Neural Paraphrase Generation with Stacked Residual LSTM Networks

Aaditya Prakash[1,2]**, Sadid A. Hasan**[2]**, Kathy Lee**[2]**, Vivek Datla**[2]**,**
Ashequl Qadir[2]**, Joey Liu**[2]**, Oladimeji Farri**[2]
[1]Brandeis University, Waltham, MA, USA
[2]Artificial Intelligence Laboratory, Philips Research North America, Cambridge, MA, USA
`{aprakash,aaditya.prakash}@{brandeis.edu,philips.com}`
`{sadid.hasan,kathy.lee_1,vivek.datla}@philips.com`
`{ashequl.qadir,joey.liu,dimeji.farri}@philips.com`

Abstract

In this paper, we propose a novel neural approach for paraphrase generation. Conventional paraphrase generation methods either leverage hand-written rules and thesauri-based alignments, or use statistical machine learning principles. To the best of our knowledge, this work is the first to explore deep learning models for paraphrase generation. Our primary contribution is a stacked residual LSTM network, where we add residual connections between LSTM layers. This allows for efficient training of deep LSTMs. We evaluate our model and other state-of-the-art deep learning models on three different datasets: `PPDB`, `WikiAnswers`, and `MSCOCO`. Evaluation results demonstrate that our model outperforms sequence to sequence, attention-based, and bi-directional LSTM models on `BLEU`, `METEOR`, `TER`, and an *embedding*-based sentence similarity metric.

1 Introduction

Paraphrasing, the act to express the same meaning in different possible ways, is an important subtask in various Natural Language Processing (NLP) applications such as question answering, information extraction, information retrieval, summarization and natural language generation. Research on paraphrasing methods typically aims at solving three related problems: (1) recognition (i.e. to identify if two textual units are paraphrases of each other), (2) extraction (i.e. to extract paraphrase instances from a thesaurus or a corpus), and (3) generation (i.e. to generate a reference paraphrase given a source text) (Madnani and Dorr, 2010). In this paper, we focus on the paraphrase generation problem.

Paraphrase generation has been used to gain performance improvements in several NLP applications, for example, by generating query variants or pattern alternatives for information retrieval, information extraction or question answering systems, by creating reference paraphrases for automatic evaluation of machine translation and document summarization systems, and by generating concise or simplified information for sentence compression or sentence simplification systems (Madnani and Dorr, 2010). Traditional paraphrase generation methods exploit hand-crafted rules (McKeown, 1983) or automatically learned complex paraphrase patterns (Zhao et al., 2009), use thesaurus-based (Hassan et al., 2007) or semantic analysis driven natural language generation approaches (Kozlowski et al., 2003), or leverage statistical machine learning theory (Quirk et al., 2004; Wubben et al., 2010). In this paper, we propose to use deep learning principles to address the paraphrase generation problem.

Recently, techniques like sequence to sequence learning (Sutskever et al., 2014) have been applied to various NLP tasks with promising results, for example, in the areas of machine translation (Cho et al., 2014; Bahdanau et al., 2015), speech recognition (Li and Wu, 2015), language modeling (Vinyals et al., 2015), and dialogue systems (Serban et al., 2016). Although paraphrase generation can be formulated as a sequence to sequence learning task, not much work has been done in this area with regard to applications of state-of-the-art deep neural networks. There are several works on paraphrase recognition (Socher et al., 2011; Yin and Schütze, 2015; Kiros et al., 2015), but those employ classification techniques and do not attempt to generate paraphrases. More recently, attention-based Long Short-Term Memory (LSTM) networks have been used for textual entailment generation (Kolesnyk et al., 2016); however, paraphrase

generation is a type of bi-directional textual entailment generation and no prior work has proposed a deep learning-based formulation of this task.

To address this gap in the literature, we explore various types of sequence to sequence models for paraphrase generation. We test these models on three different datasets and evaluate them using well recognized metrics. Along with the application of various existing sequence to sequence models for the paraphrase generation task, in this paper we also propose a new model that allows for training multiple stacked LSTM networks by introducing a residual connection between the layers. This is inspired by the recent success of such connections in a deep Convolutional Neural Network (CNN) for the image recognition task (He et al., 2015). Our experiments demonstrate that the proposed model can outperform other techniques we have explored.

Most of the deep learning models for NLP use Recurrent Neural Networks (RNNs). RNNs differ from normal perceptrons as they allow gradient propagation in time to model sequential data with variable-length input and output (Sutskever et al., 2011). In practice, RNNs often suffer from the vanishing/exploding gradient problems while learning long-range dependencies (Bengio et al., 1994). LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Cho et al., 2014) are known to be successful remedies to these problems.

It has been observed that increasing the depth of a deep neural network can improve the performance of the model (Simonyan and Zisserman, 2014; He et al., 2015) as deeper networks learn better representations of features (Farabet et al., 2013). In the vision-related tasks where CNNs are more widely used, adding many layers of neurons is a common practice. For tasks like speech recognition (Li and Wu, 2015) and also in machine translation, it is useful to stack layers of LSTM or other variants of RNN. So far this has been limited to only a few layers due to the difficulty in training deep RNN networks. We propose to add residual connections between multiple stacked LSTM networks and show that this allows us to stack more layers of LSTM successfully.

