TeaForN: Teacher-Forcing with N-grams

Sebastian Goodman
Google Research
Venice, CA 90291
seabass@google.com

Nan Ding
Google Research
Venice, CA 90291
dingnan@google.com

Radu Soricut
Google Research
Venice, CA 90291
rsoricut@google.com

Abstract

Sequence generation models trained with teacher-forcing suffer from issues related to exposure bias and lack of differentiability across timesteps. Our proposed method, Teacher-Forcing with N-grams (TeaForN), addresses both these problems directly, through the use of a stack of N decoders trained to decode along a secondary time axis that allows model-parameter updates based on N prediction steps. TeaForN can be used with a wide class of decoder architectures and requires minimal modifications from a standard teacher-forcing setup. Empirically, we show that TeaForN boosts generation quality on one Machine Translation benchmark, WMT 2014 English-French, and two News Summarization benchmarks, CNN/Dailymail and Gigaword.

1 Introduction

Many state-of-the-art sequence generation models are trained using a technique called teacher-forcing (Goodfellow et al., 2016). Teacher-forcing is popular because it improves sample efficiency and provides training stability, but models trained with teacher-forcing are known to suffer from issues such as exposure bias (Venkatraman et al., 2015; Bengio et al., 2015; Ding and Soricut, 2017) and a lack of differentiability across timesteps (i.e., training updates made when decoding at time-step t cannot fully propagate to time-step $t-1$). Previous attempts to address these issues include scheduled sampling (Bengio et al., 2015), parallel N-gram prediction (Yan et al., 2020), and sampling from previous predictions (Zhang et al., 2019).

Our proposed method, Teacher-Forcing with N-grams (TeaForN), imposes few requirements on the decoder architecture and does not require curriculum learning or sampling model outputs. TeaForN fully embraces the teacher-forcing paradigm and extends it to N-grams, thereby addressing the problem at the level of teacher-forcing itself.

The advent of large-scale pretraining has pushed the state-of-the-art on Natural Language benchmarks to impressive heights, often showing gains across many tasks at once (Devlin et al., 2019; Raffel et al., 2019; Zhang et al., 2019). A negative consequence of this is the tendency towards large, data-hungry models, which have a negative impact on energy-consumption and accessibility (Strubell et al., 2019), as well as higher latency and production costs. As such, it is of increasing importance to develop techniques that counteract these tendencies. While TeaForN does increase training cost moderately, it can be used to drive down latency and inference cost, which dominate the overall cost of a production model.

Many sequence generation models use beam search to improve generation quality (Vaswani et al., 2017; Raffel et al., 2019; Zhang et al., 2019; Yan et al., 2020). In contrast with greedy decoding, beam search keeps the k most-likely candidates at each decoding timestep. While beam search has proven to be a reliable technique for improving output quality, previous work has shown that beam search actually degrades performance for sufficiently large k (Koehn and Knowles, 2017). In addition, the inference cost of a model increases linearly with k, due to the need for multiple decodings. We conduct an analysis of the effect of beam size on models trained both with and without TeaForN. We show that models trained with TeaForN require a smaller beam size to reach similar performance, a property that can achieve significant cost-savings.

Our experiments show that TeaForN can boost performance on both Machine Translation and News Summarization tasks, provided there is sufficient model capacity. With TeaForN, Transformer $_{\text{big}}$ (Vaswani et al., 2017) improves by +.5 SacreBLEU (Post, 2018) on the WMT14 En-Fr benchmark with beam search and +.3 without. When using TeaForN for summarization, PEGASUS $_{\text{large}}$ (Zhang et al., 2019) improves by +.3 ROUGE-L on the Gigaword benchmark (Rush et al., 2015) and by +.2 on the CNN/Dailymail benchmark (Hermann et al., 2015). Further, PEGASUS $_{\text{large}}$ trained with TeaForN matches the prior ROUGE-L scores on these benchmarks *without beam search*, representing an 8x reduction in decoder inference cost.

2 Related Work

One of the standard approaches to sequence-learning training is Maximum-likelihood Estimation (MLE). Although widely used in large array of applications, MLE estimation for sequence learning suffers from the exposure-bias problem (Venkatraman et al., 2015; Ranzato et al., 2015). Exposure-bias produces brittle models due to training procedures during which the models are only exposed to their training data distribution but not to their own predictions. Possible solutions to the exposure-bias problem in neural-network settings have used “data as demonstrator” (Venkatraman et al., 2015) and “scheduled sampling” (Bengio et al., 2015) approaches. Although improving model performance in practice, such proposals have been shown to be statistically inconsistent (Huszar, 2015), and still need to perform MLE-based warm-start training, rendering such solutions unsatisfactory. Along similar lines, the “professor forcing” (Lamb et al., 2016) method uses adversarial domain adaptation to encourage network dynamics to be the same during training and inference, though it requires sampling sequences during training.

A different approach, based on reinforcement learning methods, achieves sequence learning following a policy-gradient (PG) method (Sutton et al., 1999). It directly attacks the exposure-bias problem by having the training models exposed exclusively to their own predictions while scoring them using reward functions. However, this approach introduces another issue, related to the large discrepancy between the model prediction distribution and the reward function’s values, which is especially acute during the early training stages when the predicted outputs are all equally bad. As a result, this method also requires a warm-start phase in which the model distribution achieves some local maximum with respect to a reward–free objective (e.g., MLE), followed by a model refinement phase in which reward-based PG updates are used to refine the model (Ranzato et al., 2015; Wu et al., 2016; Liu et al., 2017). Although such combinations achieve better results in practice compared to pure likelihood-based approaches, they are unsatisfactory from a theoretical and modeling perspective, as well as inefficient from a speed-to-convergence perspective. A pure PG formulation that side-steps these issues is (Ding and Soricut, 2017), which allows for both cold-start training as well as more efficient convergence properties.

The PG-based approaches have an inherent complexity that stems from the use of quirky reward functions such as ROUGE (Lin, 2004) or CIDEr (Vedantam et al., 2015), which forfeits the advantage of sample efficiency as they often cannot be efficiently computed using current accelerators like TPUs (You et al., 2019). MLE-based approaches appear to be favored due to efficiency properties, and the search for training methods that produce less brittle models is still on-going.

Another closely related idea is End-to-End Backprop (E2E) (Ranzato et al., 2015), which has a similar goal of naturally approximating sequence level training by propagating smooth model predictions instead of groundtruth inputs. TeaForN differs from E2E in several key ways. First, TeaForN learns jointly from both groundtruth and model predictions as inputs throughout the entire training duration, whereas E2E requires a training schedule to transition from groundtruths to model predictions. Second, TeaForN supports methods other than k-max for computing smooth model predictions, two of which we explore as a part of our work. Third, we introduce the notion of a discount factor, which weights the importance of immediate predictions higher than that of future predictions.

Another such work is (Yan et al., 2020), which proposes a modified Transformer for parallel N-gram prediction. While their work does address the issue of strong local correlations caused by teacher-forcing, it does not address exposure bias, as it always trains on groundtruth inputs.

Also related are models such as the one proposed by (Strubell et al., 2017), which uses a stacked of dilated convolutions to iteratively refine model predictions. Though architecturally similar, TeaForN only uses the stack at training time and solves for a fundamentally different problem.

