Self-Critial Sequence Training for Image Captioning: A Rewards Metrics Study

Zishun Yu
Complex System Informatics Laboratory
Science and Enigineering Labs West, 4227
Chicago, Illinois 60607

Abstract

In this project, we inspired by paper *self-critical sequence training for image captioning* (SCST) which is using REINFORCE algorithm with baseline to train a sequence generation model to do image captioning task. We further investigated several evaluation metrics, which is not be used in the SCST, to be directly optimized. The metrics we studied are SPICE, LTEIC and SPIDEr. We found that directly optimizing SPICE will not promise us a better results overall mainly because the testing metrics we report on are mostly syntactic-based but SPICE is semantic-based. Optimizing SPIDEr shows a slightly better performance than optimizing CIDEr, which has a decent improvement on SPICE with relatively small trade-off on BLEU and CIDEr.

1 Introduction

Image caption has been a widely studied NLP task. However, there are still two challenges when using supervised learning. The first challenge is exposure bias, mentioned by (Ranzato et al., 2015). In detail, some text generation models are trained to predict next word given the previous ground truth word, but at test time, the prediction of next word will be made given a generated word instead of the previous ground truth word. As a result, the errors will be accumulated and we refer this as exposure bias since those models are never exposed to its predictions during training. Another is that most generative models are evaluated by non-differentiable NLP metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), CIDEr (Vedantam et al., 2015) and SPICE (Anderson et al., 2016), however these models are trained by cross entropy which means the training is not directly optimizing the evaluation metrics.

For addressing these issues, (Ranzato et al., 2015) firstly applied policy gradient to do image captioning and (Bahdanau et al., 2016) employeed actor-critic (Konda and Tsitsiklis, 2000) algorithm. Afterwards, (Rennie et al., 2016) and (Liu et al., 2016) both use policy gradient methods to build sequence generation model. Among these works, *self-critical sequence training for image captioning* (Rennie et al., 2016) achieve the best results and occupied the leaderboard best result for a very time (still stay 5th place in MSCOCO leaderboard).

Hence, we decide to build our project based on the paper *self-critical sequence training for image captioning*. And for reinforcement learning training, the reward fucntion design is very crusial and in self-critical paper, they used CIDEr as the reward metric. Therefore, our motiviation is to examinate varies of metrics not tested in this paper and hopefully we can make some improvement on the results.

2 Methods

In this section, we will describe the recurrent model we used, what is reinforcement learning and what is self-critical sequence training in detail. And in this project, we kept the same models structure and REINFORCE algorithm which the self-critical sequence training paper used. Therefore, we could better focus on the reward metrics study.

2.1 Captioning model

In this project, we keep the same recurrent model with (Rennie et al., 2016) and similarly to (Vinyals et al., 2014; Karpathy and Li, 2014).

This model first extract the image F feature using a ResNet101, proposed by (He et al., 2015), and then embed it through a linear projection.Therefore we obtained the frist input x_1. Words are one-hot encoded and will be embeded with a linear embedding E (which has same output dimension as W_I). And a sequence is started with a BOS token and ended with a EOS token. Under the model, words are generated and then fed back into the LSTM, with the image treated as the first word $W_I CNN(F)$. The following updates for the hidden units and cells of an LSTM define the model (Hochreiter and Schmidhuber, 1997).

$$
\begin{aligned}
x_1 &= W_I CNN(F) \\
x_t &= E 1_{w_{t-1}} \text{ for } t > 1 \\
i_t &= \sigma\left(W_{ix} x_t + W_{ih} h_{t-1} + b_i\right) \quad \text{(Input Gate)} \\
f_t &= \sigma\left(W_{fx} x_t + W_{fh} h_{t-1} + b_f\right) \quad \text{(Forget Gate)} \\
o_t &= \sigma\left(W_{ox} x_t + W_{oh} h_{t-1} + b_o\right) \quad \text{(Output Gate)} \\
c_t &= i_t \odot \phi(W_{zx}^{\otimes} x_t + W_{zh}^{\otimes} h_{t-1} + b_z^{\otimes}) + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t) \\
s_t &= W_s h_t,
\end{aligned}
$$

where ϕ is a maxout non-linearity with 2 units ($\otimes$ denotes the units) and σ is the sigmoid function. We initialize h_0 and c_0 to zero. The LSTM outputs a distribution over the next word w_t using the softmax function:

$$w_t \sim \text{softmax}\left(\mathrm{s_t}\right) \tag{1}$$

In our architecture, the hidden states and word and image embeddings have dimension 512. Let θ denote the parameters of the model. Traditionally the parameters θ are learned by maximizing the likelihood of the observed sequence. Specifically, given a target ground truth sequence $(w_1^*, \ldots, w_T^*)$, the objective is to minimize the cross entropy loss (XE):

$$L(\theta) = -\sum_{t=1}^{T} \log(p_\theta(w_t^* | w_1^*, \ldots w_{t-1}^*)), \tag{2}$$

where $p_\theta(w_t | w_1, \ldots w_{t-1})$ is given by the parametric model in Equation (1).