The rest of the paper is organized as follows: Section 2 presents a brief overview of the sequence to sequence models followed by a description of our proposed residual deep LSTM model, Section 3 describes the datasets used in this work, Section 4 explains the experimental setup, Section 5 presents the evaluation results and analyses, Section 6 discusses the related work, and in Section 7 we conclude and discuss future work.

2   Model Description

2.1   Encoder-Decoder Model

A neural approach to sequence to sequence modeling proposed by Sutskever et al. (2014) is a two-component model, where a source sequence is first encoded into some low dimensional representation (Figure 1) that is later used to reproduce the sequence back to a high dimensional target sequence (i.e. decoding). In machine translation, an encoder operates on a sentence written in the source language and encodes its meaning to a vector representation before the decoder can take that vector (which represents the meaning) and generate a sentence in the target language. These encoder-decoder blocks can be either a vanilla RNN or its variants. While producing the target sequence, the generation of each new word depends on the model and the preceding generated word. Generation of the first word in the target sequence depends on the special 'EOS' (end-of-sentence) token appended to the source sequence.

The training objective is to maximize the log probability of the target sequence given the source sequence. Therefore, the best possible decoded target is the one that has the maximum score over the length of the sequence. To find this, a small set of hypotheses (candidate set) called *beam size* is used and the total score for all these hypotheses are computed. In the original work by Sutskever et al. (2014), they observe that although a beam size of 1 achieves good results, a higher beam size is always better. This is because for some of the hypotheses, the first word may not always have the highest score.

2.2   Deep LSTM

LSTM (Figure 2) is a variant of RNN, which computes the hidden state h_t using a different approach by adding an internal memory cell $c_t \in \mathbb{R}^n$ at every time step t. In particular, an LSTM unit considers the input state x_t at time step t, the hidden state h_{t-1}, and the internal memory state c_{t-1} at time step
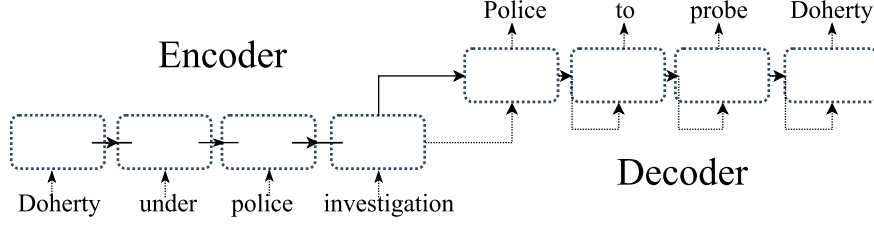
Figure 1: Encoder-Decoder framework for sequence to sequence learning.

1. Gates

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

2. Input transform

$$c\_in_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_{c\_in})$$

3. State Update

$$c_t = f_t \odot c_{t-1} + i_t \odot c\_in_t$$
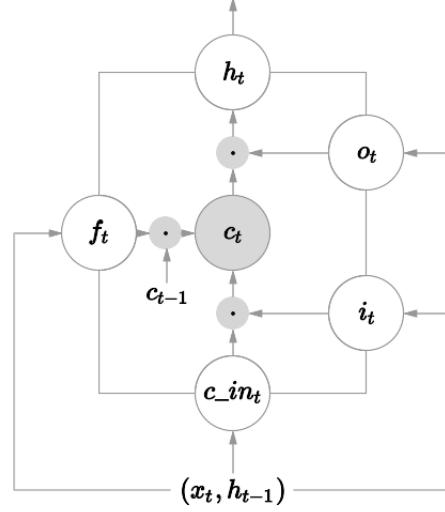$$h_t = o_t \odot \tanh(c_t)$$

Figure 2: LSTM cell (Paszke, 2015).

$t-1$ to produce the hidden state h_t and the internal memory state c_t at time step t. The memory cell is controlled via three learned gates: input i, forget f, and output o. These memory cells use the addition of gradient with respect to time and thus minimize the gradient explosion. In most NLP tasks, LSTM outperforms vanilla RNN (Sundermeyer et al., 2012). Therefore, for our model we only explore LSTM as a basic unit in the encoder and decoder. Here, we describe the basic computations in an LSTM unit, which will provide the grounding to understand the residual connections between stacked LSTM layers later.

In the equations above, $W_{x\_}, W_{h\_}$ are the learned parameters for x and h respectively. $\sigma(.)$ and $\tanh(.)$ denote element-wise sigmoid and hyperbolic tangent functions respectively. $\odot$ is the element-wise multiplication operator and b denotes the added bias.