Our TeaForN method maintains the efficiency advantages of MLE-based approaches, while ad-

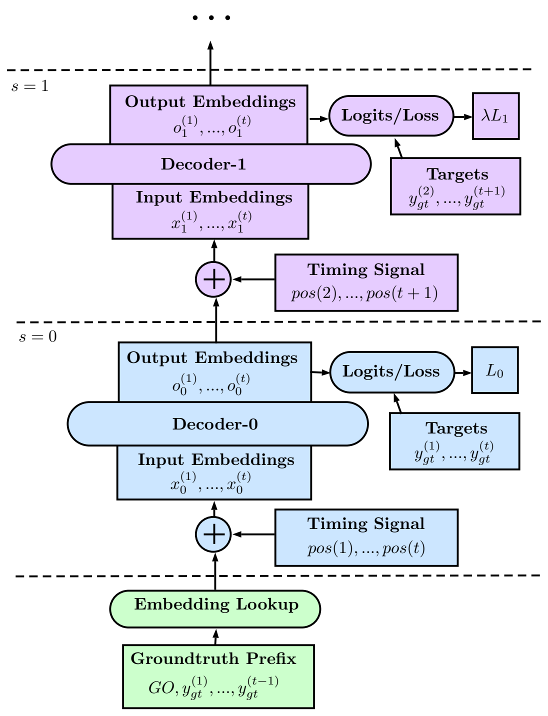

Figure 1: An illustration of TeaForN training, wherein each decoder after the first uses the outputs of the previous decoder as inputs. Decoder weights may be shared across layers in order to address exposure bias.

dressing both exposure bias and the issue of differentiability across timesteps. In addition, it is general enough to be used on a wide class of autoregressive decoders, including RNN (Hochreiter and Schmidhuber, 1997; Chung et al., 2014) and Transformer (Vaswani et al., 2017) decoders, though our experiments focus on the Transformer.

3 TeaForN

Autoregressive sequence decoders are trained to minimize the negative log likelihood of the groundtruth tokens $y_{gt}^{(t)}$. During training, previous *groundtruth tokens* are used as decoder inputs for predicting the next token. If we define the embedding matrix to be E of size $V \times D$, where V is the vocabulary size and D is the embedding size, and the embedding of the groundtruth token as $x^{(t)} = E[y_{gt}^{(t-1)}, :] := e_{gt}^{(t-1)}$ of size D, then the standard teacher-forcing loss is equal to,

$$\begin{aligned} L &= -\sum_{t=1}^{T} \log(P(y_{gt}^{(t)} | y_{gt}^{(0)}, ..., y_{gt}^{(t-1)})) \\ &= -\sum_{t=1}^{T} \log(P(y_{gt}^{(t)} | x^{(1)}, ..., x^{(t)})) \end{aligned}$$

where we define $y_{gt}^{(0)} = GO$ as the starting token.

The class probability distribution P is typically modeled as softmax-normalized logits, which are a linear projection of decoder output $o^{(t)}$ of size D onto the class embeddings using output projection matrix W of size $V \times D$:

$$P(y^{(t)} | x^{(1)}, ..., x^{(t)}) = softmax(W o^{(t)}).$$

To reduce the model parameter size, it is standard to share the parameters of the output projection matrix and the embedding matrix, such that $E = W$.

During inference, groundtruth tokens become unavailable. Therefore, previous tokens from the *model predictions* are used as decoder inputs for decoding the next token. The discrepancy between training time and inference time input distributions causes models to suffer from exposure bias, meaning that they do not learn to correct for past decoding errors (Bengio et al., 2015).

TeaForN addresses exposure bias by learning jointly how to predict from both groundtruth and past model predictions as inputs. TeaForN setups consist of a stack of N decoders, as illustrated in Figure 1. At position t, the first decoder (Decoder-0) takes input from the embedding of the previous groundtruth token $x_0^{(t)} = e_{gt}^{(t-1)}$ and learns to predict the target token $y_{gt}^{(t)}$, same as in teacher-forcing. The next decoder (Decoder-1) takes input from the output $x_1^{(t)} = o_0^{(t)}$ of the first decoder and learns to predict the next target token $y_{gt}^{(t+1)}$.

More formally, let us use subscript $s \in [0, N)$ to denote the offset within the decoder stack. We define the input to decoder s at time t as:

$$x_s^{(t)} = pos(t+s) + \begin{cases} e_{gt}^{(t-1)} & s = 0 \\ o_{s-1}^{(t)} & s > 0 \end{cases}$$

where $pos(t+s)$ is a timing signal that is added to the inputs for models such as the Transformer (Vaswani et al., 2017). This term may be omitted for models that do not expect it.

The training loss of Decoder-s at time t is the negative log likelihood of the $(t+s)^{th}$ element in the groundtruth sequence:

$$L_s = -\sum_{t=1}^{T} \log(P(y_{gt}^{(t+s)} | x_s^{(1)}, ..., x_s^{(t)}; \theta_s))$$

and the total TeaForN training loss is the sum of decoder losses

$$L = \sum_{s=0}^{N-1} \lambda^{s-1} L_s$$

where $\lambda \in (0,1]$ is a discount factor needed to weigh the risk of harming next-word accuracy against the benefits of TeaForN. During inference, TeaForN uses only the first decoder (Decoder-0) in the stack; the rest are discarded.

The intuition behind TeaForN is as follows. Under standard teacher-forcing, the decoder output $o^{(t)}$ only learns to predict the groundtruth label $y_{gt}^{(t)}$, while outputs that favor other classes are considered equally bad, and will be penalized by the loss. This is not reasonable because classes carrying similar meanings to the groundtruth label do not change the meaning of the sequence significantly, and may still lead to the correct prediction for the next label. Under TeaForN, the decoder output $o^{(t)}$ is also used as the input of a secondary decoder for decoding the next position. Therefore, all outputs that result in predicting the next groundtruth label $y_{gt}^{(t+1)}$ will have lower loss and therefore be differentiated from other outputs.

In our experiments, we allow the decoder parameters to be either shared ($\theta_0 = \theta_s, \forall s$) or unshared. In a shared-weight configuration, the model learns to predict the next groundtruth label from the class that the same model predicted in the previous position. This is similar to the inference time condition, so we expect shared-weight TeaForN to address exposure-bias better than unshared-weight TeaForN. Shared-weight configurations also have performance advantages such as lower memory consumption and faster training.

Since TeaForN solves for a more difficult problem than teacher-forcing, we expect it to work better for models with higher capacity. We later show evidence of this by comparing results for two model sizes on Machine Translation.

It is straightforward to show that TeaFor1 (N=1) and teacher-forcing are equivalent, as the inputs to the first TeaForN decoder are groundtruth sequence embeddings and $\lambda^0 = 1$. Thus, TeaForN is a natural extension of teacher-forcing to N-grams.

3.1 Embedded Top-k Stacked Decoder Input

Previously, our TeaForN model directly used the decoder output of the $(s-1)$-th stack as the input of the decoder of the s-th stack:

$$x_s^{(t)} = o_{s-1}^{(t)}. \tag{1}$$

This is an approximation to the inference-time decoder input, which (for greedy decoding) is

$$x_s^{(t)} = E[argmax(Wo_{s-1}^{(t)}), :], \tag{2}$$

where $argmax(x)$ returns the index of the V-dim vector with the maximum value.

