1}^J, s_{i-1})$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

the context vector is concatenated with the previous target word vector $\mathbf{y}_{i-1}$ and fed into a GRU:

$$s_i = \text{GRU}(s_{i-1}, c_i \bigoplus \mathbf{y}_{i-1})$$

at last these three vectors are passed through a linear layer followed by a softmax function to produce a probability for the next word:

$$p(\mathbf{y}_i) = \text{Softmax}(\text{Linear}(s_i \bigoplus c_i \bigoplus \mathbf{y}_{i-1}))$$

At inference time, we use a greedy decoder, selecting the output token with the highest probability at each time step. An overview of the model architecture is illustrated in Figure 1.

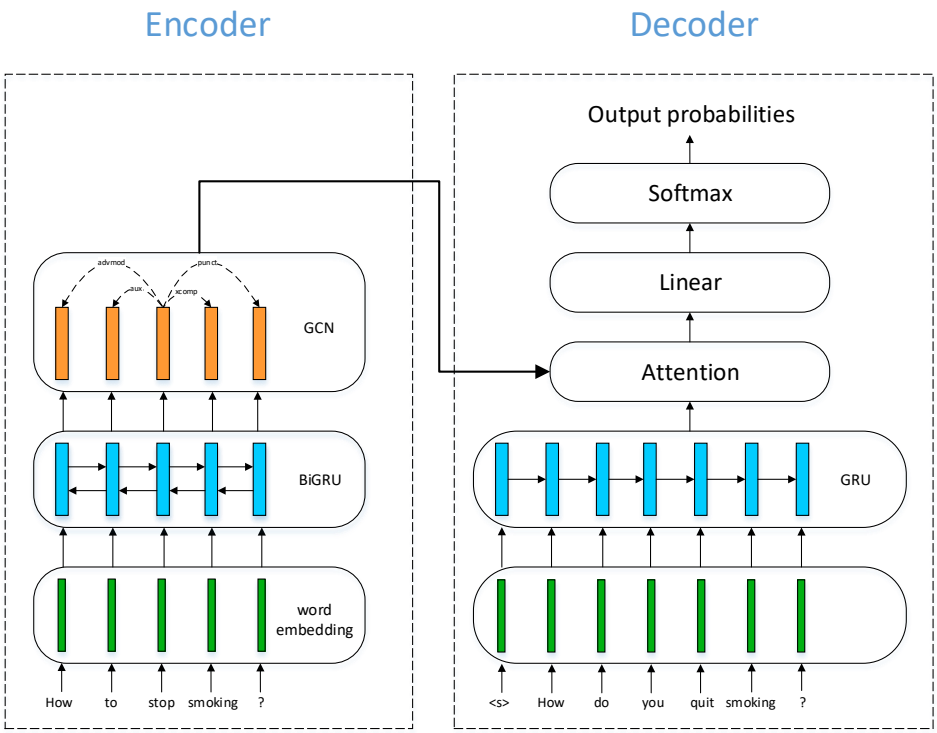


Figure 1. Model architecture. Vectors are depicted as colored rectangles, including word embeddings and hidden vectors learned by neural nets. Arrows denote information flow. For the GCN part we also show the dependency relations for the example input sentence as dashed arrows. The specific dependency types are annotated on the corresponding arrows.

4. Experiments

Experiments are performed with PyTorch[1] version 1.4.0, and we use torchtext(version 0.6.0) which is shipped with PyTorch that provide a range of convenient text processing utilities. We employ spaCy(version 2.3.2) to perform tokenization and dependency parsing, the spaCy model we use is `en-core-web-sm-2.3.1`. Encoder-decoder code is based on [23]. GCNs[2] are also implemented with PyTorch.

