CALCS: Continuously Approximating Longest Common Subsequence for Sequence Level Optimization

Semih Yavuz*
University of California, Santa Barbara
syavuz@cs.ucsb.edu

Chung-Cheng Chiu
Google Brain
chungchengc@google.com

Patrick Nguyen
Google Brain
drpng@google.com

Yonghui Wu
Google Brain
yonghui@google.com

Abstract

Maximum-likelihood estimation (MLE) is one of the most widely used approaches for training structured prediction models for text-generation based natural language processing applications. However, besides *exposure bias*, models trained with MLE suffer from *wrong objective* problem where they are trained to maximize the word-level correct next step prediction, but are evaluated with respect to sequence-level discrete metrics such as ROUGE and BLEU. Several variants of policy-gradient methods address some of these problems by optimizing for final discrete evaluation metrics and showing improvements over MLE training for downstream tasks like text summarization and machine translation. However, policy-gradient methods suffers from high sample variance, making the training process very difficult and unstable. In this paper, we present an alternative direction towards mitigating this problem by introducing a new objective (CALCS) based on a differentiable surrogate of longest common subsequence (LCS) measure that captures sequence-level structure similarity. Experimental results on abstractive summarization and machine translation validate the effectiveness of the proposed approach.

1 Introduction

Recently, deep neural networks have achieved state-of-the-art results in various tasks including computer vision, natural language processing, and speech processing. Specifically, neural text generation models, central focus of this work, have led to great progress in central downstream NLP tasks like text summarization, machine translation, and image captioning. For example, the abstractive summarization task, which has previously not been the popular choice for text summarization due to lack of appropriate text generation methods, has gained revived attention with the success of neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2015). There has been several recent work with an impressive progress on this task including (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Miao and Blunsom, 2016; See et al., 2017; Tan et al., 2017; Zhou et al., 2017). Machine translation is another central field in NLP where the emergence of neural sequence-to-sequence models has enabled viable alternative approaches (Luong et al., 2015; Bahdanau et al., 2015; Cho et al., 2014; Sutskever et al., 2014) to challenge traditional phrase-based methods (Koehn et al., 2003).

Most of the recent existing works on neural text generation are based on variants of sequence-to-sequence models with attention (Bahdanau et al., 2015) trained with Maximum-likelihood estimation (MLE) with teacher forcing. As Ranzato et al. (2016) points out in a previous work, these models have two major drawbacks. First, they are trained to maximize the probability of correct next word given the entire sequence of previous ground truth words. While, at test time, the models need to generate the entire sequence by feeding its own predictions at previous time steps. This discrepancy is called *exposure bias* and hurts the performance as the model is never exposed to its own predictions during training. The second drawback, called *wrong objective*, is due yet another discrepancy between training and testing. It refers to the critique (Ranzato et al., 2016) that MLE-trained models tend to have suboptimal performance as they are trained to maximize a convenient objective (i.e., maximum likelihood of word-level correct next step prediction) rather than a desirable sequence-level objective that correlates better with the common discrete evaluation metrics such as ROUGE (Lin and Och, 2004) for summarization,

*Work done while interning at Google Brain.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3708–3718
Brussels, Belgium, October 31 - November 4, 2018.

BLEU (Papineni et al., 2002) for translation, and word error rate for speech recognition, not log-likelihood. On the other hand, training models that directly optimize for such discrete metrics as objective is hard due to non-differentiable nature of the corresponding loss functions (Rosti et al., 2011). To address these issues, Ranzato et al. (2016) introduces an incremental learning recipe that uses a hybrid loss function combining REINFORCE (Williams, 1992) and cross-entropy. Recently, Paulus et al. (2018) also explored combining maximum-likelihood and policy gradient training for text summarization.

Towards sequence level optimization, previous works (Ranzato et al., 2016; Wu et al., 2016; Paulus et al., 2018) employ reinforcement learning (RL) with a policy-gradient approach which works around the difficulty of differentiating the reward function by using it as a weight. However, REINFORCE is known to suffer from high sample variance and credit assignment problems which makes the training process difficult and unstable besides resulting in models that are hard to reproduce (Henderson et al., 2018).

In this paper, we propose an alternative approach for sequence-level training with longest common subsequence (LCS) metric that measures the sequence-level structure similarity between two sequences. We essentially introduce a continuous approximation to the discrete LCS metric which can be directly optimized against using standard gradient-based methods. Our proposed approach has the advantage of being able to directly optimize for a surrogate reward as opposed to using the exact reward only as a weight as in RL-inspired works. Hence, it provides a viable alternative perspective to policy-gradient methods for side stepping the non-differentiability with respect to the exact reward. In addition, it simultenuously combats the *exposure bias* problem through exposing the model to its own predictions while computing our approximation to LCS metric.

To this end, we introduce a new learning recipe that incorporates the aformentioned continuous approximation to LCS metric (CALCS) as an additional objective on top of maximum-likelihood loss in existing neural text generation models. We evaluate the proposed approach on abstractive text summarization and machine translation tasks. To this end, we use recently introduced *pointer-generator* network (See et al., 2017) and *transformer* (Vaswani et al., 2017) as underlying baselines for summarization and machine translation, respectively. More precisely, we start from a pre-trained baseline model with cross-entropy loss, and continue training the model to optimize for the proposed differentiable objective based on CALCS. Using this recipe, we conduct various experiments on *CNN/Daily Mail* (Hermann et al., 2015; Nallapati et al., 2016) summarization and *WMT 2014 English-to-German* machine translation tasks. Experimental results validate the effectiveness of the proposed approach on both tasks.

