Classical Structured Prediction Losses for Sequence to Sequence Learning

Sergey Edunov[*], **Myle Ott**[*],
Michael Auli, David Grangier, Marc'Aurelio Ranzato
Facebook AI Research
Menlo Park, CA and New York, NY

Abstract

There has been much recent work on training neural attention models at the sequence-level using either reinforcement learning-style methods or by optimizing the beam. In this paper, we survey a range of classical objective functions that have been widely used to train linear models for structured prediction and apply them to neural sequence to sequence models. Our experiments show that these losses can perform surprisingly well by slightly outperforming beam search optimization in a like for like setup. We also report new state of the art results on both IWSLT'14 German-English translation as well as Gigaword abstractive summarization. On the large WMT'14 English-French task, sequence-level training achieves 41.5 BLEU which is on par with the state of the art.[1]

1 Introduction

Sequence to sequence models are usually trained with a simple token-level likelihood loss (Sutskever et al., 2014; Bahdanau et al., 2014). However, at test time, these models do not produce a single token but a whole sequence. In order to resolve this inconsistency and to potentially improve generation, recent work has focused on training these models at the sequence-level, for instance using REINFORCE (Ranzato et al., 2015), actor-critic (Bahdanau et al., 2016), or with beam search optimization (Wiseman and Rush, 2016).

Before the recent work on sequence level training for neural networks, there has been a large body of research on training linear models at the

---

[*] Equal contribution.
[1] An implementation of the losses is available as part of fairseq at:
`https://github.com/pytorch/fairseq/tree/classic_seqlevel`

sequence level. For example, direct loss optimization has been popularized in machine translation with the Minimum Error Rate Training algorithm (MERT; Och 2003) and expected risk minimization has an extensive history in NLP (Smith and Eisner, 2006; Rosti et al., 2010; Green et al., 2014). This paper revisits several objective functions that have been commonly used for structured prediction tasks in NLP (Gimpel and Smith, 2010) and apply them to a neural sequence to sequence model (Gehring et al., 2017b) (§2). Specifically, we consider likelihood training at the sequence-level, a margin loss as well as expected risk training. We also investigate several combinations of global losses with token-level likelihood. This is, to our knowledge, the most comprehensive comparison of structured losses in the context of neural sequence to sequence models (§3).

We experiment on the IWSLT'14 German-English translation task (Cettolo et al., 2014) as well as the Gigaword abstractive summarization task (Rush et al., 2015). We achieve the best reported accuracy to date on both tasks. We find that the sequence level losses we survey perform similarly to one another and outperform beam search optimization (Wiseman and Rush, 2016) on a comparable setup. On WMT'14 English-French, we also illustrate the effectiveness of risk minimization on a larger translation task. Classical losses for structured prediction are still very competitive and effective for neural models (§5, §6).

2 Sequence to Sequence Learning

The general architecture of our sequence to sequence models follows the encoder-decoder approach with soft attention first introduced in (Bahdanau et al., 2014). As a main difference, in most of our experiments we parameterize the encoder and the decoder as convolutional neural

networks instead of recurrent networks (Gehring et al., 2017a,b). Our use of convolution is motivated by computational and accuracy considerations. However, the objective functions we present are model agnostic and equally applicable to recurrent and convolutional models. We demonstrate the applicability of our objective functions to recurrent models (LSTM) in our comparison to Wiseman and Rush (2016) in §6.6.

Notation. We denote the source sentence as $\mathbf{x}$, an output sentence of our model as $\mathbf{u}$, and the reference or *target* sentence as $\mathbf{t}$. For some objectives, we choose a pseudo reference $\mathbf{u}^*$ instead, such as a model output with the highest BLEU or ROUGE score among a set of candidate outputs, $\mathcal{U}$, generated by our model.

Concretely, the encoder processes a source sentence $\mathbf{x} = (x_1, \ldots, x_m)$ containing m words and outputs a sequence of states $\mathbf{z} = (z_1, \ldots, z_m)$. The decoder takes $\mathbf{z}$ and generates the output sequence $\mathbf{u} = (u_1, \ldots, u_n)$ left to right, one element at a time. For each output u_i, the decoder computes hidden state h_i based on the previous state h_{i-1}, an embedding g_{i-1} of the previous target language word u_{i-1}, as well as a conditional input c_i derived from the encoder output $\mathbf{z}$. The attention context c_i is computed as a weighted sum of $(z_1, \ldots, z_m)$ at each time step. The weights of this sum are referred to as attention scores and allow the network to focus on the most relevant parts of the input at each generation step. Attention scores are computed by comparing each encoder state z_j to a combination of the previous decoder state h_i and the last prediction u_i; the result is normalized to be a distribution over input elements. At each generation step, the model scores for the V possible next target words u_i by transforming the decoder output h_i via a linear layer with weights W_o and bias b_o: $s_i = W_o h_i + b_o$. This is turned into a distribution via a softmax: $p(u_i|u_1, \ldots, u_{i-1}, \mathbf{x}) = \text{softmax}(s_i)$.