[1] https://pytorch.org/

[2] Source code for the GCN part is available at https://github.com/chifish/SyntacticGCN.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

4.1. Datasets

Following previous work on the task of paraphrase generation, we use two widely investigated paraphrase datasets, namely Quora[3] and ParaNMT[24]. The Quora dataset, released in January 2017, is originally developed for classifying whether question pairs have the same intent. Each line in the dataset consists of question numbers(IDs), followed by the text of each question and a binary value given by human annotators indicating whether the pair are considered duplicate or not. If the label is "1", the question pair is indeed paraphrases of each other. We further split the Quora dataset into three datasets with increasing sizes: Quora50K, Quora100K, and Quora150K[4], where each dataset is a subset of a larger dataset. The minimum frequency for building vocabulary is set to 3 for these three Quora datasets.

The ParaNMT dataset is constructed by translating the Czech side of a large English-Czech parallel corpus into English using NMT. A paraphrase is formed by pairing the English translation with the corresponding reference sentence. The main purpose of creating this large paraphrase corpus is to learn sentence representations whose superority is manifested in a semantic textual similarity task. Yet it is also shown to be useful in paraphrase generation tasks. A score is associated with each paraphrase pair indicating the level of similarity. The scores are divided into five ranges. After manually checking a random sample from the highest score range, we find that a large portion reveals a remarkable level of lexical overlap between sentence pairs. We believe this phenomenon is not desirable in generated paraphrases since generally (at least lexically) divergent sentences are deemed to be more interesting and useful. So we choose the second-highest score range (0.6–0.8) for this work. The dataset exhibits a high level of noise in various forms, so we filter out noisy sentences[5]. Finally, we got 2.3 million sentence pairs. When building the vocabulary with the training set, we keep those tokens that appear at a frequency of at least 10, resulting in a vocabulary size of about 50K.

Previous work truncates sentences in both datasets at the length of 15. This procedure would result in incomplete graph structures for models utilizing GCNs. So it is necessary to keep whole sentences. Filtering out sentences with lengths beyond 15 would result in a substantial loss in data volume. We decide to reserve sentences with a maximum length of 30 for all four datasets. We randomly sample 2K sentence pairs for development and 10K for test respectively.

4.2. Training Configuration

We use the Adam optimizer[25] with a initial learning rate of 0.001. We utilize early-stopping[26] to monitor training and mitigate overfitting on training data. The model with the highest validation BLEU is saved for testing. Word embedding size is set to 256, hidden size is set to 128. Batch size is set to 128, sentences of similar length are bucketed together to minimize padding and hence increase training efficiency. This bucketing procedure is realized through torchtext. Between layers we apply dropout with a probability of 0.2. In experiments with GCNs we also use the value of 0.2 for edge dropout. These dropout rates are tuned on the validation set. To mitigate the effect of randomness on model performance, we run each model five times, each time with a random seed chosen from 42–46. The performance scores are averaged across five runs. Considering the wide gap between the sizes of the Quora dataset and the ParaNMT dataset, we adopt different configurations in several aspects, which we introduce below in detail.

[3] https://www.kaggle.com/c/quora-question-pairs

[4] There are actually 134k sentence pairs in this dataset, we keep this naming to be consistent with previous work.

[5] Script available at https://github.com/chifish/preprocess

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

- **Quora**
  For the three Quora datasets, evaluation is performed on the validation set at the end of each epoch. The learning rate is halved if the validation loss do not decrease for two consecutive evaluations. Stopping criterion is BLEU score and the tolerance number is set to 5, so training will be terminated if BLEU scores on 5 consecutive evaluations on the development set do not improve. Since the Quora datasets are relatively small, we use one layer in both encoder(bidirectional) and decoder.
- **ParaNMT**
  For the ParaNMT dataset we evaluate the model every 2000 training steps. The criterion adopted for early-stopping is BLEU and the tolerance number is set to 10 which is the minimum to cover an entire epoch. The layer depths of both encoder and decoder are set to 2 where the encoder is again bidirectional.

