Encode, Review, and Decode: Reviewer Module for Caption Generation

Zhilin Yang Ye Yuan Yuexin Wu Ruslan Salakhutdinov William W. Cohen
School of Computer Science
Carnegie Mellon University
{zhiliny,yey1,yuexinw,rsalakhu,wcohen}@cs.cmu.edu

Abstract

We propose a novel module, *the reviewer module*, to improve the encoder-decoder learning framework. The reviewer module is generic, and can be plugged into an existing encoder-decoder model. The reviewer module performs a number of *review* steps with attention mechanism on the encoder hidden states, and outputs a *fact vector* after each review step; the fact vectors are used as the input of the attention mechanism in the decoder. We show that the conventional encoder-decoders are a special case of our framework. Empirically, we show that our framework can improve over state-of-the-art encoder-decoder systems on the tasks of image captioning and source code captioning.

1 Introduction

Encoder-decoder is a framework for learning a transformation from one representation to another. In this framework, an encoder network first encodes the input into a context vector, and then a decoder network decodes the context vector to generate the output. The encoder-decoder framework was recently introduced for sequence-to-sequence learning based on recurrent neural networks (RNNs) with applications to machine translation [3, 15], where the input is a text sequence in one language and the output is a text sequence in the other language. More generally, the encoder-decoder framework is not restricted to RNNs and text; e.g., encoders based on convolutional neural networks (CNNs) are used for image captioning [17]. Since it is often difficult to encode all the necessary information in a single context vector (often regarded as the bottleneck), *attentive encoder-decoder* introduces attention mechanism to the encoder-decoder framework. Attention mechanism breaks down the encoder-decoder bottleneck by conditioning the generative process in the decoder on the encoder hidden states, rather than on one single context vector only. Improvements by the attention mechanism are shown on various tasks, including machine translation [1], image captioning [19], and text summarization [12].

However, there remain two important issues to address for the attentive encoder-decoder models. First, the attention mechanism proceeds in a sequential manner and thus lacks global modeling abilities. More specifically, at the generation step t, the decoded token is conditioned on the attention results at the current time step $\tilde{\mathbf{h}}_t$, but has no information about future attention results $\tilde{\mathbf{h}}_{t'}$ with $t' > t$. For example, when there are multiple objects in the image, the caption tokens generated at the beginning focuses on the first one or two objects and is unaware of the other objects, which is potentially suboptimal due to the lack of global modeling abilities. Second, previous works show that discriminative supervision (e.g., predicting word occurrences in the caption) is beneficial for generative models [5], but it is not clear how to integrate discriminative supervision into the encoder-decoder framework in an end-to-end manner.

To address the above questions, we propose a novel module, *the reviewer module*, which can be integrated into existing (attentive) encoder-decoder models. The reviewer module is plugged into the middle of the encoder and the decoder. The reviewer module performs a given number of *review* steps with attention mechanism on the encoder hidden states and outputs a *fact vector* after each step, where the fact vectors are expected to capture the global *facts* in compact vector representation and are usable by the attention mechanism in the decoder. The intuition behind the reviewer module is to review all the information encoded by the encoder and learn fact vectors that are more compact, abstractive, and global representation than the original encoder hidden states. Our novel model with the reviewer module is referred to as *encoder-reviewer-decoder* models in the following sections.

The basic reviewer module does not have step-wise supervision as in the decoder RNNs. However, we can leverage discriminative supervision signals by imposing loss on the fact vectors to guide the review process. More specifically, we apply a linear transformation upon the fact vectors, and then employ a multi-label margin loss following a max pooling layer, to impose discriminative supervision, such as predicting word occurrences in the caption. We also study the effects of weight tying between the reviewer units.

We show that our model can be reduced to a conventional attentive encoder-decoder in a special case, which indicates that our model is strictly more expressive than the attentive encoder-decoders. We experiment with two different tasks, image captioning and source code captioning, using CNNs and RNNs as the encoders respectively. Our results show that the reviewer module can consistently improve the performance over attentive encoder-decoders on both datasets and obtain state-of-the-art performance.