2 Continuously Approximating Longest Common Subsequence Metric

In this work, we explore the potential use of longest common subsequence (LCS) metric from an algorithmic point of view to address the aforementioned *wrong objective* and *exposure bias* problems. LCS metric measures a sequence-level structure similarity between discrete sequences by identifying longest co-occurring in sequence n-grams and it has been shown to correlate well with human judgments for downstream text generation tasks (Lin and Och, 2004). To this end, we propose a way to continuously approximate LCS metric and use this differentiable approximation as the objective to train text generation models rather than the exact LCS measure, which is hard to optimize for due to non-differentiability of the corresponding loss function. Although such differentiable approximation provides a unique advantage from modeling and optimization perspective, the difficulty of controlling its tightness might be a potential drawback in terms of its applicability. In this section, we will first introduce our proposed approximation to LCS metric, and then provide a natural way to control its tightness.

Consider a sequence generation problem conditioned on an input sequence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ and let $\mathbf{y} = (y_1, y_2, \ldots, y_m)$ denote its corresponding ground-truth output sequence. Let

$$f(\mathbf{x}, \mathbf{\Theta}) = \mathbf{z} = (z_1, z_2, \ldots, z_k)$$

denote hypothesis sequence obtained by greedy decoding from a generic encoder-decoder architecture for input sequence $\mathbf{x}$, where $\mathbf{\Theta}$ represents model parameters. Also, let $p_1, p_2, \ldots, p_k$ be the probability distributions over vocabulary V at decoding time steps from which $z_1, z_2, \ldots, z_k$ are generated via *argmax* operator, respectively.

2.1 CALCS

In this section, we define our approach to continuously approximate the longest common subsequence measure (LCS), which is an unnormalized version of ROUGE-L metric (Lin and Och, 2004) (See Appendix B) that is commonly used for performance evaluation of text summarization models. The main intuition behind our approach is to relax the common necessity for hard inferences while computing discrete metrics by instead comparing discrete tokens in a soft way. Towards this end, we start by defining LCS metric.

Definition 1. *Given two sequences* $\mathbf{y}$ *and* $\mathbf{z}$ *of tokens, longest common subsequence* $LCS(\mathbf{y}, \mathbf{z})$ *is defined as the longest sequence of tokens that appear left-to-right (but not necessarily in a contiguous block) in both sequences.*

The most common and intuitive solution for computing longest common subsequence is via dynamic programming. We will briefly revisit this here as it will be useful in terms of both recall and notational convenience while describing our surrogate LCS measure. Let $r_{i,j}$ denote the longest common subsequence between prefix sequences $\mathbf{y}_{[:i]} = (y_1, y_2, \ldots, y_i)$ and $\mathbf{z}_{[:j]} = (z_1, z_2, \ldots, z_j)$ of $\mathbf{y}$ and $\mathbf{z}$, respectively. A dynamic programming solution is given by

$$r_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ r_{i-1,j-1} + 1 & \text{if } y_i = z_j \\ max(r_{i-1,j}, r_{i,j-1}) & \text{o/w.} \end{cases} \tag{1}$$

$r_{i,j}$ for all $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, k$. It can be computed in mk iterations using the formula in Eqn 1. After computing 2D dynamic programming matrix r, we obtain $\text{LCS}(\mathbf{y}, \mathbf{z}) = r_{m,k}$.

Towards removing the dependence on hard inference for computing LCS, we now define our approximation to longest common subsequence, which we call CALCS. At high-level, the idea is to continuously relax the original LCS measure. To this end, we leverage output probability distributions $p_1, p_2, \ldots, p_k$ as soft predictions to refine the dynamic programming formulation for original LCS. More precisely, we recursively define soft longest common subsequence $s_{i,j}$ between prefixes $\mathbf{y}_{[:i]}$ and $\mathbf{z}_{[:j]}$ in analogous to $r_{i,j}$ as follows:

$$s_{i,j} = p_j^{(y_i)}(s_{i-1,j-1} + 1) + \tag{2}$$

$$(1 - p_j^{(y_i)}) \max(s_{i-1,j}, s_{i,j-1}) \tag{3}$$

for $i, j > 0$ and $s_{i,0} = s_{0,j} = 0$, where $p_j^{(y_i)}$ denote the probability of generating y_i at j-th decoding step. Intuitively, CALCS replaces the hard token comparison $\mathbb{1}[y_i = z_j]$ in Eq. 1 with the probability $p_j^{(y_i)}$ as in Eq. 2. Interpreting the probability $p_j^{(y_i)}$ as a continuous relaxation of discrete comparison operator $\mathbb{1}[y_i = z_j]$, $s_{i,j}$ establishes a natural continuous approximation to $r_{i,j}$. Similar to LCS, after iteratively filling up $s_{i,j}$ matrix, we define

$$\text{CALCS}(\mathbf{y}, \mathbf{z}) = s_{m,k} \tag{4}$$

Although the proposed approximation is a natural way of relaxing/extending the hard binary comparison of discrete tokens, it is not clear how tight the approximation is, which is established in the next section.