4.3. Evaluation Metrics

As opposed to NMT, there is no consensus on what metrics should be used for evaluating paraphrase generation models. As a result, measurements adopted by different works vary. In this work, we report model performance on multiple metrics to provide a more comprehensive understanding.

- **BLEU**[27] counts the overlap of sentence fragments in reference translations and the candidate translation output by NMT systems. Assume there are two reference translations and a candidate translation. For each word type in the candidate, the number of times it occurs in both reference sentences are computed and the maximum of these two counts is taken as an upper bound. Then, the total count of each candidate word is clipped by this upper bound. Next, these clipped counts are summed up. Finally, this sum is divided by the total (unclipped) number of candidate words. This is the case for single words or unigrams. The score for other n-grams is computed similarly. Typically n ranges from 1 to 4.
- **METEOR**[28] calculates unigram matching in a generalized fashion. For unigrams, BLEU only takes surface form into account. In contrast, METEOR also matches stem, synonym, and paraphrase between candidate and reference.
- **ROUGE**[29] is commonly employed in the evaluation of automatically generated summaries. Yet it is also used to assess paraphrases. There are four types of ROUGE, among which ROUGE-n that deals with n-grams is most pertinent to paraphrase evaluation. Assume there are two reference sentences. For each n-gram in each of the two reference sentences, the number of times the n-gram occurs in the candidate is calculated. The maximum of these two counts is kept. Then the maximum counts for each n-gram are summed. Which is then divided by the total number of counts of n-grams in the references. We also report results with another type of ROUGE: ROUGE-L. It operates on Longest Common Subsequence. The ROUGE scores presented in this work are F-measure values, which is a trade-off between precision and recall.

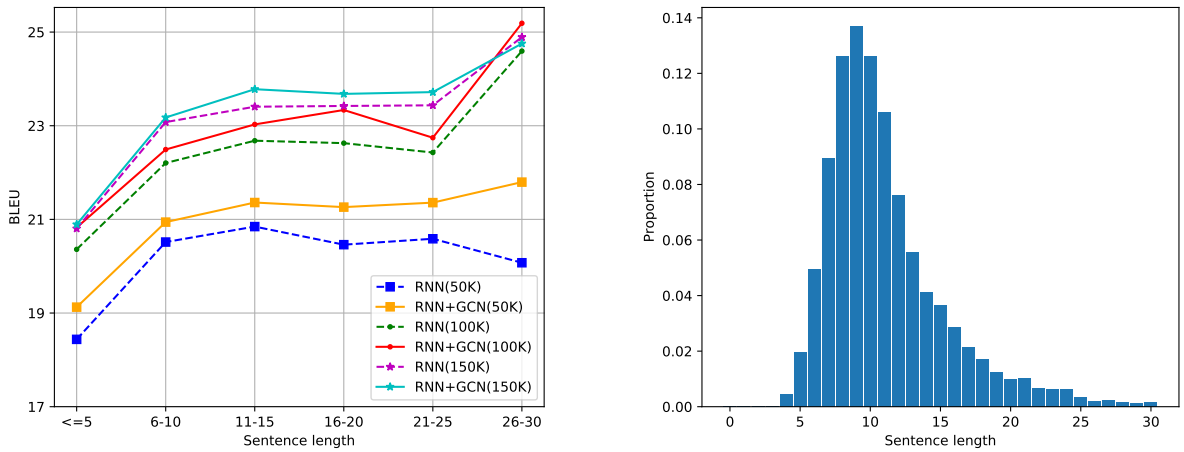
5. Results and Discussion

Now we report the experiment results for all four datasets.

