The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Sequential Copying Networks

Qingyu Zhou,[†*] **Nan Yang,**[‡] **Furu Wei,**[‡] **Ming Zhou**[‡]
[†]Harbin Institute of Technology, Harbin, China
[‡]Microsoft Research, Beijing, China
qyzhgm@gmail.com, {nanya, fuwei, mingzhou}@microsoft.com

Abstract

Copying mechanism shows effectiveness in sequence-to-sequence based neural network models for text generation tasks, such as abstractive sentence summarization and question generation. However, existing works on modeling copying or pointing mechanism only considers single word copying from the source sentences. In this paper, we propose a novel copying framework, named Sequential Copying Networks (SeqCopyNet), which not only learns to copy single words, but also copies sequences from the input sentence. It leverages the pointer networks to explicitly select a subspan from the source side to target side, and integrates this sequential copying mechanism to the generation process in the encoder-decoder paradigm. Experiments on abstractive sentence summarization and question generation tasks show that the proposed SeqCopyNet can copy meaningful spans and outperforms the baseline models.

Introduction

Recently, attention-based sequence-to-sequence (seq2seq) framework (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015) has achieved remarkable progress in text generation tasks, such as abstractive text summarization (Rush, Chopra, and Weston 2015), question generation (Zhou et al. 2017a) and conversation response generation (Vinyals and Le 2015). In this framework, an encoder is employed to read the input sequence and produce a list of vectors, which are then fed into a decoder to generate the output sequence by making word predictions one by one through the softmax operation over a fixed size target vocabulary.

It has been observed that seq2seq suffers from the unknown or rare words problem (Luong et al. 2015). Gulcehre et al. (2016) and Gu et al. (2016) makes the key observation that in tasks like summarization and response generation, rare words in the output sequence usually can be found in the input sequence. Based on this observation, they propose a copying mechanism to directly copy words to the output sequence from input, which alleviates the rare word problem. In their work, every output words can be either generated by predicting words in the target vocabulary or copied from the input sequence.

[*]Contribution during internship at Microsoft Research.

We further observe that the copied words usually form a continuous chunk of the output, exhibiting a "sequential copying" phenomenon. For example, in the Gigaword dataset of abstractive sentence summarization task, about 57.7% words are copied from the input as indicated in Figure 1. Moreover, the copied words in multi-word span account for 28.1%, which is also very common. For example, in Figure 2, there are two copied bi-grams in the output summary. Similar phenomenon has also been observed in question generation task.

However, previous methods fall into one paradigm, which we call "single word copy". At each decoding time step, the models still follow the "word by word" style to make separate decisions of whether to copy. Therefore, this "single word copy" paradigm may introduce errors due to these separate decisions. For example, the words in a phrase should be copied consecutively from the input sentence, but these separate decisions cannot guarantee to achieve this. This may cause that some unrelated words appears unexpectedly in the middle of the phrase, or the phrase is not copied completely with some words missed. Therefore, we argue that tasks such as abstractive sentence summarization and question generation can benefit from sequential copying considering the intrinsic nature of these tasks and datasets.

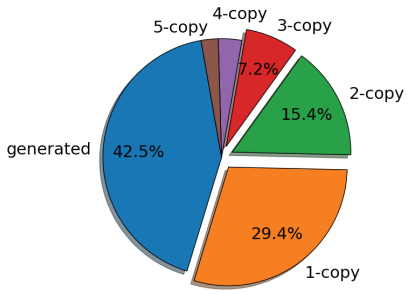

Figure 1: Percentage of generated and copied words in sentence summarization training data.

In this paper, we propose a novel copying framework, **Seq**uential **Copy**ing **Net**works (SeqCopyNet), to extend the vanilla seq2seq framework. SeqCopyNet is intended to learn not only the "single word copy" behavior, but also the "se-

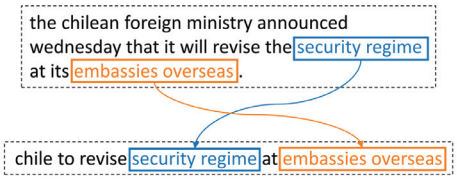

Figure 2: An example of sequential copying in abstractive sentence summarization task.

quence copy" operation as mentioned above. We design a span extractor for the decoder so it can make "sequence copy" actions during decoding. Specifically, SeqCopyNet consists of three main components, an RNN based sentence encoder, an attention-equipped decoder, and the newly designed copying module. We follow previous works to use the bidirectional RNN as the sentence encoder, and the decoder also employs an RNN with attention mechanism (Bahdanau, Cho, and Bengio 2015). To achieve the sequential copying mechanism, the copying module is integrated with the decoder to make decisions during decoding.