2.2 On the Tightness of Approximation

In this section, we first discuss the tightness of the proposed approximation, and then provide a natural way of controlling it.

2.2.1 Bounding the Approximation Error

We now present a bound on the approximation error of the proposed CALCS compared to the original LCS measure. Characterization of this bound will enable us to theoretically argue about the feasibilty of using the proposed surrogate reward function for our objective as well as controlling its tightness.

LCS measure is intrinsically monotonic by definition. We start by a lemma that establishes a similar monotonicity property for CALCS.

Lemma 1. *[Monotonicity] The following two inequalities*

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$$
$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$$

hold for all $0 \leq i < m$ *and* $0 \leq j < k$.

Proof. See Appendix A for the proof. □

Having established a certain monotonicity property for CALCS, we will discuss its approximation error to the original LCS measure. Let

$$\delta_{i,j} = s_{i,j} - r_{i,j} \tag{5}$$

denote the *approximation error* of CALCS to LCS measure between generated prefix sequence $\mathbf{y}_{[:i]}$ and the ground-truth prefix $\mathbf{z}_{[:j]}$.

Lemma 2. *Let* $P_{i,j} = \{(0,0),(i_1,j_1),\ldots,(i_{q-1},j_{q-1}),(i_q,j_q)\}$ *denote the path of dynamic programming algorithm for LCS ending at* $(i,j)=(i_q,j_q)$ *cell of* $m \times k$ *grid. Then,*

$$|\delta_{i,j}| < \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - max(p_j)) \quad (6)$$

where $max(p_j) = \max\{p_j^{(1)}, p_j^{(2)}, \ldots, p_j^{(|V|)}\}$.

Proof. We will establish the proof by investigating two cases and combining them.

CASE 1: $z_j = y_i$.
In this case, we have

$$r_{i,j} = 1 + r_{i-1,j-1} \quad (7)$$

and

$$(i_{q-1}, j_{q-1}) = (i-1, j-1) \quad (8)$$

by 1. Using Eq. 7, we get

$$\begin{aligned}
\delta_{i,j} &= s_{i,j} - r_{i,j} \\
&= \left(1 - p_j^{(y_i)}\right) \max\left(s_{i-1,j}, s_{i,j-1}\right) \\
&\quad + p_j^{(y_i)} \left(s_{i-1,j-1} + 1\right) - \left(1 + r_{i-1,j-1}\right) \\
&= \left(s_{i-1,j-1} - r_{i-1,j-1}\right) \\
&\quad + \left(1 - p_j^{(y_i)}\right) \Big[\max(s_{i-1,j}, s_{i,j-1}) \\
&\quad - \left(1 + s_{i-1,j-1}\right)\Big]
\end{aligned}$$

Using the definition of δ and triangle inequality, we get

$$\begin{aligned}
|\delta_{i,j}| &\le |\delta_{i-1,j-1}| + \left(1 - p_j^{(y_i)}\right) \Big| \left(1 + s_{i-1,j-1}\right) \\
&\quad - \max\left(s_{i-1,j}, s_{i,j-1}\right) \Big| \\
&\le |\delta_{i-1,j-1}| + \left(1 - p_j^{(y_i)}\right) \quad (9)
\end{aligned}$$

where inequality 9 follows from the monotonicity established by Lemma 1.

Moreover, $z_j = y_i$ implies $p_j^{(y_i)} = \max(p_j)$ because $\mathbf{z}$ is generated by greedy decoding. Plugging this in Eq. 9 and using Eq. 8, we can immediately conclude that

$$|\delta_{i,j}| < \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - \max(p_j)) \quad (10)$$

CASE 2: $z_j \neq y_i$.
By definition 1, we have

$$r_{i,j} = \max\left(r_{i-1,j}, r_{i,j-1}\right).$$

Using this identity, we obtain

$$\begin{aligned}
\delta_{i,j} &= s_{i,j} - r_{i,j} \\
&= \left(1 - p_j^{(y_i)}\right) \max\left(s_{i-1,j}, s_{i,j-1}\right) \\
&+ p_j^{(y_i)} \left(s_{i-1,j-1} + 1\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right) \\
&= p_j^{(y_i)} \left[\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right] \\
&+ \left[\max\left(s_{i-1,j}, s_{i,j-1}\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right)\right]
\end{aligned}$$

