On Sparsifying Encoder Outputs in Sequence-to-Sequence Models

Biao Zhang[1] **Ivan Titov**[1,2] **Rico Sennrich**[3,1]

[1]School of Informatics, University of Edinburgh
[2]ILLC, University of Amsterdam
[3]Department of Computational Linguistics, University of Zurich
B.Zhang@ed.ac.uk, ititov@inf.ed.ac.uk, sennrich@cl.uzh.ch

Abstract

Sequence-to-sequence models usually transfer *all* encoder outputs to the decoder for generation. In this work, by contrast, we hypothesize that these encoder outputs can be compressed to shorten the sequence delivered for decoding. We take Transformer as the testbed and introduce a layer of stochastic gates in-between the encoder and the decoder. The gates are regularized using the expected value of the sparsity-inducing L_0 penalty, resulting in completely masking-out a subset of encoder outputs. In other words, via joint training, the $\mathcal{L}_0$DROP layer forces Transformer to route information through a subset of its encoder states. We investigate the effects of this sparsification on two machine translation and two summarization tasks. Experiments show that, depending on the task, around 40–70% of source encodings can be pruned without significantly compromising quality. The decrease of the output length endows $\mathcal{L}_0$DROP with the potential of improving decoding efficiency, where it yields a speedup of up to 1.65× on document summarization and 1.20× on character-based machine translation against the standard Transformer. We analyze the $\mathcal{L}_0$DROP behaviour and observe that it exhibits systematic preferences for pruning certain word types, e.g., function words and punctuation get pruned most. Inspired by these observations, we explore the feasibility of specifying rule-based patterns that mask out encoder outputs based on information such as part-of-speech tags, word frequency and word position. [1]

1 Introduction

Neural sequence-to-sequence (Seq2Seq) models have dominated various text generation tasks, including machine translation (Vaswani et al., 2017) and abstractive document summarization (Gehrmann et al., 2018; Liu and Lapata, 2019). These models generally follow the encoder-decoder paradigm, where the encoder interprets source context and converts source words into vector representations such that the decoder has sufficient information to predict the target sequence. Early Seq2Seq models (Sutskever et al., 2014; Cho et al., 2014) provided only the last and/or first encoder states to the decoder. In contrast, modern approaches rely on the attention mechanism (Bahdanau et al., 2015) and implicitly make an assumption that information from *all* encoder outputs should flow to the decoder.[2] However, this assumption neglects the fact that a large portion of source words in machine translation receives just minor attention as shown in Figure 1, let alone in summarization where the input contains redundant expressions and large parts of text are not relevant to any plausible summary. Moreover, information content varies

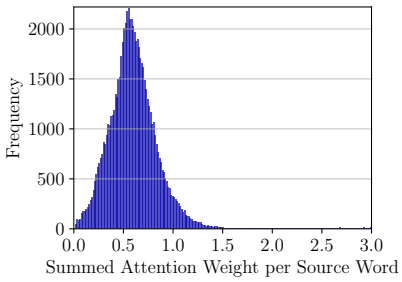


Figure 1: Distribution of the summed attention weight per source word estimated on the English-German WMT14 test set. For each (source sentence, translation) pair, we extract the attention matrices from all encoder-decoder attention sublayers in Transformer and average them over different (8) heads and (6) layers. The attention value for each source word is summed over all target words in the translation. Higher attention weights suggest larger impacts on translation. Around 49.7% source words get attention weights of less than 0.6, compared to the mean value of 1.03.

[1]Source code is available at https://github.com/bzhangGo/zero.

[2]We interchangeably use *source representation*, *encoder output* and *source encoding* unless otherwise specified.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 2888–2900
August 1–6, 2021.