Inspired by the End-to-End Backprop (E2E) (Ranzato et al., 2015), we also consider the following alternative decoder input,

$$x_s^{(t)} = E^\top softmax(top\_k(Wo_{s-1}^{(t)})), \tag{3}$$

where top_k is a function which keeps the top-k values of the vector, and masks out the others.

It is easy to verify, when $k = 1$, Eq. (3) reduces to Eq. (2); when $k = V$,

$$x_s^{(t)} = E^\top softmax(Wo_{s-1}^{(t)}).$$

Compared to Eq. (1), Eq. (3) is more computationally expensive, as it involves additional embedding matrix multiplications and/or a top-k sorting. Furthermore, we would like to emphasize a critical difference between the TeaForN and E2E (Ranzato et al., 2015). In TeaForN, the 0-th stack of every position is always clamped to the groundtruth input, while for E2E the groundtruth is completely thrown away after warm-up training. The groundtruth clamping allows the TeaForN to avoid the warm-up training which is necessary for E2E.

4 Experimental Results

Our empirical study of TeaForN is comprised of two sections. First, we present experiments on Machine Translation using the well-known Transformer model (Vaswani et al., 2017). Second, we show results for News Summarization, for which we use PEGASUS (Zhang et al., 2019), a state-of-the-art pretrained text summarization model.

We perform minimal hyperparameter tuning over the course of these experiments. This can be partly credited to the underlying models being well-tuned already, but also to TeaForN, which works out-of-the-box without much hyperparameter tuning. One exception is the tuning of the number of training steps, as we found that the number of steps used by previous settings is sometimes insufficient.

4.1 Machine Translation

In this section, we study the effects of applying TeaForN to a well-known Transformer-based Machine Translation model. We present results for two size variants of the model, Transformer$_{\text{base}}$ and Transformer$_{\text{big}}$ (Vaswani et al., 2017). The differences are summarized in Table 2.

We use the same WMT14 language-pair benchmarks originally reported in the Transformer paper:

	θ_{shared}	Greedy decoding		Beam search@k=4	
		Teacher-forcing	TeaFor2	Teacher-forcing	TeaFor2
En-De	N	$26.96 \pm .04$	$27.02 \pm .06$	$27.96 \pm .09$	$27.88 \pm .05$
	Y		**27.16** $\pm .02$		$27.90 \pm .03$
En-Fr	N	$40.20 \pm .04$	**40.32** $\pm .08$	$40.86 \pm .08$	$40.88 \pm .10$
	Y		**40.32** $\pm .04$		$40.84 \pm .07$

Table 1: A comparison of models on WMT14 language pairs En-De and En-Fr using Transformer$_{\text{base}}$. We report mean and Standard Error of SacreBLEU scores over five independent training runs. θ_{shared} refers to whether the free parameters of the decoder are shared across decoder instances (Y) or kept separate (N). The discount factor is $\lambda = .5$ for TeaFor2 models.

- **English-German (En-De)**, with 4.5M sentence pairs for training and 2,737 for testing.
- **English-French (En-Fr)**, with 36M sentence pairs for training and 3,003 for testing.

We use *SacreBLEU* (Post, 2018) with case-sensitive tokenization to score translations. We report SacreBLEU scores for beam search widths $k \in [1, 8]$ to show the interaction between TeaForN learning and beam search.

4.1.1 Transformer$_{\text{base}}$

Using Transformer$_{\text{base}}$ as our underlying model, we measure the impact of TeaForN on the Machine Translation task. We test both shared- and unshared-weight configurations, with $N = 2$ (i.e. "TeaFor2") and $\lambda = .5$. We expect weight-shared configurations to be more effective, as a more direct means of addressing exposure bias in the decoder.

All models are trained for 1M steps, and we observe no signs of overfitting. For model selection, we average the last five checkpoints, as originally done for the Transformer (Vaswani et al., 2017). We report mean and standard-error variation of SacreBLEU scores over five runs.

	Transformer$_{\text{base}}$	Transformer$_{\text{big}}$
P_{drop}	.1	.3
d_{model}	512	1024
d_{ff}	2048	4096
h	8	16

Table 2: A summary of differences between model variants Transformer$_{\text{base}}$ and Transformer$_{\text{big}}$. P_{drop} refers to dropout probability, d_{model} refers to class embedding size and hidden size, d_{ff} refers to the size of feedforward layers, and h refers to the number of self-attention heads (Vaswani et al., 2017).

Table 1 shows that TeaFor2 improves the quality of greedy decoding on both language pairs. TeaFor2 raises SacreBLEU scores by +.20 on En-De (27.16 vs 26.96) and +.12 on En-Fr (40.32 vs 40.20). Shared-weight TeaFor2 boosts performance on the En-De benchmark by +.14 SacreBLEU (27.16 vs 27.02). The small size of the En-De training set (relative to En-Fr), means that the En-De model has additional capacity for learning the TeaFor2 task. This supports our case that TeaForN with weight-sharing improves model performance, but only if there is sufficient model capacity. Table 1 also shows that TeaFor2, with or without weight-sharing for En-Fr, outperforms standard teacher-forcing by the same amount, +.12 SacreBLEU (40.32 vs 40.20). We credit the increase in performance to TeaForN's ability to make predictions that lead to better predictions in the subsequent sequence positions.

Beam search results in Table 1 show that the gains of TeaFor2 are negated by beam search with $k = 4$. Because of the small capacity of the Transformer$_{\text{base}}$ model, the benefits of TeaFor2 are minimal and only reflected in the result from greedy decoding. In the following experiment, we show that higher-capacity models benefit more from TeaForN when using beam search.

4.1.2 Transformer$_{\text{big}}$

Using Transformer$_{\text{big}}$, we now compare standard teacher-forcing against TeaForN (N=2,3).

We test against the same WMT14 language pairs as the previous experiments. We train En-Fr models for 1M steps and En-De models for 500k steps. Beyond 500k training steps, we observe that En-De models overfit the training data (see Table 4). This is likely due to a combination of larger model capacity in Transformer$_{\text{big}}$ (Table 2) and smaller training set for En-De. For model selection, we

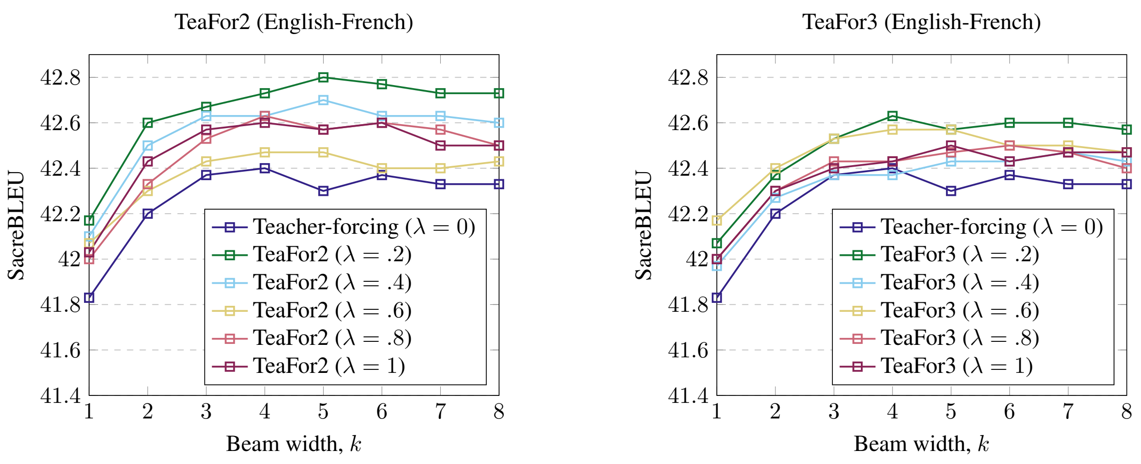

Figure 2: Beam width vs. SacreBLEU on the WMT 2014 English-French benchmark, for discount factors $\lambda \in \{0, .2, .4, .6, .8, 1\}$. Reported scores are averages over n=3 independent training runs, with error bars omitted for readability. See Appendix Tables 7 and 9 for results with standard error measurements.

average the last twenty checkpoints, as was done for the Transformer (Vaswani et al., 2017).