Applying triangle inequality on the last equation above, we get

$$\begin{aligned}
|\delta_{i,j}| &\le \\
&\quad p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \left|\max\left(s_{i-1,j}, s_{i,j-1}\right) - \max\left(r_{i-1,j}, r_{i,j-1}\right)\right| \\
&\quad \le \\
&\quad p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \max\left(\left|s_{i-1,j} - r_{i-1,j}\right|, \left|s_{i,j-1} - r_{i,j-1}\right|\right) \quad (11) \\
&\quad = \\
&\quad p_j^{(y_i)} \left|\left(1 + s_{i-1,j-1}\right) - \max\left(s_{i-1,j}, s_{i,j-1}\right)\right| \\
&+ \max\left(\left|\delta_{i-1,j}\right|, \left|\delta_{i,j-1}\right|\right) \\
&\quad \le p_j^{(y_i)} + \max\left(\left|\delta_{i-1,j}\right|, \left|\delta_{i,j-1}\right|\right) \quad (12)
\end{aligned}$$

where inequality 12 follows from again the monotonicity of $s[\cdot,\cdot]$, and inequality 11 follows from the following identity that holds true for all real numbers $a, b, c, d \ge 0$

$$|\max(a,b) - \max(c,d)| \le \max(|a-c|, |b-d|)$$

Moreover, since $z_j \neq y_i$, we know that $p_j^{(y_i)} \neq \max(p_j)$, which implies

$$p_j^{(y_i)} \le 1 - \max(p_i). \quad (13)$$

Combining 11 and 13 completes the proof for this case. Finally, two cases investigated above together establish the proof of Lemma 2. □

Lemma 2 leads to the following important corollary.

Corollary 1. *Let* $P_{i,j} = \{(0,0),(i_1,j_1),\ldots,(i_q,j_q)\}$ *be the path of dynamic programming algorithm for LCS ending at* $(i,j)=(i_q,j_q)$ *cell of* $m \times k$ *grid. Then,*

$$|\delta_{i,j}| \le \sum_{w=1}^{q} (1 - \max(p_{j_w})) \quad (14)$$

Proof. Applying Lemma 2 iteratively and using $\delta_{0,0} = 0$, we get

$$\begin{aligned}
|\delta_{i,j}| &\leq \left|\delta_{i_{q-1},j_{q-1}}\right| + (1 - \max(p_{j_q})) \\
\left|\delta_{i_{q-1},j_{q-1}}\right| &\leq \left|\delta_{i_{q-2},j_{q-2}}\right| + (1 - \max(p_{j_{q-1}})) \\
\left|\delta_{i_{q-2},j_{q-2}}\right| &\leq \left|\delta_{i_{q-3},j_{q-3}}\right| + (1 - \max(p_{j_{q-2}})) \\
&\vdots \\
|\delta_{i_1,j_1}| &\leq |\delta_{0,0}| + (1 - \max(p_{j_1})) \\
|\delta_{0,0}| &\leq 0
\end{aligned}$$

Summing $(q+1)$-many inequalities above side by side and cancelling out the same terms appearing on both sides of the resulting inequality establishes the proof of corollary. □

2.2.2 Controlling the Tightness of Approximation

Corollary 1 hints for a natural way of controlling the tightness of approximation CALCS by exploiting the peakedness of model's softmax output probability distributions. More precisely, upper bound on the approximation error is represented as a sum of $1 - \max(p_j)$'s, hence the more peaked the model's output probability distributions on average, the smaller the approximation error we are guaranteed by the established bounds.

We exploit this property to control the tightness of approximation by making a modification to computation of the proposed CALCS measure. Formally, let $l_1, l_2, \ldots, l_k$ denote the unnormalized logits of the model output before applying softmax to obtain probabilities $p_1, p_2, \ldots, p_k$ at decoding time steps, respectively. Hence,

$$p_j^{(i)} = \frac{\exp(l_j^{(i)})}{\sum_i \exp(l_j^{(i)})} \tag{15}$$

Recall that CALCS is computed using p_j's. Using *peaked softmax*, we can obtain more peaked probability distributions without causing any change in the actual generated sequence $\mathbf{z}$ via greedy decoding. This is simply because the order of probabilities for corresponding vocabulary words will not change, only the probability disribution p_j will get more peaked. So, we define *peaked softmax* operator with hyperparameter α as

$$p_j^{(i)}(\alpha) = \frac{\exp(l_j^{(i)}/\alpha)}{\sum_i \exp(l_j^{(i)}/\alpha)} \tag{16}$$

By Corollary 1, $|\delta_{i,j}| \to 0$ as $\alpha \to 0$ for CALCS measure computed with $p_j(\alpha)$. One can further attempt to use Corollary 1 as a guide to pinpoint a range of α values to force the approximation error within certain desired limits. We will use α as a hyperparameter in this work.

Corollary 1 is also useful for alternative ways of controlling the tightness of approximation such as incurring penalty for high-entropy output probability distributions or simply penalizing the maximum output probability values less than a desired threshold (that explicitly controls the tightness of the approximation). We leave such options of controlling the approximation error for future work.

With the guidance of Corollary 1 and *peaked softmax* in Eq. 16, we conclude that CALCS establishes a promising approximation for LCS measure. In the next section, we introduce a new objective function using CALCS as a continuously differentiable reward to be directly maximized.

2.3 Sequence Level Optimization via CALCS

In this section, we describe how to leverage CALCS to define a loss function for sequence level optimization. For notational consistency, we will use $f(\mathbf{x}, \mathbf{\Theta})$ to denote an encoder-decoder architecture that takes an input sequence $\mathbf{x}$ and outputs a sequence of tokens $\mathbf{z} = (z_1, z_2, \ldots, z_m)$ via greedy decoding from corresponding probability distributions $p_1, p_2, \ldots, p_m$ at each step.