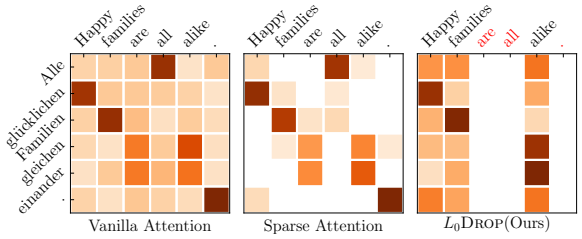


Figure 2: Encoder-decoder attention distribution of target words (y-axis) over source words (x-axis) for the vanilla attention (Vaswani et al., 2017), the sparse attention (Correia et al., 2019) and our model. Darker color indicates larger attention weight, and the white blocks denote an attention weight of 0. The source words whose encoding is pruned by $\mathcal{L}_0$DROP (receiving zero weight) are highlighted in red.

across words, for example, it is negatively correlated with event frequency (Shannon, 1948; Zipf, 1949). When moving from word-level to character-level processing, the notion of encoding information and computing attention on the level of characters also seems excessive. Previous work has proposed hierarchical architectures where character-level encodings are compressed into word-level or span-level states (Ling et al., 2015; Lee et al., 2017).

In this work, we hypothesize that encoder outputs are compressible and we can force Seq2Seq model to route information through their subset. Figure 2 illustrates our intuition as well as the difference with existing work (Vaswani et al., 2017; Correia et al., 2019). Instead of dynamically sparsifying attention weights for individual decoder steps (Correia et al., 2019), we aim at detecting uninformative source encodings and dropping them to shorten the encoding sequence before generation. To this end, we build on recent work on sparsifying weights (Louizos et al., 2018) and activations (Bastings et al., 2019) of neural networks. Specifically, we insert a differentiable neural sparsity layer ($\mathcal{L}_0$DROP) in-between the encoder and the decoder. The layer can be regarded as providing a multiplicative scalar gate for every encoder output. The gate is a random variable and, unlike standard attention, can be exactly zero, effectively masking out the corresponding source encodings. The sparsity is promoted by introducing an extra term to the learning objective, i.e. an expected value of the sparsity-inducing L_0 penalty. By varying the coefficient for the regularizer, we can obtain different levels of sparsity. Importantly, the objective remains fully end-to-end differentiable.

Given an encoding sequence of length N, the vanilla attention model attends to it recurrently for M steps at the decoding phase, leading to a computational complexity of $\mathcal{O}(NM)$ ($N = 6$, $M = 6$ in Figure 2). This could be costly if N or M is very large. With the induced sparse structure by $\mathcal{L}_0$DROP, we introduce a specialized decoding algorithm which lowers this complexity to $\mathcal{O}(N'M)$ ($N' \leq N$, and $N' = 3$ in Figure 2). As a result, $\mathcal{L}_0$DROP can improve decoding efficiency by reducing the encodings' length, especially for long inputs.

We apply $\mathcal{L}_0$DROP to Transformer (Vaswani et al., 2017), the state-of-the-art Seq2Seq model. We conduct extensive experiments on WMT translation tasks with two language pairs and document summarization tasks covering single document and multiple documents settings. We also explore character-based machine translation where the lengthy character sequence often leads to slow inference. We analyze how pruning source encodings impacts the generation quality and which word types get pruned. Inspired by the analysis of $\mathcal{L}_0$DROP, we further study rule-based sparsity patterns , such as deterministically filtering out the encodings of words with specific POS tags, high-frequency words or simply attending to every other word in the sequence.

Our main findings are summarized as follows:

- We confirm that the encoder outputs can be compressed, around 40–70% of them can be dropped without large effects on the generation quality.
- The resulting sparsity level differs across word types, the encodings corresponding to function words (such as determiners, prepositions) are more frequently pruned than those of content words (e.g., verbs and nouns).
- $\mathcal{L}_0$DROP can improve decoding efficiency particularly for lengthy source inputs. We achieve a decoding speedup of up to 1.65× on document summarization tasks and 1.20× on character-based machine translation task.
- Filtering out source encodings with rule-based sparse patterns is feasible, and confirms information-theoretic expectations, although rule-based patterns do not generalize well across tasks.

2 Related Work

Approaches to compression in Seq2Seq models fall into the category of model parameter compression (See et al., 2016), sequential knowledge dis-

tillation (Kim and Rush, 2016) or sparse attention induction that ranges from modeling hard attention (Wu et al., 2018) to developing differentiable sparse softmax functions or regularizing attention weights for sparsity (Niculae and Blondel, 2017; Correia et al., 2019; Cui et al., 2019; Zhang et al., 2019). Unfortunately, the success of all these studies builds upon the access to all source encodings in training and decoding. Learning which encoder outputs to prune in Seq2Seq models, to the best of our knowledge, has never been investigated before. Sukhbaatar et al. (2019) learn attention spans in self-attention and discard information from states outside of the span; this method is not directly applicable to encoder-decoder attention.