2 Related Work

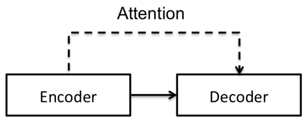
The encoder-decoder framework in the context of sequence-to-sequence learning was recently introduced for learning transformation between text sequences [3, 15], where RNNs are used for both encoding and decoding. Encoder-decoders, in general, can refer to models that learn representation transformation using two network components, an encoder and a decoder. Besides RNNs, convolutional encoders are developed to address multi-modal tasks such as image captioning [17]. Attention mechanism was later introduced to the encoder-decoder framework for machine translation, with an explanation of explicit token-level alignment between input and output sequences [1]. Different from vanilla encoder-decoders, attentive encoder-decoders condition the decoder on the encoder hidden states. At each generation step, the decoder pays attention to a specific part (token) in the encoder, and generates the next token based on both the current hidden state in the decoder and the attended hidden states in the encoder. Attention mechanism demonstrates considerable success in other applcations as well, including image captioning [19] and text summarization [12].

Our work is also related to memory networks [18, 14]. Memory networks take a question embedding as input, and perform multiple computational steps with attention on the *memory*, which is usually formed by the embeddings of a group of sentences. Dynamic memory networks extend memory networks to model sequential memories [8]. Variants of memory networks are mainly developed in the context of question answering; the reviewer module, on the other hand, is a generic module that can be integrated into existing encoder-decoder models. Moreover, the reviewer module learns fact vectors using multiple review steps, while (embedded) facts are provided as input to the memory networks. The reviewer module outputs a sequence of fact vectors, while memory networks only use the last hidden state to generate the answer.

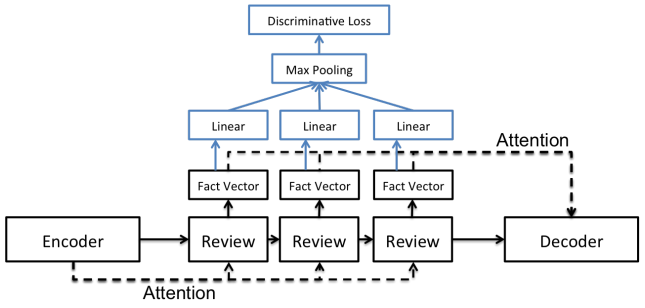
3 Model

Given the input representation $\mathbf{x}$ and the output representation $\mathbf{y}$, the goal is to learn a function mapping from $\mathbf{x}$ to $\mathbf{y}$. For example, image captioning aims to learn a mapping from an image $\mathbf{x}$ to a caption $\mathbf{y}$. For notation simplicity, we use $\mathbf{x}$ and $\mathbf{y}$ to denote both a tensor and a sequence of tensors. For example, $\mathbf{x}$ can be a 3d-tensor that represents an image with RGB channels in image captioning, or can be a sequence of 1d-tensors (i.e., vectors) $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_{T_x})$ in machine translation, where $\mathbf{x}_t$ denotes the one-of-K embedding of the t-th word in the input sequence of length T_x.

In contrast to conventional (attentive) encoder-decoder models, our model consists of three components, encoder, reviewer, and decoder. The comparison of architectures is shown in Figure 1. Now we describe the three components in detail.

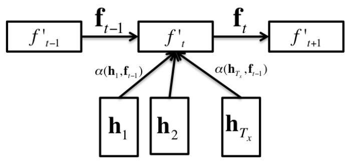


(a) Attentive Encoder-Decoder Model.

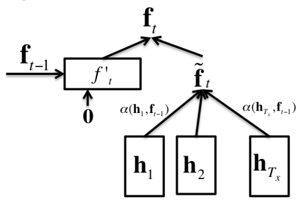


(b) Our Encoder-Reviewer-Decoder Model. Blue components denote optional discriminative supervision. The number of review steps is set to 3 in this example.

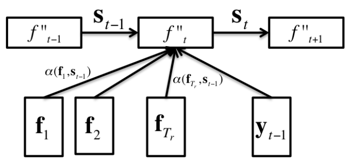
Figure 1: Model Architectures.



(a) Attentive Input Reviewer.