Our encoder and decoder use gated convolutional neural networks which enable fast and accurate generation (Gehring et al., 2017b). Fast generation is essential to efficiently train on the model output as is done in this work as sequence-level losses require generating at training time. Both encoder and decoder networks share a simple block structure that computes intermediate states based on a fixed number of input tokens and we stack several blocks on top of each other. Each block

contains a 1-D convolution that takes as input k feature vectors and outputs another vector; subsequent layers operate over the k output elements of the previous layer. The output of the convolution is then fed into a gated linear unit (Dauphin et al., 2017). In the decoder network, we rely on causal convolution which rely only on states from the previous time steps. The parameters θ of our model are all the weight matrices in the encoder and decoder networks. Further details can be found in Gehring et al. (2017b).

3 Objective Functions

We compare several objective functions for training the model architecture described in §2. The corresponding loss functions are either computed over individual tokens (§3.1), over entire sequences (§3.2) or over a combination of tokens and sequences (§3.3). An overview of these loss functions is given in Figure 1.

3.1 Token-Level Objectives

Most prior work on sequence to sequence learning has focused on optimizing token-level loss functions, i.e., functions for which the loss is computed additively over individual tokens.

Token Negative Log Likelihood (`TokNLL`)
Token-level likelihood (TokNLL, Equation 1) minimizes the negative log likelihood of individual reference tokens $\mathbf{t} = (t_1, \ldots, t_n)$. It is the most common loss function optimized in related work and serves as a baseline for our comparison.

Token NLL with Label Smoothing (`TokLS`)
Likelihood training forces the model to make extreme zero or one predictions to distinguish between the ground truth and alternatives. This may result in a model that is too confident in its training predictions, which may hurt its generalization performance. Label smoothing addresses this by acting as a regularizer that makes the model less confident in its predictions. Specifically, we smooth the target distribution with a prior distribution f that is independent of the current input $\mathbf{x}$ (Szegedy et al., 2015; Pereyra et al., 2017; Vaswani et al., 2017). We use a uniform prior distribution over all words in the vocabulary, $f = \frac{1}{V}$. One may also use a unigram distribution which has been shown to work better on some tasks (Pereyra et al., 2017). Label smoothing is equivalent to adding the KL divergence between f and the model prediction

$$\mathcal{L}_{\texttt{TokNLL}} = -\sum_{i=1}^{n} \log p(t_i|t_1,\ldots,t_{i-1},\mathbf{x}) \tag{1}$$

$$\mathcal{L}_{\texttt{TokLS}} = -\sum_{i=1}^{n} \log p(t_i|t_1,\ldots,t_{i-1},\mathbf{x}) - D_{KL}(f\|p(t_i|t_1,\ldots,t_{i-1},\mathbf{x})) \tag{2}$$

$$\mathcal{L}_{\texttt{SeqNLL}} = -\log p(\mathbf{u}^*|\mathbf{x}) + \log \sum_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} p(\mathbf{u}|\mathbf{x}) \tag{3}$$

$$\mathcal{L}_{\texttt{Risk}} = \sum_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} \text{cost}(\mathbf{t},\mathbf{u}) \frac{p(\mathbf{u}|\mathbf{x})}{\sum_{\mathbf{u}'\in\mathcal{U}(\mathbf{x})} p(\mathbf{u}'|\mathbf{x})} \tag{4}$$

$$\mathcal{L}_{\texttt{MaxMargin}} = \max\left[0, \text{cost}(\mathbf{t},\hat{\mathbf{u}}) - \text{cost}(\mathbf{t},\mathbf{u}^*) - s(\mathbf{u}^*|\mathbf{x}) + s(\hat{\mathbf{u}}|\mathbf{x})\right] \tag{5}$$

$$\mathcal{L}_{\texttt{MultiMargin}} = \sum_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} \max\left[0, \text{cost}(\mathbf{t},\mathbf{u}) - \text{cost}(\mathbf{t},\mathbf{u}^*) - s(\mathbf{u}^*|\mathbf{x}) + s(\mathbf{u}|\mathbf{x})\right] \tag{6}$$

$$\mathcal{L}_{\texttt{SoftmaxMargin}} = -\log p(\mathbf{u}^*|\mathbf{x}) + \log \sum_{\mathbf{u}\in\mathcal{U}(\mathbf{x})} \exp\left[s(\mathbf{u}|\mathbf{x}) + \text{cost}(\mathbf{t},\mathbf{u})\right] \tag{7}$$