We use the differentiable L_0-relaxation which was first introduced by Louizos et al. (2018) in the context of pruning individual neural network parameters. It was previously used to prune heads in multi-head attention (Voita et al., 2019). Our work is more similar in spirit to Bastings et al. (2019) where they used the L_0 relaxations to construct interpretable classifiers, i.e. models that can reveal which words they rely on when predicting a class. In their approach, the information from dropped words is lost rather than rerouted into the states of retained words, as desirable for interpretability but problematic in the text generation set-up.

The number of the source encodings selected by $\mathcal{L}_0$DROP is sentence-dependent, which differs from the linear-time model of Wang et al. (2019), although both can accelerate decoding. Our study of rule-based sparsity patterns is in line with the sparse Transformer (Child et al., 2019) though we also explore the use of external linguistic information (POS tag) in our sparsification rules, and focus on encoder outputs instead of self-attention.

Character-based translation gained increasing popularity due to its capability of handling out-of-vocabulary issues while avoiding tokenization and subword segmentation (Ling et al., 2015; Costa-jussà and Fonollosa, 2016; Sennrich, 2017; Cherry et al., 2018). Recent efforts often focus on closing the performance gap against its subword-level counterpart (Libovický and Fraser, 2020; Gao et al., 2020), but little study explores solutions to improve its inefficient inference resulted from the long character sequences. In this respect, Cherry et al. (2018) proposed to use conditional computation to dynamically compress encoder states. Similar to our results, they also observed a trade-off between the translation quality and the degree of compression.

3 Background: Transformer

We take Transformer (Vaswani et al., 2017) as our testbed. Transformer uses the dot-product attention network as its backbone to handle intra- and inter-sequence dependencies:

$$\textsc{Att}(\mathbf{H}, \mathbf{M}) = \mathbf{A}\mathbf{V} = \textsc{Sm}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}, \quad (1)$$

where $\mathbf{Q}, \mathbf{K}, \mathbf{V} = \mathbf{H}\mathbf{W}_q, \mathbf{M}\mathbf{W}_k, \mathbf{M}\mathbf{W}_v$. The input $\mathbf{H} \in \mathbb{R}^{J\times d}$ of length J queries and summarizes task-relevant clues from the memory $\mathbf{M} \in \mathbb{R}^{I\times d}$ of length I based on their dot-product semantic matching $\mathbf{A} \in \mathbb{R}^{J\times I}$. $\textsc{Sm}$ denotes the softmax function, d is the model dimension, and $\mathbf{W}_q$, $\mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d\times d}$ are trainable model parameters. Vaswani et al. (2017) also extend this mechanism to multi-head attention.

Given a source sequence $X = (x_1, x_2, \ldots, x_N)$, Transformer maps it to the target sequence $Y = (y_1, y_2, \ldots, y_M)$ following the encoder-decoder paradigm (Bahdanau et al., 2015):[3]

$$\begin{aligned}\mathbf{X}^L &= \text{Encoder}\left(\mathbf{X}^0\right) \quad (2)\\ &\triangleq \overset{L}{\underset{l=1}{}} \textsc{Ffn}\left(\textsc{Att}(\mathbf{X}^{l-1}, \mathbf{X}^{l-1})\right),\\ \mathbf{Y}^L &= \text{Decoder}\left(\mathbf{Y}^0, \mathbf{X}^L\right) \quad (3)\\ &\triangleq \overset{L}{\underset{l=1}{}} \textsc{Ffn}\left(\textsc{Att}\left(\overline{\textsc{Att}}(\mathbf{Y}^{l-1}, \mathbf{Y}^{l-1}), \mathbf{X}^L\right)\right),\end{aligned}$$

where $\mathbf{X}^0 \in \mathbb{R}^{N\times d}$ and $\mathbf{Y}^0 \in \mathbb{R}^{M\times d}$ stand for the source and the shifted target sequence embedding, respectively, enriched with positional encoding (Vaswani et al., 2017). $\textsc{Ffn}(\cdot)$ is a point-wise feed-forward network. $\overline{\textsc{Att}}(\cdot,\cdot)$ in the decoder denotes masked $\textsc{Att}(\cdot,\cdot)$ which prevents access to future target words. Both the encoder and the decoder involve a stack of L identical layers, with the encoder output $\mathbf{X}^L$ fed to the decoder via an encoder-decoder attention sublayer, i.e. the $\textsc{Att}(\cdot,\cdot)$ in Eq. (3). Based on the decoder output $\mathbf{Y}^L$, Transformer performs the next-word prediction and adopts the maximum likelihood loss for training.