5.1. Prediction Scores

We evaluate paraphrases generated by our models given test set source sentences as input. Scores are computed between prediction sentence and ground-truth pairs on the corpus level. We present the average scores across five random runs for each model and dataset combination. The set of random seeds are repeated for each combination. Standard deviations are shown in parentheses that follow. Significance test is performed against RNN. "†": significantly better than RNN ($p < 0.05$). "††": significantly better than RNN ($p < 0.01$).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1



(a) Validation BLEU per sentence length. **(b)** Length distribution

Figure 2. Effect of sentence length for Quora datasets. (a) The relationship between BLEU score and sentence length on three Quora datasets. The sizes of datasets are given in parentheses. (b) Distribution of sentence length for Quora50K, the patterns for the two larger Quora datasets are similar.

As can be seen in the table, our GCN-enhanced models almost uniformly outperform the RNN baseline models at a significance level of $p < 0.01$. The only exception is BLEU score for Quora150K.

Table 1. Test performance in five metrics.

dataset	model	BLEU	METEOR	ROUGE-1	ROUGE-2	ROUGE-L
Quora50K	RNN	20.57(±0.12)	47.3(±0.14)	48.63(±0.15)	28.12(±0.15)	46.84(±0.13)
	RNN+GCN	21.09(±0.07)††	48.41(±0.1)††	49.84(±0.1)††	28.93(±0.09)††	48.0(±0.1)††
Quora100K	RNN	22.38(±0.08)	49.85(±0.07)	51.64(±0.12)	30.57(±0.08)	49.56(±0.09)
	RNN+GCN	22.74(±0.1)††	50.59(±0.1)††	52.72(±0.09)††	31.42(±0.09)††	50.67(±0.11)††
Quora150K	RNN	23.19(±0.06)	50.89(±0.05)	52.86(±0.07)	31.71(±0.02)	50.7(±0.06)
	RNN+GCN	23.4(±0.13)†	51.4(±0.19)††	53.68(±0.11)††	32.31(±0.16)††	51.53(±0.11)††
ParaNMT	RNN	16.15(±0.37)	43.32(±0.49)	46.66(±0.34)	22.58(±0.35)	43.48(±0.33)
	RNN+GCN	17.09(±0.15)††	44.52(±0.27)††	47.65(±0.19)††	23.5(±0.18)††	44.49(±0.2)††

5.2. Effect of Sentence Length

We expect that GCN networks would show a larger advantage for longer sentences since they contain long-distance syntactic dependencies that would be challenging for RNNs to model but can be easily captured via syntactic connections encoded by GCNs. To verify this hypothesis, we split the sentences predicted by models trained on each Quora dataset into six buckets and compute BLEU scores for each separate bucket. Figure 2a shows that GCNs do outperform syntax-agnostic RNN models by a larger margin for longer sentence on the Quora50K dataset. For Quora100K the advantage of GCN models is relatively uniform. For Quora150K the distinction between two models is marginal when sentence length goes to two extremes. One explanation is data sparsity for those two length ranges. As can be seen in Figure 2b most sentences fall into two buckets(6–10 and 11–15), which account for 53% and 31% respectively, for all three datasets. Sentences which contain no more than 5 words take up a share of only 2.4% and sentences fall into the longest bucket is most scarce, with a low proportion of less than 1%. We also draw the standard deviation of each performance scores, the variation for buckets at both ends are indeed much higher.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

5.3. Discussion

Splitting the Quora dataset into different sizes allows us to study another dimension of GCNs: the effect of dataset size on model performance. A close look at Figure 2a tells us that as the dataset grows, the performance gains of GCN models over baseline RNNs declines. This aspect can also be deduced from Table 1, where smaller datasets show a higher level of significance difference between the two model architectures. To be specific, the p-values(lower values signify higher significance levels) for the three datasets are in the order of magnitude of 10^{-5}, 10^{-4} and 10^{-2} respectively. Our hypothesis is that when given sufficient data, RNNs could model some aspects of syntactic structure, hence the advantage of GCN-based models gradually disappears. This is presumably the case for the Quora dataset which mainly contains questions with simple syntactic patterns. Work by Bastings et al.[30] involved datasets of differing sizes. Specifically, for the English-German machine translation task, they investigated two dataset sizes: 226K and 4.5M. The BLEU_1 score gain they monitored dropped from 2.3 to 1.6, and the gain in BLEU_4 dropped from 1.2 to 0.6, respectively.