The sequential copying module in SeqCopyNet contains three main components, namely, the copy switch gate network, the pointer network and the copy state transducer. The copy switch gate network is used to make decisions of whether to copy according to the current decoding states. Its output is not a binary value, but a scalar range in [0, 1], which is the probability of choosing to copy. The pointer network is then used to extract a span from the input sentence. We maintains a copying state in the copying module so that the pointer network can make predictions based on it. In detail, the pointer network predicts the start and end positions of the span. The start position is predicted using the start copying state. Then the copy state transducer will update the copying state so that the pointer network can predict the end position. This transduction process is made by an RNN so that it can remember related information such as the start position, and guide the pointer to the corresponding end copying position.

We conduct experiments on abstractive sentence summarization and question generation tasks to verify the effectiveness of SeqCopyNet. On both tasks, SeqCopyNet outperforms the baseline models and the case study show that it can copy meaningful spans.

Sequential Copying Networks

As shown in Figure 3, our SeqCopyNet consists of three main components, namely the encoder, the copying module and the decoder. Like in vanilla seq2seq frameworks, the encoder leverages two Gated Recurrent Unit (GRU) (Cho et al. 2014) to read the input words, and the decoder is modeled with GRU with attention mechanism. The copying module consists of a copy switch gate network, a pointer network and a recurrent copy state transducer. At each decoding time step, the copying module will make a decision of whether to copy or generate. If it decides to copy, the pointer network and the copy state transducer will cooperate to copy a sub-span from the input sentence by predicting the start and end positions of it. After the copying action, if the copied sequence contains more than one word, the decoder will apply "Copy Run" to update its states accordingly.

Encoder

The role of the sentence encoder is to read the input sentence and construct the basic sentence representation. Here we employ a bidirectional GRU (BiGRU) as the recurrent unit, where GRU is defined as:

$$z_i = \sigma(\mathbf{W}_z[x_i, h_{i-1}]) \quad (1)$$

$$r_i = \sigma(\mathbf{W}_r[x_i, h_{i-1}]) \quad (2)$$

$$\widetilde{h}_i = \tanh(\mathbf{W}_h[x_i, r_i \odot h_{i-1}]) \quad (3)$$

$$h_i = (1 - z_i) \odot h_{i-1} + z_i \odot \widetilde{h}_i \quad (4)$$

where $\mathbf{W}_z$, $\mathbf{W}_r$ and $\mathbf{W}_h$ are weight matrices.

The BiGRU consists of a forward GRU and a backward GRU. The forward GRU reads the input sentence word embeddings from left to right and gets a sequence of hidden states, $(\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_n)$. The backward GRU reads the input sentence embeddings reversely, from right to left, and results in another sequence of hidden states, $(\overleftarrow{h}_1, \overleftarrow{h}_2, \ldots, \overleftarrow{h}_n)$:

$$\vec{h}_i = \text{GRU}(x_i, \vec{h}_{i-1}) \quad (5)$$

$$\overleftarrow{h}_i = \text{GRU}(x_i, \overleftarrow{h}_{i+1}) \quad (6)$$

The initial states of the BiGRU are set to zero vectors, i.e., $\vec{h}_1 = 0$ and $\overleftarrow{h}_n = 0$. After reading the sentence, the forward and backward hidden states are concatenated, i.e., $h_i = [\vec{h}_i; \overleftarrow{h}_i]$, to get the basic sentence representation.

Sequential Copying Mechanism

To model the sequential copying mechanism, SeqCopyNet needs three key abilities: a) at decoding time step t, the model needs to decide whether to copy or not; b) if the model decides to copy, it will need to select a sub-span from the input; c) the decoder should switch between the generate mode and copy mode smoothly. To enable our SeqCopyNet of the first two functions, we design the copying module as a three-part component, i.e., the copy switch gate network, the pointer network and the copy state transducer. The last ability is enabled by Copy Run method, which is described in the next section. The copy switch gate network decides whether to copy during decoding. If the model goes to generate mode, then it will generate next words as same as the vanilla attention-base seq2seq model. If the model choose to copy, the pointer network will predict a sub-span.