Graves (2013) explored the advantages of deep LSTMs for handwriting recognition and text generation. There are multiple ways of combining one layer of LSTM with another. For example, Pascanu et al. (2013) explored multiple ways of combining them and discussed various difficulties in training deep LSTMs. In this work, we employ vertical stacking where only the output of the previous layer of LSTM is fed to the input, as compared to the stacking technique used by Sutskever et al. (2014), where hidden states of all LSTM layers are fully connected. In our model, all but the first layer input at time step t is passed from the hidden state of the previous layer h_t^l, where l denotes the layer. This is similar to stacked RNN proposed by Bengio et al. (1994) but with LSTM units. Thus, for a layer l the activation is described by:

$$\boldsymbol{h}_t^{(l)} = f_h^l(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l)})$$

where hidden states $\boldsymbol{h}$ are recursively computed and $\boldsymbol{h}_t^{(l)}$ at $t = 0$ and $l = 0$ is given by the LSTM equation of h_t.
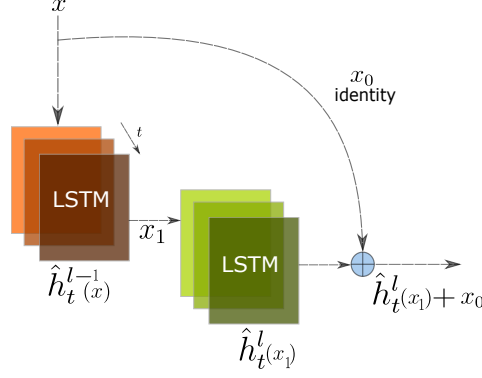
Figure 3: A unit of stacked residual LSTM.

2.3 Stacked Residual LSTM

We take inspiration from a very successful deep learning network *ResNet* (He et al., 2015) with regard to adding residue for the purpose of learning. With theoretical and empirical reasoning, He et al. (2015) have shown that the explicit addition of the residue x to the function being learned allows for deeper network training without overfitting the data.

When stacking multiple layers of neurons, the network often suffers through a *degradation* problem (He et al., 2015). The degradation problem arises due to the low convergence rate of training error and is different from the vanishing gradient problem. Residual connections can help overcome this issue. We experimented with four-layers of stacked LSTM for each of the model. Residual connections are added at layer two as the pointwise addition (see Figure 3), and thus it requires the input to be in the same dimension as the output of h_t. Principally because of this reason, we use a simple last hidden unit stacking of LSTM instead of a more intricate way as shown by Sutskever et al. (2014). This allowed us to clip the h_t to match the dimension of x_{t-2} where they were not the same. Similar results could be achieved by padding x to match the dimension instead. The function $\hat{h}$ that is being learned for the layer with residual connection is therefore:

$$\hat{\boldsymbol{h}}_t^{(l)} = f_h^l(\boldsymbol{h}_t^{(l-1)}, \boldsymbol{h}_{t-1}^{(l)}) + x_{l-n}$$

where $\hat{h}$ for layer l is updated with residual value x_{l-n} and x_i represents the input to layer $i+1$. Residual connection is added after every n layers. However, for stacked LSTM, $n > 3$ is very expensive in terms of computation. In this paper we experimented with $n = 2$. Note that, when $n = 1$, the resulting function learned is a standard LSTM with bias that depends on the input x. That is why, it is not necessary to add the residual connection after every stacked layer of LSTM. The addition of residual connection does not add any learnable parameters. Therefore, this does not increase the complexity of the model unlike bi-directional models which double the number of LSTM units.

3 Datasets

We present the performance of our model on three datasets, which are significantly different in their characteristics. So, evaluating our paraphrase generation approach on these datasets demonstrates the versatility and robustness of our model.

PPDB (Pavlick et al., 2015) is a well known paraphrase dataset used for various NLP tasks. It comes in different sizes and the precision of the paraphrases degrades with the size of the dataset. We use the size L dataset from PPDB 2.0, which comes with over $18M$ paraphrases including lexical, phrasal and syntactic types. We have omitted the syntactic paraphrases and the instances which contain numbers, as they increase the vocabulary size significantly without giving any advantage of a larger dataset. This dataset contains relatively short paraphrases (86% of the data is less than four words), which makes it suitable for synonym generation and phrase substitution to address lexical and phrasal paraphrasing (Madnani and Dorr, 2010). For some phrases, PPDB has one-to-many paraphrases. We collect all such phrases to make a set of paraphrases and sampling without replacement was used to obtain the source

and reference phrases.

WikiAnswers (Fader et al., 2013) is a large question paraphrase corpus created by crawling the WikiAnswers website[1], where users can post questions and answers about any topic. The paraphrases are different questions, which were tagged by the users as similar questions. The dataset contains approximately 18M word-aligned question pairs. Sometimes, there occurs a loss of specialization between a given source question and its corresponding reference question when a paraphrase is tagged as similar to a reference question. For example, "prepare a *three month* cash budget" is tagged to "how to prepare a cash budget". This happens because general questions are typically more popular and get answered. So, specific questions are redirected to the general ones due to a comparative lack of interest in the very specific questions. It should be noted that this dataset comes preprocessed and lemmatized. We refer the reader to the original paper for more details.

MSCOCO (Lin et al., 2014) dataset contains human annotated captions of over $120K$ images. Each image contains five captions from five different annotators. While there is no guarantee that the human annotations are paraphrases, the nature of the images (which tends to focus on only a few objects and in most cases one prominent object or action) allows most annotators describe the most obvious things in an image. In fact, this is the main reason why neural networks for generating captions obtain better BLEU scores (Vinyals et al., 2014), which confirms the suitability of using this dataset for the paraphrase generation task.