We use weight-sharing for all TeaForN setups in this section. Transformer $_{\text{big}}$ has more capacity than Transformer $_{\text{base}}$, so it is expected to perform better in a shared-weight configuration.

Figure 2 shows that TeaForN outperforms standard-teacher forcing on the En-Fr benchmark, across all beam widths up to $k = 8$ and all discount factors $\lambda \in \{.2, .4, .6, .8, 1\}$. With beam 2, TeaFor2 achieves a higher score on En-Fr than teacher-forcing achieves with any beam size up to 8 (42.6 vs 42.4) and significantly outperforms it with beam size 5 (42.8 vs 42.4). TeaFor3 performs as well as Teacher-forcing but worse than TeaFor2

	En-De	En-Fr
(Ott et al., 2018)	28.6	41.4
(So et al., 2019)	29.2	-
Transformer$_{\text{big}}$	29.20 ± 0.12	42.33 ± 0.07
TeaFor2	29.30 ± 0.05	$\mathbf{42.73} \pm 0.05$
TeaFor3	29.23 ± 0.05	42.43 ± 0.03

Table 3: A comparison of SacreBLEU scores of Machine Translation models on WMT14 En-De and En-Fr benchmarks. Results for our models are shown below the horizontal line. We report mean and standard error over n=3 independent training runs. We set beam width $k = 8$ for all models; we tune λ against the validation set (selected values of λ are .4, .2, .4, and .4 left-to-right and top-to-bottom). See Appendix for test and validation scores with standard error measurements.

on the En-Fr benchmark, for nearly every discount factor tested. This shows that TeaForN can be used to train models with higher quality for any given beam size or, alternatively, train models of similar quality but lower inference cost (i.e., faster).

In contrast with the results for lower-capacity models, Fig. 2 shows that beam search does not erase gains due to TeaForN training. The Teacher-forcing setup gains +.6 SacreBLEU from beam search (42.4 vs 41.8) compared to +.6 for TeaFor2 (42.8 vs 42.2), in spite of a +.4 SacreBLEU higher baseline. Provided sufficient model capacity, TeaForN is seen to improve the quality of the underlying model, so that greedy decoding is more effective, but not at the expense of beam search.

Intuitively, discount factors that are too high may interfere with prediction quality, as they decrease the relative importance of next word prediction. We see this on the English-German benchmark, shown in Fig. 3, where the highest discount factor tested ($\lambda = 1$) significantly reduces greedy performance (27.9/27.8 from 28.1) and peak performance (28.9/28.8 vs 29.2). In all of our Transformer$_{\text{big}}$ experiments, the best performing discount factor is either .2 or .4, which are the lowest values tested.

Table 3 shows our results compared to current state-of-the-art Machine Translation models. To allow for a fair comparison, we select our discount factor $\lambda \in \{.2, .4, .6, .8, 1\}$ to maximize performance against a development set, WMT12. On the En-De benchmark, TeaForN setups perform similarly to Teacher-forcing, at 29.0 SacreBLEU.

On En-Fr, TeaFor2 outperforms Teacher-forcing significantly, by +.4 SacreBLEU (42.7 vs 42.3).

4.1.3 Top-K Approximation

Up to this point, TeaForN setups have used Eq. (1) to approximate the inference-time decoder input.

We now share results for an alternative approximation called Top-K, described by Eq. (3) and inspired by (Ranzato et al., 2015), which feeds the embedding expectation of the decoder output. If $K = V$, Top-K is an exact expectation. If $K < V$, Top-K approximates the expectation as the probability-weighted embeddings of the K most likely outputs.

In this experiment, we try $K \in \{4, V\}$ and $N \in \{2, 3\}$ using Transformer$_{\text{big}}$ as our base model. We report results on both WMT14 benchmarks. We use discount factor $\lambda = .2$ for all setups.

Figure 4 shows that Top-K does not work as well as the original TeaForN approximation described by Eq. (1). Top-K with $K = 4$ performs worse than TeaForN on the En-De benchmark but not the En-Fr benchmark. When $K = V$, the situation is the exact opposite, with Top-K performing better on the En-De benchmark but not the En-Fr benchmark.

4.1.4 Word Drop Regularization

TeaForN could potentially have regularization-like effects by solving for a more difficult task than standard teacher-forcing. TeaForN trains models to decode not just from groundtruth prefixes, but also from past model predictions.

To see whether regularization-like effects are responsible for the gains seen using TeaForN, we perform a regularization experiment using Transformer$_{\text{big}}$. In particular, we randomly sample a set of groundtruth decoder input words in each example with probability $P_{drop} \in \{0, .1, .2, .3\}$. For each selected word, we apply the $word\_drop$ regularization by masking all its embedding elements to zero.

Fig. 5 shows that word drop regularization increases performance against the En-De benchmark but reduces performance against En-Fr. These results are in stark contrast with the results of TeaForN, which only improves performance in the En-Fr case. Though TeaForN may have regularization-like effects, they are likely different from the effects of word drop regularization.

		1x iterations	1.5x iterations
En-De	greedy	$28.10 \pm .12$	$27.97 \pm .20$
	beam	$29.30 \pm .20$	$29.20 \pm .16$
En-Fr	greedy	$41.83 \pm .08$	$41.90 \pm .12$
	beam	$42.40 \pm .04$	$42.40 \pm .12$

Table 4: SacreBLEU scores of Transformer$_{\text{big}}$ on WMT14 En-De and En-Fr benchmarks. We report mean and standard error over n=3 independent training runs.

4.1.5 Additional Compute

TeaForN uses more compute resources than Teacher-forcing when inference-time architecture and number of training iterations are the same, as is the case in our Transformer$_{\text{big}}$ experiments.

To enable a fair comparison in terms of training-time compute, we conduct an experiment where we train Transformer$_{\text{big}}$ so that the total device time is about the same. We train the baseline for 1.5x iterations, a figure which was estimated from the observed training speeds of Transformer$_{\text{big}}$ and TeaFor2 (4.5 iterations/sec and 6.8 iterations/sec).

Table 4 shows that this additional training does not significantly benefit Transformer$_{\text{big}}$, for either language pair. Based on these results, we conclude that the benefits of TeaForN do not likely derive from additional compute.

4.2 News Summarization

We now present our experiment on News Summarization using PEGASUS$_{\text{large}}$ (Zhang et al., 2019) as our base model.

We test on two News Summarization tasks, CNN/Dailymail and Gigaword:

- **CNN/Dailymail** (Hermann et al., 2015) consists of 93k CNN articles and 220k Daily Mail articles, where publishers provide bullet-style summaries with each article.

- **Gigaword** (Rush et al., 2015) contains 4M articles from seven publishers, where article headlines serve as the summary.