4 Neural Sparsity Layer: $\mathcal{L}_0$DROP

In this section, we introduce a neural sparsity layer ($\mathcal{L}_0$DROP), which we use to prune encoder outputs.

[3]Each sublayer ($\textsc{Att}$/$\overline{\textsc{Att}}$/$\textsc{Ffn}$) in the encoder and decoder is wrapped with residual connection (He et al., 2015) followed by layer normalization (Ba et al., 2016), which are dropped in Eq. (2) and (3) for clarity.

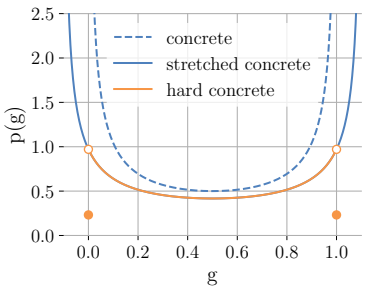


Figure 3: Hard concrete distribution (orange curve): samples are first stretched from the binary concrete distribution (dashed blue curve) to a stretched distribution (solid blue curve), and then rectified to collapse the probability mass of the shadow areas into $\{0\}$ and $\{1\}$ (solid orange points).

At inference time, only retained encoder outputs will be used as input to the decoder.

4.1 Training with $\mathcal{L}_0$DROP

Intuitively, $\mathcal{L}_0$DROP assigns each encoder output $\mathbf{x}_i^L$ a gate $g_i \in [0, 1]$ ($i \in \{1, \ldots, N\}$)

$$\mathcal{L}_0\text{DROP}(\mathbf{x}_i^L) = g_i \mathbf{x}_i^L, \tag{4}$$

and prunes encodings by closing their gates, i.e. $g_i = 0$, relying on adding a differentiable sparsity-inducing penalty to the objective.

More formally, to achieve sparsity, each gate is assumed to be a random variable and its value is drawn from the HardConcrete distribution:

$$g_i \sim \text{HardConcrete}(\alpha_i, \beta, \epsilon), \tag{5}$$

where α_i, β and ϵ are shape parameters of the distribution. HardConcrete (Louizos et al., 2018) is a parameterized family of mixed discrete-continuous distributions over the closed interval $[0, 1]$. These distributions have point mass at 0 and 1 and continuous density in-between, i.e. in $(0, 1)$, as shown in Figure 3. Thus, the gates will have a non-zero probability of being exactly 0, corresponding to masking out the input completely.

Specifically, the sample from HardConcrete distribution is obtained by stretching and rectifying samples from BinaryConcrete distributions (Maddison et al., 2017; Jang et al., 2017):

$$s_i \sim \text{BinaryConcrete}(\alpha_i, \beta) \tag{6}$$

$$\bar{s}_i = s_i\,(1 + 2\epsilon) - \epsilon, \tag{7}$$

$$g_i = \min\left(1, \max\left(0, \bar{s}_i\right)\right). \tag{8}$$

In the above expression, we first obtain a sample from the BinaryConcrete distribution (Eq. (6)), then stretch it from $(0, 1)$ to $(-\epsilon, 1 + \epsilon)$ (Eq. (7), $\epsilon > 0$), and finally rectify with a hard sigmoid to the closed interval $[0, 1]$ (Eq. (8)).

