
Differentiable lower bound for expected BLEU score

Vlad Zhukov

Neural Networks and Deep Learning Lab
Moscow Institute of Physics and Technology
Russia
vladzhukovtirko@gmail.com

Maksim Kreto

Neural Networks and Deep Learning Lab
Moscow Institute of Physics and Technology
Russia
kretovmk@gmail.com

Abstract

In natural language processing tasks performance of the models is often measured with some non-differentiable metric, such as BLEU score. To use efficient gradient-based methods for optimization, it is a common workaround to optimize some surrogate loss function. This approach is effective if optimization of such loss also results in improving target metric. The corresponding problem is referred to as loss-evaluation mismatch. In the present work we propose a method for calculation of differentiable lower bound of expected BLEU score that does not involve computationally expensive sampling procedure such as the one required when using REINFORCE rule from reinforcement learning (RL) framework.

1 Introduction

One of the features of natural language processing tasks is complicated quality metrics. We often cannot design and calculate such metrics rigorously, for example it is hard to quantify readability of text as perceived by human evaluator. So benchmark metrics are commonly used, such as BLEU [1] or ROUGE [2]. Although these metrics do not address the problem mentioned above [3] in full, they are interpretable and easy to compute.

A problem with these metrics is that they are usually non-differentiable hence gradient-based methods for optimization cannot be applied directly. This complication can be addressed at least in two principle ways:

- Design a "reasonable" differentiable surrogate loss function that provides some intuition and correlates with target metric. An example is optimization of cross-entropy loss with teacher forcing [4].
- Appeal to RL-like techniques (REINFORCE rule [5] and its generalizations [6]) in order to optimize expectation of target metric [7, 8]. This approach is typically data-inefficient, and a number of tricks are required to ensure reasonably fast convergence [9].

In the present article we are going to talk about the second approach. Specifically we want to eliminate sampling procedure from the technique at the cost of optimizing lower bound instead of exact expectation value. We illustrate our approach on a toy example.

2 Calculation of BLEU score

2.1 BLEU

In this section we briefly describe how BLEU score between candidate and reference texts is calculated. In order to calculate BLEU score modified n-gram precisions [1] are computed. Then counts of all n-grams are computed, which are further used to calculate overlap O^n for n-gram:

$$O_n = \sum_{\text{n-gram} \in C} \min(\text{Count}_C(\text{n-gram}), \text{Count}_R(\text{n-gram})) \quad (1)$$

Precisions are calculated as follows:

$$p_n = \frac{O_n}{\sum_{C \in \{Candidates\}} \sum_{\text{n-gram} \in C} \text{Count}(\text{n-gram})} \quad (2)$$

Where denominator is number of all possible n-grams in translation. BLEU score is calculated as geometric mean of precisions multiplied by weights w_n :

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^N w_n \log p_n\right) \quad (3)$$

Where $w_n = 1/N$, $N = 4$ and BP is brevity penalty, which is calculated as follows:

$$\text{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & c \leq r \end{cases} \quad (4)$$

Where r is effective *Reference* corpus length, and c is the total length of the *Candidate* corpus.

2.2 Matrix form

In order to work with BLEU score let us now rewrite formulas in more convenient matrix form. For simplicity we will consider the case of one pair of generated text *Candidate* and ground truth *Reference* with vocabulary of size v . Denote by x the matrix with shape $[\text{len}_x \times v]$ which contains one-hot encoded vectors for each word in *Candidate*. Similarly denote by y the matrix with shape $[\text{len}_y \times v]$ which contains one-hot encoded vectors for each word in reference.

We compute matrix $S^1 = xx^T$ and $P^1 = yc^T$ with shape $[\text{len}_x \times \text{len}_x]$, and $[\text{len}_y \times \text{len}_x]$, correspondingly:

$$S_{i,j}^1 = x_i x_j \quad P_{i,j}^1 = y_i x_j$$

In the similar manner matrices for all n-grams ($n \in [1..N]$) are defined:

$$S_{i,j}^n = \prod_{k=0}^{n-1} x_{i+k} x_{j+k} \quad P_{i,j}^n = \prod_{k=0}^{n-1} y_{i+k} x_{j+k} \quad (5)$$