2.2 Reinforcement learning

Formulate image captioning as reinforcement learning problem: As we previously discussed, supervised model are usaslly trained by cross entropy and will encounter the exposure bias issue. Although scheduled sampleing (Bengio et al., 2015a) has been proposed to address the exposure bias and has been proved very effective. The different training metric, cross entropy, and evaluation metrics, i.e. BLEU, CIDEr, SPICE etc., will still be an issue preventing image captioning improving.

In order to directly optimize evaluation mertics and address the exposure bias, we could formulate our model as (Ranzato et al., 2015) did in a reinforcement learning way. In detail, the LSTM is fomulated as *agent* and the input image, words are foumulated as *environment*. *Agent* will interact with *environment* and choose its action based on its *policy* p_θ, where θ is the network weights. Once the *Agent* choose a EOS token, it will receive a reward, for example a CIDEr score, based on the sentence the *Agent* generated. We denote the reward as r. And our goal is to maximize the expected reward $R(\theta)$, for convience of later fomulating, we use negative expected reward, $L(\theta)$, then our goal is to minimize the loss:

$$L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[r(w^s)\right] \tag{3}$$

where $w^s = (w_1^s, \ldots w_T^s)$ and w_t^s is the word sampled from the model at the time step t. And $L(\theta)$ can be estimated by the negative reward of a sampled sequence:

$$L(\theta) \approx -r(w^s), \; w^s \sim p_\theta \tag{4}$$

REINFORCE algorithm: We use the REINFORCE(Williams, 1992) algorithm (see also (Sutton and Barto, 2018)) to directly optimize the evaluation mertics. In general, REINFORCE is a policy gradient method which iteratively updates agent's parameters by computing policy gradient. Before discussing the algorithm in detail, we will firstly introduce the Policy Gradient Theorem (Sutton et al., 2000).

Theorem 1
for any differentiable policy p_θ, any of policy objective functions $J = J_1, J_{avR}, or (1/1-\gamma) J_{avR}$ the policy gradient is

$$\nabla_\theta J(\theta) = \mathbb{E}_{p_\theta}\left[\nabla_\theta \log p_\theta(s,a) Q^{p_\theta}(s,a)\right]$$

where s is the state, a is the action and $Q^{p_\theta}(s,a)$ is the long-term value of an ation in a given state instead of instantaneous reward r.

In our application, the policy objective function $J(\theta)$ is the loss function we previously fomulated in Equation (4). Therefore, the gradient $\nabla_\theta L(\theta)$ of our task can be computed as below:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[\nabla_\theta \log p_\theta(w^s) r(w^s)\right] \tag{5}$$

where $w^s = (w_1^s, \dots w_T^s)$ and w_t^s is the word sampled from the model at the time step t.
In practice, the expected gradient $\nabla_\theta L(\theta)$ can be estimated by using sampled sequence $w^s = (w_1^s, \dots w_T^s)$

$$\nabla_\theta L(\theta) \approx -r(w^s)\nabla_\theta \log p_\theta(w^s) \tag{6}$$

Therefore, we are able to update our generative model by the following algorithm (we update the policy sequencely in this algorithm, normally it should be updated batchly, it will be included in SCST section):

Algorithm 1 REINFORCE algorithm

initialize θ arbitrarily
for each image input **do**
  embed image as the first word w_1
  timestep $t = 1$
  while EOS token not generated **do**
    timestep $t = t + 1$
    next word $w_t = p_\theta(w_1, \dots, w_{t-1})$
  end while
  calculate reward r for sampled sequence w^s
  $\theta \leftarrow \theta - \alpha r(w^s)\nabla_\theta \log p_\theta(w^s)$
end for

REINFORCE algorithm with baseline: REINFORCE algorithm (Monte-Carlo policy gradient) still has high variance. There are varies way to reduce the variance. Actor-critic (Konda and Tsitsiklis, 2000) is a widely used approach to reduce variance. There are some image captioning works (Bahdanau et al., 2016) using actor-critic. In our project, we use REINFORCE with baseline to address the variance issue. The policy gradient $\nabla_\theta L(\theta)$ given by (5) can be generalized as following:

$$\nabla_\theta L(\theta) = -\mathbb{E}_{w^s \sim p_\theta}\left[(r(w^s) - b)\nabla_\theta \log p_\theta(w^s)\right] \tag{7}$$

where b is a baseline, then $\nabla_\theta L(\theta)$ is no longer only associated with the sampled sequences but also the baseline b.
The goal of adding a baseline is to reduce the variance but at the same time, the baseline should not change the gradient $\nabla_\theta L(\theta)$. Indeed, as long as the baseline does not depend on the "actions" w^s, it will not change the expected gradient. And the prove (Sutton et al., 1998) is shown following:

$$\begin{aligned}\mathbb{E}_{w^s \sim p_\theta}\left[b\nabla_\theta \log p_\theta(w^s)\right] &= b\sum_{w_s} \nabla_\theta p_\theta(w^s) \\ &= b\nabla_\theta \sum_{w_s} p_\theta(w^s) \\ &= b\nabla_\theta 1 = 0 \end{aligned} \tag{8}$$