(b) Attentive Output Reviewer.



(c) Decoder.

Figure 2: Illustrations of modules in the encoder-reviewer-decoder model. f' and f'' denote LSTM units.

3.1 Encoder

The encoder encodes the input $\mathbf{x}$ into a context vector $\mathbf{c}$ and a set of hidden states $H = \{\mathbf{h}_t\}_t$. We discuss two types of encoders, RNN encoders and CNN encoders.

RNN Encoder: Let $T_x = |H|$ be the length of the input sequence. An RNN encoder processes the input sequence $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_{T_x})$ sequentially. At time step t, the RNN encoder updates the hidden state by

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}).$$

In this work, we implement f using an LSTM unit. The context vector is defined as the final hidden state $\mathbf{c} = \mathbf{h}_{T_x}$. The cell state and hidden state $\mathbf{h}_0$ of the first LSTM unit are initialized as zero.

CNN Encoder: We take a widely-used CNN architecture—VGGNet [13]—as an example to describe how we use CNNs as encoders. Given a VGGNet, we use the output of the last fully connected layer `fc7` as the context vector $\mathbf{c} = \text{fc7}(\mathbf{x})$, and use $14 \times 14 = 196$ columns of 512d convolutional output `conv5` as hidden states $H = \text{conv5}(\mathbf{x})$. In this case $T_x = |H| = 196$.

3.2 Reviewer

Let T_r be a hyperparameter that specifies the number of review steps. The intuition behind the reviewer module is to review all the information encoded by the encoder and learn fact vectors that are more compact, abstractive, and global representation than the original encoder hidden states. The reviewer performs T_r review steps on the encoder hidden states H and outputs a fact vector $\mathbf{f}_t$ after each step. More specifically,

$$\mathbf{f}_t = g_t(H, \mathbf{f}_{t-1}),$$

where g_t is a modified LSTM unit with attention mechanism at review step t. We study two variants of g_t, attentive input reviewer and attentive output reviewer.

3.2.1 Attentive Input Reviewer

At each review step t, the attentive input reviewer first applies an attention mechanism on H and use the attention result as the input to an LSTM unit (Cf. Figure 2a). Let $\tilde{\mathbf{f}}_t = \text{att}(H, \mathbf{f}_{t-1})$ be the attention result at step t. The attentive input reviewer is formulated as

$$\tilde{\mathbf{f}}_t = \text{att}(H, \mathbf{f}_{t-1}) = \sum_{i=1}^{|H|} \frac{\alpha(\mathbf{h}_i, \mathbf{f}_{t-1})}{\sum_{i'=1}^{|H|} \alpha(\mathbf{h}_{i'}, \mathbf{f}_{t-1})} \mathbf{h}_i, \quad g_t(H, \mathbf{f}_{t-1}) = f'_t(\tilde{\mathbf{f}}_t, \mathbf{f}_{t-1}), \tag{1}$$

where $\alpha(\mathbf{h}_i, \mathbf{f}_{t-1})$ is a function that determines the weight for the i-th hidden state. $\alpha(\mathbf{x}_1, \mathbf{x}_2)$ can be implemented as a dot product between $\mathbf{x}_1$ and $\mathbf{x}_2$ or a multi-layer perceptron (MLP) that takes the concatenation of $\mathbf{x}_1$ and $\mathbf{x}_2$ as input [9]. f'_t is an LSTM unit at step t.

3.2.2 Attentive Output Reviewer

In contrast to the attentive input reviewer, the attentive output reviewer uses a zero vector as input to the LSTM unit, and the fact vector is computed as the weighted sum of the attention results and the output of the LSTM unit (Cf. Figure 2b). More specifically, the attentive output reviewer is formulated as

$$\tilde{\mathbf{f}}_t = \text{att}(H, \mathbf{f}_{t-1}), \quad g_t(H, \mathbf{f}_{t-1}) = f'_t(\mathbf{0}, \mathbf{f}_{t-1}) + \mathbf{W}\tilde{\mathbf{f}}_t,$$

where the attention mechanism `att` follows the definition in Eq. (1), $\mathbf{0}$ denotes a zero vector, $\mathbf{W}$ is a model parameter matrix, and f'_t is an LSTM unit at step t. We note that performing attention on top of an RNN unit is commonly used in sequence-to-sequence learning [1, 9, 12]. We apply a linear transformation with a matrix $\mathbf{W}$ since the dimensions of $f'_t(\cdot, \cdot)$ and $\tilde{\mathbf{f}}_t$ can be different.