The PEGASUS approach has been shown to work better on News Summarization tasks when pretrained on HugeNews, a dataset of 1.5B news-like articles scraped from the web between 2013 and 2019. We use the same pretraining procedure as originally described for PEGASUS$_{\text{large}}$ (HugeNews), which uses teacher-forcing to learn based on an unsupervised Gap Sentence Generation task (Zhang et al., 2019).

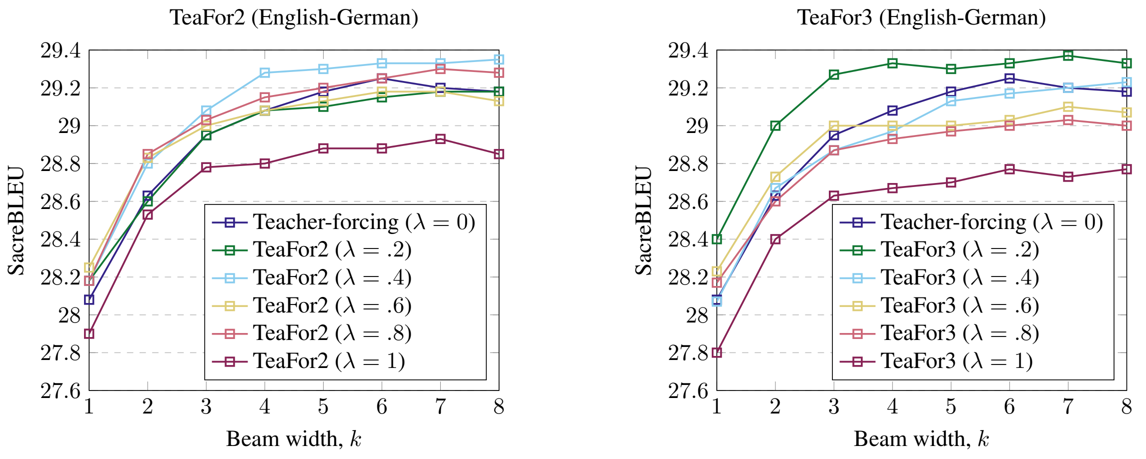

Figure 3: Beam width vs. SacreBLEU on the WMT 2014 English-German benchmark, for discount factors $\lambda \in \{0, .2, .4, .6, .8, 1\}$. Left and right plots show TeaFor2 and TeaFor3, respectively. Reported scores are averages over n=3 independent training runs, with error bars omitted for readability. See Appendix Tables 11 and 13 for results with standard error measurements.

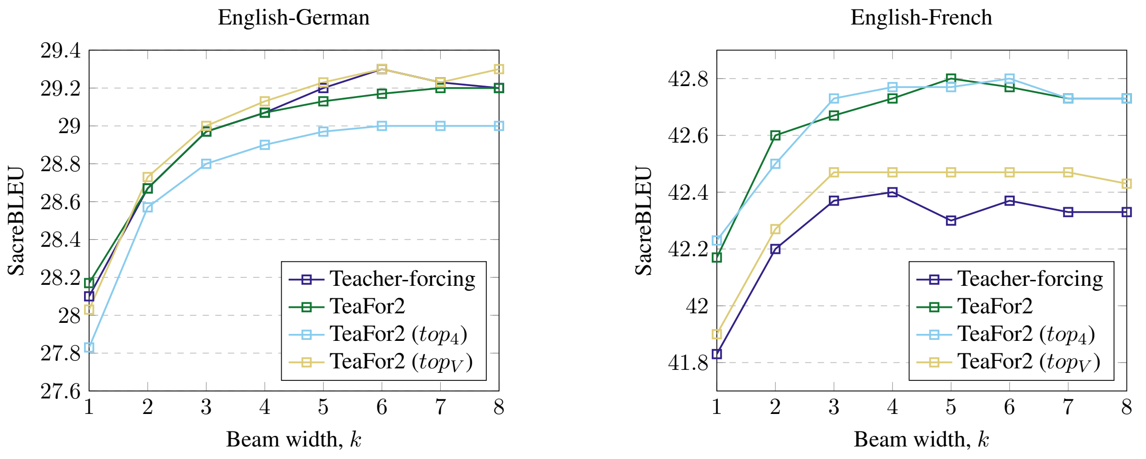

Figure 4: Beam width vs. SacreBLEU on WMT 2014 benchmarks comparing approximation methods Top-K. Reported scores are averages over n=3 independent training runs, with error bars omitted for readability. See Appendix Tables 15 and 16 for results with standard error measurements.

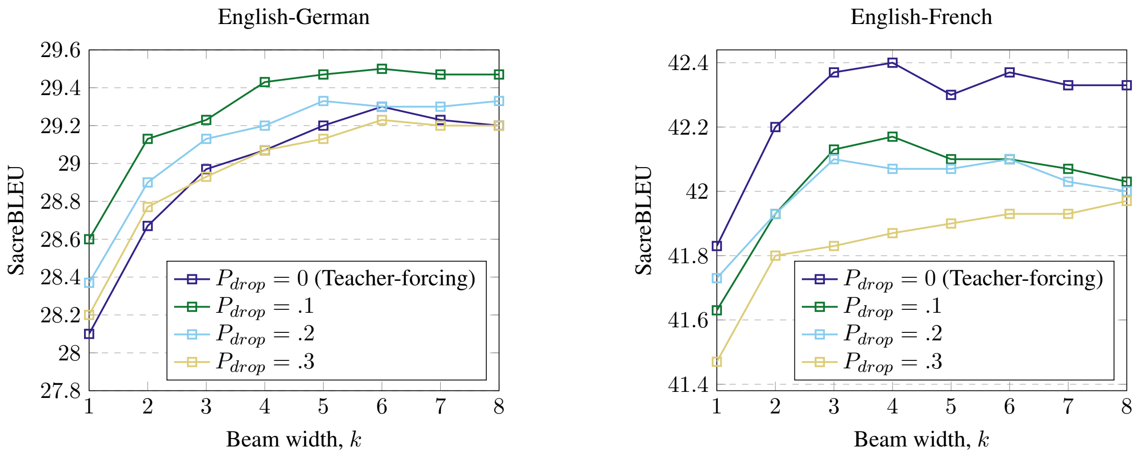

Figure 5: Beam width vs. SacreBLEU on WMT 2014 benchmarks using Word Drop Regularization. Reported scores are averages over n=3 independent training runs. See Appendix Tables 17 and 18 for results with standard error measurements.

R1/R2/RL	CNN/Dailymail	Gigaword
BERTShare (Rothe et al., 2019)	39.25/18.09/36.45	38.13/19.81/35.62
MASS (Song et al., 2019)	42.12/19.50/39.01	38.73/19.71/35.96
UniLM (Dong et al., 2019)	43.33/20.21/40.51	38.45/19.45/35.75
BART (Lewis et al., 2019)	44.16/21.28/40.90	-
T5 (Raffel et al., 2019)	43.52/21.55/40.69	-
PEGASUS (Zhang et al., 2019)	44.17/21.47/41.11	39.12/19.86/36.24
(Greedy) TeaFor3+PEGASUS	43.90/20.36/41.20	39.10/19.40/36.30
(Beam@k=8) TeaFor3+PEGASUS	44.20/**21.70/41.32**	39.16/**20.16/36.54**

Table 5: A comparison of News Summarization models on CNN/Dailymail and Gigaword benchmarks. Scores are ROUGE-1/ROUGE-2/ROUGE-L F-measures. PEGASUS is shorthand for PEGASUS$_{\text{large}}$ (HugeNews) and uses beam width $k = 8$ for both tasks. We use TeaFor3 with $\lambda = .5$ and weight-sharing.