Therefore, the estimated gradient can be modified as:

$$\nabla_\theta L(\theta) \approx -(r(w^s) - b)\nabla_\theta \log p_\theta(w^s) \tag{9}$$

Final Gradient Expression: From chain rule, we have:

$$\nabla_\theta L(\theta) = \sum_{t=1}^{T} \frac{\partial L(\theta)}{\partial s_t}\frac{\partial s_t}{\partial \theta} \tag{10}$$

where s_t is the softmax socres for the entire vocabulary, i.e. the input to the softmax function.
Given by (Zaremba and Sutskever, 2015), the gradient of $\frac{\partial L(\theta)}{\partial s_t}$ can be estimated by the following:

$$\frac{\partial L(\theta)}{\partial s_t} \approx (r(w^s) - b)(p_\theta(w_t|h_t) - 1_{w_t^s}) \tag{11}$$

2.3 Self-critical sequence training (SCST)

Self-critical sequence training (SCST) is the key approach of this project, proposed by (Rennie et al., 2016). The idea of SCST is using the current model under inference algorithm, i.e. greedy decoding, as a baseline for the REINFORCE algorithm. Therefore, the gradient can be written as following:

$$\frac{\partial L(\theta)}{\partial s_t} \approx (r(w^s) - r(\hat{w}))(p_\theta(w_t|h_t) - 1_{w_t^s}) \tag{12}$$

$$\hat{w}_t = \arg\max_{w_t} p(w_t \mid h_t) \tag{13}$$

where $r(\hat{w})$ is the reward obtained by the current model under inference mode, i.e. picking the word with maximum probablity instead of sampling word. Therefore if the sampled senetence obtain a higher reward $r(w^s)$ compared with greddy decoding reward $r(\hat{w})$. The probabily of w^s will be therefore increased. (See figure 1 for details)

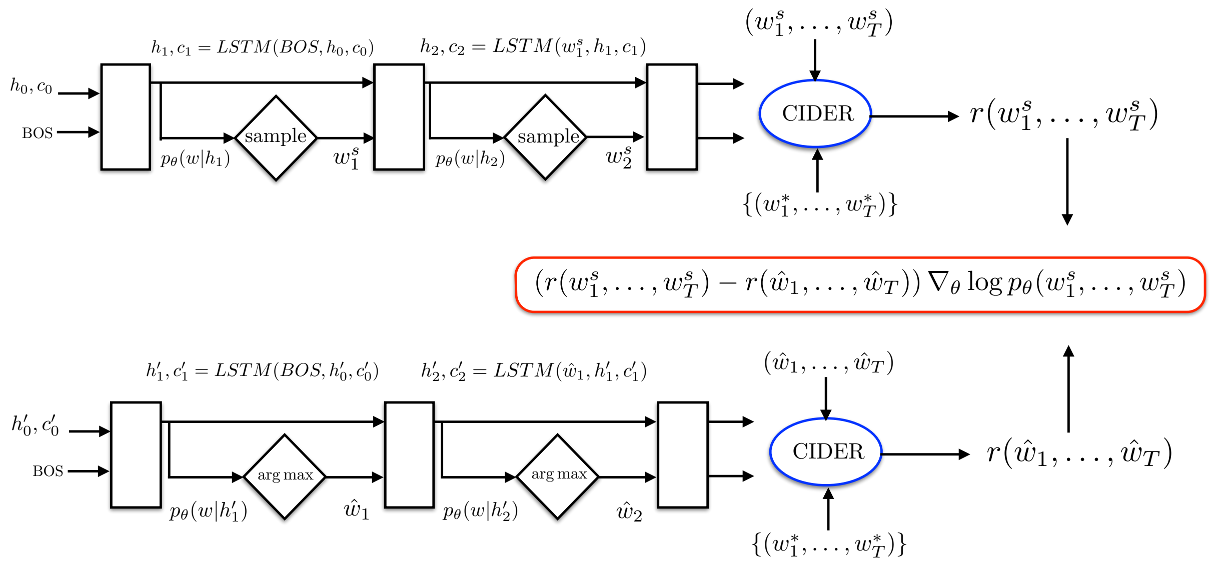


Figure 1: Self-critical sequence training (SCST). The weight put on words of a sampled sentence from the model is deter- mined by the difference between the reward for the sampled sentence and the reward obtained by the estimated sentence under the test-time inference procedure (greedy inference depicted). This harmonizes learning with the inference procedure, and lowers the variance of the gradients, improving the training procedure.