With shapes $[\text{len}_x - n + 1 \times \text{len}_x - n + 1]$ and $[\text{len}_y - n + 1 \times \text{len}_x - n + 1]$. Elements of these matrices have the following form:

$$S_{i,j}^n = \begin{cases} 1 & \text{if } \text{Cand}_i^n = \text{Cand}_j^n \\ 0 & \text{otherwise} \end{cases} \quad P_{i,j}^n = \begin{cases} 1 & \text{if } \text{Ref}_i^n = \text{Cand}_j^n \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Where $Cand_i^n(Ref_i^n)$ is n-gram on the position i in $Candidate(Reference)$. For every index $Candidate$ we compute count of n-grams in $Reference$ and $Candidate$ identical to the n-gram which corresponds to that index.

$$v_i^{x,n} = \sum_{j=0}^{\text{len}_x-1} S_{j,i}^n \quad v_i^{y,n} = \sum_{j=0}^{\text{len}_y-1} P_{j,i}^n \quad (7)$$

Finally we combine these vectors to get formula for overlap:

$$O_n = \sum_{i=1}^{\text{len}_x} \frac{\min(v_i^{x,n}, v_i^{y,n})}{v_i^{x,n}} = \quad (8)$$

$$\sum_{i=1}^{\text{len}_x} \min\left(1, \frac{v_i^{y,n}}{v_i^{x,n}}\right) = \sum_{i=1}^{\text{len}_x} \min\left(1, \frac{\sum_{j=0}^{\text{len}_y-1} \prod_{k=0}^{n-1} y_{i+k} x_{j+k}}{\sum_{j=0}^{\text{len}_x-1} \prod_{k=0}^{n-1} x_{i+k} x_{j+k}}\right) \quad (9)$$

See Appendix A for the proof that formulas (1) and (8) are equivalent. Now we can calculate corresponding precisions:

$$p_n = \frac{O_n}{\text{len}_x - n + 1}$$

Thus formulas for calculation of BLEU score are rewritten in matrix form and we can now proceed with derivation of lower bound for expected BLEU score.

3 Optimization of BLEU score

3.1 REINFORCE rule

Given that BLEU score is non-differentiable, common approach is to use techniques from RL, specifically REINFORCE rule [5] for optimization of its expectation:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [R(\tau)] \Rightarrow \nabla_\theta J(\theta) = \mathbb{E}_\tau \left[\sum_t R_t \nabla_\theta \log \pi_\theta(a_t | s_t) \right]$$

Here R_t is sum of rewards for all future steps starting from t , $\pi_\theta(a_t | s_t)$ is stochastic policy, which in our case forms a probability distribution over words at state s_t . Problem with this formula is that random variable in integrand usually shows high variance [9]. Numerical estimation of this integral is challenging and requires a number of tricks [9], such as complex control variates, see for example [10].

While for typical reinforcement learning (RL) tasks this integral is indeed very complex and requires numerical estimation, we reckon in some specific tasks, such as calculation of BLEU score, analytical formula for lower bound on this integral can be derived.

3.2 Lower Bound

Consider somewhat simplified version of decoding task [11]. Assume decoder does not have any stochastic nodes in computational graph [6], and once we encoded context then decoder can issue matrix of probabilities of words p_x with size $[\text{len}_x, v]$. Many standard models fall into this category, for example any sequence-to-sequence [11] model with soft attention [12] trained by teacher forcing approach [4]. Denote by p_y matrix of stacked one-hot vectors that represent reference text. We assume that all the words in reference text are unique.

High-level plan of attack for deriving lower bound:

1. Write down explicit formula for mathematical expectation of BLEU score using distributions p_x and degenerate distribution p_y .

2. Introduce assumptions that allow deriving analytical formula for lower bound on the integral.
3. Show that in case of degenerate distribution p_x derived lower bound coincides with exact BLEU score.
4. Consider values of p_x as parameters of loss function (negative lower bound of BLEU in our case). Derivatives of loss function w.r.t. parameters can be calculated with automatic differentiation software.

Let us now elaborate the plan. Hereinafter we denote expected score as BLEU and score for specific input sentences x and y as $\text{BLEU}(x, y)$. First, we write down formula for BLEU:

$$\text{BLEU} = \mathbb{E}_{x \sim p_x, y \sim p_y} [\text{BLEU}(x, y)] = \text{BP} \cdot \mathbb{E}_{x \sim p_x, y \sim p_y} \left[\exp \left(\sum_{n=1}^N \omega_n \log p_n \right) \right] \quad (10)$$

Re-write formula in terms of overlaps for unigrams only:

$$\text{BLEU}_1 \sim \mathbb{E}_{x \sim p_x, y \sim p_y} \left[\sum_{i=1}^{\text{len}_x} O_1^i \right] = \mathbb{E}_{x \sim p_x, y \sim p_y} \left[\sum_{i=1}^{\text{len}_x} \frac{\min(v^{y,1}, v^{x,1})}{v^{x,1}} \right] \quad (11)$$

