An Empirical Evaluation Of Attention And Pointer Networks For Paraphrase Generation[⋆]

Varun Gupta[1] and Adam Krzyżak[1,2]

[1] Department of Computer Science and Software Engineering
Concordia University
Montreal, Canada H3G 1M8
[2] Department of Electrical Engineering
Westpomeranian University of Technology
70-313 Szczecin, Poland
varun9208@yahoo.com, krzyzak@cs.concordia.ca

Abstract. In computer vision, one of the common practices to augment the image dataset is by creating new images using geometric transformation preserving similarity. This data augmentation was one of the most significant factors for winning the Image Net competition in 2012 with vast neural networks. Unlike in computer vision and speech data, there have not been many techniques explored to augment data in natural language processing (NLP). The only technique explored in the text data is by lexical substitution, which only focuses on replacing words by synonyms.

In this paper, we investigate the use of different pointer networks with the sequence-to-sequence models, which have shown excellent results in neural machine translation (NMT) and text simplification tasks, in generating similar sentences using a sequence-to-sequence model and of the paraphrase dataset (PPDB). The evaluation of these paraphrases is carried out by augmenting the training dataset of IMDb movie review dataset and comparing its performance with the baseline model. To our best knowledge, this is the first study on generating paraphrases using these models with the help of PPDB dataset.

Keywords: Paraphrase generation · Data augmentation · Attention networks · Pointer networks

1 Paraphrase Theory

A standard ability of human language communication is the ability of humans to communicate the equivalent information in multiple ways. These dialects are known paraphrases, which in the literature have been also referred to as reformulations, restating and other diversity of phenomena.

⋆ Supported by the Natural Sciences and Engineering Research Council of Canada. Part of this research was carried out by the second author during his visit of the Westpomeranian University of Technology while on sabbatical leave from Concordia University

There can be many variations of paraphrases, but in this work, we try to limit generation of paraphrases to such, which can be carried out by linguistic and semantic knowledge to produce similar sentences. Here are some examples [3]

1. **S:** The couple wants to **purchase** a home.
   P: The couple wants to **buy** a home.
2. **S: It was** a Honda **that** John **sold to** Aman.
   P: John **sold** a Honda **to** Aman.
3. **S:** Aman **bought** a house **from** John.
   P: John **sold** a house **to** Aman.
4. **S:** The flood **last year** was a terrible catastrophe in which many people died.
   P: The flood **in 2006** was a terrible catastrophe in which many people died.

In all the examples mentioned above, we only require linguistic, lexical, referential and structural knowledge to generate paraphrases. Example (1), is generated using knowledge of synonym words which comes under the lexical category. Example (2) is generated using structural information, which comes under syntactic knowledge. This type of transformation is described in the theory of transformational grammar [6]. Example (3) is an illustration of alternation which can be carried out by syntactic transformation. Example (4) is an instance of referential paraphrase.

One common thing about all the above-generated paraphrases is that we do not need any domain knowledge or domain is common in both the original sentence and in its paraphrase sentence, i.e., 'English literature.' These things become more tricky when we try to generate paraphrases where the original sentence is in one domain, but we want to generate paraphrase in a domain other than the original domain. Here is an example of these kinds of paraphrases.

5. **S :** Nearest neighbor is good.
   P (Literature Domain) : The closest neighbor is good.
   P (Machine learning Domain) : The closest neighbor is good.

As we can see in the above example when generating paraphrase in one domain for example in 'English literature' (as described in sample 5) 'nearest neighbour' is a synonym of the 'closest neighbour.' However, when generating paraphrase in machine learning domain, it might not be a good idea to convert 'nearest neighbour' to 'closest neighbour' as 'nearest neighbour' has a technical or reserved meaning in machine learning context. This means context or domain knowledge is also required in generating paraphrases.

In this work, we focus on evaluating past methods on generating similar sentences using linguistic, lexical, referential and structural knowledge.

[3] Here 'S' represents the original sentence, and 'P' represents paraphrase of it. Furthermore, bold words are the primary information used in generating paraphrase.