At each time-step t, the decoder GRU holds its previous hidden state s_{t-1}, the previous output word y_{t-1} and the previous context vector c_{t-1}. With these previous states, the decoder GRU updates its states as given by formula 7 . To initialize the GRU hidden state, we use a linear layer with the last backward encoder hidden state $\overleftarrow{h}_1$ as input:

$$s_t = \text{GRU}(y_{t-1}, c_{t-1}, s_{t-1}) \quad (7)$$

$$s_0 = \tanh(\mathbf{W}_d \overleftarrow{h}_1 + b) \quad (8)$$

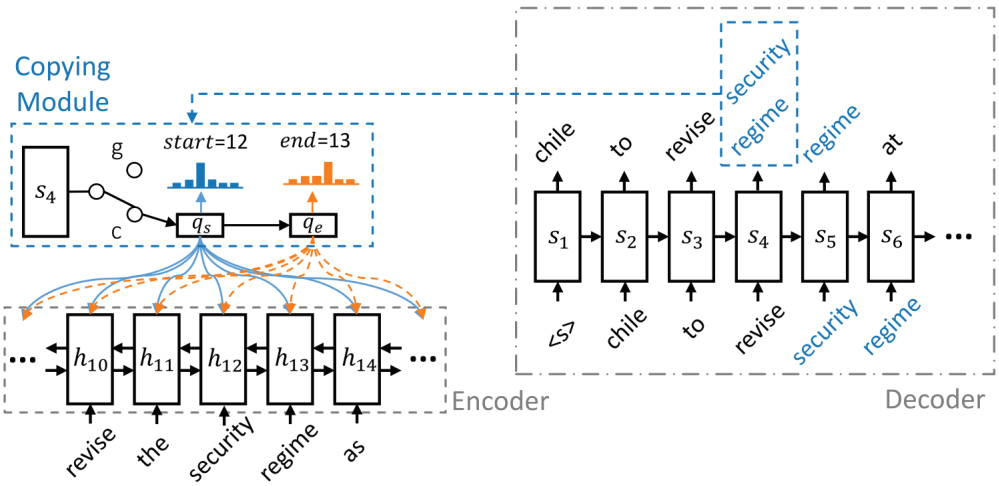

Figure 3: The overview diagram of SeqCopyNet. For simplicity, we omit some units and connections. The copying process of the sequence "security regime" is magnified as indicated in the copying module part.

With the new decoder hidden state s_t, the context vector c_t for current time step t is computed through the concatenate attention mechanism (Luong, Pham, and Manning 2015), which matches the current decoder state s_t with each encoder hidden state h_i to get an importance score. The importance scores are then normalized to get the current context vector by weighted sum:

$$e_{t,i} = v_a^\top \tanh(\mathbf{W}_a s_t + \mathbf{U}_a h_i) \tag{9}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^{n} \exp(e_{t,i})} \tag{10}$$

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{11}$$

where $\mathbf{W}_a$ and $\mathbf{U}_a$ are learnable parameters.

We then construct a new state vector and name it as decoder memory vector m_t, which is the concatenation of the embedding of previous output word y_{t-1}, the decoder GRU hidden vector s_t and the current context vector c_t:

$$m_t = \begin{bmatrix} y_{t-1} \\ s_t \\ c_t \end{bmatrix} \tag{12}$$

In SeqCopyNet, the decoder memory vector m_t plays an important role. In the copying module, the copy switch gate network makes decisions based on m_t. Specifically, the copy switch gate network ($\mathcal{G}$) is a Multilayer Perceptron (MLP) with two hidden layers:

$$\mathcal{G}(x) = \sigma\left(\mathbf{W_2}(\tanh(\mathbf{W_1}x + b_1)) + b_2\right) \tag{13}$$

where $\mathbf{W_1}$, $\mathbf{W_2}$, b_1 and b_2 are learnable parameters. The activation function of the first hidden layer is hyperbolic tangent (tanh). To produce a probability of whether to copy, we use the sigmoid function ($\sigma(\cdot)$ in Equation 13) as the activation function of the last hidden layer. The copy probability p_c and generate probability p_g are defined as:

$$p_c = \mathcal{G}(m_t) \tag{14}$$

$$p_g = 1 - p_c \tag{15}$$

Generate Mode If the copy switch gate network decides to generate, SeqCopyNet will generate the next word using the decoder memory vector m_t. The decoder first generates a readout state r_t and then pass it through a maxout hidden layer (Goodfellow et al. 2013) to predict the next word with a softmax layer over the decoder vocabulary.

$$r_t = \mathbf{W}_r w_{y-1} + \mathbf{U}_r c_t + \mathbf{V}_r s_t \tag{16}$$

$$r'_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]_{j=1,\ldots,d}^\top \tag{17}$$

$$p(y_t|y_{<t}) = \text{softmax}(\mathbf{W}_o r'_t) \tag{18}$$

where $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_o$ are weight matrices. Readout state r_t is a $2d$-dimensional vector, and the maxout layer (Equation 17) picks the max value for every two numbers in r_t and produces a d-dimensional maxout vector r'_t. We then apply a linear transformation on r'_t to get a target vocabulary size vector and predict the next word y_t with the softmax operation.