Figure 1: Token and sequence negative log-likelihood (Equations 1 and 3), token-level label smoothing (Equation 2), expected risk (Equation 4), max-margin (Equation 5), multi-margin (Equation 6), softmax-margin (Equation 7). We denote the source as $\mathbf{x}$, the reference target as $\mathbf{t}$, the set of candidate outputs as $\mathcal{U}$ and the best candidate (pseudo reference) as $\mathbf{u}^*$. For max-margin we denote the candidate with the highest model score as $\hat{\mathbf{u}}$.

$p(\mathbf{u}|\mathbf{x})$ to the negative log likelihood (`TokLS`, Equation 2). In practice, we implement label smoothing by modifying the ground truth distribution for word u to be $q(u) = 1 - \epsilon$ and $q(u') = \frac{\epsilon}{V}$ for $u' \neq u$ instead of $q(u) = 1$ and $q(u') = 0$ where ϵ is a smoothing parameter.

3.2 Sequence-Level Objectives

We also consider a class of objective functions that are computed over entire sequences, i.e., sequence-level objectives. Training with these objectives requires generating and scoring multiple candidate output sequences for each input sequence during training, which is computationally expensive but allows us to directly optimize task-specific metrics such as BLEU or ROUGE.

Unfortunately, these objectives are also typically defined over the entire space of possible output sequences, which is intractable to enumerate or score with our models. Instead, we compute our sequence losses over a subset of the output space, $\mathcal{U}(\mathbf{x})$, generated by the model. We discuss approaches for generating this subset in §4.

Sequence Negative Log Likelihood (`SeqNLL`)

Similar to `TokNLL`, we can minimize the negative log likelihood of an entire sequence rather than individual tokens (`SeqNLL`, Equation 3). The log-

likelihood of sequence $\mathbf{u}$ is the sum of individual token log probabilities, normalized by the number of tokens to avoid bias towards shorter sequences:

$$p(\mathbf{u}|\mathbf{x}) = \exp \frac{1}{n} \sum_{i=1}^{n} \log p(u_i|u_1,\ldots,u_{i-1},\mathbf{x})$$

As target we choose a pseudo reference[2] amongst the candidates which maximizes either BLEU or ROUGE with respect to $\mathbf{t}$, the gold reference:

$$\mathbf{u}^*(\mathbf{x}) = \underset{\mathbf{u}\in\mathcal{U}(\mathbf{x})}{\arg\max}\, \text{BLEU}(\mathbf{t},\mathbf{u})$$

As is common practice when computing BLEU at the sentence-level, we smooth all initial counts to one (except for unigram counts) so that the geometric mean is not dominated by zero-valued n-gram match counts (Lin and Och, 2004).

Expected Risk Minimization (`Risk`)

This objective minimizes the expected value of a given cost function over the space of candidate sequences (`Risk`, Equation 4). In this work we use task-specific cost functions designed to maximize BLEU or ROUGE (Lin, 2004), e.g., $\text{cost}(\mathbf{t},\mathbf{u}) =$

---

[2]Another option is to use the gold reference target, $\mathbf{t}$, but in practice this can lead to degenerate solutions in which the model assigns low probabilities to nearly all outputs. This is discussed further in §4.

$1 - \mathrm{BLEU}(\mathbf{t}, \mathbf{u})$, for a given a candidate sequence $\mathbf{u}$ and target $\mathbf{t}$. Different to SeqNLL (§3.2), this loss may increase the score of several candidates that have low cost, instead of focusing on a single sequence which may only be marginally better than any alternatives. Optimizing this loss is a particularly good strategy if the reference is not always reachable, although compared to classical phrase-based models, this is less of an issue with neural sequence to sequence models that predict individual words or even sub-word units.

The Risk objective is similar to the REIN-FORCE objective used in Ranzato et al. (2015), since both objectives optimize an expected cost or reward (Williams, 1992). However, there are a few important differences: (1) whereas REINFORCE typically approximates the expectation with a single sampled sequence, the Risk objective considers multiple sequences; (2) whereas REINFORCE relies on a *baseline reward*[3] to determine the sign of the gradients for the current sequence, for the Risk objective we instead estimate the expected cost over a set of candidate output sequences (see §4); and (3) while the baseline reward is different for every word in REINFORCE, the expected cost is the same for every word in risk minimization since it is computed on the sequence level based on the actual cost.