2 Similarity Measures

Before evaluating the existing methods first, we should clarify the conditions which should be fulfilled by the constructed sentence to be considered as a paraphrase. Here are a few criteria for assessing paraphrases.

1. **Syntax Level:** The first minimal requirement for generating paraphrase from the source sentence is to have a valid syntax in the given language. In other words, it should follow all the syntax rules defined by the given language while generating a natural language paraphrase.
2. **Semantics Level:** The second minimal requirement which must be followed by the generated paraphrase in natural language generation is its meaning and interpretation of output or target sentence. The output sentence must be meaningful.
3. **Lexical Level:** Lexical level is a way to convert characters or words into tokens, which can be identified by machines (numbers). These tokens dimensions capture several likeness measures of a word, a context of a word and other things. There can be many ways to convert these characters or words into these tokens, for example, n-grams, similarity coefficients, alter remove, etc. This type of measure is useful to find the similarity or to generate more interpretations from a given source sentence. However, in our case, while generating similar sentences, they should also be checked for contextual meaning of the source sentence.
4. **Same Meaning:** The main property of paraphrase sentences is to have the same meaning in a given context. To better understand how two sentences can have the same meaning, let us describe two key terms: **Connotation** and **Denotation**. Connotation is the emotional and imaginative association surrounding a word. For example, connotations for the word snake could include evil or danger. Denotation is the strict dictionary meaning of word. For example, if we look up the word "snake“ in a dictionary, we discover that one of its denotative meanings is "any of numerous scaly, legless, sometimes venomous reptiles having a long, tapering, cylindrical body and found in most tropical and temperate regions" [7].
   In this work, we assess some of the existing methods which performed well in NMT and text simplification by generating paraphrases in the given context.

3 Paraphrase Generation

Paraphrase generation task is is a particular case of neural machine translation (NMT) task in which, given the source sentence we need to generate an output sentence which has the same meaning. The only anomaly in a paraphrase generation and NMT is that in the former case, output sentence is also in the same language as the source sentence.

Paraphrase generator models are given an input interpretation as a source sentence, and they produce more than one (depending on the beam size) similar interpretations which are then given a score based on some criteria. In general, there are two techniques to generate paraphrases.

3.1 Various Approaches to Paraphrase Generation

In this subsection, we survey various strategies for paraphrase generation.

Bootstrapping This method does not need any machine translation. We generate paraphrases using templates. This technique can only be used when the input and output sentences are templates and it is applied on a large monolingual corpus. We start with retrieving the sentences in the corpus that contain seed pairs which match to the template we wish to generate. Filtering techniques are used to filter out candidate paraphrases, which are not useful enough. Next, after obtaining these templates, we look into the corpus again for the sentences which match these new templates. More seed values are extracted from new sentences, and more iterations are used to generate more templates and more seeds. This process is repeated until no new seed can be obtained or limitation on number of iterations is reached.

In this method, if the slot values can be identified reliably, then one can obtain initial seed slot values automatically by retrieving direct sentences that match the original templates. There are many well-known methods for bootstrapping; one of them is template extraction anchor set extraction (TEAS). It has been used in many information extraction patterns [1]. There are other methods which require corpora annotated with instances of particular types of events to be extracted [3].

Statistical Machine Translation (SMT) As mentioned earlier, the paraphrase generation can be seen as a particular case of the machine translation problem. In general, most of the generation tasks rely on statistical machine translation (SMT), which is based on a large corpus. Next we define SMT.

Let S be an input sentence, whose words are w_1, w_2, w_3 .... $\mathrm{w}_{|\mathrm{S}|}$ and let N be an instance of one candidate translation or in our case it is a candidate for good paraphrase which has words a_1, a_2, a_3 .... a_i. If we have more than one instance of such N, our aim is to find the best $N*$ from the list of N, which has maximum probability of being a translation or paraphrase of S (Source) sentence. This can be represented as follows:

$$N^* = \text{argmax}_N P(N|S) = \text{argmax}\frac{P(N)P(S|N)}{P(S)} = \text{argmax}_N P(N)P(S|N) \quad (1)$$

In Equation (1), using the conditional probability formula argmax$P(N|S)$ can be further expanded as shown below. The source sentence, i.e., S is fixed, so, $P(S)$ is fixed across all translations N, hence can be removed from the denominator. $P(N|S)$ is the probability of translation given source sentence. $P(N)$ is the language model which is used to find out the probability of being a correct sentence of output sentences. Also, $P(S|N)$ is probability of translation or paraphrase model.