Copy Mode If the copy switch gate network decides to copy, then SeqCopyNet uses its pointer network to predict a sub-span in the input sentence. In detail, the pointer network makes two predictions, i.e., the start position prediction and the end position prediction. The pointer network makes these predictions based on the decoder state. Before deciding the start position, the copying module first generate a start query vector q_s using the decoder memory vector m_t:

$$q_s = \tanh(\mathbf{W}_s m_t + b) \tag{19}$$

Using the start query vector q_s, the pointer network pre-

dicts the start position copy_s of the sub-span:

$$e_{s,i} = v_p^\top \tanh(\mathbf{W}_p q_s + \mathbf{U}_p h_i) \tag{20}$$

$$\alpha_{s,i} = \frac{\exp(e_{s,i})}{\sum_{i=1}^{n} \exp(e_{s,i})} \tag{21}$$

$$\text{copy}_\text{s} = \arg\max_i \alpha_{s,i} \tag{22}$$

$$p_{\text{copy\_s}} = \alpha_{s,\text{copy}_\text{s}} \tag{23}$$

$$c_s = \sum_{i=1}^{n} \alpha_{s,i} h_i \tag{24}$$

where c_s is the copying context state vector, $p_{\text{copy\_s}}$ is the probability of copy_s being the start copying position.

After predicting the start position, the copy state transducer will generate the end position query vector q_e. It is modeled with a single layer MLP and a GRU. In detail, the MLP first produces an initial GRU hidden state cst. Then the GRU generates an end position query based on this hidden state cst and the copying context state c_s:

$$cst = \tanh(\mathbf{W}_e m_t + b) \tag{25}$$

$$q_e = \text{GRU}(cst, c_s) \tag{26}$$

$$e_{e,i} = v_p^\top \tanh(\mathbf{W}_p q_e + \mathbf{U}_p h_i) \tag{27}$$

$$\alpha_{e,i} = \frac{\exp(e_{e,i})}{\sum_{i=1}^{n} \exp(e_{e,i})} \tag{28}$$

$$\text{copy}_\text{e} = \arg\max_i \alpha_{e,i} \tag{29}$$

$$p_{\text{copy\_e}} = \alpha_{e,\text{copy}_\text{e}} \tag{30}$$

where copy_e is the end position and $p_{\text{copy\_e}}$ is the probability of copy_e being the end copying position given the condition that copy_s is the start position.

After predicting the start and end positions, we can calculate the probability of copying this sub-span:

$$\begin{aligned} p(\text{copy\_mode}, y_t = (x_{\text{copy}_\text{s}}, \ldots, x_{\text{copy}_\text{e}}) | y_{<t}) \\ = p_c * p_{\text{copy\_s}} * p_{\text{copy\_e}} \end{aligned} \tag{31}$$

Copy Run for Multi-word Span

Since SeqCopyNet may choose to copy a long sequence (copy mode) or generate a word from target vocabulary (generate mode), the decoder should adapt itself to smoothly switching between these two modes. Suppose the decoder just copies a five-word span and the next word is generated, if we just skip these five copied words, the decoder RNN cannot remember the previous words when generating the next word. This may lead to ill decoder RNN hidden states and poor performance of the sentence fluency and quality. To solve this problem, we make the decoder GRU keep its state updated when it copies a long sequence. Inspired by Tang et al. (2016), we propose a method, named Copy Run, to solve this problem in both training and testing phases.

During training, Copy Run solves this problem by keeping the decoder running over all output words. By doing this, the decoder can learn from a complete sequence so the lack of decoder states update problem can be avoided.

During testing, Copy Run maintains the decoder GRU states by running over the copied words. According to our model architecture, SeqCopyNet decoder consumes the last generated word y_{t-1} and GRU state s_{t-1} at time step t. Therefore, the Copy Run only needs to be applied if the decoder copies a sequence whose length is larger than 1. For a copied sequence with length $l \geq 2$, the Copy Run updates the decoder states for $l-1$ times.

For example in Figure 3, at time step 4, the decoder decides to copy a two-word ($l = 2$) sequence (x_{12}, x_{13}). Using Copy Run, the decoder will feed the first $l-1$ words into the GRU and generate a pseudo state s_5. Specifically, after the copying of (x_{12}, x_{13}), the copy run will execute the decoder as mentioned in Equation 7, 9, 10 and 11 to update the GRU state s_t and the context vector c_t, to prepare for the next time step. For the rare words in copied sequence, the copy run will instead feed the embedding of $\langle unk \rangle$ to update the decoder GRU states.