Distribution over y is omitted for brevity hereinafter. Now we derive lower bound for i -th component of this sum (see Appendix B for details):

$$\mathbb{E}_{x \sim p_x} O_1^i \geq \sum_n p_x^{in} \cdot \min \left(1, \frac{\sum_j y_j^n}{1 + \sum_{k \neq i} p_x^{kn}} \right) \quad (12)$$

Thus, components of overlap O_1 are easily calculated using p_x and p_y . BLEU score is proportional to individual terms corresponding to n-grams. At first it is transformed into product of expectations (see Appendix C):

$$\mathbb{E}_{x \sim p_x} \exp \left(\sum_{n=1}^N w_n \log p_n \right) \geq \prod_{n=1}^N (\mathbb{E}_{x \sim p_x} p_n)^{\omega_n} - N + 1 \quad (13)$$

Lower bound for n -th component of this product results in similar formulas as for unigrams (see details on derivation in Appendix D):

$$\mathbb{E}_{x \sim p_x} [O_n^i] \geq \sum_{m_0} p_x^{i, m_0} \cdot \sum_{m_{n-1}} p_x^{i+n-1, m_{n-1}} \min \left(1, \frac{\sum_j \prod_{k=0}^{n-1} y_{j+k, m_k}}{1 + \sum_{l \neq i} \prod_{k=0}^{n-1} p_x^{l+k, m_k}} \right) \quad (14)$$

Derived lower bound in equation above for each of the individual term in Eq. 13 is tight in the sense that if p_x is deterministic distribution (probabilities of words in candidate text are either 1 or 0) derived lower bound coincides with exact term. So if we want to optimize just one type of n-grams (say, 4-grams only) and not the aggregated BLEU score, we can use expression Eq. 14. If we optimize usual BLEU score with all n-grams up to order n simultaneously, we use sum of such lower bounds, but this sum do not coincide with exact BLEU score even for degenerate distribution due to Eq. 13.

4 Results and discussion

To test our approach we designed the following toy task: distribution p_x of words in generated text and reference text r is given. We also assume that all the words in r are unique and encoded as one-hot vectors. Both text are of same size 10. Vocabulary size set to 10000. Training examples are generated randomly:

- Elements of matrix p_x are generated by sampling logits from $\mathcal{N}(0, 1)$ and then *softmax* function is applied row-wise to obtain probability distribution for words.

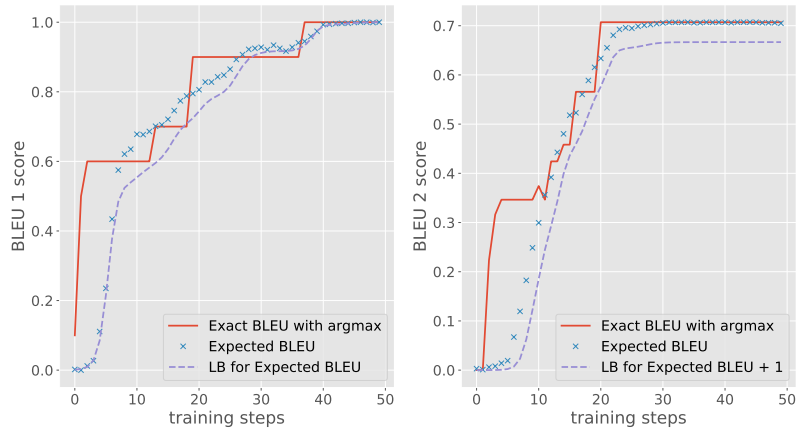


Figure 1: Example of learning curve for unigram BLEU score. Red line is exact BLEU score obtained by applying $argmax$ to p_x row-wise and calculating BLEU score. Purple dotted line is lower bound for expected BLEU score. Crosses denote expected BLEU score calculated by sampling from distribution p_x

- Indexes of reference text r are generated by uniform sampling with replacement from vocabulary.

In such setting, our target is to optimize parameters (elements of matrix p_x) to maximize expected BLEU score between two corpuses of text: one is sampled from p_x and another is r . For optimization we used lower bound of expected BLEU score.

Learning curve is presented at Figure 1. Chart indicates correlated behavior of lower bound and exact BLEU score of reference text.