Hence, obviously SCST keeps the adavantages of REINFORCE algorithm, i.e. directly optimizing evalution metrics, avoid exposure bias issue, but also avoid using an estimated baseline as actor-critic alorithm do. It directly improves the inference mode performance which is used at testing. This emphersize the consistency between training and testing similar to approaches *Professor Forcing* (Lamb et al., 2016), *E2E* (Ranzato et al., 2015) and *Data as Demonstrator* (Bengio et al., 2015b).

3 Reward Metrics

In this project, we keep the self-critical (SCST) sequence training algorithm and mainly focus on studying what reward metric would be better compared with CIDEr (Vedantam et al., 2015), what has been originally used in SCST. Since normally reinforcement learning algorithm is highly rely on the reward function design, the intuition is a better reward metric would lead a better model. In the meanwhile, there are some new metrics emerged recently, such as SPICE (Anderson et al., 2016), *learning to evaluate image captioning* (Cui et al., 2018) and *Learning-based Composite Metrics for Improved Caption Evaluation* (Sharif et al., 2018). SPICE is widely used in image captioning researches and *learning to evaluate image captioning* yield their methods have a much better human correlation than metrics like CIDEr and SPICE. Therefore, these two metrics will be used in our project.

3.1 Semantic propositional image caption evaluation

SPICE is a automatic image caption evalution metric which compares the semantic propositional content.

Semantic parsing: Before calculating the SPICE score, we need to parse the captions to scene graphs. Given a set of object classed C, a set of relation R, a set of attribute types A and a caption c, the scene graph is define as below:

$$G(c) = \{O(c), E(c), K(c)\} \tag{14}$$

where $O(c)$ is the objects appeared in the caption c, $E(c) \subseteq O(c) \times R \times O(c)$ is the relation types appeared in c (relation between O(c)) and $K(c) \subseteq O(c) \times A$ is the attributes associated with objects $O(c)$ appeared in c. Besides, there is no pre-defined objects, relation or attributes sets, the sets will be extended once new object/relation/attribute appears.

For example, given a caption c, *A young girl standing on top of a tennis court.*, the scene graph should be as shown below:

Table 1: semantic parsing table

A young girl standing on top of a tennis court.		
$O(c)$	E(c)	K(c)
(girl), (court)	(girl, on-top-of, court)	(girl, young), (girl, standing), (court, tennis)

SPICE score calculation: SPICE score is designed to evaluate the scene graph similarity between candidate and reference captions. Denote the union of all tuples from a scene graph as T:

$$T(G(c)) = O(c) \cup E(c) \cup K(c) \quad (15)$$

where again $G(c)$ is the scene graph, $O(c)$ is the set of objects appears in the caption c, $E(c)$ is the set of relations and $K(c)$ is the set of attributes.
Authors of SPICE define a binary matching operator $\otimes$ which return the matching tuples in two scene graphs. In order to evaluate the candidate and references similarity, authors take both precision P and recall R into consideration. The precision, recall and SPICE scores will be calculated as following:

$$P(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(c))|} \quad (16)$$

$$R(c, S) = \frac{|T(G(c)) \otimes T(G(S))|}{|T(G(S))|} \quad (17)$$

$$SPICE(c, S) = F_1(c, S) = \frac{2 \times P(c, S) \times R(c, S)}{P(c, S) + R(c, S)} \quad (18)$$

where c is the candidate caption and S is the set of reference captions.

Issue encountered: We encountered some issue when directly optimizing SPICE score. There are two main issues, the frist one is the time efficiency and the second is that the SPICE ignores the repeated or duplicate tuples which will result our model generating sentence with repeating words or phrases.

Improve the time efficiency: The default SPICE package is designed for evaluating the entire testing set at the same time. But in our application scenario, we need to calculate the SPICE score for a minibatch every iteration. At beginning, it would take more than 10 seconds to finish a interation, it would take more than one month to finish 40 epochs (model trained on GTX1080Ti) which is not acceptable at all. Firstly, we modified the I/O interface. It would be orginally called as a subprocess every time we need to do a evaluation and therefore will initilized the standford pipeline frequently which is super time consuming. We modified the package as a sever and will be waiting for evaluating request until receiving an ending request. Besides, we noticed the package will frequently generate or re-generate some objects. So we modified the workflow, pre-generate all required objects when the server starts, it also reduces decent evaluating time. After modification, it takes less than 1 seconds to evaluate an iteratioin which is around 10 times faster and is able to finish the entire training, 40 epochs, within 5 days (although it is not super fast but at least acceptable compared to the default package).

Penalty on duplicates tuples: As observed by (Liu et al., 2016), using reinforcement learning to directly optimize SPICE score will result a ungrammatical results, for example:

Table 2: ungrammatically captions

Image	SPICE caption
	a red double decker bus on a city street on a street with a bus on the street with a bus on the street in front of a bus on
	a group of people walking down a street with a man on a street holding a traffic light and a traffic light on a city street with a city street

This is because the default SPICE package ignores and doesn't put penalty on the repeated tuples. In detail:

Table 3: SPICE score on repeated tuples

candidate caption	
a kitchen and dining room and living room.	a kitchen and dining room and living room and dining room.
score: 0.129	score: 0.129
reference captions	
the room is empty other than the furniture. an open floor plan displays a modern kitchen, dining and living room arrangement. a kitchen, dining table and a living room looks like a small space. a small apartment is lit by modern style lamps.	