4 Experimental Settings

4.1 Data Selection

For PPDB we remove the phrases that contain numbers including all syntactic phrases. This gives us a total of $5.3M$ paraphrases from which we randomly select 90% instances for training. For testing, we randomly select $20K$ pairs of paraphrases from the remaining 10% data. Although WikiAnswers comes with over $29M$ instances, we randomly select $4.8M$ for training to keep the training size similar to PPDB (see Table 1). $20K$ instances were randomly selected from the remaining data for testing. Note that, for the WikiAnswers dataset, we clip the vocabulary size[2] to $50K$ and use the special *UNK* symbol for the words outside the vocabulary. MSCOCO dataset has five captions for every image. This dataset comes with separate subsets for training and validation: *Train 2014* contains over $82K$ images and *Val 2014* contains over $40K$ images. From the five captions accompanying each image, we randomly omit one caption and use the other four as training instances (by creating two source-reference pairs). Thus, we obtain a collection of over $330K$ instances for training and $20K$ instances for testing. Because of the free form nature of the caption generation task (Vinyals et al., 2014), some captions were very long. We reduced those captions to the size of 15 words (by removing the words beyond the first 15) in order to reduce the training complexity of the models.

Dataset	Training	Test	Vocabulary Size
PPDB	4,826,492	20,000	38,279
WikiAnswers	4,826,492	20,000	50,000
MSCOCO	331,163	20,000	30,332

Table 1: Dataset details.

Models	Reference
Sequence to Sequence	(Sutskever et al., 2014)
With Attention	(Bahdanau et al., 2015)
Bi-directional LSTM	(Graves et al., 2013)
Residual LSTM	Our proposed model

Table 2: Models.

4.2 Models

We experimented with four different models (see Table 2). For each model, we experimented with two- and four-layers of stacked LSTMs. This was motivated by the state-of-the-art speech recognition systems that also use three to four layers of stacked LSTMs (Li and Wu, 2015). In encoder-decoder models, the size of the beam search used during inference is very important. Larger beam size always gives higher

---

[1]http://wiki.answers.com

[2]WikiAnswers dataset had many spelling errors yielding a very large vocabulary size (approximately $250K$). Hence, we selected the most frequent $50K$ words in the vocabulary to reduce the computational complexity.
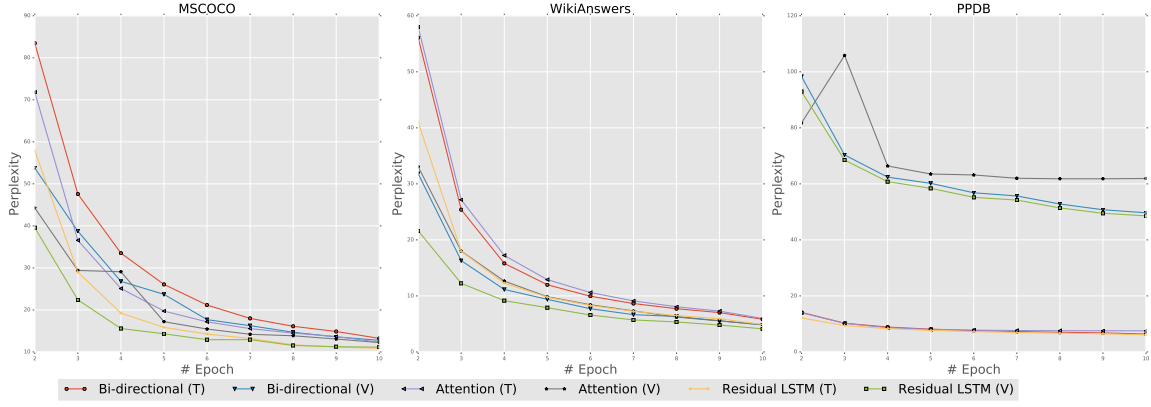
Figure 4: Perplexity during training (T) and validation (V) for various models [shared legend]. A lower perplexity represents a better model.

accuracy but is associated with a computational cost. We experimented with beam sizes of 5 and 10 to compare the models, as these are the most common beam sizes used in the literature (Sutskever et al., 2014). The bi-directional model used half of the number of layers shown for other models. This was done to ensure similar parameter sizes across the models.

4.3 Training

We used a one-hot vector approach to represent the words in all models. Models were trained with a stochastic gradient descent (SGD) algorithm. The learning rate began at 1.0, and was halved after every third training epoch. Each network was trained for ten epochs. In order to allow exploration of a wide variety of models, training was restricted to a limited number of epochs, and no hyper-parameter search was performed. A standard dropout (Srivastava et al., 2014) of 50% was applied after every LSTM layer. The number of LSTM units in each layer was fixed to 512 across all models. Training time ranged from 36 hours for `WikiAnswers` and `PPDB` to 14 hours for `MSCOCO` on a *Titan X* with *CuDNN 5* using Theano version $0.9.0dev1$ (Theano Development Team, 2016).