3.2.3 Weight Tying

We study two variants of weight tying for the reviewer units. Let $\mathbf{w}_t$ denote the parameters for the unit f'_t. The first variant follows the common setting in RNNs, where weights are shared among all the units; i.e., $\mathbf{w}_1 = \cdots = \mathbf{w}_{T_r}$. We also observe that the reviewer unit does not have sequential input, so we experiment with the second variant where weights are untied; i.e. $\mathbf{w}_i \neq \mathbf{w}_j, \forall i \neq j$.

The cell state and hidden state of the first unit f'_1 are initialized as the context vector $\mathbf{c}$. The cell states and hidden states are passed through all the reviewer units in both cases of weight tying.

3.3 Decoder

Let $F = \{\mathbf{f}_t\}_t$ be the set of fact vectors output by the reviewer. The decoder is formulated as an LSTM network with attention on the fact vectors F (Cf. Figure 2c). Let $\mathbf{s}_t$ be the hidden state of the t-th LSTM unit in the decoder. The decoder is formulated as follows:

$$\tilde{\mathbf{s}}_t = \text{att}(F, \mathbf{s}_{t-1}), \quad \mathbf{s}_t = f''([\tilde{\mathbf{s}}_t; \mathbf{y}_{t-1}], \mathbf{s}_{t-1}), \quad y_t = \arg\max_y \text{softmax}_y(\mathbf{s}_t), \tag{2}$$

where $[\cdot;\cdot]$ denotes the concatenation of two vectors, f'' denotes the decoder LSTM, softmax$_y$ is the probability of word y given by a softmax layer, y_t is the t-th decoded token, and $\mathbf{y}_t$ is the word embedding of y_t. The attention mechanism `att` follows the definition in Eq. (1). The initial cell state and hidden state $\mathbf{s}_0$ of the decoder LSTM are both set to the *review vector* $\mathbf{r} = \mathbf{W}'[\mathbf{f}_{T_r}; \mathbf{c}]$, where $\mathbf{W}'$ is a model parameter matrix.

3.4 Discriminative Supervision

In conventional encoder-decoders, supervision is provided in a generative manner; i.e., the model aims to maximize the conditional probability of generating the sequential output $p(\mathbf{y}|\mathbf{x})$. However, discriminative supervision has been shown to be useful in [5], where the model is guided to predict discriminative objectives, such as the word occurrences in the output $\mathbf{y}$.

We argue that the reviewer module provides a natural way of incorporating discriminative supervision into the model. Here we take word occurrence prediction for example to describe how to incorporate discriminative supervision. As shown in the blue components in Figure 1b, we first apply a linear layer on top of the fact vector to compute a score for each word at each review step. We then apply a max-pooling layer over all the review units to extract the most salient signal for each word, and add a multi-label margin loss as discriminative supervision. Let s_i be the score of word i after the max pooling layer, and W be the set of all words that occur in $\mathbf{y}$. The discriminative loss can be written as

$$\mathcal{L}_d = \frac{1}{Z} \sum_{j \in W} \sum_{i \neq j} \max(0, 1 - (s_j - s_i)), \tag{3}$$

where Z is a normalizer that counts all the valid i, j pairs. We note that when the discriminative supervision is derived from the given data (i.e., predicting word occurrences in captions), we are not using extra information.