Therefore, we add a simple penalty on the SPICE evaluation package, denote as penalty factor $pf(c)$:

$$pf(c) = 1 - 0.15(|T(G(c))| - |T(G(c))_{minimal}|) \quad (19)$$

$$SPICE_p(c, S) = pf(c) * SPICE(c, S) \quad (20)$$

where $T(G(c))_{minimal}$ is the set removing all duplicate tuples and $SPICE_p(c, S)$ is the SPICE score with penalty and will be used in training (we use original package for testing). And a simple example is shown below:

Table 4: SPICE with penalty on repeated tuples

candidate caption	
a kitchen and dining room and living room.	a kitchen and dining room and living room and dining room.
score: 0.129	score: 0.110

3.2 Learning to evaluate image captioning

So far, all evaluation mertics mentioned are rule-based automatic metric. Researchers (Cui et al., 2018) (Sharif et al., 2018) start to get interested in learning-based evaluation metrics because learning-based methods not require expert-level linguistic knowledge but can be very powerful sometimes. Therefore, we choose the *Learning to evaluate image captioning* (LTEIC) to examinate how learning-based metric would performe as a reward metric.

Model overview: The intuition of LTEIC is to build a model to distinguish between machine-generated and human-written captions. Therefore we could build a discriminator accepting image, candidate caption, groud truth caption(s) and return a score of how likely the candidate is human-written. Hence, the LTEIC model have the arichiture as 2 shown:

It embeds the input image with a ResNet101 and embeds the reference captions with a LSTM. Image feature and reference captions representation are concatenated as the context vector. The candidate caption will also be fed into the same LSTM. The context representation and the candidate caption representation will be combine by compact bilinear pooling. Afterwards, a softmax classifier will score, as below, the combined feature from 0 to 1, the higher the better.

$$score_\theta(\hat{c}, C(I, S)) = P(\hat{c}\, is\, human\, written | C(I, S), \theta) \quad (21)$$

where I is the image, S is the reference captions, $C(I, S)$ denote the context, $\hat{c}$ is the candidate caption and θ is the network parameters.

Model performance: we trained the discriminator using *show attend and tell* (Xu et al., 2015) model, and tested our discriminator on *show attend and tell* itself, *neural talk* (Karpathy and Fei-Fei, 2015), *show and tell* (Vinyals et al., 2015) and human captions. Results are shown below:

Table 5: discriminator performance on test set

	show attend and tell	*neural talk*	*show and tell*	human
LTEIC score	**0.074**	0.342	0.408	**0.797**
Accuracy	**0.968**	0.651	0.564	**0.886**

This discriminator shows very accurate results on *show attend and tell* model and human captions but it struggles on captions generated by *neural talk* and *show and tell*. Which indicate a large bias, so we don't really expect it working as a reward metric (but we still use this frozen model as a scorer later even though it has a large bias.)

Some thoughts: It turns out the idea of using a discriminator as reward metrics is very close to the idea of Generative Adversarial Networks(GAN) (Goodfellow et al., 2014). But due to the time limit, We didn't try it but we believe it would be a good idea to improve our generative model and there were a few works (Dai et al., 2017) (Liang

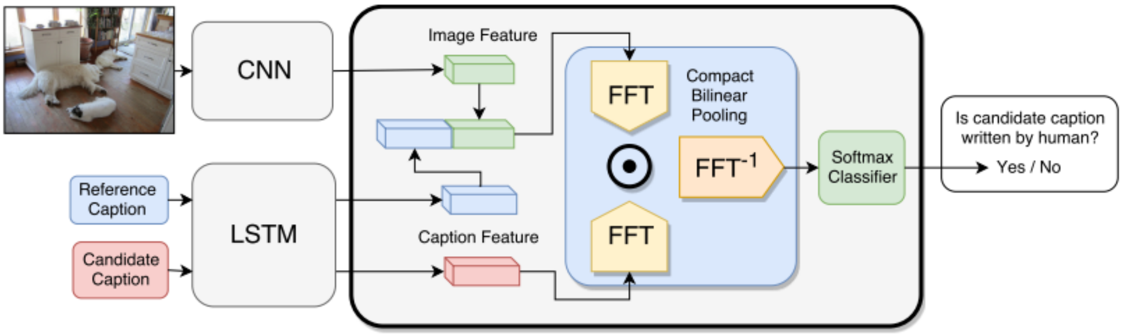


Figure 2: LTEIC model architecture

et al., 2017) using GAN emerged. Therefore, using GAN would be a better choice instead of using a frozen pretained discriminator.