The probability of g_i being exactly 0 ($p(g_i = 0|\alpha_i, \beta, \epsilon)$) equals the probability of $\bar{s}_i$ hitting $(-\epsilon, 0)$ and is available in a closed form (Louizos et al., 2018):

$$p(g_i = 0|\alpha_i, \beta, \epsilon) = \sigma(\beta \log \frac{\epsilon}{1+\epsilon} - \log \alpha_i),$$

where $\sigma(\cdot)$ denotes the sigmoid function. The parameter α_i (i.e. the location parameter of BinaryConcrete) is predicted relying on the encoder output $\mathbf{x}_i$:

$$\log \alpha_i = \mathbf{x}_i^L \mathbf{w}^T, \tag{9}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a learned parameter vector; the temperature β and the stretch degree ϵ are treated as hyperparameters. By adjusting α_i the model can change the shape of the HardConcrete distribution, and dynamically decide which outputs to pass to the decoder and which to prune.

Note that the sum

$$\mathcal{L}_0(X) = \sum_{i=1}^{N} 1 - p(g_i = 0|\alpha_i, \beta, \epsilon), \tag{10}$$

yields the expected number of open gates, or, equivalently, the expected L_0 loss on gate vector $(g_1, \ldots, g_N)$. Minimizing the loss encourages the model to prune encoder outputs.

Once $\mathcal{L}_0$DROP is integrated as a new layer into Transformer, the decoder, previously defined in Eq. (3), becomes:

$$\mathbf{Y}^L = \text{Decoder}(\mathbf{Y}^0, \mathcal{L}_0\text{DROP}(\mathbf{X}^L)). \tag{11}$$

Other components in Transformer are kept intact, except for using a modified objective $\mathcal{L}(X, Y)$:

$$\begin{aligned}
&\mathcal{L}_{\text{MLE}}(X, Y) + \lambda \mathcal{L}_0(X) \\
&= -\log \mathbb{E}_{\mathbf{g} \sim p(\mathbf{g}|\phi)} \left[p(Y, \mathbf{g}|X)\right] + \lambda \mathcal{L}_0(X) \\
&\leq \mathbb{E}_{\mathbf{g} \sim p(\mathbf{g}|\phi)} \left[-\log p(Y, \mathbf{g}|X)\right] + \lambda \mathcal{L}_0(X) \\
&= \mathcal{L}(X, Y)
\end{aligned} \tag{12}$$

where ϕ is short for $(\alpha, \beta, \epsilon)$, $\lambda \in \mathbb{R}^+$ is a hyperparameter defining the level of sparsity. The bound is derived by applying Jensen's inequality.

Importantly, the objective remains fully differentiable as we can rely on the reparameterization technique (Kingma and Welling, 2013) to sample $\tilde{\mathbf{g}}$ for computing unbiased estimates of the gradients. Adding $\mathcal{L}_0$DROP and the regularizer introduces only a negligible computational overhead to training compared to the original Transformer.

Algorithm 1 Decoding algorithm for $\mathcal{L}_0$DROP

Input: Source encodings, $\mathbf{X}^L \in \mathbb{R}^{N\times d}$;
Gates, $\hat{\mathbf{g}} \in \mathbb{R}^N$;
Query state, $\mathbf{y}_j^l \in \mathbb{R}^d$;
Output: Attention vector for the query
▷ *step 1: reorganize source-side inputs*
▷ *note this step can be done before the decoder*
1: $I \leftarrow \{i|\hat{g}_i \neq 0\}$ ▷ $N' \leftarrow |I|$
2: $\hat{\mathbf{g}}' \in \mathbb{R}^{N'}, \mathbf{X'}^L \in \mathbb{R}^{N'\times d} \leftarrow \hat{\mathbf{g}}[I], \mathbf{X}^L[I]$
3: $c \leftarrow N - N'$
4: $\bar{\mathbf{X}}^L \leftarrow [\mathbf{0} \in \mathbb{R}^d, \mathbf{X'}^L \odot \hat{\mathbf{g}}'], \mathbf{c} \leftarrow [c, \mathbf{1} \in \mathbb{R}^{N'}]$
▷ *step 2: attention with counts*
5: $\mathbf{q}, \mathbf{K}, \mathbf{V} \leftarrow \mathbf{y}_j^l\mathbf{W}_q, \bar{\mathbf{X}}^L\mathbf{W}_k, \bar{\mathbf{X}}^L\mathbf{W}_v$
6: $\mathbf{e} \in \mathbb{R}^{N'+1} \leftarrow \mathbf{q}\mathbf{K}^T/\sqrt{d}$
▷ *perform softmax with counts*
7: $\mathbf{a} \leftarrow \mathbf{c} \odot \exp(\mathbf{e}) / \sum_i (c_i \exp(e_i))$
8: $\mathbf{v} \in \mathbb{R}^d \leftarrow \mathbf{a}\mathbf{V}$
9: **return** $\mathbf{v}$