In the candidate sentence, each word probability is dependent on its precedence word. So, the total probability of P(N) becomes:

$$P(N) = P(a_1) * P(a_2|a_3) * P(a_3|a_1, a_2).....P(a_N|a_{N-2}, a_{N-1}) \tag{2}$$

This language model has a smoothing mechanism. Smoothing mechanism is needed to handle cases, where n-grams that are unique or do not exist in the corpus, which can lead to the language model where probability of the whole sentence is 0, i.e., $P(N) = 0$. There is some progress seen in utilizing long short-term memory (LSTM) models to produce paraphrases in this case [11]. The model consists of encoder and decoder, both utilizing varieties of the stacked remaining LSTM.

These models are ubiquitous and are very generic for a wide variety of different generation tasks in natural language processing, for example, in question answering, paraphrase generation and text summarizing. Also, this is the state-of-the-art in most of the generation task. In this work, we have used these models for generating paraphrases of the input sentence.

Parsing Syntactic transfer in machine translation may also be used [9] to generate paraphrases. In this approach, we first need to parse the input expression. Then to generate output paraphrase sentence, these parse tree or expression are modified in a way which preserves the meaning of the syntax. There may be errors induced while parsing the input sentence.

4 Encoder-Decoder Networks for NLP

Encoder-decoders are the neural network approaches, which are genuinely ongoing models for deep learning in NLP. These models in some cases outperform classical statistical machine translation methods. The Encoder-Decoder architecture has become an effective and standard approach for both neural machine translation (NMT) and sequence-to-sequence (seq2seq) prediction tasks which involve task like paraphrase generation, question answering and language modeling. Encoder-decoder normally consists of two components: (1) an encoder to encode input sentence into a context vector, (2) the decoder which decodes the context vector to output sequence. The key advantage of seq2seq model is the capacity to train a solitary end-to-end model right on the source and target sentences, and the capacity to deal with sentences of variable length to yield sequence of content. They were first presented autonomously as Recurrent Neural Network (RNN) encoder-decoder [4] and sequence-to-sequence [12] nets for machine interpretation. This group of encoder-decoder models are regularly referred to as seq2seq, regardless of their particular execution. The seq2seq model tries to learn the conditional distribution

$$p(y_1, y_2, ..., y_{T'}|x_1, x_2, ..., x_T) \tag{3}$$

where, y is the output sequence conditioned on the input sequence x or source sequence, $y_{T'}$ denotes the word generated by the model at time step T', T' is the length of the output sentence and T is the length of the input sentence. T' and sequence length T are not necessarily same. A seq2seq model first encodes the entire variable x input with its encoder RNN into a fixed size vector c known as context vector. Then, a decoder RNN generates output $y_1, ...; y'_T$ conditioned on previous predictions and context vector c:

$$p(y_1, y_2, ..., y_{T'}|x_1, x_2, ..., x_T) = \prod_{t=1}^{T'} p(y_t|y_1, y_2, ..., y_{t-1}, c) \tag{4}$$

There are two different ways to define dependency of output sequence y on context vector c. One way is to condition y on c at the first output from the decoder [12]. Another way is to condition every generation of $y_{T'}$ on the same context vector c, thus forming the basis to our model. For the simplicity, we modify equation for vanilla RNN version to get the hidden state s at time step t, denoted by s_t. Modifying hidden state equation leads to

$$s_t = f_h(W_s x_t^d + U_s s_{t-1} + Cc + b_s). \tag{5}$$

Here and elsewhere C is a parameter matrix.

As demonstrated in [12] performance of seq2seq model while generating text can be improved by giving the input sentence in reverse order. The framework accomplished a bilingual evaluation understudy (BLEU) score of 34.81, which is a decent score contrasted with the standard score of 33.30 achieved by STM. This is the first case of a neural machine interpretation framework that defeated a phrase-based statistical machine translation baseline on a large scale problem. However, this work has not accounted for reverse order of sentences.