4 Experiments and Results

4.1 Dataset

We use the same dataset, MSCOCO dataset (Lin et al., 2014), and keep the same dataset split with the SCST paper. The training set contains 113,287 images and 5 reference captions each image, validation set contains 5k images and results are reported on a testing set contains 5k images. And our results will be reported on 5 widely used image caption evalution mertics, including BLEU4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-L (Lin, 2004), CIDEr and SPICE.

4.2 Experiments

Since self-critical is trying to use the model itself as a baseline and it is difficult to start with a weak baseline which requires a huge amount of training, we therefore pretained a model with cross entropy for 30 epochs. And starts from the pretained model, we employeed the self-critical model and run another 40 epochs by directly optimizing CIDEr. And we are able to replicated the SCST paper results as the table below shows:

Table 6: replicated results on optimizing CIDEr

orginal paper				
	CIDEr	BLEU4	ROUGE-L	METEOR
XE	94.0	28.6	52.3	24.1
CIDEr	106.3	31.9	54.3	25.5
replicated results				
	CIDEr	BLEU4	ROUGE-L	METEOR
XE	92.5	29.5	52.8	24.6
CIDEr	106.7	32.5	54.5	25.6

At beginning, we hope directly optimizing SPICE could outperform the baseline, directly optimizing CIDEr, but it turns out directly optimizing SPICE might lead other metircs drop. A quick guess is, most of evaluation metrics we use is syntactic but SPICE is semantic based as figure 3 shows. Therefore, optimizing SPICE may not improve the syntactic quality, i.e. cause other scores droping.

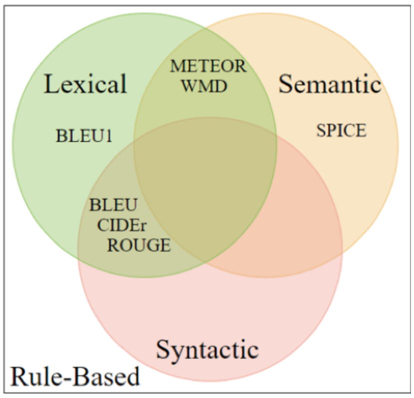


Figure 3: evalution metrics overview

Inspired by (Liu et al., 2016), a linear combination of both CIDEr and SPICE might cover all aspects, lexical, syntactic and semantic. Hence, we further employeed the linear combination of CIDEr and SPICE (SPIDEr) as a reward metric. Hence, we report 5 sets experiments results, including CIDEr, SPICE, SPIDEr(.5), SPIDEr(.8) and LTEIC (where .5 and .8 is the weight of SPICE score).

4.3 Results

We trained our 5 sets experiments 40 epoches each and evaluate the model on test set every 6000 iterations. The last evaluation results on test set are reported in table 7:

In general, directly optimzing SPICE might cause a worse results overall as we previously discussed

Table 7: experiment results, **bold font** indicate the best result in all experiments, underline indicate outperforming the baseline

Experiments	BLEU4	METEOR	ROUGE-L	CIDEr	SPICE
CIDEr(baseline)	**0.323**	0.256	0.544	**1.063**	0.188
SPICE	0.186	0.250	0.471	0.602	**0.241**
SPIDEr(.5)	0.321	0.258	**0.545**	1.058	0.196
SPIDEr(.8)	0.312	**0.261**	0.541	1.040	0.207
LTEIC	0.141	0.086	0.300	0.023	0.031