4.2 Decoding with $\mathcal{L}_0$DROP

At test time we do not sample gate values but estimate their expected value g_i, following Louizos et al. (2018):

$$\hat{g}_i = \min(1, \max(0, \sigma(\log \alpha_i)(1+2\epsilon) - \epsilon)), \quad (13)$$

which often turns out to be exactly either 0 or 1, albeit being in-between in some cases. Encodings corresponding to non-zero $\hat{g}_i$ are preserved and simply weighted by the expectation $\hat{g}_i$.

To leverage the induced sparse structure, we revise the decoding procedure as in Algorithm 1. The notation $[\cdot, \cdot]$ refers to row-wise concatenation, $[I]$ stands for extracting elements with the indices I, $\odot$ is element-wise multiplication, and $\mathbf{1} \in \mathbb{R}^{N'}$ indicates a vector of ones of length N'. We first reorganize the gates $\hat{\mathbf{g}} \in \mathbb{R}^N$ and the source encodings $\mathbf{X}^L \in \mathbb{R}^{N\times d}$ by eschewing the entries corresponding to closed gates ($\hat{g}_i = 0$, line 1-2). We augment the compressed sequence $\mathbf{X'}^L \in \mathbb{R}^{N'\times d}$ with a dummy zero encoding vector $\mathbf{0} \in \mathbb{R}^d$ to represent all pruned encodings, and record their count into a counting vector $\mathbf{c} \in \mathbb{R}^{N'+1}$ (line 4).[4] Notice that this step is decoder-agnostic, which only relies on the source encodings and $\mathcal{L}_0$DROP gates. We then modify the attention process to enable the inclusion of this counting information (line 5-8) for correctly estimating the attention weights. Note that the shortened source sequence $\bar{\mathbf{X}}^L$ is reused across decoder layers and steps. $\mathcal{L}_0$DROP changes the dependency of the encoder-decoder attention on source sequence from $\mathcal{O}(NM)$ to $\mathcal{O}(N'M)$, and allows for efficiency gains even with moderate sparsity, especially for large L, N and M.

[4] Note that $N' \leq N$. $\mathcal{L}_0$DROP could increase the sequence length if no source encoding is pruned, which is not observed in our experiments.

5 Experimental Setup

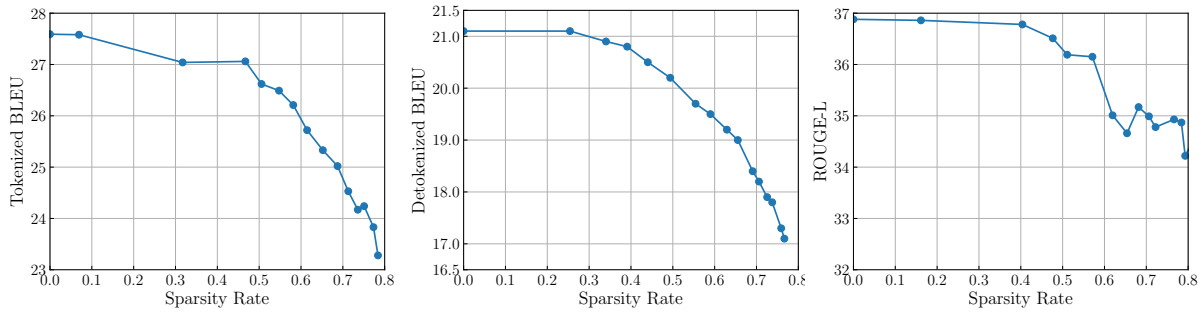
We study $\mathcal{L}_0$DROP on machine translation tasks (WMT14 English-German (En-De) (Bojar et al., 2014) and WMT18 Chinese-English (Zh-En) (Bojar et al., 2018)[5]) and document summarization tasks (CNN/Daily Mail (Hermann et al., 2015)[6] and WikiSum (Liu et al., 2018)[7]). We adopt BLEU (Papineni et al., 2002) and ROUGE-L (Lin, 2004) to evaluate the translation and summarization quality, respectively. Other details, including model settings, are given in the Appendix A. For character-based translation, we employ the same model architecture and hyperparameters for training and decoding as specified in Appendix A, except that we adopt larger-batch training (∼85K characters) and encourage longer sequence decoding (length penalty of 1.0).