The encoder-decoder framework has one disadvantage, which is as the length of a sentence increases the performance of seq2seq model decreases. We also use other variations of RNN like long short term memory (LSTM) or Gated Recurrent unit (GRU) for better performance on long sentences.

5 Encoder-Decoder with Attention

In this section, we present the attention mechanism to improve the poor performance of seq2seq model on longer sentences [2], [8].

The problem occurs while generating the word in decoder network. It looks at the entire input sentence every time while generating a new word. The basic concept of attention is to only focus on a particular part of the sentence. Each time the model predicts an output word, it only uses parts of an input where the most relevant information is concentrated instead of an entire sentence. In other words, it only pays attention to some input words.

The basic structure of the model is the same as the encoder-decoder discussed in the previous section. However, the main difference after adding attention mechanism in seq2seq model can be observed when generating the next word in the decoder network. In seq2seq model with attention mechanism, we determine the hidden state of the decoder at the current time by taking the previous output, previous hidden state and context vector. Further, note that here we are not using the single context vector c for generating all the words in the decoder network, but a separate context vector c_i for each target word $y_{T'}$.

The encoder first encodes input sentence represented by its word embedding sequence x, into a single context vector c (which is a combination of all the hidden units in encoder and represented by $c = q(h_1, ..., h_{T_x}))$ and a hidden state $h_t = f(x_t, h_{t-1})$. Typically decoder network predicts the sequence by predicting one word at a time denoted by y_t, where each y_t output is conditioned on the previous outputs $y_1, ..., y_{t-1}$ and the context vector c, maximizing the following joint probability

$$p(y) = \prod_{t=1}^{T'} p(y_t | y_1, ..., y_{t-1}, c). \tag{6}$$

In the context of RNNs, the conditional probability of each y_t in the joint probability of (6) is modeled as a nonlinear function g with input y_{t-1} context vector c and hidden state s_{t-1}:

$$p(y_t | y_1, ..., y_{t-1}, c) = g(y_{t-1}, s_{t-1}, c). \tag{7}$$

Then [2] proposes to use unique vector c_t for each decoding time step, redefining the decoder conditional probability for each word y_t as:

$$p(y_t | y_1, ..., y_{t-1}, x) = g(y_{t-1}, s_{t-1}, c_t), \tag{8}$$

where the context vector c_t is a weighted sum over all input hidden states $(h_1, ..., h_T)$:

$$c_t = \sum_{j=1}^{T} a_{tj} h_j. \tag{9}$$

Here, attention weights a_{tj} are calculated as

$$a_{tj} = \frac{exp(e_{tj})}{\sum_{k=1}^{T_x} exp(e_{tk})}. \tag{10}$$

$$e_{tj} = a(s_{t-1}, h_j) \tag{11}$$

Here the scoring function (11), is a pair-wise scoring function which is used for scoring the relation between decoder hidden state s_{t-1} and encoder's hidden state h_j. This scoring is learned jointly while training the whole seq2seq model with the help of a feedforward network.

There are many different kinds of attention mechanism, out of which in this work we have tried two different variations proposed in [8] and [2]. Moreover, we have used seq2seq model with an attention mechanism to compare with seq2seq model with pointer network in generating similar sentences.

6 Encoder-Decoder with Pointer Network

Encoder-Decoder network also suffers from two other problems, which are reproducing factual details or unknown words or rare word inaccurately, and also they tend to repeat themselves by generating the same words again and again. The second problem can be resolved by attention and coverage mechanisms [13].

Typically, we create a seq2seq model when we have to define the maximum value of vocabulary length, which is represented by their word embedding. Usually, this vocabulary length varies from 10,000 to 50,000, containing the maximum number of most frequent words. Note that an increase in the maximum length of vocabulary also increases computation complexity of seq2seq model and makes the training process slower. All the other words or tokens which are not incorporated under vocabulary are marked as '<UNK>' or unknown words and all these tokens have the same word embedding. Therefore whenever decoder is generating an output word of embedding <UNK> token, then decoder outputs <UNK> as a token. This is known as unknown words problem and can be very problematic in the case of paraphrase generation. To solve this unknown words problem, we use pointer network in seq2seq model.