3.5 Training

The training loss for a single training instance $(\mathbf{x}, \mathbf{y})$ is defined as a weighted sum of the negative conditional log likelihood and the discriminative loss. Let T_y be the length of the output sequence $\mathbf{y}$. The loss can be written as

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = \frac{1}{T_y} \sum_{t=1}^{T_y} - \log \text{softmax}_{y_t}(\mathbf{s}_t) + \lambda \mathcal{L}_d,$$

where the definition of softmax_y and $\mathbf{s}_t$ follows Eq. (2), and the formulation of $\mathcal{L}_d$ follows Eq. (3). λ is a constant weighting factor. Since $\mathcal{L}_d$ is optional, we set $\mathcal{L}_d = 0$ when unused. We adopt adaptive stochastic gradient descent (AdaGrad) [4] to train the model in an end-to-end manner. The loss of a training batch is averaged over all instances in the batch.

3.6 Connection to Encoder-Decoders

We now show that our model can be reduced to the conventional (attentive) encoder-decoders in a special case. In attentive encoder-decoders, the decoder takes the context vector $\mathbf{c}$ and the set of encoder hidden states $H = \{\mathbf{h}_t\}_t$ as input, while in our encoder-reviewer-decoder model, the input of the decoder is instead the review vector $\mathbf{r}$ and the set of fact vectors $F = \{\mathbf{f}_t\}_t$. To show that our model can be reduced to attentive encoder-decoders, we only need to construct a case where $H = F$ and $\mathbf{c} = \mathbf{r}$.

Since $\mathbf{r} = \mathbf{W}'[\mathbf{f}_{T_r}; \mathbf{c}]$, it can be reduced to $\mathbf{r} = \mathbf{c}$ with a specific setting of $\mathbf{W}'$. We further set $T_r = T_x$, and define each reviewer unit as an identity mapping $g_t(H, \mathbf{f}_{t-1}) = \mathbf{h}_t$, which can fit into the definition of both the attentive input reviewer and the attentive output reviewer with untied weights. With the above setting, we have $\mathbf{h}_t = \mathbf{f}_t, \forall t = 1, \cdots, T_x$; i.e., $H = F$. Thus our model can be reduced to attentive encoder-decoders in a special case. Similarly we can show that our model can be reduced to vanilla encoder-decoders (without attention) by constructing a case where $\mathbf{r} = \mathbf{c}$ and $\mathbf{f}_t = \mathbf{0}$. Therefore, our model is more expressive than (attentive) encoder-decoders.

Though we set $T_r = T_x$ in the above construction, in practice, we set the number of review steps T_r to be much smaller compared to T_x, since we find that the reviewer module can learn more compact and effective representation.

4 Experiments

We experiment with two datasets of different tasks, image captioning and source code captioning. Since these two tasks are essentially different, we can use them to test the robustness and generalizability of our model.

4.1 Image Captioning

4.1.1 Data and Settings

We evaluate our model on the MSCOCO benchmark dataset [2] for image captioning. The dataset contains 123,000 images with at least 5 captions for each image. We use the same data split as in [7, 19, 20], where we reserve 5,000 images for development and test respectively and use the rest for training. The models are evaluated using the official MSCOCO evaluation scripts. We report three widely used automatic evaluation metrics, BLEU-4, METEOR, and CIDEr.

We remove all the non-alphabetic characters in the captions, transform all letters to lowercase, and tokenize the captions using white space. We replace all words occurring less than 5 times with an unknown token `<UNK>` and obtain a vocabulary of 9,520 words. We truncate all the captions longer than 30 tokens.

Table 1: Comparison of model variants on MSCOCO dataset. Results are obtained with a single model using VGGNet. Scores in the brackets are without beam search. We use RNN-like tied weights for the reviewer module unless otherwise indicated. "Disc Sup" means discriminative supervision.

Model	BLEU-4	METEOR	CIDEr
Attentive Encoder-Decoder	27.8 (25.5)	22.9 (22.3)	84.0 (79.3)
Encoder-Reviewer-Decoder (ERD)	28.2 (25.9)	23.3 (22.7)	85.2 (81.6)
ERD + Disc Sup	28.7 (26.4)	**23.8** (**23.2**)	87.9 (83.3)
ERD + Disc Sup + Untied Weights	**29.0** (**26.8**)	23.7 (**23.2**)	**88.6** (**85.2**)

Table 2: Comparison with state-of-the-art results on MSCOCO dataset. v means VGGNet, g means GoogLeNet, † indicates using task-specific hand-engineered features/attributes, ‡ indicates ensemble models (single-model results are not reported in these works), $*$ indicates using unpublished data splits. Higher is better in all columns.