For a pair of input sequence $\mathbf{x}$ and its corresponding ground-truth output sequence $\mathbf{y}$, we define

$$J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) = -\log\left(\frac{\text{CALCS}(\mathbf{y}, f(\mathbf{x}, \mathbf{\Theta}))}{|\mathbf{y}|}\right) \tag{17}$$

as the loss function for a sample $(\mathbf{x}, \mathbf{y})$ based on the CALCS, where $|\mathbf{y}|$ denote the length of sequence $\mathbf{y}$. It is important to note here that while computing probability distribution p_t at decoding step t, we feed model's own prediction z_{t-1} at the previous time step to fight *exposure bias*.

It is important to observe here that $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ is differentiable in $p_1, p_2, \ldots, p_k$ by definition and each p_i is differentiable in model parameters $\mathbf{\Theta}$. Hence, $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ is differentiable in model parameters $\mathbf{\Theta}$, which allows us to directly optimize the network parameters with respect to LCS metric. The bound we established on the approximation error and our proposed strategy to control it theoretically ensures the feasibility of using the introduced loss function J_{CALCS} to optimize for LCS metric.

3 Model

In this section, we first briefly revisit the pointer-generator (See et al., 2017) and transformer (Vaswani et al., 2017) networks that are used as the underlying baselines in our experiments. Subsequently, we describe how the proposed objective function and its variants are used to train new summarization and machine translation models.

3.1 Baseline Models

Pointer-Generator Network. We use pointer-generator network (See et al., 2017) as our baseline sequence-to-sequence model for text summarization. It is essentially a hybrid between sequence-to-sequence model with attention (Bahdanau et al., 2015) and a pointer network (Vinyals et al., 2015) that supports two decoding modes, copying and generating, via a soft switch mechanism. This enables the model to copy a word from the input sequence based on the attention distribution. On each decoding time step t, the decoder LSTM is fed the word embedding of the previous word, and computes a decoder state s_t, an attention distribution a^t over the words of input article, and a probability $P_{\text{vocab}}(w)$ of generating word w for summary from output vocabulary V, which is then softly combined with the *copy* mode's probability distribution $P_{\text{copy}}(w)$ via soft switch probability $p_{\text{gen}} \in [0, 1]$ by

$$p_t^{(w)} = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) P_{\text{copy}}(w)$$

and

$$P_{\text{copy}}(w) = \sum_{\{i:w_i=w\}} a_i^t$$

where a_i^t indicates the attention probability on i-th word of the input article. The whole network is then trained end-to-end with the negative log-likelihood loss function of

$$J_{\text{PG}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) = -\frac{1}{|y|} \sum_{t=1}^{|y|} \log(p_t^{(y_t)})$$

for a sample article-summary pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Theta}$ denote the learnable model parameters. It is important to note here that we do not use the coverage mechanism introduced by the original work (See et al., 2017) to prevent the potential repetition problem in the summaries generated by the model.

Transformer Network. For machine translation, we use the transformer network (Vaswani et al., 2017), which is a recently published model that achieved state-of-the-art results on *WMT 2014 English-to-German* MT task with less computational time owing to its highly parallelizable architecture. The core idea behind this model is to use stacked self-attention mechanisms along with point-wise, fully connected layers for both encoder and decoder to represent its input and output. For the sake of brevity, we refer the reader to (Vaswani et al., 2017) for further details regarding the architecture. Similar to previously defined loss functions, let $J_{\text{TF}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$ denote the per-example loss function of transformer networks for an input-output translation pair $(\mathbf{x}, \mathbf{y})$ where $\mathbf{\Theta}$ is again indicating the learnable model parameters.

3.2 Model Variants and Training

Let $\{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{N}$ denote the set of training examples, where $\mathbf{x}^{(l)}$'s are input sequences, and $\mathbf{y}^{(l)}$'s are their corresponding ground-truth output sequences. Before optimizing for the introduced objective J_{CALCS}, we first train the corresponding baseline network by minimizing

$$J_{\{\text{PG,TF}\}}(\mathbf{\Theta}) = \frac{1}{N} \sum_{l=1}^{N} J_{\{\text{PG,TF}\}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}).$$

Unlike J_{CALCS}, loss functions $J_{\{\text{PG,TF}\}}$ for baseline models are computed by teacher forcing, feeding the previous ground-truth word at each decoding step. We will denote the baseline models by POINTGEN for pointer-generator network and TRANSFORMER for transformer network.

To optimize for the proposed objective J_{CALCS}, we initialize the model parameters $\mathbf{\Theta}$ from the pre-trained baseline network and continue training the model by minimizing the joint loss

$$J(\mathbf{\Theta}) = \lambda J_{\text{CALCS}}(\mathbf{\Theta}) + (1 - \lambda) J_{\{\text{PG,TF}\}}(\mathbf{\Theta}) \tag{18}$$

$$J_{\text{CALCS}}(\mathbf{\Theta}) = \frac{1}{N} \sum_{l=1}^{N} J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta}) \tag{19}$$

where λ is a hyperparameter controlling the balance between the two losses. During the training with the joint loss, we compute $J_{\text{CALCS}}(\mathbf{x}, \mathbf{y}; \mathbf{\Theta})$, defined in Eq. 17, by performing $|y|$-many decoding steps as a simple strategy to prevent the model from gaming the training objec-