Objective Function

Given a training dataset with n input-output pairs $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$, where the k-th pair is $(x^{(k)}, y^{(k)}) = \left((x_1^{(k)}, \ldots, x_{T_x}^{(k)}), (y_1^{(k)}, \ldots, y_{T_y}^{(k)})\right)$ and T_x and T_y are the lengths of $x^{(k)}$ and $y^{(k)}$ respectively. For the k-th training instance $(x^{(k)}, y^{(k)})$, the set C_k contains the copied spans in $y^{(k)}$. Our training objective is to minimize the negative log likelihood loss $\mathcal{L}$ with respect to the learnable model parameter θ:

$$\mathcal{L} = -\frac{1}{n} \sum_{i=1}^{n} \Big(\sum_{t=1}^{T_y} \log p_g p(y_t) + \sum_{span \in C_k} \log p_c p_{\text{start}} p_{\text{end}} \Big) \tag{32}$$

Beam Search

Beam search is a common practical searching strategy to improve results for many tasks such as machine translation and dependency parsing. We report both greedy search and beam search result in the experiments. During beam search, we normalize the score of a beam path with its length. For vanilla seq2seq models, this length equals to the number of generated words. For SeqCopyNet, the length is defined as the summation of generated words and copied sequences. Taking the output in Figure 2 as an example, the output sentence is " chile to revise [security regime] at [embassies overseas]", the length normalization term is 6, which means 4 generated words and 2 copied sequences. We empirically set beam size to 8 in our experiments.

Experiments

We conduct experiments on the abstractive sentence summarization and question generation tasks to demonstrate the effectiveness of SeqCopyNet.

Abstractive Sentence Summarization

Sentence summarization aims to shorten a given sentence and produce a brief summary of it, and the models can be roughly categorized into extractive and abstractive methods. Extractive methods select words from given inputs to form

final summary sentence, while abstractive methods generate output summary after reading the input. These two methods have their merits, the extractive methods use exactly the same words so the summary sentence is accurate, the abstractive methods can perform paraphrasing so the output can be more diverse. After analyzing the dataset, we found that copying a sequence from input sentence happens in 57.5% sentence-summary pairs (as shown in Figure 1). Therefore, we conduct experiments on this task.

Dataset We conduct abstractive sentence summarization experiment on English Gigaword dataset, as mentioned in Rush, Chopra, and Weston (2015). The parallel corpus is produced by pairing the first sentence and the headline in the news article with some heuristic rules. We modify the script released by Rush, Chopra, and Weston (2015) to pre-process and extract the training and development datasets. We obtain the test set used by Rush, Chopra, and Weston (2015).

However, like previous works mentioned (Chopra, Auli, and Rush 2016; Zhou et al. 2017b), there are many invalid lines in it and the scores reported on it cannot fully demonstrate the performance of the models. So we acquired the test set sampled by Zhou et al. (2017b). But we find that this test set is processed similar to Rush, Chopra, and Weston (2015), that the rare words and numbers have already been replaced by $\langle unk \rangle$ and # symbol. On test set like this, the copying methods can barely work since the unknown words and numbers in the references are already replaced, which is as shown in the experimental results. Therefore, we further sample and release[1] a new test set, in which the sentence-summary pairs are remained untouched. During training, we set the maximum copying length to 5.

Baseline We compare SeqCopyNet with the following baselines on abstractive sentence summarization task:

ABS Rush, Chopra, and Weston (2015) use an attentive CNN encoder and NNLM decoder for this task.

ABS+ Based on ABS model, Rush, Chopra, and Weston (2015) further tune ABS using DUC 2003 dataset.

RAS-Elman As an extension of ABS, Chopra, Auli, and Rush (2016) use a convolutional attention-based encoder and RNN decoder, which outperforms the ABS model.

Feats2s Nallapati et al. (2016) use a full RNN sequence-to-sequence encoder-decoder model and add some features to enhance the encoder, such as POS tag, NER, and so on.

Luong-NMT Neural machine translation model of Luong, Pham, and Manning (2015) with two-layer encoder-decoder implemented in Chopra, Auli, and Rush (2016).

s2s+att We also implement a sequence-to-sequence model with attention as our baseline and denote it as "s2s+att".

NMT + UNK_PS (single copy) We implement the UNK pointer softmax (PS) proposed by Gulcehre et al. (2016).

SEASS Selective encoding model for abstractive sentence summarization proposed by Zhou et al. (2017b).