We expected GCN models to outperform RNNs by a narrower margin as the dataset size rises to the order of millions. Surprisingly, the BLEU score margin for GCN models increases to nearly 1 point on the ParaNMT dataset with over 2 million training samples. This again testifies the efficacy of GCN-based models for encoding syntax information.

6. Related Work

6.1. Linguistic Aware Methods for Paraphrase Generation

Various researchers exploit linguistic knowledge in a paraphrase generation model, be it lexical or syntactic or semantic knowledge. [13] make use of information from an off-the-shelf dictionary. Specifically, they extract appropriate word-level and phrase-level paraphrase pairs from the PPDB database, while taking the original sentence into account. Instead of naively replacing words in the source sentence with their counterpart in the paraphrase pairs, they use these paraphrase pairs to construct edit vectors. Edit vectors are responsible for deletion and insertion operations in paraphrase generation.

[14] also employ an off-the-shelf dictionary containing word-level paraphrase pairs(synonyms). In addition to using these synonym pairs to guide the decision on whether to generate a new word or replace it with a synonym as in [13], they integrate information of word location with a positional encoding layer in Transformer. An interesting part of their work is that they formulate the locating of synonym candidates as a synonym labeling task, which is first trained independently. Then the synonym labeler and the paraphrase generator are trained simultaneously to perform multi-task learning, where the two tasks share a common encoder.

[31] propose to interact with distributed word representations instead of the corresponding words per se. The hidden vector from a neural network is used as a query. A dictionary is constructed which is basically a word embedding lookup table with its keys and values swapped. When the network is making predictions, a matching score is computed between the query and the embedding keys, the key obtaining the highest score is retrieved and the associated value(word) is returned. Hence the model generates the words in a retrieval style. Yet they do not experiment with standard paraphrase generation but two closely related tasks: text simplification and short text abstractive summarization.

Another line of research takes advantage of linguistic knowledge at the syntactic level. [17] propose to incorporate linearized constituency parses from paraphrase pairs to control the syntax of generated paraphrases. In this way, their model learns the syntactic transformations that naturally occur in paraphrase data. Samples produced by the method could fool traditional paraphrase detection models due to syntactic variations. Thus such adversarial examples could be used to augment the training data for paraphrase detection models so as to improve their robustness.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 March 2021 doi:10.20944/preprints202103.0754.v1

[18] adopt a syntactic exemplar sentence as an alternative to the explicit target syntactic form used in [17]. The syntactic exemplar is composed such that its semantics deviate from the input sentence to be paraphrased. The generated paraphrase follows the semantics of the first sentence and syntax of the second sentence(syntactic exemplar).

Similarly, [19] also used a syntactic exemplar to conduct paraphrase generation with controlled syntax. Their model utilizes full exemplar syntactic tree information and is capable of regulating the granularity level of syntactic control.

[15] add semantic information in their paraphrase generation model. PropBank style semantic information is obtained through an off-the-shelf frame-semantic parser. Each token in an input sentence is associated with a frame and a role label, resulting in three input channels which are fed into three separate encoders. The results are then aggregated with a linear layer and fed into the decoder to generate a paraphrase.

6.2. GCN for NLP

Graph Convolutional Networks has been applied in a range of NLP problems. Here we briefly overview some relevant works employing syntactic GCNs. Marcheggiani and Titov[22] proposed to adopt syntactic GCNs for semantic role labeling. Word embeddings are first fed into a BiLSTM network to obtain hidden word representations. Then these hidden word vectors are re-encoded by the following GCN network. Finally, the hidden vectors are passed to a classifier to predict a semantic label.