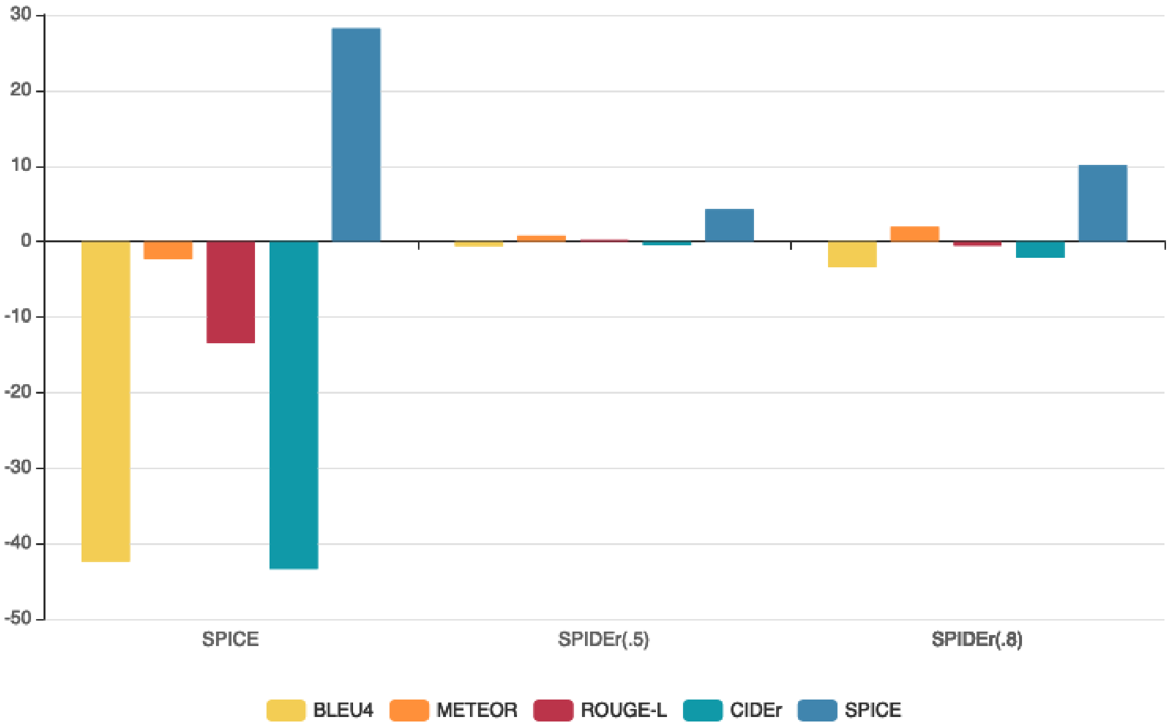


Figure 4: improvement% compared with baseline (optmizing CIDEr)

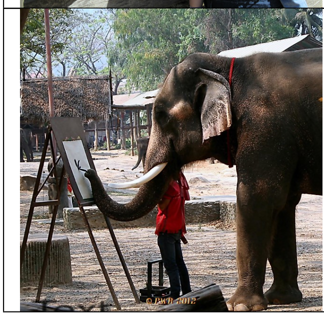
but end with a very high SPICE score. Optimzing LTEIC does not work at all as we expected. Optimzing CIDEr, SPIDEr(.5) and SPIDEr(.8) seems have very close results, no one is dominating others. Therefore we compared these 3 experiments, taking optimizing CIDEr as baseline, in figure 4. It shows optimizing SPIDEr has a little sacrifices on BLEU4 and CIDEr but will lead a decent improvement on SPICE (and small improvement on METEOR).
Besides, example captions from different models on 10 randomly selected COCO images can be found in table 8.

5 Conclusion

In conclusion, in this project, we inspired by the paper, *self-critical sequence training for image captioning*, and investigated several new evaluation metrics to be directly optimizing. The metrics we studied are SPICE, LTEIC and SPIDEr. We found that directly optimizing SPICE will not promise us a better results overall mainly because the testing metrics we report on are mostly syntactic-based but SPICE is semantic-based. And LTEIC itself has a very strong bias, therefore it doesn't work and indeed we didn't expect it to work. However, it turns out using a learning-based reward metric is the idea of GAN and we believe using GAN to formulate our generative model might be very petential and can be consider as a furture work. Optimizing SPIDEr shows a slightly better performance than optimizing CIDEr, which has a decent improvement on SPICE with relatively small trade-off on BLEU and CIDEr. We are also expecting a novel and solid metric to be emerged which would be very helpful for not only improving reinforcement learning based image captioning but also a lot of NLP tasks.

Acknowledgments

I acknowledgment Dr. Natalie Parde for lectures, instructions on this project. Thanks Ruotian Luo for high-quality implementation of *self-critical sequence training*. Thanks Yin Cui for releasing their code for *learning to evaluate image captioning* And I would like to thank Dr. Natalie Parde, Dr. Siqi Liu for advices on improving SPICE time efficiency.