A beam search algorithm was used to generate optimal paraphrases by exploiting the trained models in the testing phase (Sutskever et al., 2014). We used perplexity as the loss function during training. Perplexity measures the uncertainty of the language model, corresponding to how many bits on average would be needed to encode each word given the language model. A lower perplexity indicates a better score. While `WikiAnswers` and `MSCOCO` had a very good correlation between training and validation perplexity, overfitting was observed with `PPDB` that yielded a worse validation perplexity (see Figure 4).

5 Evaluation

5.1 Metrics

To quantitatively evaluate the performance of our paraphrase generation models, we use the well-known automatic evaluation metrics[3] for comparing parallel corpora: `BLEU` (Papineni et al., 2002), `METEOR` (Lavie and Agarwal, 2007), and Translation Error Rate (`TER`) (Snover et al., 2006). Even though these metrics were designed for machine translation, previous works have shown that they can perform well for the paraphrase recognition task (Madnani et al., 2012) and correlate well with human judgments in evaluating generated paraphrases (Wubben et al., 2010).

Although there exists a few automatic evaluation metrics that are specifically designed for paraphrase generation, such as `PEM` (Paraphrase Evaluation Metric) (Liu et al., 2010) and `PINC` (Paraphrase In N-gram Changes) (Chen and Dolan, 2011), they have certain limitations. `PEM` relies on large in-domain bilingual parallel corpora along with sample human ratings for training while it can only model paraphrasing up to the phrase-level granularity. `PINC` attempts to solve these limitations by proposing a method that is essentially the inverse of `BLEU`, as it calculates the n-gram difference between the source

---

[3]We used the software available at `https://github.com/jhclark/multeval`

and the reference sentences. Although `PINC` correlates well with human judgments in lexical dissimilarity assessment, `BLEU` has been shown to correlate better for semantic equivalence agreements at the sentence-level when a sufficiently large number of reference sentences are available for each source sentence (Chen and Dolan, 2011).

`BLEU` considers exact matching between reference paraphrases and system generated paraphrases by considering n-gram overlaps while `METEOR` improves upon this measure via stemming and synonymy using WordNet. `TER` measures the number of edits required to change a system generated paraphrase into one of the reference paraphrases. As suggested in Clark et al. (2011), we used a stratified approximate randomization (AR) test. AR calculates the probability of a metric score providing the same reference sentence by chance. We report our p-values at `95%` Confidence Intervals (CI).

The major limitation of these evaluation metrics is that they do not consider the meaning of the paraphrases, and hence, are not able to capture paraphrases of entities. For example, these metrics do not reward the paraphrasing of *"London"* to *"Capital of UK"*. Therefore, we also evaluate our models on a sentence similarity metric[4] proposed by Rus et al. (2012). This metric uses word embeddings to compare the phrases. In our experiments, we used Word2Vec embeddings pre-trained on the Google News Corpus (Mikolov et al., 2014). This is referred to as *'Emb Greedy'* in our results table.

5.2 Results

Table 3 presents the results from various models across different datasets. ↑ denotes that higher scores represent better models while ↓ means that a lower score yields a better model. Although our focus is on stacked residual LSTM, which is applicable only when there are more than two layers, we still present the scores from two-layer LSTM as a baseline. This provides a good comparison against deeper models. The results demonstrate that our proposed model outperforms other models on `BLEU` and `TER` for all datasets. On *Emb Greedy*, our model outperforms other models in all datasets except the Attention model when beam size is 10. On `METEOR`, our model outperforms other models on `MSCOCO` and `WikiAnswers`; however, for `PPDB`, the simple sequence to sequence model performs better. Note that these results were obtained by using single models and no ensemble of the models was used.

To calculate `BLEU` and `METEOR`, four references were used for `MSCOCO`, and five for `PPDB` and `WikiAnswers`. In some instances, `WikiAnswers` did not have up to five reference paraphrases for every source, hence, those were calculated on reduced references. In Table 4, we present the variance due to the test set selection. This is calculated using bootstrap re-sampling for each optimizer run (Clark et al., 2011). Variance due to optimizer instability was less than 0.1 in all cases. p-value of these tests are less than 0.05 in all cases. Thus, comparison between two models is significant at 95% CI if the difference in their score is more than the variance due to test set selection (Table 4).

5.3 Analysis

Scores on various metrics vary a lot across the datasets, which is understandable due to their inherent differences. `PPDB` contains very small phrases and thus does not score well with metrics like `BLEU` and `METEOR` which penalize shorter phrases. As shown in Figure 5, more than 50% of `PPDB` contains one or two words. This leads to a substantial difference between training and validation errors, as shown in Figure 4. The results demonstrate that deeper LSTMs consistently improve performance over shallow models. For beam size of 5 our model outperforms other models in all datasets. For beam size of 10, the attention-based model has a marginally better *Emb Greedy* score than our model. When we look at the qualitative results, we notice that the bias in the dataset is exploited by the system which is a side effect of any form of learning on a limited dataset. We can see this effect in Table 5. For example, *an OBJECT* is mostly paraphrased with *an OBJECT* (e.g. *bowl*, *motorcycle*). Shorter sentences mostly generate shorter paraphrases and the same is true for longer sequences. Based on our results, the embedding-based metric correlates well with statistical metrics. Figure 4 and the results from Table 5 suggest that perplexity is a good loss function for training paraphrase generation models. However, a more ideal metric to fully encode the fundamental objective of paraphrasing should also reward novelty and penalize redundancy during paraphrase generation, which is a notable limitation of the existing paraphrase evaluation metrics.