Max-Margin

MaxMargin (Equation 5) is a classical margin loss for structured prediction (Taskar et al., 2003; Tsochantaridis et al., 2005) which enforces a margin between the model scores of the highest scoring candidate sequence $\hat{\mathbf{u}}$ and a reference sequence. We replace the human reference $\mathbf{t}$ with a pseudo-reference $\mathbf{u}^*$ since this setting performed slightly better in early experiments; $\mathbf{u}^*$ is the candidate sequence with the highest BLEU score. The size of the margin *varies* between samples and is given by the difference between the cost of $\mathbf{u}^*$ and the cost of $\hat{\mathbf{u}}$. In practice, we scale the margin by a hyper-parameter β determined on the validation set: $\beta(\mathrm{cost}(\mathbf{t}, \hat{\mathbf{u}}) - \mathrm{cost}(\mathbf{t}, \mathbf{u}^*))$. For this loss we use the unnormalized scores computed by the model before the final softmax:

$$s(\mathbf{u}|\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} s(u_i|u_1, \ldots, u_{i-1}, \mathbf{x})$$

---

[3]Ranzato et al. (2015) estimate the baseline reward for REINFORCE with a separate linear regressor over the model's current hidden state.

Multi-Margin

MaxMargin only updates two elements in the candidate set. We therefore consider MultiMargin (Equation 6) which enforces a margin between *every* candidate sequence $\mathbf{u}$ and a reference sequence (Herbrich et al., 1999), hence the name Multi-Margin. Similar to MaxMargin, we replace the reference $\mathbf{t}$ with the pseudo-reference $\mathbf{u}^*$.

Softmax-Margin

Finally, SoftmaxMargin (Equation 7) is another classic loss that has been proposed by Gimpel and Smith (2010) as another way to optimize task-specific costs. The loss augments the scores inside the exp of SeqNLL (Equation 3) by a cost. The intuition is that we want to penalize high cost outputs proportional to their cost.

3.3 Combined Objectives

We also experiment with two variants of combining sequence-level objectives (§3.2) with token-level objectives (§3.1). First, we consider a weighted combination (Weighted) of both a sequence-level and token-level objective (Wu et al., 2016), e.g., for TokLS and Risk we have:

$$\mathcal{L}_{\texttt{Weighted}} = \alpha \mathcal{L}_{\texttt{TokLS}} + (1 - \alpha)\mathcal{L}_{\texttt{Risk}} \quad (8)$$

where α is a scaling constant that is tuned on a held-out validation set.

Second, we consider a constrained combination (Constrained), where for any given input we use either the token-level or sequence-level loss, but not both. The motivation is to maintain good token-level accuracy while optimizing on the sequence-level. In particular, a sample is processed with the sequence loss if the token loss under the current model is at least as good as the token loss of a baseline model $\mathcal{L}_{\texttt{TokLS}}^{b}$. Otherwise, we update according to the token loss:

$$\mathcal{L}_{\texttt{Constrained}} = \begin{cases} \mathcal{L}_{\texttt{Risk}} & \mathcal{L}_{\texttt{TokLS}} \leq \mathcal{L}_{\texttt{TokLS}}^{b} \\ \mathcal{L}_{\texttt{TokLS}} & \text{otherwise} \end{cases}$$

$$(9)$$

In this work we use a fixed baseline model that was trained with a token-level loss to convergence.

4 Candidate Generation Strategies

The sequence-level objectives we consider (§3.2) are defined over the entire space of possible output sequences, which is intractable to enumerate or

score with our models. We therefore use a subset of K candidate sequences $\mathcal{U}(\mathbf{x}) = \{u_1, \ldots, u_K\}$, which we generate with our models.

We consider two search strategies for generating the set of candidate sequences. The first is *beam search*, a greedy breadth-first search that maintains a "beam" of the top-K scoring candidates at each generation step. Beam search is the *de facto* decoding strategy for achieving state-of-the-art results in machine translation. The second strategy is *sampling* (Chatterjee and Cancedda, 2010), which produces K independent output sequences by sampling from the model's conditional distribution. Whereas beam search focuses on high probability candidates, sampling introduces more diverse candidates (see comparison in §6.5).