6.1 COPYNET Network

CopyNet was proposed in [5] to incorporate copying mechanism in seq2seq model. This mechanism has shown good results on text summarization tasks on different datasets. The model uses bidirectional RNN as an encoder which transforms or encodes the variable length of the sentence into a fixed size of context vector. It has the same setup as proposed in [2]. The difference is in the way how model copies words from the input sentence in a decoder network.

The model has two sets of vocabularies: $V = V_1, ..., V_n$, which also includes <UNK> for out of vocabulary (OOV) words and the source vocabulary $A = a_1, ..a_N$, which includes unique vocabulary from input sentence. Source vocabulary makes COPYNET to copy OOV words in the output sentence.

At time t, the decoder RNN state is represented by s_t. The probability of a generated word y_t is given by

$$p(y_t|s_t, y_{t-1}, c_t, H) = p(y_t, g|s_t, y_{t-1}, c_t, H) + p(y_t, c|s_t, y_{t-1}, c_t, H), \tag{12}$$

where H is a combination of hidden states h_r of the encoder network, c_t is a context vector at t. Here g stands for generative mode and c stands for copy mode, and these probabilities are calculated as follows

$$p(y_t, g|\cdot) = \begin{cases} \frac{1}{F} e^{\Psi_g(y_t)} & y_t \in V \\ 0 & y_t \in A \cap \bar{V} \\ \frac{1}{F} e^{\Psi_g(UNK)} & y_t \notin V \cap A \end{cases}$$

$$p(y_t, c|\cdot) = \begin{cases} \frac{1}{F} \sum\limits_{j: x_j = y_t} e^{\Psi_c(x_j)} & y_t \in A \\ 0 & \text{otherwise,} \end{cases}$$

where F is a normalization term, and $e^{\Psi_c(\cdot)}$ and $e^{\Psi_g(\cdot)}$ are scoring functions for copy mode and generate mode respectively. Because of the shared normalization term both generate mode and copy mode probabilities are competing through softmax function (12). The scoring functions are calculated as follows

$$\Psi_g(y_t = v_i) = v_i^T W_o s_t, v_i \in V \cap UNK \tag{13}$$

and

$$\Psi_c(y_t = x_j) = \sigma(h_j^T W_c), x_j \in X, \tag{14}$$

where X is an input sentence, x_j is a word at j position, v_i and W_0 are one-hot indicator vector for word v_i from the vocabulary. Here σ is a nonlinear activation function.

COPYNET updates decoder RNN state at every time step t, using previous hidden state s_{t-1}, predicted word y_{t-1} and context vector c as follows

$$s_t = f(y_{t-1}, s_{t-1}, c). \tag{15}$$

However, if y_{t-1} is copied over to the output sentence then the decoder RNN states are updated by changing y_{t-1} to $[w(y_{t-1}); S(y_{t-1})]^T$, where $w(y_{t-1})$ is the word embeddings of y_{t-1} and $S(y_{t-1})$ is the weighted sum of hidden states in H or in encoder RNN network corresponding to y_t.

$$S(y_{t-1}) = \sum_{r=1}^{T} ptr \cdot h_r \tag{16}$$

$$ptr = \begin{cases} \frac{1}{K} p(x_i, c | s_{t-1}, H) & x_i = y_{t-1} \\ 0 & \text{otherwise.} \end{cases}$$

Here K is a normalizing term. Pointer network (ptr) is only concentrated on one location from source sentence. Although $S(y_{t-1})$ helps decoder to copy over subsequence from source sentence and is named as "selective read."