Vashishth et al.[32] employ syntactic GCNs to learn word embeddings. They generalize the continuous-bag-of-words (CBOW)[33] model by substituting the sequential context (words within a fixed window size) with a syntactic context (neighbor words in a dependency tree). By exploiting syntax information in selecting and weighting context words, their approach produced more meaningful word representations.

Bastings et al.[30] proposed to use syntactic GCNs for English-German and English-Czech machine translation. They fixed the decoder as a recurrent network and explored baseline models in combination with GCN to compose the encoder. Specifically, the baselines they examined are a bag-of-words encoder, a recurrent encoder, and a convolutional encoder. When stacking a GCN network on each of the baseline encoders, significant performance gains were observed for both translation tasks.

7. Conclusions

In this paper, we proposed to encode syntax information for paraphrase generation via graph convolutional networks. GCNs allow us to encode sentences via dependency trees. By stacking GCNs on top of recurrent networks on the encoder side of our paraphrasing model, we achieved significantly better results on four paraphrase datasets of varying sizes and genres. We hypothesize that dependency relations contain rich syntactic information which is valuable in learning sentence representation. Besides, convolutional operations in GCNs have complementary power to recurrent connections in RNNs. Through analyzing the experiment results in multiple evaluation metrics on a range of paraphrase datasets, we demonstrated the efficacy of our approach. We also studied the effects of sentence length and size of training data on model performance, and find that recurrent networks enhanced by GCNs are more effective for smaller datasets when the dependency patterns are less diverse. But for sentences that are either too short or too long, it becomes challenging for GCNs to learn more meaningful representations. This is mainly due to data sparsity. We plan to apply GCNs in other text generation tasks such as question answering and dialog systems in future work.

Author Contributions: Conceptualization, Y.X. and X.C.; methodology, X.C.; software, X.C.; validation, X.C.; writing—original draft preparation, X.C.; writing—review and editing, X.C.; visualization, X.C.; supervision, Y.X.; project administration, Y.X.; funding acquisition, Y.X.

Funding: This work is supported by General Program of National Natural Science Foundation of China under Grant No.72071145.

Data Availability Statement: The datasets investigated in this work are publicly available at https://www.kaggle.com/c/quora-question-pairs and https://drive.google.com/file/d/1rbF3daJjCsa1-fu2GANeJd2FBXos1ugD/view

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NLP	Natural Language Processing
GCN	Graph Convolutional Network
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
NMT	Neural Machine Translation
CBOW	continuous-bag-of-words