Model	ROUGE-1	ROUGE-L
(Nallapati et al., 2016)	35.46	32.65
w/o coverage (See et al., 2017)	36.44	33.42
w/ coverage (See et al., 2017)	39.53	36.38
LEAD-3 baseline (See et al., 2017)	**40.34**	36.57
RL (Paulus et al., 2018)	**41.16**	39.08
ML + RL (Paulus et al., 2018)	39.87	36.90
Our Models		
POINTGEN*	39.11	26.97**
POINTGEN*+SS	39.33	26.94**
POINTGEN*+SS+CALCS	**40.37**	**29.18****

Table 1: ROUGE F_1 results on *CNN/Daily Mail* summarization dataset. Our reimplementation of POINTGEN* corresponds to w/o coverage (See et al., 2017). ** sign near ROUGE-L results reported for our models indicates a difference in our ROUGE-L evaluation as explained below.

tive by generating longer and longer hypotheses instead of incurring an additional length penalty. We will refer to the resulting model trained with the loss function in Eq. 18 as {POINTGEN, TRANSFORMER}+CALCS depending on the baseline model.

4 Experiments

We numerically evaluate the proposed method on two sequence generation benchmarks: abstractive document-summarization and machine translation. We compare the results of the proposed method against the recently proposed strong baseline models (See et al., 2017) for summarization and and (Vaswani et al., 2017) for machine translation tasks.

4.1 Abstractive Summarization

We use a modified version of the CNN/Daily Mail dataset (Hermann et al., 2015) that is first used for summarization by (Nallapati et al., 2016). However, we follow the processing script provided by (See et al., 2017) to obtain *non-anonymized* version of the data that contains 287,226 training pairs, 13,368 validation pairs, and 11,490 test pairs of news articles (781 tokens on average) and their corresponding ground-truth summaries (56 tokens on average). We refer the reader to (See et al., 2017) for further details of the difference of their version from (Nallapati et al., 2016).

For training our baseline model, we use single layer LSTM encoder (bi-directional) and decoder with hidden dimensions of 512 and 1024, respectively. We use a vocabulary of 50k words for both source and target. Following the original paper, we also do not pre-train word embeddings, which are learned with the rest of model parameters during training. We use the Adam (Kingma and Ba, 2015) optimizer with a learning rate of 0.00001 for training. We pre-train the baseline model for 20k steps by applying greedy scheduled sampling (Bengio et al., 2015) with fixed ground-truth feeding probability of 75%. Once the baseline model training is complete, we start optimizing for CALCS objective as described in the previous section. Also, we set $\lambda = 1.0$ and $\alpha = 1.0$, which are tuned on the development set.

In Table 1, we report our main results on the summarization task. POINTGEN+SS refers to the baseline model trained with scheduled sampling. POINTGEN+SS+CALCS corresponds our model trained with CALCS starting from POINTGEN+SS model. Experimental results demonstrate that training with our proposed objective provides an improvement of 2.2 points in ROUGE-L score. This also provides empirical evidence to justify that our approximate CALCS effectively captures what the original LCS metric is supposed to measure, recalling ROUGE-L is a normalized LCS. The reason why ROUGE-L scores of our models are lower than previously reported is that we evaluate ROUGE-L score by taking the entire summary as a single sequence instead of splitting it into sentences, which is also the way we compute CALCS objective during the model training process. The main motivation behind this approach is to encourage the model to preserve the sentence order within a summary, and evaluate its performance in the same way. We consider the capability of preserving the order across produced sentences as an important attribute a multi-sentence summarization model should have in terms of readability and fluency of its generated summaries as a whole. When POINTGEN*+SS and POINTGEN*+SS+CALCS are evaluated by splitting the generated summaries into sentences, their corresponding ROUGE-L scores become 35.38 and 35.12, respectively. We also observe a nice side-improvement of 1.0 point in ROUGE-1 score over the baseline, which achieves a comparable performance with the long-overdue LEAD-3 baseline score. It might also be comparable to the recently reported state-of-the-art ROUGE-1 result on CNN/DailyMail dataset by Paulus et al. (2018) as they used a different dataset processing pipeline, which makes it difficult to directly compare with ours.

Model	BLEU
GNMT (Wu et al., 2016)	24.61
GNMT+RL (Wu et al., 2016)	24.60
TRANSFORMER (Vaswani et al., 2017)	27.3
WEIGHTED TRANSFORMER (Ahmed et al., 2018)	28.4
TRANSFORMER*	27.6
TRANSFORMER*+CALCS	**27.8**

Table 2: Machine translation results on *WMT 2014 English-to-German* task. TRANSFORMER* corresponds to our training of the original model in (Vaswani et al., 2017).

4.2 Machine Translation

We also evaluate our sequence-level training approach on the *WMT 2014 English-to-German* machine translation task, which contains 4.5M pairs of sentences.

To train our baseline transformer model, we closely follow the small model in the original transformer paper (Vaswani et al., 2017). We use a vocabulary of size 32k. Our encoder and decoder consist of $N = 6$ identical layers each. Following the notation in the original paper, we set the other parameters as $d_{\text{model}} = 512$, $d_{\text{ff}} = 2048$, $h = 8$, $P_{\text{drop}} = 0.1$. We set $\lambda = 0.3$ and $\alpha = 1.0$, which are tuned on the development set.