The COPYNET network is fully differentiable and can be trained end-to-end exactly like seq2seq model. It minimizes the negative log-likelihood as an objective loss function given by

$$J = -\frac{1}{N} \sum_{k=1}^{N} \sum_{t=1}^{T} \log[p(y_t | y_1, y_2, ...y_{t-1}, X)]. \tag{17}$$

6.2 Pointer Softmax Network

The Pointer Softmax Network (PS) was proposed in [14]. The idea is to use attention mechanism and attention weights to select a word or token from the input sequence as the output instead of using it to blend hidden units of an encoder to a context vector at each decoder step. This setup was able to copy a word from the input sentence to the output sentence, which is not present in seq2seq model vocabulary or is not seen by the model in the training process.

This approach shows the improvement in two tasks, i.e., neural machine translation on the Europarl English to French parallel corpora and text summarization on the Gigaword dataset.

The model learns two main things: 1) to predict whether the pointing mechanism is required at each time step 't'; 2) to point to the correct location of the word in source sentence which needs to be copied over to target sentence. The model uses two different softmax output layers, first, is *shortlist softmax layer*, and the second one is the *location softmax layer*. The first one is the softmax layer, which is used in the attention mechanism over all the vocabulary to generate a word from the model vocabulary. The second softmax is a location softmax, which is a pointer network in seq2seq model, where each of the output dimension corresponds to the location of a word in the context sequence. Consequently, the output dimension of the location softmax varies according to the length of the given source sequence. The decision whether the pointer word should be used or shortlisted is made by a switching network. The switching network is a multilayer perceptron network which takes the representation of source sequence and the previous hidden state from decoder RNN as an input and outputs the value of binary variable z_t which decides whether to shortlist softmax layer (when $z_t == 1$) or location softmax layer (when $z_t == 0$). When the word is not in the shortlist softmax and not even in the location softmax layer, this switching network chooses to shortlist softmax and gives <UNK> as the next token.

7 Experiments

7.1 Dataset

In the first part of the experiment, we compared seq2seq with attention model, pointer softmax model (PS) and the COPYNET network described in Section 5 and 6 for paraphrase generation. We train our model on the PPDB dataset [10]. Note that for higher precision, we have used medium size of PPDB dataset which has almost 9,103,492 pair sentences, which means their score for being a paraphrase is high. Before training the model, we preprocessed the PPDB dataset. We only took the sentence pair where the maximum number of words in a sentence is 50. We also removed all the punctuation signs from the source and target sentences. Consequently the word "I'm" becomes "I m," and "I haven't" becomes "I havent" and we consider them as two different words. After preprocessing we partition the dataset into three different categories with 80 percent of samples in the training set and 10-10 percentage of samples in testing and validation sets, respectively. The same dataset was used for both models for training, testing and validation.

7.2 Models

Here we compare three different models for paraphrase generation described in Section 5 and 6. The first model described in Section 5 only uses attention to generate paraphrases. The model is trained on the preprocessed PPDB dataset [10]. We have used word-level embedding in the model to represent the whole sentence. We first created a vocabulary of 30,000 most frequent words in training samples and represented them with a unique index. This vocabulary is also augmented with four extra tokens that are <UNK> for representing unknown words or the words which are not covered in the vocabulary of the model. <PAD> this is used to add extra spacing to a sentence to make it equal to the length of source sentence if the target or source sentence is smaller than length 50, <SOS> and <EOS> representing the start of sentence and end of sentence, respectively. Both <SOS> and <EOS> were added before the sentence starts and at the end of the sentence respectively. Using these unique indices, the words in source text sentence are converted into the list of integers, which is then input to the seq2seq model described in 5.

Furthermore, the weights and bias parameters of the model were learned by backpropagating the error at training time. This model was then tested and validated using test samples and validation dataset with different hyper-parameters like the number of hidden units, type of RNN cell used in the encoder-decoder model, batch size and different attention types. The output from the decoder is an index value of a generated token which is then converted back to a word by matching it to the model vocabulary and then combined to form one complete sentence. We have used the beam of size 3, which means we picked the top 3 sentences with the highest probabilities.

We followed the same preprocessing for the COPYNET and the pointer softmax model with different variants of coping mechanism in seq2seq model. Therefore while training the PS model with the encoder-decoder network, a separate multi-layer perceptron model was also trained and used for binary classification. We used standard binary cross-entropy as a loss function for backpropagating error in the model. We have also tried this model with different hyper-parameters of the model described in the previous section.