References

1. Androutsopoulos, I.; Malakasiotis, P. A survey of paraphrasing and textual entailment methods. *Journal of Artificial Intelligence Research* **2010**, *38*, 135–187.
2. Madnani, N.; Dorr, B.J. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics* **2010**, *36*, 341–387.
3. Shinyama, Y.; Sekine, S. Paraphrase acquisition for information extraction. Technical report, NEW YORK UNIV NY DEPT OF COMPUTER SCIENCE, 2003.
4. Wallis, P. Information retrieval based on paraphrase. Proceedings of PACLING Conference. Citeseer, 1993.
5. Yan, Z.; Duan, N.; Bao, J.; Chen, P.; Zhou, M.; Li, Z.; Zhou, J. Docchat: An information retrieval approach for chatbot engines using unstructured documents. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2016, pp. 516–525.
6. Fader, A.; Zettlemoyer, L.; Etzioni, O. Open question answering over curated and extracted knowledge bases. Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 1156–1165.
7. Berant, J.; Liang, P. Semantic parsing via paraphrasing. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 1415–1425.
8. Zhang, C.; Sah, S.; Nguyen, T.; Peri, D.; Loui, A.; Salvaggio, C.; Ptucha, R. Semantic sentence embeddings for paraphrasing and text summarization. 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP). IEEE, 2017, pp. 705–709.
9. Zhao, S.; Meng, R.; He, D.; Andi, S.; Bambang, P. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193* **2018**.
10. Guo, H.; Pasunuru, R.; Bansal, M. Dynamic Multi-Level Multi-Task Learning for Sentence Simplification. Proceedings of the 27th International Conference on Computational Linguistics, 2018, pp. 462–476.
11. Prakash, A.; Hasan, S.A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; Farri, O. Neural Paraphrase Generation with Stacked Residual LSTM Networks. Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, 2016, pp. 2923–2934.
12. Cao, Z.; Luo, C.; Li, W.; Li, S. Joint copying and restricted generation for paraphrase. Thirty-First AAAI Conference on Artificial Intelligence, 2017.
13. Huang, S.; Wu, Y.; Wei, F.; Luan, Z. Dictionary-guided editing networks for paraphrase generation. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, Vol. 33, pp. 6546–6553.
14. Lin, Z.; Li, Z.; Ding, N.; Zheng, H.T.; Shen, Y.; Wang, W.; Zhao, C.Z. Integrating Linguistic Knowledge to Sentence Paraphrase Generation. AAAI, 2020, pp. 8368–8375.
15. Wang, S.; Gupta, R.; Chang, N.; Baldridge, J. A task in a suit and a tie: paraphrase generation with semantic augmentation. Proceedings of the AAAI Conference on Artificial Intelligence, 2019, Vol. 33, pp. 7176–7183.
16. Palmer, M.; Gildea, D.; Kingsbury, P. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics* **2005**, *31*, 71–106.
17. Iyyer, M.; Wieting, J.; Gimpel, K.; Zettlemoyer, L. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 1875–1885.
18. Chen, M.; Tang, Q.; Wiseman, S.; Gimpel, K. Controllable Paraphrase Generation with a Syntactic Exemplar. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5972–5984.
19. Kumar, A.; Ahuja, K.; Vadapalli, R.; Talukdar, P. Syntax-guided Controlled Generation of Paraphrases. *arXiv preprint arXiv:2005.08417* **2020**.

 doi:10.20944/preprints202103.0754.v1

20. Sutskever, I. Sequence to Sequence Learning with Neural Networks. 27th Conference on Neural Information Processing Systems (NIPS 2014), 2014, pp. 1–9.
21. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 3rd International Conference on Learning Representations (ICLR 2015), 2015, pp. 1–15, [arXiv:1409.0473v7].
22. Marcheggiani, D.; Titov, I. Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 1506–1515.
23. Bastings, J. The Annotated Encoder-Decoder with Attention, 2018.
24. Wieting, J.; Gimpel, K. ParaNMT-50M: Pushing the Limits of Paraphrastic Sentence Embeddings with Millions of Machine Translations. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 451–462.
25. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. 3rd International Conference on Learning Representations (ICLR 2015), 2015.
26. Caruana, R.; Lawrence, S.; Giles, C.L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Advances in neural information processing systems, 2001, pp. 402–408.
27. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: a method for automatic evaluation of machine translation. Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.
28. Banerjee, S.; Lavie, A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, 2005, pp. 65–72.
29. Lin, C.Y. Rouge: A package for automatic evaluation of summaries. Text summarization branches out, 2004, pp. 74–81.
30. Bastings, J.; Titov, I.; Aziz, W.; Marcheggiani, D.; Sima'an, K. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 1957–1967.
31. Ma, S.; Sun, X.; Li, W.; Li, S.; Li, W.; Ren, X. Query and Output: Generating Words by Querying Distributed Word Representations for Paraphrase Generation. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), 2018, pp. 196–206.
32. Vashishth, S.; Bhandari, M.; Yadav, P.; Rai, P.; Bhattacharyya, C.; Talukdar, P. Incorporating Syntactic and Semantic Information in Word Embeddings using Graph Convolutional Networks. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 3308–3318.
33. Mikolov, T.; Corrado, G.; Kai, C.; Dean, J. Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013.