6 Results and Analysis

How much can encoder outputs be sparsified? We answer this question by analyzing the impact of pruning source encodings on the generation quality. We first train a baseline Transformer model, and then finetune this model using $\mathcal{L}_0$DROP (Eq. (12)) with varied λ to explore different levels of sparsity. We sample λ with a range of $(0, 1.5]$ and a step size of 0.1, and finetune WMT14 En-De and WMT18 Zh-En models for extra 50K steps, and CNN/Daily Mail for extra 20K steps. We use the *sparsity rate* to measure the sparsity; we define it as the ratio of the pruned source encoding number $\#(\hat{g}_i = 0)$ to the total number of source words.

Figure 4 shows the results. The generation quality exhibits a negative correlation with the sparsity rate across different tasks, reflecting the usefulness of encoder outputs for generation. However, the fact that we can remove about 40% source encodings without largely degrading the generation performance (-0.5 BLEU and -0.1 ROUGE-L) supports our hypothesis that we can force Seq2Seq

[5] https://www.statmt.org/wmt14(8)/translation-task.html
[6] https://github.com/harvardnlp/sent-summary
[7] https://github.com/nlpyang/hiersumm



(a) WMT14 En-De (b) WMT18 Zh-En (c) CNN/Daily Mail

Figure 4: Generation quality (BLEU and ROUGE-L, evaluated on test set) as a function of sparsity rate for WMT14 En-De, WMT18 Zh-En and CNN/Daily Mail. Pruning about 40% source encodings results in marginal performance loss on all tasks.

Task	Time	Speedup	Sparsity	Quality
WMT14 En-De	68.89	1.00×	0.00%	27.59
	68.38	1.01×	46.7%	27.06
WMT14 En-De (Char)	1418	1.00×	0.00%	25.40
	1186	1.20×	46.1%	25.35
WMT18 Zh-En	116.3	1.00×	0.00%	21.10
	118.3	0.98×	39.1%	20.80
CNN/Daily Mail	3909	1.00×	0.00%	36.88
	3227	1.21×	47.6%	36.51
WikiSum	70505	1.00×	0.00%	39.20
	42669	1.65×	71.5%	38.75

Table 1: Decoding results for different tasks when finetuning with $\lambda = 0.3$. "Time": the decoding time (in seconds) of the whole test set. "Sparsity": the sparsity rate, 0.00% indicates the Transformer baseline. "Speedup": the decoding acceleration over the baseline. "Quality": BLEU for WMT tasks and ROUGE-L for summarization tasks. "Char": the character-level models, where we set $\lambda = 0.02$. We evaluate the decoding time on GeForce GTX 1080 Ti, with a batch size of 32 for WMT tasks and 10 for summarization tasks.

model to route information through a subset of its source encodings. We also observe that the compressibility seems relatively language independent (the curves of WMT14 En-De 4(a) and WMT18 Zh-En 4(b) are similar) but clearly task dependent. Compared to translation tasks, the summarization task seems less sensitive to the pruning of source encodings. We ascribe this to the property of summarization where the summary only reflects a part of the input document, rather than the entire document.

Does $\mathcal{L}_0$DROP improve the decoding speed? With appropriate finetuning, $\mathcal{L}_0$DROP can shorten the encoding sequence fed to the decoder, reducing the calculation amount of the encoder-decoder attention. However, the encoder-decoder attention corresponds to about 1/3 of the decoder calculations,[8] and Algorithm 1 also brings in extra overhead, such as gathering and indexing operations. Thus, a speed-up is not guaranteed, and we report empirical decoding time across different tasks.