We also consider both online and offline candidate generation settings in §6.4. In the online setting, we regenerate the candidate set every time we encounter an input sentence $\mathbf{x}$ during training. In the offline setting, candidates are generated before training and are never regenerated. Offline generation is also embarrassingly parallel because all samples use the same model. The disadvantage is that the candidates become stale. Our model may perfectly be able to discriminate between them after only a single update, hindering the ability of the loss to correct eventual search errors.[4]

Finally, while some past work has added the reference target to the candidate set, i.e., $\mathcal{U}'(\mathbf{x}) = \mathcal{U}(\mathbf{x}) \cup \{\mathbf{t}\}$, we find this can destabilize training since the model learns to assign low probabilities nearly everywhere, ruining the candidates generated by the model, while still assigning a slightly higher score to the reference (cf. Shen et al. (2016)). Accordingly we do not add the reference translation to our candidate sets.

5 Experimental Setup

5.1 Translation

We experiment on the IWSLT'14 German to English (Cettolo et al., 2014) task using a similar setup as Ranzato et al. (2015), which allows us to compare to other recent studies that also adopted this setup, e.g., Wiseman and Rush (2016).[5] The training data consists of 160K sentence pairs and the validation set comprises 7K sentences ran-

domly sampled and held-out from the train data. We test on the concatenation of *tst2010, tst2011, tst2012, tst2013* and *dev2010* which is of similar size to the validation set. All data is lowercased and tokenized with a byte-pair encoding (BPE) of 14,000 types (Sennrich et al., 2016) and we evaluate with case-insensitive BLEU.

We also experiment on the much larger WMT'14 English-French task. We remove sentences longer than 175 words as well as pairs with a source/target length ratio exceeding 1.5 resulting in 35.5M sentence-pairs for training. The source and target vocabulary is based on 40K BPE types. Results are reported on both newstest2014 and a validation set held-out from the training data comprising 26,658 sentence pairs.

We modify the *fairseq-py* toolkit to implement the objectives described in §3.[6] Our translation models have four convolutional encoder layers and three convolutional decoder layers with a kernel width of 3 and 256 dimensional hidden states and word embeddings. We optimize these models using Nesterov's accelerated gradient method (Sutskever et al., 2013) with a learning rate of 0.25 and momentum of 0.99. Gradient vectors are renormalized to norm 0.1 (Pascanu et al., 2013).

We train our baseline token-level models for 200 epochs and then anneal the learning by shrinking it by a factor of 10 after each subsequent epoch until the learning rate falls below 10^{-4}. All sequence-level models are initialized with parameters of a token-level model before annealing. We then train sequence-level models for another 10 to 20 epochs depending on the objective. Our batches contain 8K tokens and we normalize gradients by the number of non-padding tokens per mini-batch. We use weight normalization for all layers except for lookup tables (Salimans and Kingma, 2016). Besides dropout on the embeddings and the decoder output, we also apply dropout to the input of the convolutional blocks at a rate of 0.3 (Srivastava et al., 2014). We tuned the various parameters above and report accuracy on the test set by choosing the best configuration based on the validation set.

We length normalize all scores and probabilities in the sequence-level losses by dividing by the number of tokens in the sequence so that scores are comparable between different lengths. Ad-

---

[4]We can mitigate this issue by regenerating infrequently, i.e., once every b batches but we leave this to future work.

[5]Different to Ranzato et al. (2015) we train on sentences of up to 175 rather than 50 tokens.

[6]https://github.com/pytorch/fairseq/tree/classic_seqlevel.

ditionally, when generating candidate output sequences during training we limit the output sequence length to be less than 200 tokens for efficiency. We generally use 16 candidate sequences per training example, except for the ablations where we use 5 for faster experimental turnaround.

5.2 Abstractive Summarization

For summarization we use the Gigaword corpus as training data (Graff et al., 2003) and pre-process it identically to Rush et al. (2015) resulting in 3.8M training and 190K validation examples. We evaluate on a Gigaword test set of 2,000 pairs identical to the one used by Rush et al. (2015) and report F1 ROUGE similar to prior work. Our results are in terms of three variants of ROUGE (Lin, 2004), namely, ROUGE-1 (RG-1, unigrams), ROUGE-2 (RG-2, bigrams), and ROUGE-L (RG-L, longest-common substring). Similar to Ayana et al. (2016) we use a source and target vocabulary of 30k words. Our models for this task have 12 layers in the encoder and decoder each with 256 hidden units and kernel width 3. We train on batches of 8,000 tokens with a learning rate of 0.25 for 20 epochs and then anneal as in §5.1.