[1]We release the preprocessing script and this test set at http://res.qyzhou.me

Evaluation Metric We employ ROUGE (Lin 2004) as our evaluation metric. ROUGE measures the quality of summary by computing overlapping lexical units, such as unigram, bigram, trigram, and longest common subsequence (LCS). It becomes the standard evaluation metric for DUC shared tasks and popular for summarization evaluation. Following previous work, we use ROUGE-1 (unigram), ROUGE-2 (bigram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

Model Parameters and Training The input and output vocabularies are collected from the training data with omitting the words appearing less than 20 times, which have 67,171 and 36,444 word types respectively. We set the word embedding size to 300 and all GRU hidden state sizes to 512. We use dropout (Srivastava et al. 2014) with probability $p = 0.4$. We initialize model parameters randomly using a Gaussian distribution with Xavier scheme (Glorot and Bengio 2010). We use Adam (Kingma and Ba 2015) as our optimizing algorithm. For the hyperparameters of Adam optimizer, we set the learning rate $\alpha = 0.001$, two momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ respectively, and $\epsilon = 10^{-8}$. During training, we test the model performance (ROUGE-2 F1) on development set for every 2,000 batches. We use the learning rate decay method, which is to halve the Adam learning rate α if the ROUGE-2 F1 score drops for six consecutive tests on development set. We also apply gradient clipping (Pascanu, Mikolov, and Bengio 2013) with range $[-5, 5]$ during training. To both speed up the training and converge quickly, we use mini-batch size 64 by grid search. During test, we do post-processing by replacing the $\langle unk \rangle$ with the token that has the highest attention score.

Models	RG-1	RG-2	RG-L
ABS (beam)‡	29.55	11.32	26.42
ABS+ (beam)‡	29.76	11.88	26.96
Feats2s (beam)‡	32.67	15.59	30.64
RAS-Elman (greedy)‡	33.10	14.45	30.25
RAS-Elman (beam)‡	33.78	15.97	31.15
Luong-NMT (beam)‡	33.10	14.45	30.71
s2s+att (greedy)	34.95	16.51	32.54
s2s+att (beam)	35.77	17.34	33.24
NMT + UNK_PS (greedy)	34.97	16.51	32.53
NMT + UNK_PS (beam)	35.67	17.44	33.19
SeqCopyNet (greedy)	35.33	16.66	32.90
SeqCopyNet (beam)	**35.93**	**17.51**	**33.35**

Table 1: Full length ROUGE F1 evaluation results on the English Gigaword test set used by Rush, Chopra, and Weston (2015). RG in the Table denotes ROUGE. Results with ‡ mark are taken from the corresponding papers.

Results We use the official ROUGE script (version 1.5.5) [2] to evaluate the summarization quality in our experiments.

[2]http://www.berouge.com/

	Test set in Zhou et al. (2017b)			Our internal test set		
Models	**RG-1**	**RG-2**	**RG-L**	**RG-1**	**RG-2**	**RG-L**
ABS‡	37.41-	15.87-	34.70-	-	-	-
s2s+att (greedy)	46.21	24.02	43.30	45.46	22.83	42.66
s2s+att (beam)	47.08	25.11	43.81	46.54	24.18	43.55
NMT + UNK_PS (greedy)	45.64	23.38	42.67	45.21	23.01	42.38
NMT + UNK_PS (beam)	47.05	24.82	43.87	46.52	24.41	43.58
SEASS (greedy)‡	45.27	22.88	42.20	-	-	-
SEASS (beam)‡	46.86	24.58	43.53	-	-	-
SeqCopyNet (greedy)	46.51	24.14	43.20	46.08	23.99	43.26
SeqCopyNet (beam)	**47.27**	25.07	**44.00**	**47.13**	**24.93**	**44.06**

Table 2: Full length ROUGE F1 evaluation on English Gigaword test set of Zhou et al. (2017b) and our internal test set. Results with ‡ mark are taken from the corresponding papers. The ABS baseline fails to run with the latest stable Torch framework, so we omit its performance on our internal test set.

For English Gigaword test sets the outputs have different lengths so we evaluate the system with F1 metric [3].

Table 1 and 2 give the performance in terms of ROUGE-F1 of the SeqCopyNet and the baseline models on three test sets, i.e., Rush, Chopra, and Weston (2015) test set, Zhou et al. (2017b) test set and our internal test set. As indicated in these tables, SeqCopyNet performs better than all the baseline models. As previously mentioned, the performance of copying methods performs comparably to the s2s+att baseline on test sets of Rush, Chopra, and Weston (2015) and Zhou et al. (2017b), since the rare words and numbers have already been replaced in the references.

Therefore, on the untouched test set, which is more likely in the real application scenarios, the copying methods performs better than the vanilla seq2seq baseline. SeqCopyNet achieves 47.13 ROUGE-1, 24.93 ROUGE-2 and 44.06 ROUGE-L F1 scores on this test set and performs better than the "single word copy" model.

Case Study Table 3 gives three summarization examples generated by SeqCopyNet. These examples show that SeqCopyNet can choose the correct span and generate meaningful summaries. We also observe that the copying of named entities are very common as shown in the examples. We can see that SeqCopyNet can copy a sequence more completely while the "single copy" model occasionally misses some words such as the first and second examples.