	Decoder Layers	Steps/sec	HBM usage (GB)
Teacher-Forcing	16	1.97	8.64
TeaFor2	32	1.25	8.96
TeaFor3	48	.986	10.33

Table 6: Performance of TeaForN with weight-sharing during PEGASUS$_{\text{large}}$ (HugeNews) pretraining. Steps/sec refers to the number of training batches processed per second. High-Bandwidth Memory usage refers to the consumption of Google Cloud TPU device memory.

We use TeaFor3 with $\lambda = .5$ and weight-sharing. For model selection, we use the checkpoint with the highest ROUGE-L F-score on the validation set, with evaluations every 1k steps. We stop training on Gigaword after 160k steps and CNN/Dailymail after 400k steps.

Final scores in Table 5 show the benefits of TeaForN on summarization tasks. Using just greedy decoding, TeaFor3 setups match or exceed the previous state-of-the-art ROUGE-L score on both CNN/Dailymail and Gigaword benchmarks, with an 8x cheaper decoder. Using beam search, TeaForN increases performance on the CNN/Dailymail task by +.23 ROUGE-2 (21.70 vs 21.47) and +.21 ROUGE-L (41.32 vs 41.11) and the Gigaword task by +.30 ROUGE-2 (20.16 vs 19.86) and +.30 ROUGE-L (36.54 vs 36.24).

4.3 Training Performance

Table 6 shows how TeaForN affects training performance, using Google Cloud TPUs (You et al., 2019). TeaFor2 slows down training by 37% compared to standard teacher-forcing (1.25 steps/sec vs 1.97) and TeaFor3 by 50% (.986 steps/sec vs 1.97). TeaFor2 increases High-Bandwidth Memory (HBM) usage by 4% compared to teacher-forcing (8.96GB vs 8.64) and TeaFor3 by 20%.

While training cost and speed are moderately impacted by TeaForN, we note that inference cost is significantly reduced by virtue of producing models that reach similar quality with fewer beams, enabling significant cost savings for production models, in addition to overall stronger models.

5 Conclusion

In this work, we introduce a new technique for sequence generation models called Teacher-Forcing with N-grams (TeaForN), which (a) addresses exposure bias, (b) allows the decoder to better take into account future decisions, and (c) requires no curriculum training.

We show empirical evidence of the efficacy of TeaForN on several sequence generation tasks. With Transformer $_{\text{big}}$ (Vaswani et al., 2017), we boost the performance of Transformer$_{\text{big}}$ significantly on the En-Fr benchmark. With PEGASUS $_{\text{large}}$ (Zhang et al., 2019), we improve upon the existing ROUGE-L scores the Gigaword and CNN/Dailymail benchmarks (Rush et al., 2015). Further, we show that TeaForN can match the prior state-of-the-art ROUGE-L scores on the summarization benchmarks without beam search, representing an 8x reduction in decoder cost at inference.

Overall, TeaForN is a promising approach for improving quality and/or reducing inference costs in sequence generation models.

References

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 28*, pages 1171–1179.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Nan Ding and Radu Soricut. 2017. Cold-start reinforcement learning with softmax policy gradients. In *NIPS*.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. *arXiv e-prints*, page arXiv:1905.03197.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. *arXiv e-prints*, page arXiv:1506.03340.

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

Ferenc Huszar. 2015. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *CoRR*, abs/1511.05101.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. *arXiv e-prints*, page arXiv:1706.03872.

Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4601–4609. Curran Associates, Inc.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv e-prints*, page arXiv:1910.13461.

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

Siqi Liu, Zhenhai Zhu, Ning Ye, Sergio Guadarrama, and Kevin Murphy. 2017. Optimization of image description metrics using policy gradient methods. In *International Conference on Computer Vision (ICCV)*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling Neural Machine Translation. *arXiv e-prints*, page arXiv:1806.00187.

Matt Post. 2018. A call for clarity in reporting bleu scores.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2019. Leveraging Pre-trained Checkpoints for Sequence Generation Tasks. *arXiv e-prints*, page arXiv:1907.12461.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. *arXiv e-prints*, page arXiv:1509.00685.

David R. So, Chen Liang, and Quoc V. Le. 2019. The Evolved Transformer. *arXiv e-prints*, page arXiv:1901.11117.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: Masked Sequence to Sequence Pre-training for Language Generation. *arXiv e-prints*, page arXiv:1905.02450.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics.

Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. 2017. Fast and accurate sequence labeling with iterated dilated convolutions. *CoRR*, abs/1702.02098.

RS Sutton, D McAllester, S Singh, and Y Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

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

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. CIDEr: Consensus-based image description evaluation. In *Proceedings of CVPR*.

Arun Venkatraman, Martial Hebert, and J. Andrew Bagnell. 2015. Improving multi-step prediction of learned time series models. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 3024–3030. AAAI Press.

Y. Wu, M. Schuster, and al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training.

Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. 2019. Fast deep neural network training on distributed systems and cloud tpus. *IEEE Trans. Parallel Distrib. Syst.*, 30(11):2449–2462.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the Gap between Training and Inference for Neural Machine Translation. *arXiv e-prints*, page arXiv:1906.02448.

A Appendix

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	41.83 ± 0.05	42.17 ± 0.07	42.10 ± 0.08	42.07 ± 0.07	42.00 ± 0.05	42.03 ± 0.05
2	42.20 ± 0.09	42.60 ± 0.05	42.50 ± 0.08	42.30 ± 0.05	42.33 ± 0.10	42.43 ± 0.07
3	42.37 ± 0.07	42.67 ± 0.03	42.63 ± 0.05	42.43 ± 0.03	42.53 ± 0.05	42.57 ± 0.03
4	42.40 ± 0.05	42.73 ± 0.05	42.63 ± 0.03	42.47 ± 0.03	42.63 ± 0.03	42.60 ± 0.05
5	42.30 ± 0.08	42.80 ± 0.05	42.70 ± 0.05	42.47 ± 0.03	42.57 ± 0.03	42.57 ± 0.03
6	42.37 ± 0.07	42.77 ± 0.03	42.63 ± 0.03	42.40 ± 0.00	42.60 ± 0.00	42.60 ± 0.05
7	42.33 ± 0.07	42.73 ± 0.05	42.63 ± 0.03	42.40 ± 0.00	42.57 ± 0.03	42.50 ± 0.05
8	42.33 ± 0.07	$\mathbf{42.73 \pm 0.05}$	42.60 ± 0.05	42.43 ± 0.03	42.50 ± 0.05	42.50 ± 0.05