6 Results

6.1 Comparison of Sequence Level Losses

First, we compare all objectives based on a weighted combination with token-level label smoothing (Equation 8). We also show the likelihood baseline (MLE) of Wiseman and Rush (2016), their beam search optimization method (BSO), the actor critic result of Bahdanau et al. (2016) as well as the best reported result on this dataset to date by Huang et al. (2017). We show a like-for-like comparison to Wiseman and Rush (2016) with a similar baseline model below (§6.6).

Table 1 shows that all sequence-level losses outperform token-level losses. Our baseline token-level results are several points above other figures in the literature and we further improve these results by up to 0.61 BLEU with Risk training.

6.2 Combination with Token-Level Loss

Next, we compare various strategies to combine sequence-level and token-level objectives (cf. §3.3). For these experiments we use 5 candidate sequences per training example for faster experimental turnaround. We consider Risk as

	test	std
MLE (W & R, 2016) [T]	24.03	
BSO (W & R, 2016) [S]	26.36	
Actor-critic (B, 2016) [S]	28.53	
Huang et al. (2017) [T]	28.96	
Huang et al. (2017) (+LM) [T]	29.16	
TokNLL [T]	31.78	0.07
TokLS [T]	32.23	0.10
SeqNLL [S]	32.68	0.09
Risk [S]	32.84	0.08
MaxMargin [S]	32.55	0.09
MultiMargin [S]	32.59	0.07
SoftmaxMargin [S]	32.71	0.07

Table 1: Test accuracy in terms of BLEU on IWSLT'14 German-English translation with various loss functions cf. Figure 1. W & R (2016) refers to Wiseman and Rush (2016), B (2016) to Bahdanau et al. (2016), [S] indicates sequence level-training and [T] token-level training. We report averages and standard deviations over five runs with different random initialization.

	valid	test
TokLS	33.11	32.21
Risk only	33.55	32.45
Weighted	33.91	32.85
Constrained	33.77	32.79
Random	33.70	32.61

Table 2: Validation and test BLEU for loss combination strategies. We either use token-level TokLS and sequence-level Risk individually or combine them as a weighted combination, a constrained combination, a random choice for each sample, cf. §3.3.

sequence-level loss and label smoothing as token-level loss. Table 2 shows that combined objectives perform better than pure Risk. The weighted combination (Equation 8) with $\alpha = 0.3$ performs best, outperforming constrained combination (Equation 9). We also compare to randomly choosing between token-level and sequence-level updates and find it underperforms the more principled constrained strategy. In the remaining experiments we use the weighted strategy.

6.3 Effect of initialization

So far we initialized sequence-level models with parameters from a token-level model trained with label smoothing. Table 3 shows that initializing weighted Risk with token-level label smoothing

	valid	test
`TokNLL`	32.96	31.74
`Risk` init with `TokNLL`	33.27	32.07
Δ	+0.31	+0.33
`TokLS`	33.11	32.21
`Risk` init with `TokLS`	33.91	32.85
Δ	+0.8	+0.64

Table 3: Effect of initializing sequence-level training (`Risk`) with parameters from token-level likelihood (`TokNLL`) or label smoothing (`TokLS`).

	valid	test
Online generation	33.91	32.85
Offline generation	33.52	32.44

Table 4: Generating candidates online or offline.

achieves 0.7-0.8 better BLEU compared to initializing with parameters from token-level likelihood. The improvement of initializing with `TokNLL` is only 0.3 BLEU with respect to the `TokNLL` baseline, whereas, the improvement from initializing with `TokLS` is 0.6-0.8 BLEU. We believe that the regularization provided by label smoothing leads to models with less sharp distributions that are a better starting point for sequence-level training.

6.4 Online vs. Offline Candidate Generation

Next, we consider the question if refreshing the candidate subset at every training step (online) results in better accuracy compared to generating candidates before training and keeping the set static throughout training (offline). Table 4 shows that offline generation gives lower accuracy. However the online setting is much slower, since regenerating the candidate set requires incremental (left to right) inference with our model which is very slow compared to efficient forward/backward over large batches of pre-generated hypothesis. In our setting, offline generation has 26 times higher throughput than the online generation setting, despite the high inference speed of fairseq (Gehring et al., 2017b).

6.5 Beam Search vs. Sampling and Candidate Set Size

So far we generated candidates with beam search, however, we may also sample to obtain a more diverse set of candidates (Shen et al., 2016). Fig-
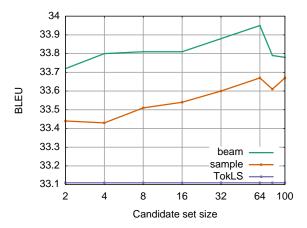

Figure 2: Candidate set generation with beam search and sampling for various candidate set sizes during sequence-level training in terms of validation accuracy. Token-level label smoothing (TokLS) is the baseline.