Question Generation

Automatic question generation from natural language text aims to generate questions taking text as input, which has the potential value of education purpose (Heilman 2011). As the reverse task of question answering, question generation also has the potential for providing a large scale corpus of question-answer pairs (Zhou et al. 2017a). In this task, we also found that sequential copying is frequent so it may benefit from this.

[3]The ROUGE evaluation option is the same as Rush, Chopra, and Weston (2015), -m -n 2 -w 1.2

Model Following Zhou et al. (2017a), we change the encoder to a feature-rich encoder and combine it with the SeqCopyNet. The feature-rich encoder reads the sentence words and handcrafted features to produce a sequence of word-and-feature vectors. In detail, Zhou et al. (2017a) use four features, namely answer position, word case, POS tag and NER tag. We follow this work and change the encoder in SeqCopyNet accordingly.

Dataset and Evaluation Metric We use the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al. 2016) as our training data. SQuAD is composed of more than 100K questions posed by crowd workers on 536 Wikipedia articles. Following (Zhou et al. 2017a), we acquired their training, development and test sets, which contain 86,635, 8,965 and 8,964 triples respectively. We also use the same Stanford CoreNLP v3.7.0 (Manning et al. 2014) to annotate POS and NER tags in sentences with its default configuration and pre-trained models. We evaluate the model using BLEU-4 score (Papineni et al. 2002). According to the results in Zhou et al. (2017b) there is correlation between BLEU score and human evaluation, therefore we only report the model performance with BLEU metric.

Baseline We compare SeqCopyNet with the following baselines on question generation task:

PCFG-Trans The rule-based system modified on the code released by Heilman (2011). We modified the code so that it can generate question based on a given word span.

s2s+att The seq2seq with attention mechanism without rich features as the baseline method.

NQG The s2s+att baseline with the feature-rich encoder.

NQG+ (single copy) Based on NQG, pointing mechanism (Gulcehre et al. 2016) is used to deal with rare words problem.

Results We report BLEU-4 scores on both development and test sets in Table 4. Note that the s2s+att baseline performs poorly since it does know the answer position information. SeqCopyNet outperforms the baseline models with

Input:	david ortiz homered and scored three times , including the go-ahead run in the eighth inning , as the boston *red sox* beat the toronto *blue jays 10-9* in the american league on tuesday .
Reference:	david ortiz helps red sox beat blue jays 10-9
SingleCopy:	ortiz homers as red sox beat blue jays
SeqCopyNet:	[*red sox*] beat [*blue jays 10-9*]
Input:	*guyana 's president cheddi jagan* , a long-time marxist turned free - marketeer , died here early thursday , an embassy spokeswoman said . he was 78 .
Reference:	guyana 's president cheddi jagan marxist turned marketeer dies at 78
SingleCopy:	guyana 's president jagan dies at 78
SeqCopyNet:	[*guyana 's president cheddi jagan*] dies at 78
Input:	china topped myanmar 's marine *product exporting countries annually* in the past decade among over 40 's , the local voice weekly quoted the marine products producers and exporters association as reporting sunday .
Reference:	china tops myanmars marine product exporting countries in past
SingleCopy:	china tops myanmar 's marine product export
SeqCopyNet:	china tops myanmar 's marine [*product exporting countries annually*]

Table 3: Examples of generated summaries. The highlighted italic words in brackets are copied as a sequence by SeqCopyNet.

Model	Dev set	Test set
PCFG-Trans‡	9.28	9.31
s2s+att‡	3.01	3.06
NQG‡	10.06	10.13
NQG+‡ (single copy)	12.30	12.18
SeqCopyNet	**13.13**	**13.02**

Table 4: BLEU-4 evaluation results. Models with ‡ mark are taken from the corresponding papers.

13.02 BLEU-4 score on test set and achieves the state-of-the-art result on this dataset.

Related Work

Sequence-to-Sequence Learning

Sutskever, Vinyals, and Le (2014) propose sequence-to-sequence framework with Long Short-Term Memory (LSTM). It uses LSTM to encode the input words to a single hidden vector. Another LSTM is used to decode this sentence meaning vector to produce the target sentence. Bahdanau, Cho, and Bengio (2015) extend it by introducing attention mechanism to align source and target words. The encoder produces a list hidden vectors instead of single vector so the decoder can dynamically attended to different encoded vectors. This brings huge improvements in many seq2seq learning tasks, such as NMT (Bahdanau, Cho, and Bengio 2015; Luong, Pham, and Manning 2015), syntactic parsing (Vinyals et al. 2015) and abstractive text summarization (Rush, Chopra, and Weston 2015).