5 Conclusions

We proposed a computationally efficient and differentiable lower bound for BLEU score. The approach is tested on toy examples and it is empirically shown that optimizing such objective results in improving target BLEU score. Implementation is available at https://github.com/deepmipt/expected_bleu.

Proposed technique allows to eliminate data-inefficient sampling procedure from pipeline of direct optimization of target metric as in RL setting at the cost of optimizing lower bound instead of exact metric.

Next steps are to validate the applicability of derived bound to real-world tasks such as Machine Translation and Dialogue Generation.

Acknowledgments

This work was supported by National Technology Initiative and PAO Sberbank project ID 0000000007417F630002.

References

- [1] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. 2002.
- [2] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries, 2004.
- [3] Chia-Wei Liu, Ryan Lowe, Iulian V. Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation, 2016.
- [4] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. 1989.
- [5] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. 1992.
- [6] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs, 2015.
- [7] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation, 2016.
- [8] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.
- [9] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. 2004.
- [10] George Tucker, Andriy Mnih, Chris J. Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models, 2017.
- [11] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.
- [12] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation, 2015.

A Equivalence of formulas for overlaps O_n

W denote set of n-grams for *Candidate*. I_ω denote set of start indices for n-gram ω in *Candidate*:

$$\begin{aligned} O_n &= \sum_{\omega \in W} \sum_{i \in I_\omega} \frac{\min(v_i^{y,n}, v_i^{x,n})}{v_i^{x,n}} = \left\{ \begin{array}{l} \forall i, j \in I_\omega \\ v_i^{x,n} = v_j^{x,n} = |I_\omega| = \text{Count}_C(\omega) \\ v_i^{y,n} = v_j^{y,n} = \text{Count}_R(\omega) \end{array} \right\} = \\ &= \sum_{\omega \in W} \sum_{i \in I_\omega} \frac{\min(\text{Count}_R(\omega), \text{Count}_C(\omega))}{\text{Count}_C(\omega)} = \\ &= \sum_{\omega \in W} \min(\text{Count}_R(\omega), \text{Count}_C(\omega)) \end{aligned}$$

B Unigram lower bound

$$\mathbb{E}_{x \sim p_x, y \sim p_y} O_1^i = \mathbb{E}_{x \sim p_x, y \sim p_y} \left[\min \left(1, \frac{\sum_j x_i y_j}{\sum_k x_i x_k} \right) \right] \quad (15)$$

Break down expectation into two integrals, one is over random variable x_i and the rest is over all other. Distribution over y is omitted hereinafter:

$$= \mathbb{E}_{x_k, k \neq i} \left[\sum_n p_x^{in} \min \left(1, \frac{\sum_j y_{jn}}{\sum_k x_i x_k} \right) \right] = \sum_n p_x^{in} \cdot \mathbb{E}_{x_k, k \neq i} \left[\min \left(1, \frac{\sum_j y_{jn}}{1 + \sum_{k \neq i} x_i x_k} \right) \right] \quad (16)$$

Expression in square brackets can be considered as a function of variable x_k with domain $[0, 1]^n$. This function is concave (see Appendix E), so we can apply Jensen's inequality to derive lower bound:

$$= \sum_n p_x^{in} \cdot \mathbb{E}_{x_k, k \neq i} \left[\min \left(1, \frac{\sum_j y_{jn}}{1 + \sum_{k \neq i} x_i x_k} \right) \right] \geq \sum_n p_x^{in} \cdot \min \left(1, \frac{\sum_j y_{jn}}{1 + \sum_{k \neq i} p_x^{kn}} \right) \quad (17)$$

C BLEU product

Using Jensen's inequality once again in the second line:

$$\begin{aligned} &\mathbb{E}_{x \sim p_x} \exp \left(\sum_{n=1}^N w_n \log p_n \right) = \mathbb{E}_{x \sim p_x} \prod_{n=1}^N p_n^{\omega_n} = \\ &= \mathbb{E}_{x \sim p_x} \prod_{n=1}^{N-1} p_n^{\omega_n} \cdot \mathbb{E}_{x \sim p_x} p_N^{\omega_N} + \text{cov} \left(\prod_{n=1}^{N-1} p_n^{\omega_n}, p_N^{\omega_N} \right) \geq \mathbb{E}_{x \sim p_x} \prod_{n=1}^{N-1} p_n^{\omega_n} \cdot \mathbb{E}_{x \sim p_x} p_N^{\omega_N} - 1 \geq \dots \\ &\dots \geq \prod_{n=1}^{N-1} \mathbb{E}_{x \sim p_x} p_n^{\omega_n} - N \geq \prod_{n=1}^{N-1} (\mathbb{E}_{x \sim p_x} p_n)^{\omega_n} - N + 1 \end{aligned}$$