Table 7: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 on the WMT14 English-French benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	31.10 ± 0.00	31.67 ± 0.03	31.53 ± 0.03	31.53 ± 0.03	31.63 ± 0.14	31.57 ± 0.05
2	31.40 ± 0.05	32.00 ± 0.05	31.83 ± 0.07	31.97 ± 0.03	31.97 ± 0.11	31.97 ± 0.05
3	31.50 ± 0.05	32.10 ± 0.05	31.97 ± 0.05	31.90 ± 0.05	32.00 ± 0.12	32.03 ± 0.03
4	31.53 ± 0.03	32.10 ± 0.05	31.90 ± 0.05	31.97 ± 0.05	32.03 ± 0.10	32.03 ± 0.03
5	31.57 ± 0.03	32.07 ± 0.03	31.93 ± 0.03	31.97 ± 0.05	32.03 ± 0.10	32.00 ± 0.00
6	31.47 ± 0.05	32.10 ± 0.08	31.90 ± 0.05	31.97 ± 0.03	32.00 ± 0.08	31.93 ± 0.03
7	31.43 ± 0.07	32.07 ± 0.07	31.87 ± 0.03	31.87 ± 0.03	31.97 ± 0.11	31.93 ± 0.03
8	31.43 ± 0.03	$\mathbf{32.07 \pm 0.07}$	31.87 ± 0.10	31.87 ± 0.07	31.93 ± 0.14	31.90 ± 0.00

Table 8: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 on the WMT12 English-French benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	41.83 ± 0.05	42.07 ± 0.05	41.97 ± 0.03	42.17 ± 0.07	42.00 ± 0.05	42.00 ± 0.05
2	42.20 ± 0.09	42.37 ± 0.10	42.27 ± 0.05	42.40 ± 0.08	42.30 ± 0.05	42.30 ± 0.08
3	42.37 ± 0.07	42.53 ± 0.03	42.37 ± 0.03	42.53 ± 0.05	42.43 ± 0.05	42.40 ± 0.08
4	42.40 ± 0.05	42.63 ± 0.07	42.37 ± 0.03	42.57 ± 0.03	42.43 ± 0.10	42.43 ± 0.03
5	42.30 ± 0.08	42.57 ± 0.12	42.43 ± 0.03	42.57 ± 0.03	42.47 ± 0.12	42.50 ± 0.05
6	42.37 ± 0.07	42.60 ± 0.08	42.43 ± 0.03	42.50 ± 0.05	42.50 ± 0.05	42.43 ± 0.03
7	42.33 ± 0.07	42.60 ± 0.09	42.47 ± 0.03	42.50 ± 0.05	42.47 ± 0.07	42.47 ± 0.03
8	42.33 ± 0.07	42.57 ± 0.10	$\mathbf{42.43 \pm 0.03}$	42.47 ± 0.03	42.40 ± 0.05	42.47 ± 0.03

Table 9: Mean SacreBLEU and Standard Error (n=3) of TeaFor3 on the WMT14 English-French benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	31.10 ± 0.00	31.40 ± 0.00	31.70 ± 0.05	31.60 ± 0.05	31.50 ± 0.08	31.57 ± 0.05
2	31.40 ± 0.05	31.80 ± 0.05	32.00 ± 0.05	31.83 ± 0.07	31.80 ± 0.08	31.83 ± 0.03
3	31.50 ± 0.05	31.83 ± 0.05	32.10 ± 0.05	31.93 ± 0.10	31.80 ± 0.09	32.00 ± 0.05
4	31.53 ± 0.03	31.87 ± 0.03	32.03 ± 0.07	31.93 ± 0.10	31.80 ± 0.09	31.97 ± 0.12
5	31.57 ± 0.03	31.87 ± 0.03	32.00 ± 0.09	31.90 ± 0.09	31.80 ± 0.14	31.93 ± 0.07
6	31.47 ± 0.05	31.83 ± 0.03	31.93 ± 0.05	31.87 ± 0.10	31.77 ± 0.10	31.93 ± 0.10
7	31.43 ± 0.07	31.80 ± 0.00	31.93 ± 0.03	31.87 ± 0.10	31.80 ± 0.09	31.93 ± 0.10
8	31.43 ± 0.03	31.77 ± 0.03	$\mathbf{31.90 \pm 0.05}$	31.87 ± 0.10	31.77 ± 0.10	31.83 ± 0.07

Table 10: Mean SacreBLEU and Standard Error (n=3) of TeaFor3 on the WMT12 English-French benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	28.10 ± 0.08	28.17 ± 0.07	28.17 ± 0.03	28.27 ± 0.10	28.10 ± 0.08	27.83 ± 0.07
2	28.67 ± 0.07	28.67 ± 0.07	28.80 ± 0.08	28.87 ± 0.07	28.83 ± 0.03	28.47 ± 0.07
3	28.97 ± 0.10	28.97 ± 0.07	29.03 ± 0.05	29.07 ± 0.03	29.00 ± 0.08	28.73 ± 0.07
4	29.07 ± 0.10	29.07 ± 0.07	29.27 ± 0.07	29.13 ± 0.03	29.13 ± 0.07	28.77 ± 0.07
5	29.20 ± 0.12	29.13 ± 0.10	29.23 ± 0.03	29.17 ± 0.05	29.20 ± 0.08	28.80 ± 0.05
6	29.30 ± 0.12	29.17 ± 0.07	29.27 ± 0.03	29.20 ± 0.00	29.23 ± 0.07	28.80 ± 0.05
7	29.23 ± 0.12	29.20 ± 0.05	29.30 ± 0.05	29.20 ± 0.05	29.27 ± 0.05	28.87 ± 0.03
8	29.20 ± 0.12	29.20 ± 0.05	$\mathbf{29.30 \pm 0.05}$	29.13 ± 0.03	29.23 ± 0.07	28.80 ± 0.00

Table 11: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 on the WMT14 English-German benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	22.07 ± 0.03	22.10 ± 0.05	22.20 ± 0.09	22.10 ± 0.05	22.13 ± 0.03	21.97 ± 0.07
2	22.40 ± 0.05	22.50 ± 0.12	22.50 ± 0.05	22.30 ± 0.00	22.37 ± 0.03	22.40 ± 0.05
3	22.50 ± 0.05	22.57 ± 0.07	22.60 ± 0.08	22.30 ± 0.05	22.47 ± 0.03	22.43 ± 0.07
4	22.43 ± 0.07	22.57 ± 0.07	22.57 ± 0.10	22.30 ± 0.05	22.37 ± 0.03	22.43 ± 0.07
5	22.50 ± 0.09	22.57 ± 0.07	22.53 ± 0.07	22.37 ± 0.05	22.37 ± 0.03	22.37 ± 0.05
6	22.43 ± 0.07	22.57 ± 0.07	22.53 ± 0.07	22.33 ± 0.07	22.30 ± 0.05	22.40 ± 0.05
7	22.43 ± 0.07	22.53 ± 0.10	22.47 ± 0.10	22.30 ± 0.05	22.30 ± 0.05	22.37 ± 0.05
8	22.37 ± 0.10	22.43 ± 0.10	$\mathbf{22.53 \pm 0.07}$	22.30 ± 0.05	22.27 ± 0.03	22.30 ± 0.05