Pointing / Copying Mechanism

Gu et al. (2016) and Gulcehre et al. (2016) propose similar copying / pointing mechanisms. Gu et al. (2016) propose to score the source words and target vocabulary words together. This method builds an extended vocabulary which is the union of target vocabulary, source words and $\{unk\}$. Then they apply a softmax over this extended vocabulary to decide the next output from this extended vocabulary. Gulcehre et al. (2016) use a pointer softmax to handle the $\langle unk \rangle$ or rare words problem in seq2seq learning. During decoding, they use a copying gate to predict the copying probability and use the attention mechanism to choose the copied word. They tried UNK pointer and entity pointer, and show that UNK pointer is better than entity pointer for abstractive sentence summarization task.

Vinyals, Fortunato, and Jaitly (2015) propose Pointer Network to model the pointing behavior. The pointer network is modeled with RNN and the pointing score is calculated with attention mechanism. Wang and Jiang (2016) propose match-LSTM and leverage pointer networks for Stanford Question Answering competition (Rajpurkar et al. 2016). The match-LSTM first matches the words in the query and passage, and then the pointer network predicts the start and end positions of a answer span.

Incorporating Phrase Information in NMT

Phrase-based machine translation performs very well among the SMT systems (Chiang 2005; Koehn 2009). Recently, much work has been done to incorporate phrase information in NMT to further boost translation quality. Tang et al. (2016), Wang et al. (2017), and Dahlmann et al. (2017) propose similar approaches which use NMT decoder to select phrases generated by SMT system.

Conclusion

In this paper, we propose Sequential Copying Networks (SeqCopyNet) to model the sequential copying phenomenon in sequence-to-sequence generation. It leverages the pointer networks as the prediction component to dynamically extract spans from input sentence during the decoding process. Experiments on abstractive sentence summarization and question generation tasks show that SeqCopyNet has ability of copying a sub-span from input sentence. In the future, we will apply SeqCopyNet to other tasks such as multi-turn dialog response generation.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of 3rd ICLR*.

Chiang, D. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*, ACL '05, 263–270. Stroudsburg, PA, USA: Association for Computational Linguistics.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP*, 1724–1734. Doha, Qatar: Association for Computational Linguistics.

Chopra, S.; Auli, M.; and Rush, A. M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*, 93–98. San Diego, California: Association for Computational Linguistics.

Dahlmann, L.; Matusov, E.; Petrushkov, P.; and Khadivi, S. 2017. Neural machine translation leveraging phrase-based models in a hybrid search. *arXiv preprint arXiv:1708.03271*.

Glorot, X., and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, 249–256.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A. C.; and Bengio, Y. 2013. Maxout networks. *ICML (3)* 28:1319–1327.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1631–1640. Berlin, Germany: Association for Computational Linguistics.

Gulcehre, C.; Ahn, S.; Nallapati, R.; Zhou, B.; and Bengio, Y. 2016. Pointing the unknown words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 140–149. Berlin, Germany: Association for Computational Linguistics.

Heilman, M. 2011. *Automatic factual question generation from text*. Ph.D. Dissertation, Carnegie Mellon University.

Kingma, D., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd ICLR*.

Koehn, P. 2009. *Statistical machine translation*. Cambridge University Press.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Luong, T.; Sutskever, I.; Le, Q.; Vinyals, O.; and Zaremba, W. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*, 11–19.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Nallapati, R.; Zhou, B.; glar Gulçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of CoNLL*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, 311–318.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. *ICML (3)* 28:1310–1318.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*, 2383–2392. Austin, Texas: Association for Computational Linguistics.

Rush, A. M.; Chopra, S.; and Weston, J. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP*, 379–389. Lisbon, Portugal: Association for Computational Linguistics.

Srivastava, N.; Hinton, G. E.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

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

Tang, Y.; Meng, F.; Lu, Z.; Li, H.; and Yu, P. L. 2016. Neural machine translation with external phrase memory. *arXiv preprint arXiv:1606.01792*.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Vinyals, O.; Kaiser, Ł.; Koo, T.; Petrov, S.; Sutskever, I.; and Hinton, G. 2015. Grammar as a foreign language. In *NIPS*, 2773–2781.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 2692–2700.

Wang, S., and Jiang, J. 2016. Machine comprehension using match-lstm and answer pointer. In *Proceedings of ICLR*.

Wang, X.; Tu, Z.; Xiong, D.; and Zhang, M. 2017. Translating phrases in neural machine translation. *arXiv preprint arXiv:1708.01980*.

Zhou, Q.; Yang, N.; Wei, F.; Tan, C.; Bao, H.; and Zhou, M. 2017a. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792*.

Zhou, Q.; Yang, N.; Wei, F.; and Zhou, M. 2017b. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1095–1104. Vancouver, Canada: Association for Computational Linguistics.