In Table 2, we show our empirical results on machine translation task. Our first observation is that our trained baseline transformer network achieves a better performance than the one reported in the original paper (Vaswani et al., 2017) by 0.3 BLEU score, which might be solely due to hyperparameter tuning. More importantly, we observe that training with our proposed CALCS objective leads to noticeable 0.2 BLEU point improvements over the baseline, which further reinforces our confidence in effectiveness of our proposed sequence-level training approach and its applicability to other sequence prediction tasks. It is also interesting to note that optimizing for LCS metric via its continuous approximation leads to improvements in evaluation with another discrete metric BLEU. On the other had, optimizing for the exact discrete metric BLEU via reinforcement learning strategy may not improve the evaluation performance in BLEU as reported by (Wu et al., 2016). As a final remark, we would like to note that our proposed approach is orthogonal to advancements in more expressive and powerful architecture designs. Hence it has the potential to provide further improvements over the recently proposed models such as WEIGHTED TRANSFORMER (Ahmed et al., 2018).

5 Related Work

Text Summarization. Before the successful application of neural generative models, most of the existing works on text summarization (Dorr et al., 2003; Durrett et al., 2016) have focused on extractive methods. While some of the early approaches have used a rich set of heuristic rules or sparse features to select textual units to include in the summary, more recent works (Cheng and Lapata, 2016; Nallapati et al., 2017) leverage neural models to select words and sentences from the original text. With the emergence of sequence-to-sequence models (Sutskever et al., 2014) and large-scale datasets like *CNN/Daily Mail* (Hermann et al., 2015; Nallapati et al., 2016) and *NYT* (Paulus et al., 2018), abstractive summarization of longer text have become a more feasible and popular task. Several recent approaches have been proposed to tackle abstractive summarization problem, where Nallapati et al. (2016) exploits hierarchical encoders, See et al. (2017) proposes *pointer-generator* network and coverage mechanism to overcome OOV and repetition problems, Tan et al. (2017) introduces a graph-based attention mechanism and hierarchical beam search strategy, and (Paulus et al., 2018) proposes to optimize for ROUGE metric via reinforcement learning. Although impressive progress has been achieved for sentence-level summarization, attempts on abstractive document summarization task are still in early stages where the simple LEAD-3 baseline performance is only very recently matched (Paulus et al., 2018).

Neural Machine Translation. With the recent success of encoder-decoder architectures (Sutskever et al., 2014; Bahdanau et al., 2015), neural machine translation systems has gained a a lot of attention both from academia (Cho et al., 2014; Luong et al., 2015; Luong and Manning, 2016) and industry (Wu et al., 2016; Vaswani et al., 2017; Ahmed et al., 2018) over statistical machine translation, which has been the dominating translation paradigm for years. Most of these works has focused more on enhancing the architecture design aspect to tackle with various challenges such as different attention mechanisms (Bahdanau et al., 2015; Luong et al., 2015), a character-level decoder (Chung et al., 2016), a translation coverage mechanism (Tu et al., 2016), and so on. However, only very recently, a few works (Wu et al., 2016; Ranzato et al., 2016;

Norouzi et al., 2016; Shen et al., 2016; Bahdanau et al., 2017; Zhukov and Kretov, 2017; Casas et al., 2018) have investigated sequence-level optimization by training to maximize BLEU score.

Neural Sequence Generation with RL. Most neural sequence generation models are trained with the objective of maximizing the probability of the next correct word. However, this results in a major discrepancy between training and test settings of these models because they are trained with cross-entropy loss at word-level, but evaluated based on sequence-level discrete metrics such as ROUGE (Lin and Och, 2004) or BLEU (Papineni et al., 2002). On the other hand, directly optimizing for such evaluation metrics is hard due to non-differentiable nature of the exact objective (Rosti et al., 2011). Recent works (Ranzato et al., 2016; Wu et al., 2016; Bahdanau et al., 2017; Paulus et al., 2018) address the difficulty of differentiating with respect to rewards based on such discrete metrics using variants of reinforcement learning. These methods essentially propose to mitigate the problem by optimizing the reward weighted log-likelihood of the hypothesis sequences generated by the model distribution. In this paper, we propose an alternative solution to tackle this problem by introducing a differentiable approximation to exact LCS metric that can be directly optimized by standard gradient-based methods without RL, while still addressing the *exposure bias* problem.

6 Conclusion and Future Work

In this work we explored an alternative approach for training text generation models with sequence-level optimization to combat *wrong objective* and *exposure bias* problems. We introduced a new objective function based on a continuous approximation of LCS metric that measures sequence-level structure similarity between sentences. We applied our proposed approach to *CNN/Daily Mail* dataset for long document summarization and *WMT 2014 English-to-German* machine translation task. By extending the objectives of strong neural baseline models with our proposed objective, we empirically demonstrated its effectiveness on these two tasks. Our proposed approach suggests a promising alternative to policy-gradient methods to side step the difficulty of differentiating w.r.t reward function while directly optimizing for sequence-level discrete metrics.