All three models were fine-tuned on hyperparameters and then compared against each other for paraphrase generation by finding the loss in dev dataset, BLEU and METEOR scores. To save time in training we have fixed some parameters like we used teacher forcing while training encoder-decoder model, dropout ratio was fixed at 0.2, vocabulary size was set to 30,000 most frequent words in training time, and the batch size was set to 250. Note that to our best knowledge, these models were previously compared only on summarizing of paragraphs and not on paraphrase generation of sentences.

As codes for all these model were not made publicly available, we have implemented all these models in Pytorch. We trained our models on GPU provided by Helios Calcul Quebec, which has 15 compute nodes, each of which has eight K20 GPU's from NVIDIA, and 6 compute nodes and eight NVIDIA K80 boards each. Each K80 board contains two GPU, for a total of 216 GPUs for the cluster.

8 Results and Analysis

Due to time limitation, to compare models, we first tained the attention model, PS and COPYNET model with only one epoch and with different hyper-parameters. It was done to find out the best configuration for the models to proceed using the first iteration. Then we trained every model over several epochs and after looking at the training and validation perplexity, we concluded that the models converged after 15 iterations. Tables below summarize the results.

Seq2Seq Model with Attention			
Hidden Layer	Number of Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	GRU	27.1790
128	1	LSTM	27.3762
256	1	GRU	28.1144
512	1	GRU	26.5589
128	**2**	**GRU**	**26.5401**
128	2	LSTM	26.7232

Table 1. Results of seq2seq with Attention model with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

Seq2Seq Model with Pointer softmax network			
Hidden Layer	Number of Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	LSTM	29.9218
128	**1**	**GRU**	**26.5936**
256	1	GRU	27.4747
512	1	GRU	26.8019
128	2	GRU	28.2140

Table 2. Results for seq2seq with Pointer softmax model with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

Seq2Seq Model with COPYNET network			
Hidden Layer	# Layers in RNN	Type of RNN Cell	Valid Perplexity
128	1	LSTM	26.6721
128	1	GRU	26.7842
256	1	GRU	26.9701
512	1	GRU	26.6891
128	**2**	**GRU**	**25.9625**
128	2	GRU	26.3713

Table 3. Results for seq2seq with COPYNET Pointer network with different hyper parameters on PPDB test dataset. Smaller perplexity indicates better performance

Model Comparison		
Model	BLEU-Score	METEOR-Score
$Seq2Seq_{attn}$	0.4538	0.3035
$Seq2Seq_{attn+COPYNET}$	**0.4547**	**0.3464**
$Seq2Seq_{attn+PS}$	0.2922	0.3219

Table 4. BLEU and METEOR score on test dataset of PPDB dataset with attention, COPYNET and Pointer softmax. Higher scores indicate better performance.

Tables 1, 2 and 3 demonstrate the validation perplexity on the PPDB dataset. Looking at the results, COPYNET outperforms the other two models by a small margin. Switching layer in the pointer softmax model did not help much in the generation of paraphrases. Table 4 compares performance of all different models on the test dataset consisting of 25,000 examples. The BLEU and METEOR scores were slightly better for COPYNET network versus other models.

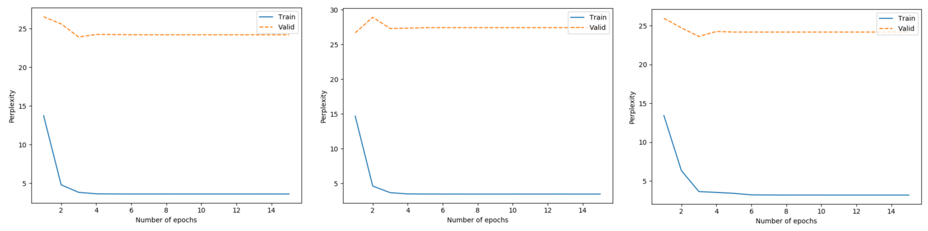

Fig. 1. Plots of perplexity of seq2seq model with attention model (left), pointer softmax model(middle) and COPYNET pointer network model on PPDB training and test data set.