Model	BLEU-4	METEOR	CIDEr
BRNN [7]^v	23.0	19.5	66.0
Soft Attention [19]^v	24.3	23.9	—
Hard Attention [19]^v	25.0	23.0	—
MS Research [5]^{v†*}	25.7	23.6	—
Google NIC [17]^{g‡*}	27.7	23.7	85.5
Semantic Attention [20]^{g†‡}	30.4	24.3	—
ERD + VGGNet (this paper)^v	29.0	23.7	88.6
ERD + GoogLeNet (this paper)^g	29.8	24.0	89.5

We set the number of review steps $T_r = 8$, the weighting factor $\lambda = 10.0$, the dimension of word embeddings to be 100, the learning rate to be $1\mathrm{e}{-2}$, and the dimension of LSTM hidden states to be $1,024$. These hyperparameters are tuned on the development set. We also use early stopping strategies to prevent overfitting. More specifically, we stop the training procedure when the BLEU-4 score on the development set reaches the maximum. We use an MLP with one hidden layer of size 512 to define the function $\alpha(\cdot, \cdot)$ in the attention mechanism, and use an attentive input reviewer in our experiments to be consistent with visual attention models [19]. We use beam search with beam size 3 for decoding. We guide the model to predict the words occurring in the caption through the discriminative supervision $\mathcal{L}_d$ without introducing extra information. We fix the parameters of the CNN encoders during training.

4.1.2 Results

We compare our model with encoder-decoders to study the effectiveness of the reviewer module. We also compare different variants of our model to evaluate the effects of different weight tying strategies and discriminative supervision. Results are reported in Table 1. All the results in Table 1 are obtained using VGGNet [13] as encoders as described in Section 3.1.

From Table 1, we can see that the reviewer module can improve the performance over conventional attentive encoder-decoders consistently on all the three metrics We also observe that adding discriminative supervision can boost the model performance, which indicates that the reviewer module provides an effective way of incorporating discriminative supervision in an end-to-end manner. Untying the weights between the reviewer units can further improve the performance. Our conjecture is that the models with untied weights are more expressive than shared-weight models since each unit can have its own parametric function to compute the fact vector.

We also compare our model with state-of-the-art results on the MSCOCO dataset. Since VGGNet and GoogLeNet [16] are both used as the image encoders in previous work, we additionally report the performance of our model using GoogLeNet as the encoder for fair comparison. The results are reported in Table 2. Different from some of the other approaches that use model ensembles, the results of our encoder-reviewer-decoder models are obtained with a single model. Table 2 shows that our models outperform all of the other approaches except for Semantic Attention [20]. Though Semantic Attention performs slightly better than our models, we note that they use model ensembles, while the results of a single model are not published. Since model ensembles can usually boost the