Images	Ground Truth Captions	Generated Captions
	1. a plate with bacon, eggs and hamburger topped with bananas 2. a plate of chicken fried steak with bananas on top, eggs and bacon. 3. a plate with meat and bananas on top 4. banana pieces placed on beacon and sausage on a white plate 5. french bread on a plate with eggs, bacon and banana slices atop the bread.	• CIDEr: a plate of food on a table • SPICE: a white plate of food with a slice of cake sitting on top of a table • SPIDEr(.5): a plate of food on a table with a cake • SPIDEr(.8): a white plate of food on top of a table
	1. there are zebras that are lying on the ground 2. two zebra laying on a ground next to each other. 3. a few zebras lounge around at a zoo. 4. two zebras in an enclosure laying on the ground. 5. two zebras laying down in an enclosure as people watch nearby.	• CIDEr: a couple of zebras and a zebra laying on the ground • SPICE: two zebras in a dirt field of grass together a fence in a tree • SPIDEr(.5): two zebras and a zebra standing in the grass • SPIDEr(.8): two zebras laying on the ground in a field
	1. a city street filled with traffic next to tall buildings. 2. a person on a red motorcycle rides down a city street. 3. a motorcyclist on a red motorcycle speeds down an urban road towards a taxi and a mail truck. 4. a person on a motor bike on a street. 5. a red motorcycle being ridden down the road	• CIDEr: a man riding a motorcycle down a city street • SPICE: a man riding a motor bike riding a city street together a road • SPIDEr(.5): a man riding a motorcycle down a city street • SPIDEr(.8): a man riding a motorcycle down a city street
	1. a man takes a swings at a tennis ball on a court 2. a young man playing tennis with a basketball court in the background. 3. a man rared back at a tennis ball on a court. 4. a man prepares to hit a tennis ball with a tennis racquet. 5. on a court shared with a basketball hoop, a tennis player runs to return the ball.	• CIDEr: a man holding a tennis ball on a tennis court • SPICE: a young man holding a tennis racket holding a tennis ball on a tennis court • SPIDEr(.5): a man holding a tennis racket on a tennis court • SPIDEr(.8): a man holding a tennis racket on a tennis court
	1. a man standing next to a commercial truck. 2. a white truck with a blue canopy with fancy designs 3. a happy man stands by a white truck. 4. a man standing next to a white truck 5. a white truck with a colorful item on it's flatbed.	• CIDEr: a truck parked on the side of a street • SPICE: a large white delivery truck parked on a side of a road together a parking lot • SPIDEr(.5): a white truck parked on the side of a road • SPIDEr(.8): a white truck parked in a parking lot
	1. a woman holding a child are looking at a cow over a fence. 2. a woman holding a child next to a large cow. 3. the young child is close enough to pet the cow. 4. a cow standing up against a wooden fence near a woman and child. 5. a girl and her mom are petting a cow	• CIDEr: a little girl standing next to a cow • SPICE: a little girl and a child holding a brown horse in a fence • SPIDEr(.5): a little girl standing next to a cow • SPIDEr(.8): a little girl petting a cow in a fence
	1. the elephant is well known for his artistic ability. 2. an elephant holds a brush with its trunk and paints. 3. the elephant is using his trunk to paint a picture. 4. an elephant painting a picture with a brush in its trunk 5. a person helps an elephant to paint a picture.	• CIDEr: an elephant standing next to a field • SPICE: a large elephant in a trunk together a tree together a field • SPIDEr(.5): an elephant standing next to a man • SPIDEr(.8): a large elephant standing next to a tree

	1. a double decked bus parked next to an old church. 2. a city bus is parked on the side walk 3. a large red double decker bus driving past a church. 4. a stone church with a double decker bus out front. 5. a double deckered bus on a narrow street	• CIDEr: a red double decker bus driving down a street • SPICE: a large red double decker bus on a city street together a building together a road • SPIDEr(.5): a red double decker bus driving down a city street with a building • SPIDEr(.8): a red double decker bus driving down a city street with a building
	1. a red and yellow fire truck and some buildings 2. An overhead view shows a fire engine in the street. 3. A red and yellow fire truck with ladders on top 4. A firetruck is parked in the street in between stop lights. 5. A fire truck (ladder truck) drives down a street in the city.	• CIDEr: a red fire truck parked on the side of a street • SPICE: a red fire truck parked on a city street together a building together a man • SPIDEr(.5): a red fire truck parked on the side of a street • SPIDEr(.8): a red fire truck parked in front of a street
	1. A woman walking on a city street in a red coat. 2. A group of people that are standing on the side of a street. 3. A woman in a red jacket crossing the street 4. a street light some people and a woman wearing a red jacket 5. A blonde woman in a red coat crosses the street with her friend.	• CIDEr: a woman walking down a street with a traffic light • SPICE: a group of people and a woman walking down a city street with a traffic light • SPIDEr(.5): a group of people walking down a street • SPIDEr(.8): a group of people walking down a city street

Table 8: Example captions from different models on random COCO images.

References

Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2016. Spice: Semantic propositional image caption evaluation. In *European Conference on Computer Vision*, pages 382–398. Springer.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015a. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015b. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Yin Cui, Guandao Yang, Andreas Veit, Xun Huang, and Serge Belongie. 2018. Learning to evaluate image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5804–5812.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

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

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137.

Andrej Karpathy and Fei-Fei Li. 2014. Deep visual-semantic alignments for generating image descriptions. *CoRR*, abs/1412.2306.

Vijay R Konda and John N Tsitsiklis. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances In Neural Information Processing Systems*, pages 4601–4609.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P Xing. 2017. Recurrent topic-transition gan for visual paragraph generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3362–3371.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2016. Optimization of image description metrics using policy gradient methods. *CoRR*, abs/1612.00370.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2016. Self-critical sequence training for image captioning. *CoRR*, abs/1612.00563.

Naeha Sharif, Lyndon White, Mohammed Bennamoun, and Syed Afaq Ali Shah. 2018. Learning-based composite metrics for improved caption evaluation. In *Proceedings of ACL 2018, Student Research Workshop*, pages 14–20, Melbourne, Australia. Association for Computational Linguistics.

Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2014. Show and tell: A neural image caption generator. *CoRR*, abs/1411.4555.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057.

Wojciech Zaremba and Ilya Sutskever. 2015. Reinforcement learning neural turing machines-revised. *arXiv preprint arXiv:1505.00521*.