Figure 1 presents training and validation perplexity curves. Seq2seq model with attention mechanism and with COPYNET network model both show the best performance at iteration 3, and they have minimum validation perplexity at this point, i.e., 23.9142 and 23.6172 respectively. On the other hand, the pointer softmax model gave the best result at one iteration, where we got minimum validation perplexity of 26.6837.

We next show examples of paraphrases generated by different models. Note, that source sentences were picked randomly and were not in the PPDB test dataset. Below, 'sentence' represents the original sentence which was given as an input to the model, while Paraphrase(X) represents the paraphrase generated by "X".

1. **Sentence**: Economy is a big problem for the Bush administration
   Paraphrase(Attn): <UNK> government problem is <UNK> <EOS>
   Paraphrase(COPYNET): Bush government problem is economy <EOS>
   Paraphrase(PS): George's government problem is big <EOS>
2. **Sentence**: Language is complex and the process of reading and understanding language is difficult for many groups of people
   Paraphrase(Attn): Language is difficult for <UNK> and <UNK> <eos>
   Paraphrase(COPYNET): Language is difficult for reading and understanding <eos>
   Paraphrase(PS): Speech is complex for understanding for many people<eos>

Glancing at the examples above, we conclude that seq2seq model with COPYNET network model generated better paraphrases, which is consistent with our earlier results.

9 Conclusions

We performed experiments on seq2seq model with attention and two different variants of pointer networks under the supervision of PPDB dataset and compared the results using BLEU and METEOR score metrics. In this experiment, COPYNET outperforms pointer softmax pointer network by a small margin. Observing examples of paraphrases generated by these models it can be concluded that COPYNET pointer network generates the best paraphrases among compared models.

References

1. I. Androutsopoulos and P. Malakasiotis, A survey of paraphrasing and textual entailment methods, Journal of Artificial Intelligence Research, vol. 38, pp. 135–187, 2010.
2. D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, Proceedings of The International Conference on Learning Representations (ICLR'2015), pp. 1–15, 2015.
3. M. E. Califf and R. J. Mooney, Bottom-up relational learning of pattern matching rules for information extraction, Journal of Machine Learning Research, vol. 4, pp. 177–210, 2003.
4. K. Cho et al, Learning phrase representations using RNN encoder-decoder for statistical machine translation, Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1724–1734, 2014.
5. J. Gu, Z. Lu, H. Li, and V. O. Li, Incorporating copying mechanism in sequence-to-sequence learning, in Proceedings of the 54-th Annual Meeting of the Association for Computational Linguistics, pp. 1631–1640, 2016.
6. Z. S. Harris, Transformational Theory. Springer Netherlands, 1970.
7. D. Inkpen, Building a lexical knowledge-base of near-synonym differences, Computational Linguistics, vol. 32, pp. 223–262, 2004.
8. M. Luong, H. Pham, and C. D. Manning, Effective approaches to attentionbased neural machine translation, vol. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421, 2015.
9. K. R. McKeown, Paraphrasing questions using given and new information, Journal of Computational Linguistic, vol. 9, no. 1, pp. 1–10, 1983.
10. E. Pavlick and C. Callison-Burch, Simple PPDB: A paraphrase database for simplification, Proceedings of the 54-th Annual Meeting of the Association for Computational Linguistics, pp. 143–148, 2016.
11. A. Prakash et al., Neural paraphrase generation with stacked residual LSTM networks, Proceedings of the 26-th International Conference on Computational Linguistics COLING'2016, pp. 2923–2934, 2016.
12. I. Sutskever, O. Vinyals, and Q. V. Le, Sequence to sequence learning with neural networks, Advances in Neural Information Processing Systems (NIPS'2014), pp. 3104–3112, 2014.
13. Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, Coverage-based neural machine translation, Proceedings of the 54-th Annual Meeting of the Association for Computational Linguistics, pp. 76–85–430, 2016.
14. O. Vinyals, M. Fortunato, and N. Jaitly, Pointer networks, Advances in Neural Information Processing Systems (NIPS'2015), pp. 2692–2700, 2015.