References

Karim Ahmed, Nitish Shirish Keskar, and Richard Socher. 2018. Weighted transformer network for machine translation. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengi. 2017. An Actor-Critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

N. Casas, M R. Costa-juss, and J.A. R. Fonollosa. 2018. A differentiable bleu loss. analysis and first results. *Workshop of the 6th International Conference on Learning Representations*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengi. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Sumit Chopra, Michael Auli, and M. Alexander Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *The North American Chapter of the Association for Computational Linguistics (NAACL)*.

J. Chung, K. Cho, , and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text Summarization Workshop - Volume 5*.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. 2018. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

C.Y. Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh-Thang Luong and Christopher D. Manning. 2016. A hybrid Word-Character approach to open vocabulary neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Yishu Miao and Phil Blunsom. 2016. Discrete generative models for sentence compression. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *AAAI Conference on Artificial Intelligence (AAAI)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Computational Natural Language Learning (CoNLL)*.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. 2016. Reward augmented maximum likelihood for neural structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Antti-Veikko I. Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2011. Expected bleu training for graphs: Bbn system description for wmt11 system combination task. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*.

M. Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods on Natural Language Processing (EMNLP)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Z. Tu, Z. Lu, Y. Liu, X. Liu, and H Li. 2016. Coverage-based neural machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Shazeer Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*.

R. J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Norouzim Mohammad, Wolfgang Macherey, Maxim Krikun, Cao Yuan, Qin Gao, and et al. Macherey, Klaus. 2016. Googles neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Qingyu Zhou, Na Yang, Fur Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Vlad Zhukov and Maksim Kretov. 2017. Differentiable lower bound for expected BLEU score. In *NIPS Workshop on Conversational AI*.

A Proof of Lemma 1

[Monotonicity] The following two inequalities

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$$
$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$$

hold for all $0 \leq i < m$ and $0 \leq j < k$.

Proof. We will prove this lemma by induction on $i + j$.

Base Case: $i + j = 0$. In this case, we have $i = j = 0$. Since $s_{0,0} = s_{1,0} = s_{0,1} = 0$ by definition, both $s_{0,0} \leq s_{0,1} \leq s_{0,0} + 1$ and $s_{0,0} \leq s_{1,0} \leq s_{0,0} + 1$ hold.

Inductive Step: Assume for $i + j = l$ that

$$s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1 \tag{20}$$
$$s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1 \tag{21}$$

hold.

We will now prove that the inequalities of inductive hypothesis hold for $i + j = l + 1$.

We will start by showing $s_{i,j+1} \geq s_{i,j}$. By definition, we have

$$s_{i,j+1} = p_{j+1}^{(y_i)}(s_{i-1,j} + 1) + \tag{22}$$
$$(1 - p_{j+1}^{(y_i)}) \max(s_{i-1,j+1}, s_{i,j}) \tag{23}$$
$$\geq p_{j+1}^{(y_i)}(s_{i-1,j} + 1) + (1 - p_{j+1}^{(y_i)}) s_{i,j} \tag{24}$$
$$\geq p_{j+1}^{(y_i)} s_{i,j} + (1 - p_{j+1,y_i}) s_{i,j} \tag{25}$$
$$\geq s_{i,j} \tag{26}$$

where inequality 24 follows from the definition of $\max$ operator, and inequality 25 follows from induction assumption 20 because $(i - 1) + j = l$. Hence, final inequality 26 establishes the proof of $s_{i,j+1} \geq s_{i,j}$.

Now, we will show that $s_{i,j+1} \leq s_{i,j} + 1$ holds. Again by definition, we have

$$s_{i,j+1} = p_{j+1}^{(y_i)}(s_{i-1,j} + 1) + \tag{27}$$
$$(1 - p_{j+1}^{(y_i)}) \max(s_{i-1,j+1}, s_{i,j}) \tag{28}$$
$$\leq p_{j+1}^{(y_i)}(s_{i-1,j} + 1) + \tag{29}$$
$$(1 - p_{j+1}^{(y_i)})(s_{i-1,j} + 1) \tag{30}$$
$$\leq s_{i-1,j} + 1 \tag{31}$$
$$\leq s_{i,j} + 1 \tag{32}$$

where inequalities 30 and 32 follow from inequalities 20 and 21 of inductive step as $(i-1)+j = l$.

Note that 26 and 32 completes the proof of $s_{i,j} \leq s_{i,j+1} \leq s_{i,j} + 1$ for $i + j = l + 1$. Following similar arguments, one can easily establish the correctness of $s_{i,j} \leq s_{i+1,j} \leq s_{i,j} + 1$ for $i+j = l+1$, which completes the proof of Lemma by induction. □

B Definition of Rouge-L

Definition 2. *ROUGE-L is a discrete similarity metric that takes into account sentence level structure similarity by identifying longest co-occurring in-sequence n-grams automatically via longest common subsequence measure. Formally, given two sequences* $\mathbf{y}$ *and* $\mathbf{z}$ *of tokens, we define ROUGE-L*$(\mathbf{y}, \mathbf{z})$ *as the harmonic mean of precision* $\frac{LCS(\mathbf{y},\mathbf{z})}{k}$ *and recall* $\frac{LCS(\mathbf{y},\mathbf{z})}{m}$ *based on LCS measure, where* $k = |z|$ *and* $m = |y|$.