	BLEU	Δ
MLE	24.03	
+ BSO	26.36	+2.33
MLE Reimplementation	23.93	
+ `Risk`	26.68	+2.75

Table 5: Comparison to Beam Search Optimization. We report the best likelihood (MLE) and BSO results from Wiseman and Rush (2016), as well as results from our MLE reimplementation and training with `Risk`. Results based on unnormalized beam search ($k = 5$).

ure 2 compares beam search and sampling for various candidate set sizes on the validation set. Beam search performs better for all candidate set sizes considered. In other experiments, we rely on a candidate set size of 16 which strikes a good balance between efficiency and accuracy.

6.6 Comparison to Beam-Search Optimization

Next, we compare classical sequence-level training to the recently proposed Beam Search Optimization (Wiseman and Rush, 2016). To enable a fair comparison, we re-implement their baseline, a single layer LSTM encoder/decoder model with 256-dimensional hidden layers and word embeddings as well as attention and input feeding (Luong et al., 2015). This baseline is trained with Adagrad (Duchi et al., 2011) using a learning rate of 0.05 for five epochs, with batches of 64 sequences. For sequence-level training we initialize weights with the baseline parameters and train

	RG-1	RG-2	RG-L
ABS+ [T]	29.78	11.89	26.97
RNN MLE [T]	32.67	15.23	30.56
RNN MRT [S]	36.54	16.59	33.44
WFE [T]	36.30	17.31	33.88
SEASS [T]	36.15	17.54	33.63
DRGD [T]	36.27	17.57	33.62
TokLS	36.53	18.10	33.93
+ Risk RG-1	36.96	17.61	34.18
+ Risk RG-2	36.65	18.32	34.07
+ Risk RG-L	36.70	17.88	34.29

Table 6: Accuracy on Gigaword abstractive summarization in terms of F-measure Rouge-1 (RG-1), Rouge-2 (RG-2), and Rouge-L (RG-L) for token-level label smoothing, and Risk optimization of all three ROUGE F1 metrics. [T] indicates a token-level objective and [S] indicates a sequence level objectives. ABS+ refers to Rush et al. (2015), RNN MLE/MRT (Ayana et al., 2016), WFE (Suzuki and Nagata, 2017), SEASS (Zhou et al., 2017), DRGD (Li et al., 2017).

	valid	test
TokLS	34.06	40.58
+ Risk	34.20	40.95
TokLS + selfatt	34.24	41.02
+ in domain	34.51	41.26
+ Risk	34.30	41.22
+ Risk in domain	34.50	41.47

Table 7: Test and valid BLEU on WMT'14 English-French with and without decoder self-attention.

and then switch to sequence-level training for another epoch. Table 7 shows that sequence-level training can improve an already very strong model by another +0.37 BLEU. Next, we improve the baseline by adding *self-attention* (Paulus et al., 2017; Vaswani et al., 2017) to the decoder network (TokLS + selfatt) which results in a smaller gain of +0.2 BLEU by Risk. If we train Risk only on the news-commentary portion of the training data, then we achieve state of the art accuracy on this dataset of 41.5 BLEU (Xia et al., 2017).

6.8 Abstractive Summarization

Our final experiment evaluates sequence-level training on Gigaword headline summarization. There has been much prior art on this dataset originally introduced by Rush et al. (2015) who experiment with a feed-forward network (ABS+). Ayana et al. (2016) report a likelihood baseline (RNN MLE) and also experiment with risk training (RNN MRT). Different to their setup we did not find a softmax temperature to be beneficial, and we use beam search instead of sampling to obtain the candidate set (cf. §6.5). Suzuki and Nagata (2017) improve over an MLE RNN baseline by limiting generation of repeated phrases. Zhou et al. (2017) also consider an MLE RNN baseline and add an additional gating mechanism for the encoder. Li et al. (2017) equip the decoder of a similar network with additional latent variables to accommodate the uncertainty of this task.

Table 6 shows that our baseline (TokLS) outperforms all prior approaches in terms of ROUGE-2 and ROUGE-L and it is on par to the best previous result for ROUGE-1. We optimize all three ROUGE metrics separately and find that Risk can further improve our strong baseline. We also compared Risk only training to Weighted on this dataset (cf. §6.2) but accuracy was generally lower on the validation set: RG-1

with Adam (Kingma and Ba, 2014) for another 10 epochs with learning rate 0.00003 and 16 candidate sequences per training example. We conduct experiments with Risk since it performed best in trial experiments.