D Generalization for n-grams

Write down explicitly mathematical expectation over random variables $\{x_{i+p} : p \in \{0..n-1\}\}$ for i -th component of this sum:

$$\mathbb{E}_{x \sim p_x} [O_n^i] = \mathbb{E}_{x_k, k \notin \{i+p : p \in \{0..n-1\}\}} \left[\sum_{m_0} p_x^{i, m_0} \dots \sum_{m_{n-1}} p_x^{i+n-1, m_{n-1}} O_n^i(x, y) \right] \quad (18)$$

Let s be some index and $s \notin \{i+p : p \in \{0..n-1\}\}$:

Consider two sets $A_{s,i} = \{l : l \neq i, s \in [l..l+n-1]\}$ and $B_{s,i} = \{l : l \neq i, s \notin [l..l+n-1]\}$ and rewrite

$$O_n^i(x, y) = \min\left(1, \frac{\sum_j \prod_{k=0}^{n-1} y_{j+k, m_k}}{1 + \sum_{A_{s,i}} \prod_{k=0}^{n-1} x_{i+k, l+k} + \sum_{B_{s,i}} \prod_{k=0}^{n-1} x_{i+k, l+k}}\right)$$

Change order of integration and write down separately expectation over p_x^s ,

$$\sum_{m_0} p_x^{i, m_0} \dots \sum_{m_{n-1}} p_x^{i+n-1, m_n} \mathbb{E}_{x_{k, k \notin \{i+p: p \in \{0..n-1\}, k \neq s\}}} \mathbb{E}_{p_x^s} O_n^i(x, y) \quad (19)$$

Mathematical expectation over random variable x^s has the following form:

$$\min\left(1, \frac{A}{1 + B + \vec{c}x_s}\right)$$

Where A, B are some constants and \vec{c} is constant vector:

$$\frac{A}{1 + B + \vec{c}x_s}$$

Since this is convex function (see Appendix E), we can apply Jensen's inequality sequentially for all $s \notin \{i+p : p \in \{0..n-1\}\}$.

New O_n^i will be

$$O_i'^n(x, y) = \min\left(1, \frac{\sum_j \prod_{k=0}^{n-1} y_{j+k, m_k}}{1 + \sum_{l \neq i} \prod_{k=0}^{n-1} p_x^{l+k, m_k}}\right)$$

And whole inequality is:

$$\mathbb{E}_{x \sim p_x} [O_n^i] \geq \sum_{m_0} p_x^{i, m_0} \dots \sum_{m_{n-1}} p_x^{i+n-1, m_n} O_i'^n(x, y)$$

E Proof of convexity

Define a as fixed vector, and A, B are some positive constants:

$$\frac{A}{B + a((1-\alpha)x_s + \alpha y_s)} \leq \frac{(1-\alpha)A}{B + ax_s} + \frac{\alpha A}{B + ay_s} \quad (20)$$

Without loss of generality:

$$\frac{1}{1 + a((1-\alpha)x_s + \alpha y_s)} \leq \frac{1-\alpha}{1 + ax_s} + \frac{\alpha}{1 + ay_s} \quad (21)$$

Let us denote $S = ax_s, T = ay_s$:

$$\frac{1}{1 + (1-\alpha)S + \alpha T} \leq \frac{1-\alpha}{1+S} + \frac{\alpha}{1+T}$$

$$\frac{1}{1 + (1-\alpha)S + \alpha T} \leq \frac{(1-\alpha)(1+T) + \alpha(1+S)}{(1+T)(1+S)}$$

$$\frac{1}{1 + (1-\alpha)S + \alpha T} \leq \frac{1 + (1-\alpha)T + \alpha S}{(1+T)(1+S)}$$

$$\ln(1+T) + \ln(1+S) \leq \ln(1 + (1-\alpha)S + \alpha T) + \ln(1 + (1-\alpha)T + \alpha S)$$

Since $\ln(1+x)$ is convex:

$$\ln(1 + (1-\alpha)S + \alpha T) \geq (1-\alpha) \ln(1+S) + \alpha \ln(1+T)$$

$$\ln(1 + (1-\alpha)T + \alpha S) \geq (1-\alpha) \ln(1+T) + \alpha \ln(1+S)$$