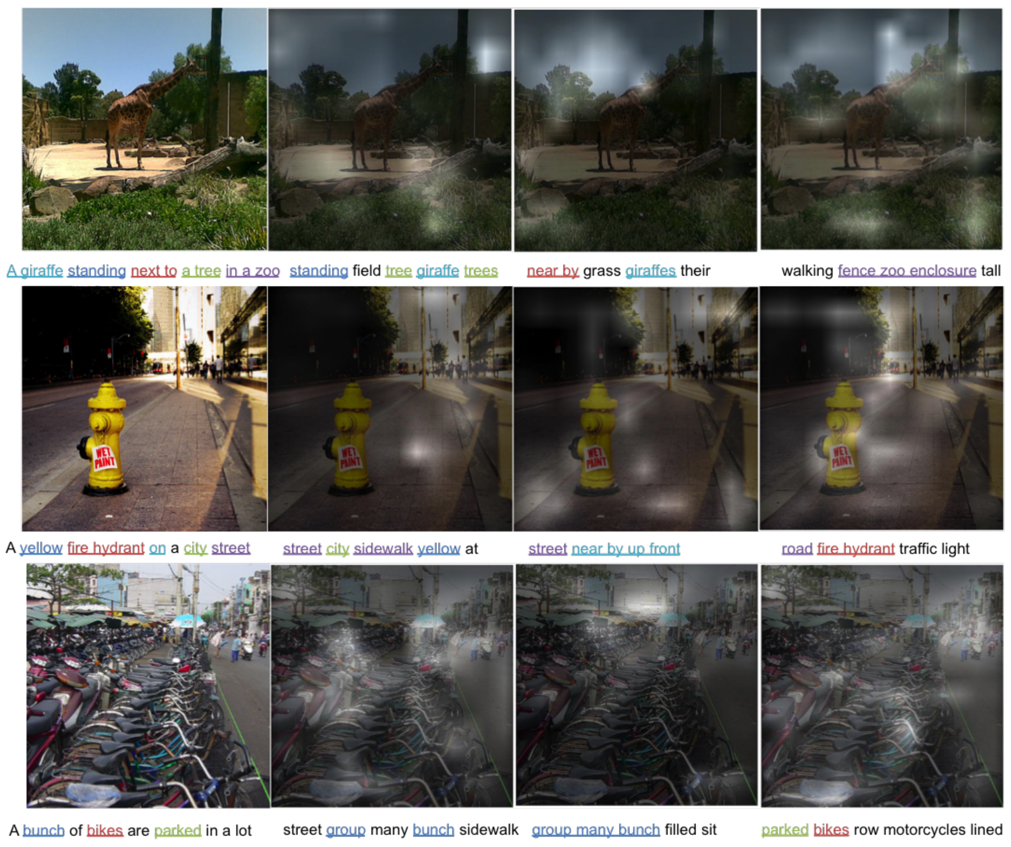

A giraffe standing next to a tree in a zoo
standing field tree giraffe trees
near by grass giraffes their
walking fence zoo enclosure tall
A yellow fire hydrant on a city street
street city sidewalk yellow at
street near by up front
road fire hydrant traffic light
A bunch of bikes are parked in a lot
street group many bunch sidewalk
group many bunch filled sit
parked bikes row motorcycles lined

Figure 3: Visualizing the reviewer module. Each row corresponds to a test image: the first is the original image with the caption output by our model, and the following three images are the visualized attention weights of the first three reviewer units. We also list the top-5 words with highest scores for each unit. Colors indicate semantically similar words.

performance dramatically (by a few BLEU-4 points [17]), our results are competitive and possibly better. Moreover, we note that Semantic Attention makes use of task-specific attributes, while our model does not.

4.1.3 Case Study and Visualization

To better understand the reviewer module, we visualize the attention weights α in the reviewer module in Figure 3. The visualization is based on the encoder-reviewer-decoder model with untied weights and discriminative supervision. We also list the top-5 words with highest scores (computed based on the fact vectors) at each reviewer unit.

We find that the top words with highest scores can uncover the reasoning procedure underlying the reviewer module. For example, in the first image (a giraffe in a zoo), the first reviewer focuses on the motion of the giraffe and the tree around, the second reviewer analyzes the relative position between the giraffe and the tree, and the third reviewer looks at the big picture and infers that the scene is in a zoo based on recognizing the fences and enclosures. All the above information is stored in the fact vectors and decoded as natural language by the decoder.

Different from attentive encoder-decoders [19] that attend to a single object at a time during generation, it can be clearly seen from Figure 3 that the reviewer module captures more global signals, usually combining multiple objects into one fact, including objects not finally shown in the caption (e.g., “traffic light” and “motorcycles”). The facts are sometimes abstractive, such as motion (“standing”), relative position (“near”, “by”, “up”), quantity (“bunch”, “group”), and scene (“city”, “zoo”). Also, the order of review is not restricted by the order in natural language.

Table 3: Comparison of model variants on HabeasCorpus code captioning dataset. "Bidir" indicates using bidirectional RNN encoders, "LLH" refers to log-likelihood, "CS-k" refers to top-k character savings.