Different from other sequence-level experiments (§5), we rescale the BLEU scores in each candidate set by the difference between the maximum and minimum scores of each sentence. This avoids short sentences dominating the sequence updates, since candidate sets for short sentences have a wider range of BLEU scores compared to longer sentences; a similar rescaling was used by Bahdanau et al. (2016).

Table 5 shows the results from Wiseman and Rush (2016) for their token-level likelihood baseline (MLE), best beam search optimization results (BSO), as well as our reimplemented baseline. Risk significantly improves BLEU compared to our baseline at +2.75 BLEU, which is slightly better than the +2.33 BLEU improvement reported for Beam Search Optimization (cf. Wiseman and Rush (2016)). This shows that classical objectives for structured prediction are still very competitive.

6.7 WMT'14 English-French results

Next, we experiment on the much larger WMT'14 English-French task using the same model setup as Gehring et al. (2017b). We TokLSfor 15 epochs

(36.59 `Risk` only vs. 36.67 `Weighted`), RG-2 (17.34 vs. 18.05), and RG-L (33.66 vs. 33.98).

7 Conclusion

We present a comprehensive comparison of classical losses for structured prediction and apply them to a strong neural sequence to sequence model. We found that combining sequence-level and token-level losses is necessary to perform best, and so is training on candidates decoded with the current model.

We show that sequence-level training improves state-of-the-art baselines both for IWSLT'14 German-English translation and Gigaword abstractive sentence summarization. Structured prediction losses are very competitive to recent work on reinforcement or beam optimization. Classical expected risk can slightly outperform beam search optimization (Wiseman and Rush, 2016) in a like-for-like setup. Future work may investigate better use of already generated candidates since invoking generation for each batch slows down training by a large factor, e.g., mixing with fresh and older candidates inspired by MERT (Och, 2003).

References

Ayana, Shiqi Shen, Yu Zhao, Zhiyuan Liu, Maosong Sun, et al. 2016. Neural headline generation with sentence-wise optimization. *arXiv preprint arXiv:1604.01904* .

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. In *arXiv preprint arXiv:1607.07086*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *Proc. of IWSLT*.

Samidh Chatterjee and Nicola Cancedda. 2010. Minimum error rate training by sampling the translation lattice.

Yann N. Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language Modeling with Gated Convolutional Networks. In *Proc. of ICML*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12(Jul):2121–2159.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2017a. A Convolutional Encoder Model for Neural Machine Translation. In *Proc. of ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017b. Convolutional Sequence to Sequence Learning. In *Proc. of ICML*.

Kevin Gimpel and Noah Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proc. of ACL*.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia* .

Spence Green, Daniel Cer, and Christopher Manning. 2014. An Empirical Comparison of Features and Tuning for Phrase-based Machine Translation. In *Proc. of WMT*. Association for Computational Linguistics.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support vector learning for ordinal regression .

Po-Sen Huang, Chong Wang, Dengyong Zhou, and Li Deng. 2017. Neural Phrase-based Machine Translation. In *arXiv preprint arXiv:1706.05565*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *Proc. of ICLR* .

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. *arXiv* .

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. of COLING*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of EMNLP*.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. Sapporo, Japan, pages 160–167.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of The 30th International Conference on Machine Learning*. pages 1310–1318.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. In *Proc. of ICLR Workshop*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level Training with Recurrent Neural Networks. In *Proc. of ICLR*.

Antti-Veikko I Rosti, Bing Zhang, Spyros Matsoukas, and Richard Schwartz. 2010. BBN System Description for WMT10 System Combination Task. In *Proc. of WMT*. Association for Computational Linguistics, pages 321–326.

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

Tim Salimans and Diederik P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *arXiv preprint arXiv:1602.07868* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proc. of 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 *Proc. of ACL*.

David A. Smith and Jason Eisner. 2006. Minimum Risk Annealing for Training Log-Linear Models. In *Proc. of ACL*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent Neural Networks from overfitting. *JMLR* 15:1929–1958.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proc. of NIPS*. pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. *arXiv preprint arXiv:1701.00138* .

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. Rethinking the inception architecture for computer vision. *arXiv* .

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *NIPS*.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research* 6:1453—-1484.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv* .

R. J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229—-256.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proc. of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv preprint arXiv:1609.08144* .

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proc. of NIPS*.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. *arXiv* .