---

[4]We used the software available at `https://github.com/julianser/hed-dlg-truncated/`

		Beam size = 5				Beam size = 10			
#Layers	Model	BLEU↑	METEOR↑	Emb Greedy↑	TER↓	BLEU↑	METEOR↑	Emb Greedy↑	TER↓
		PPDB							
2	Sequence to Sequence	12.5	21.3	32.55	82.9	12.9	20.5	32.65	83.0
	With Attention	13.0	21.2	32.95	82.2	13.8	20.6	32.29	81.9
4	Sequence to Sequence	18.3	**23.5**	33.18	82.7	18.8	**23.5**	33.78	82.1
	Bi-directional	19.2	23.1	34.39	77.5	19.7	23.2	34.56	84.4
	With Attention	19.9	23.2	34.71	83.8	20.2	22.9	**34.90**	77.1
	Residual LSTM	**20.3**	23.1	**34.77**	**77.1**	**21.2**	23.0	34.78	**77.0**
		WikiAnswers							
2	Sequence to Sequence	19.2	26.1	62.65	35.1	19.5	26.2	62.95	34.8
	With Attention	21.2	22.9	63.22	37.1	21.2	23.0	63.50	37.0
4	Sequence to Sequence	33.2	29.6	73.17	28.3	33.5	29.6	73.19	28.3
	Bi-directional	34.0	30.8	73.80	27.3	34.3	30.7	73.95	27.0
	With Attention	34.7	31.2	73.45	27.1	34.9	31.2	73.50	27.1
	Residual LSTM	**37.0**	**32.2**	**75.13**	**27.0**	**37.2**	**32.2**	**75.19**	**26.8**
		MSCOCO							
2	Sequence to Sequence	15.9	14.8	54.11	66.9	16.5	15.4	55.81	67.1
	With Attention	17.5	16.6	58.92	63.9	18.6	16.8	59.26	63.0
4	Sequence to Sequence	28.2	23.0	67.22	56.7	28.9	23.2	67.10	56.3
	Bi-directional	32.6	24.5	68.62	53.8	32.8	24.9	68.91	53.7
	With Attention	33.1	25.4	69.10	54.3	33.4	25.2	**69.34**	53.8
	Residual LSTM	**36.7**	**27.3**	**69.69**	**52.3**	**37.0**	**27.0**	69.21	**51.6**

Table 3: Evaluation results on PPDB, WikiAnswers, and MSCOCO (Best results are in **bold**).

Dataset	σ^2[BLEU]	σ^2[METEOR]	σ^2[TER]	σ^2[Emb Greedy]
PPDB	2.8	0.2	0.4	0.000100
WikiAnswers	0.3	0.1	0.1	0.000017
MSCOCO	0.2	0.1	0.1	0.000013

Table 4: Variance due to test set selection.

	PPDB	WikiAnswers	MSCOCO
Source	south eastern	what be the symbol of magnesium sulphate	a small kitten is sitting in a bowl
Reference	the eastern part	chemical formulum for magnesium sulphate	a cat is curled up in a bowl
Generated	south east	do magnesium sulphate have a formulum	a cat that is sitting on a bowl
Source	organized	what be the bigggest galaxy know to man	an old couple at the beach during the day
Reference	managed	how many galaxy be there in you known universe	two people sitting on dock looking at the ocean
Generated	arranged	about how many galaxy do the universe contain	a couple standing on top of a sandy beach
Source	counselling	what do the ph of acid range to	a little baby is sitting on a huge motorcycle
Reference	be kept informed	a acid have ph range of what	a little boy sitting alone on a motorcycle
Generated	consultations	how do acid affect ph	a baby sitting on top of a motorcycle

Table 5: Example paraphrases generated using the 4-layer Residual LSTM with beam size 5.

6 Related Work

Prior approaches to paraphrase generation have applied relatively different methodologies, typically using knowledge-driven approaches or statistical machine translation (SMT) principles. Knowledge-driven methods for paraphrase generation (Madnani and Dorr, 2010) utilize hand-crafted rules (McKeown, 1983) or automatically learned complex paraphrase patterns (Zhao et al., 2009). Other paraphrase generation methods use thesaurus-based (Hassan et al., 2007) or semantic analysis-driven natural language generation approaches (Kozlowski et al., 2003) to generate paraphrases. In contrast, Quirk et al., (2004)
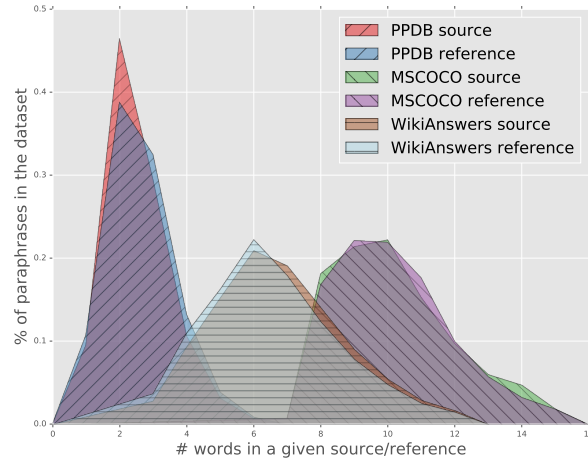
Figure 5: Distribution of sequence length (in number of words) across datasets.