Table 12: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 on the WMT12 English-German benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	28.10 ± 0.08	28.40 ± 0.12	28.07 ± 0.14	28.23 ± 0.05	28.17 ± 0.10	27.80 ± 0.05
2	28.67 ± 0.07	29.00 ± 0.08	28.67 ± 0.07	28.73 ± 0.03	28.60 ± 0.08	28.40 ± 0.05
3	28.97 ± 0.10	29.27 ± 0.05	28.87 ± 0.07	29.00 ± 0.00	28.87 ± 0.05	28.63 ± 0.03
4	29.07 ± 0.10	29.33 ± 0.07	28.97 ± 0.07	29.00 ± 0.05	28.93 ± 0.07	28.67 ± 0.05
5	29.20 ± 0.12	29.30 ± 0.09	29.13 ± 0.05	29.00 ± 0.00	28.97 ± 0.07	28.70 ± 0.05
6	29.30 ± 0.12	29.33 ± 0.07	29.17 ± 0.03	29.03 ± 0.03	29.00 ± 0.05	28.77 ± 0.10
7	29.23 ± 0.12	29.37 ± 0.07	29.20 ± 0.08	29.10 ± 0.05	29.03 ± 0.05	28.73 ± 0.07
8	29.20 ± 0.12	29.33 ± 0.10	$\mathbf{29.23 \pm 0.05}$	29.07 ± 0.05	29.00 ± 0.05	28.77 ± 0.10

Table 13: Mean SacreBLEU and Standard Error (n=3) of TeaFor3 on the WMT14 English-German benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	$\lambda = 0$	$\lambda = 0.2$	$\lambda = 0.4$	$\lambda = 0.6$	$\lambda = 0.8$	$\lambda = 1$
1	22.07 ± 0.03	22.17 ± 0.03	22.07 ± 0.03	22.10 ± 0.05	22.07 ± 0.03	22.03 ± 0.03
2	22.40 ± 0.05	22.43 ± 0.05	22.50 ± 0.05	22.40 ± 0.00	22.43 ± 0.03	22.30 ± 0.05
3	22.50 ± 0.05	22.33 ± 0.03	22.50 ± 0.05	22.43 ± 0.03	22.40 ± 0.00	22.37 ± 0.03
4	22.43 ± 0.07	22.37 ± 0.03	22.57 ± 0.05	22.43 ± 0.05	22.33 ± 0.03	22.27 ± 0.03
5	22.50 ± 0.09	22.30 ± 0.00	22.50 ± 0.09	22.47 ± 0.07	22.33 ± 0.03	22.27 ± 0.05
6	22.43 ± 0.07	22.27 ± 0.03	22.47 ± 0.10	22.37 ± 0.05	22.30 ± 0.00	22.20 ± 0.05
7	22.43 ± 0.07	22.20 ± 0.00	22.43 ± 0.07	22.40 ± 0.05	22.30 ± 0.00	22.17 ± 0.03
8	22.37 ± 0.10	22.20 ± 0.00	$\mathbf{22.40 \pm 0.05}$	22.33 ± 0.05	22.20 ± 0.00	22.13 ± 0.05

Table 14: Mean SacreBLEU and Standard Error (n=3) of TeaFor3 on the WMT12 English-German benchmark, for all beam sizes k and discount factors λ tested. Bold font indicates the configuration reported in Table 3.

k	Teacher-forcing	TeaForN	Top-4	Top-V
1	41.83 ± 0.05	42.17 ± 0.07	42.23 ± 0.11	41.90 ± 0.08
2	42.20 ± 0.09	42.60 ± 0.05	42.50 ± 0.16	42.27 ± 0.05
3	42.37 ± 0.07	42.67 ± 0.03	42.73 ± 0.12	42.47 ± 0.05
4	42.40 ± 0.05	42.73 ± 0.05	42.77 ± 0.14	42.47 ± 0.05
5	42.30 ± 0.08	42.80 ± 0.05	42.77 ± 0.10	42.47 ± 0.05
6	42.37 ± 0.07	42.77 ± 0.03	42.80 ± 0.12	42.47 ± 0.05
7	42.33 ± 0.07	42.73 ± 0.05	42.73 ± 0.12	42.47 ± 0.07
8	42.33 ± 0.07	42.73 ± 0.05	42.73 ± 0.12	42.43 ± 0.07

Table 15: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 ($\lambda = .2$) on the WMT14 English-French benchmark using different approximation methods, for all beam sizes k.

k	Teacher-forcing	TeaForN	Top-4	Top-V
1	28.10 ± 0.08	28.17 ± 0.07	27.83 ± 0.10	28.03 ± 0.10
2	28.67 ± 0.07	28.67 ± 0.07	28.57 ± 0.10	28.73 ± 0.07
3	28.97 ± 0.10	28.97 ± 0.07	28.80 ± 0.05	29.00 ± 0.00
4	29.07 ± 0.10	29.07 ± 0.07	28.90 ± 0.05	29.13 ± 0.03
5	29.20 ± 0.12	29.13 ± 0.10	28.97 ± 0.05	29.23 ± 0.03
6	29.30 ± 0.12	29.17 ± 0.07	29.00 ± 0.08	29.30 ± 0.05
7	29.23 ± 0.12	29.20 ± 0.05	29.00 ± 0.08	29.23 ± 0.07
8	29.20 ± 0.12	29.20 ± 0.05	29.00 ± 0.05	29.30 ± 0.05

Table 16: Mean SacreBLEU and Standard Error (n=3) of TeaFor2 ($\lambda = .2$) on the WMT14 English-German benchmark using different approximation methods, for all beam sizes k.

k	Transformer$_{\text{big}}$	$P_{drop} = .01$	$P_{drop} = .02$	$P_{drop} = .03$
1	41.83 ± 0.05	41.63 ± 0.07	41.73 ± 0.12	41.47 ± 0.03
2	42.20 ± 0.09	41.93 ± 0.07	41.93 ± 0.07	41.80 ± 0.08
3	42.37 ± 0.07	42.13 ± 0.05	42.10 ± 0.09	41.83 ± 0.07
4	42.40 ± 0.05	42.17 ± 0.07	42.07 ± 0.12	41.87 ± 0.07
5	42.30 ± 0.08	42.10 ± 0.05	42.07 ± 0.07	41.90 ± 0.09
6	42.37 ± 0.07	42.10 ± 0.05	42.10 ± 0.08	41.93 ± 0.07
7	42.33 ± 0.07	42.07 ± 0.07	42.03 ± 0.05	41.97 ± 0.10
8	42.33 ± 0.07	42.03 ± 0.07	42.00 ± 0.08	41.97 ± 0.12

Table 17: Mean SacreBLEU and Standard Error (n=3) on the WMT14 English-French benchmark using word drop regularization, for all beam sizes k.

	Transformer$_{\text{big}}$	$P_{drop} = .01$	$P_{drop} = .02$	$P_{drop} = .03$
1	28.10 ± 0.08	28.60 ± 0.12	28.37 ± 0.07	28.20 ± 0.08
2	28.67 ± 0.07	29.13 ± 0.15	28.90 ± 0.00	28.77 ± 0.07
3	28.97 ± 0.10	29.23 ± 0.20	29.13 ± 0.03	28.93 ± 0.05
4	29.07 ± 0.10	29.43 ± 0.17	29.20 ± 0.05	29.07 ± 0.03
5	29.20 ± 0.12	29.47 ± 0.10	29.33 ± 0.03	29.13 ± 0.03
6	29.30 ± 0.12	29.50 ± 0.12	29.30 ± 0.00	29.23 ± 0.03
7	29.23 ± 0.12	29.47 ± 0.12	29.30 ± 0.00	29.20 ± 0.05
8	29.20 ± 0.12	29.47 ± 0.10	29.33 ± 0.03	29.20 ± 0.05

Table 18: Mean SacreBLEU and Standard Error (n=3) on the WMT14 English-German benchmark using word drop regularization, for all beam sizes k.