Model	LLH	CS-1	CS-2	CS-3	CS-4	CS-5
Language Model	-5.34	0.2382	0.2846	0.3091	0.3239	0.3356
Encoder-Decoder	-5.25	0.2515	0.2911	0.3126	0.3280	0.3407
Encoder-Decoder (Bidir)	-5.14	0.2710	0.3125	0.3337	0.3476	0.3592
Attentive Encoder-Decoder (Bidir)	-5.05	0.2835	0.3246	0.3469	0.3620	0.3747
Encoder-Reviewer-Decoder (Bidir)	**-4.99**	**0.2947**	**0.3351**	**0.3591**	**0.3751**	**0.3875**

4.2 Source Code Captioning

4.2.1 Data and Settings

The task of source code captioning is to predict the code comment given the source code, which can be framed under the problem of sequence-to-sequence learning. We experiment with a benchmark dataset for source code captioning, HabeasCorpus [11]. HabeasCorpus collects nine popular open-source Java code repositories, such as Apache Ant and Lucene. The dataset contains $6,734$ Java source code files with $7,903,872$ source code tokens and $251,565$ comment word tokens. We randomly sample 10% of the files as the test set, and use the rest for training. We randomly sample a different extra validation split for tuning hyperparameters.

Our evaluation follows previous works on source code language modeling [10] and captioning [11]. We report the log-likelihood of generating the actual code captions based on the learned models. We also evaluate the approaches from the perspective of code comment completion, where we compute the percentage of characters that can be saved by applying the models to predict the next token. More specifically, we use a metric of *top-k character savings* [11] (CS-k). Let n be the minimum number of prefix characters needed to be filtered such that the actual word ranks among the top-k based on the given model. Let L be the length of the actual word. The number of saved characters is then $L - n$. We compute the average percentage of saved characters per comment to obtain the metric CS-k.

We follow the tokenization used in [11], where we transform camel case identifiers into multiple separate words (e.g., "binaryClassifierEnsemble" to "binary classifier ensemble"), and remove all non-alphabetic characters. We truncate code sequences and comment sequences longer than 300 tokens. We use an RNN encoder and an attentive output reviewer with tied weights. We set the number of review steps $T_r = 8$, the dimension of word embeddings to be 50, and the dimension of the LSTM hidden states to be 256.

4.2.2 Results

We report the log-likelihood and top-k character savings of different model variants in Table 3. The baseline model "Language Model" is an LSTM decoder whose output is not sensitive to the input code sequence. Our preliminary experiment shows that the LSTM decoder significantly outperforms the N-gram models used in [11] (+3% in CS-2), so we use the LSTM decoder as a baseline for comparison. We also compare with different variants of encoder-decoders, including incorporating bidirectional RNN encoders and attention mechanism. It can be seen from Table 3 that both bidirectional encoders and attention mechanism can improve over vanilla encoder-decoders. Encoder-reviewer-decoder model can improve attentive encoder-decoders consistently in all the metrics, which indicates that the reviewer module is effective at learning useful representation.

5 Conclusion

We present a novel module, the reviewer module, to improve the encoder-decoder learning framework. The reviewer module perform multiple review steps with attention on the encoder hidden states, and computes a set of fact vectors that summarize the global information in the input. We empirically show consistent improvement over conventional encoder-decoders on the tasks of image captioning and source code captioning. In the future, it will be interesting to apply our model to more tasks that can be modeled under the encoder-decoder framework.

Acknowledgements

This work was funded by the NSF under grants CCF-1414030 and IIS-1250956, and by Google.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[2] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.

[3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *ACL*, 2014.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[5] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *CVPR*, pages 1473–1482, 2015.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.

[8] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*, 2016.

[9] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *ACL*, 2015.

[10] Chris J Maddison and Daniel Tarlow. Structured generative models of natural source code. In *ICML*, 2014.

[11] Dana Movshovitz-Attias and William W Cohen. Natural language models for predicting programming comments. In *ACL*, 2013.

[12] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015.

[13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[14] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *NIPS*, pages 2431–2439, 2015.

[15] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[16] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[17] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *CVPR*, pages 3156–3164, 2015.

[18] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

[19] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

[20] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. Image captioning with semantic attention. In *CVPR*, 2016.