show the effectiveness of SMT techniques for paraphrase generation given adequate monolingual parallel corpus extracted from comparable news articles. Wubben et al., (2010) propose a phrase-based SMT framework for sentential paraphrase generation by using a large aligned monolingual corpus of news headlines. Zhao et al., (2008) propose a combination of multiple resources to learn phrase-based paraphrase tables and corresponding feature functions to devise a log-linear SMT model. Other models generate application-specific paraphrases (Zhao et al., 2009), leverage bilingual parallel corpora (Bannard and Callison-Burch, 2005) or apply a multi-pivot approach to output candidate paraphrases (Zhao et al., 2010).

Applications of deep learning for paraphrase generation tasks have not been rigorously explored. We utilized several sources as potential large datasets. Recently, Weiting et al. (2015) took the PPDB dataset (size XL) and annotated phrases based on their paraphrasability. This dataset is called *Annotated*-PPDB and contains 3000 pairs in total. They also introduced another dataset called *ML-Paraphrase* for the purpose of evaluating bigram paraphrases. This dataset contains 327 instances. Microsoft Research Paraphrase Corpus (MSRP) (Dolan et al., 2005) is another widely used dataset for paraphrase detection. MSRP contains 5800 pairs of sentences (obtained from various news sources) accompanied with human annotations. These datasets are too small and therefore, we did not use them for training our deep learning models.

To the best of our knowledge, this is the first work on using residual connections with recurrent neural networks. Very recently, we found that Toderici et al. (2016) used residual GRU to show an improvement in image compression rates for a given quality over JPEG. Another variant of residual network called *DenseNet* (Huang et al., 2016), which uses dense connections over every layer, has been shown to be effective for image recognition tasks achieving state-of-the-art results in CIFAR and SVHN datasets. Such works further validate the efficacy of adding residual connections for training deep networks.

7   Conclusion and Future Work

In this paper, we described a novel technique to train stacked LSTM networks for paraphrase generation. This is an extension to sequence to sequence learning, which has been shown to be effective for various NLP tasks. Our model outperforms state-of-the-art models for sequence to sequence learning. We have shown that stacking of residual LSTM layers is useful for paraphrase generation, but it may not perform equally well for machine translation because not every word in a source sequence needs to be substituted for paraphrasing. Residual connections help retain important words in the generated paraphrases.

We experimented on three different large scale datasets and reported results using various automatic evaluation metrics. We showed the use of the well-known MSCOCO dataset for paraphrase generation and demonstrated that the models can be trained effectively without leveraging the images. The presented experiments should set strong baselines for neural paraphrase generation on these datasets, enabling future researchers to easily compare and evaluate subsequent works in paraphrase generation.

Recent advances in neural networks with regard to learnable memory (Sukhbaatar et al., 2015; Graves et al., 2014) have enabled models to get one step closer to learning comprehension. It may be helpful to explore such networks for the paraphrase generation task. Also, it remains to be explored how unsupervised deep learning could be harnessed for paraphrase generation. It would be interesting to see if researchers working on image-captioning can employ neural paraphrase generation to augment their dataset.

Acknowledgment

References

D. Bahdanau, K. Cho, and Y. Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of ICLR*, pages 1–15.

C. Bannard and C. Callison-Burch. 2005. Paraphrasing with Bilingual Parallel Corpora. In *Proceedings of ACL*, pages 597–604.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

D. Chen and W. B. Dolan. 2011. Collecting Highly Parallel Data for Paraphrase Evaluation. In *Proceedings of ACL-HLT*, pages 190–200.

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of EMNLP*, pages 1724–1734.

J. H. Clark, C. Dyer, A. Lavie, and N. A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of ACL-HLT*, pages 176–181.

B. Dolan, C. Brockett, and C. Quirk. 2005. Microsoft Research Paraphrase Corpus. *Retrieved March*, 29:2008.

A. Fader, L. S Zettlemoyer, and O. Etzioni. 2013. Paraphrase-Driven Learning for Open Question Answering. In *ACL*, pages 1608–1618. ACL.

C. Farabet, C. Couprie, L. Najman, and Y. LeCun. 2013. Learning Hierarchical Features for Scene Labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.

A. Graves, N. Jaitly, and A. Mohamed. 2013. Hybrid Speech Recognition with Deep Bidirectional LSTM. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE.

A. Graves, G. Wayne, and I. Danihelka. 2014. Neural Turing Machines. In *arXiv:1410.5401*.

A. Graves. 2013. Generating Sequences with Recurrent Neural Networks. *arXiv preprint arXiv:1308.0850*.

S. Hassan, A. Csomai, C. Banea, R. Sinha, and R. Mihalcea. 2007. UNT: SubFinder: Combining Knowledge Sources for Automatic Lexical Substitution. In *Proceedings of SemEval*, pages 410–413.

K. He, X. Zhang, S. Ren, and J. Sun. 2015. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*.

S. Hochreiter and J. Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

G. Huang, Z. Liu, and K. Q. Weinberger. 2016. Densely Connected Convolutional Networks. *arXiv preprint arXiv:1608.06993*.

R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.

V. Kolesnyk, T. Rocktäschel, and S. Riedel. 2016. Generating Natural Language Inference Chains. *CoRR*, abs/1606.01404.

R. Kozlowski, K. F. McCoy, and K. Vijay-Shanker. 2003. Generation of Single-sentence Paraphrases from Predicate/Argument Structure Using Lexico-grammatical Resources. In *Proceedings of the 2nd International Workshop on Paraphrasing*, pages 1–8.

A. Lavie and A. Agarwal. 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*, pages 228–231.

X. Li and X. Wu. 2015. Constructing Long Short-term Memory based Deep Recurrent Neural Networks for Large Vocabulary Speech Recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4520–4524. IEEE.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755. Springer.

C. Liu, D. Dahlmeier, and H. T. Ng. 2010. PEM: A Paraphrase Evaluation Metric Exploiting Parallel Texts. In *Proceedings of EMNLP*, pages 923–932.

N. Madnani and B. J. Dorr. 2010. Generating Phrasal and Sentential Paraphrases: A Survey of Data-driven Methods. *Computational Linguistics*, 36(3):341–387.

N. Madnani, J. Tetreault, and M. Chodorow. 2012. Re-examining Machine Translation Metrics for Paraphrase Identification. In *Proceedings of NAACL-HLT*, pages 182–190.

K. R. McKeown. 1983. Paraphrasing Questions Using Given and New Information. *Computational Linguistics*, 9(1):1–10.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2014. Word2Vec. https://code.google.com/p/word2vec. Online; accessed 2014-04–15.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL*, pages 311–318.

R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. 2013. How to Construct Deep Recurrent Neural Networks. *arXiv preprint arXiv:1312.6026*.

A. Paszke. 2015. LSTM Implementation Explained: apaszke.github.io/lstm-explained.html , 2016-07-15.

E. Pavlick, P. Rastogi, J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL-IJCNLP)*, pages 425–430.

C. Quirk, C. Brockett, and W. Dolan. 2004. Monolingual Machine Translation for Paraphrase Generation. In *Proceedings of EMNLP*, pages 142–149.

V. Rus and M. Lintean. 2012. A Comparison of Greedy and Optimal Assessment of Natural Language Student Input using Word-to-Word Similarity Metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 157–162. ACL.

I. V. Serban, T. Klinger, G. Tesauro, K. Talamadupula, B. Zhou, Y. Bengio, and A. Courville. 2016. Multiresolution Recurrent Neural Networks: An Application to Dialogue Response Generation. *arXiv preprint arXiv:1606.00776*.

K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

R. Socher, E. H. Huang, J. Pennington, A. Y. Ng, and C. D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems*, pages 1–9.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

S. Sukhbaatar, J. Weston, R. Fergus, et al. 2015. End-to-End Memory Networks. In *Advances in neural information processing systems*, pages 2440–2448.

M. Sundermeyer, R. Schlüter, and H. Ney. 2012. LSTM Neural Networks for Language Modeling. In *Interspeech*, pages 194–197.

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating Text with Recurrent Neural Networks. In *Proceedings of ICML*, pages 1017–1024.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Theano Development Team. 2016. Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints*, abs/1605.02688, May.

G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell. 2016. Full Resolution Image Compression with Recurrent Neural Networks. *arXiv preprint arXiv:1608.05148*.

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. 2014. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015. Grammar as a Foreign Language. In *Advances in Neural Information Processing Systems*, pages 2773–2781.

J. Wieting, M. Bansal, K. Gimpel, and K. Livescu. 2015. From Paraphrase Database to Compositional Paraphrase Model and Back. *Transactions of the ACL (TACL)*.

S. Wubben, A. van den Bosch, and E. Krahmer. 2010. Paraphrase Generation As Monolingual Translation: Data and Evaluation. In *Proceedings of INLG*, pages 203–207.

W. Yin and H. Schütze. 2015. Convolutional Neural Network for Paraphrase Identification. In *Proceedings of NAACL-HLT*, pages 901–911.

S. Zhao, C. Niu, M. Zhou, T. Liu, and S. Li. 2008. Combining Multiple Resources to Improve SMT-based Paraphrasing Model. In *Proceedings of ACL-HLT*, pages 1021–1029.

S. Zhao, X. Lan, T. Liu, and S. Li. 2009. Application-driven Statistical Paraphrase Generation. In *Proceedings of ACL-IJCNLP*, pages 834–842.

S. Zhao, H. Wang, X. Lan, and T. Liu. 2010. Leveraging Multiple MT Engines for Paraphrase Generation. In *Proceedings of COLING*, pages 1326–1334.