Results in Table 1 show that $\mathcal{L}_0$DROP only marginally improves the decoding speed for subword-based machine translation, despite a high sparsity rate of 46.7% (WMT14 En-De) and 39.1% (WMT18 Zh-En). By contrast, $\mathcal{L}_0$DROP yields a speedup of 1.21× and 1.65× on CNN/Daily Mail and WikiSum, respectively. One explanation lies at the significant difference in target sequence length, where the average length per summary is >60, compared to ∼25 in machine translation. Note that $\mathcal{L}_0$DROP achieves a substantially higher sparsity rate of 71.5% on WikiSum with the same $\lambda = 0.3$. This is because the input paragraphs overlap in content; the information about redundant words does not need to be routed into other encoder states, making it easier to prune them.

When it comes to character-based translation, we observe that $\mathcal{L}_0$DROP performs comparably to the baseline (-0.05 BLEU) with a high sparsity rate of 46.1%. This echoes with our findings on subword-based translation, except that $\mathcal{L}_0$DROP also improves inference efficiency here due to the higher sequence length with a decoding speedup of 1.20×. In comparison to adaptive compression (Cherry et al., 2018), $\mathcal{L}_0$DROP delivers higher compression ratio with almost no quality loss. This further demonstrates the effectiveness of $\mathcal{L}_0$DROP in handling lengthy inputs. We notice that character-level models still underperform their subword-level counterparts with the standard Transformer, but

[8] At decoding, the encoder-decoder attention accounts for about 34.4% decoder time according to profiling on our implementation.

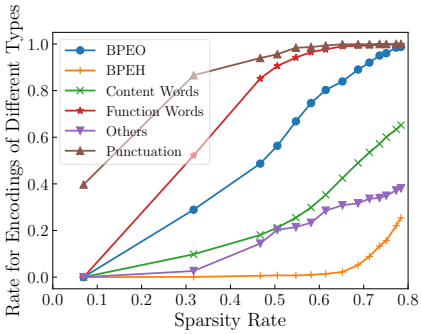


Figure 5: Curves for sparsity rate of different types of encoding on the WMT14 En-De test set. x-axis denotes the overall sparsity rate. The encoding of content words and BPEH is more valuable for generation, compared to that of function words and punctuation.

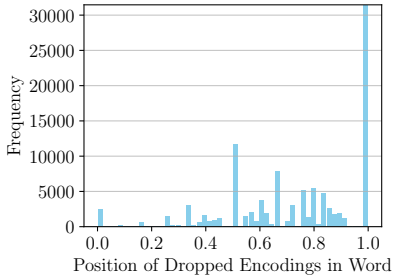


Figure 6: Distribution of the position of pruned source characters within a word on the WMT14 En-De test set. x-axis: 0.0 and 1.0 corresponds to the beginning and ending of one word, respectively.

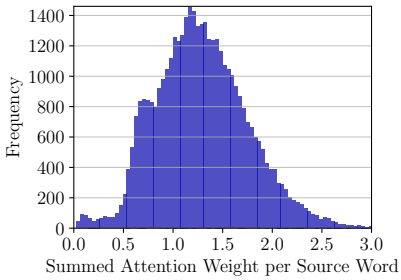


Figure 7: Distribution of the summed attention weight per source word on the WMT14 En-De test set for Transformer with $\mathcal{L}_0$DROP (sparsity rate 47%, BLEU 27.06). Only 4.5% source words get attention weights of less than 0.6.

closing this performance gap is beyond the scope of this study.

Note that the pretraining-then-finetuning schema is mainly used for saving training efforts. By scheduling λ linearly with training steps, we can train models with $\mathcal{L}_0$DROP (Eq. (12)) from scratch, and obtain a BLEU score of 27.03 ($\lambda = 0.2$, warm-up step of 200K) on WMT14 En-De (using subwords), comparable to using finetuning (27.04).

What types of source encoding are required for generation? Our goal here it to understand encodings of which types of tokens are retained. For each source encoding, we regard the POS of its corresponding word as its type. We take WMT14 En-De as our benchmark, where we annotate POS for source sentences in the test set using the Stanford POS tagger (Toutanova et al., 2003). We handle subwords separately by labeling its first piece as *BPEH* while the others as *BPEO*, regardless of the POS of its unsegmented form. We group different POS tags into 6 categories for the sake of analysis: BPEH, BPEO, function words, content words, punctuation and the rest.[9]

Figure 5 shows how the sparsity rate of each encoding type changes as a function of the overall sparsity rate. We find that $\mathcal{L}_0$DROP first choose to eliminate the encoding of punctuation, followed by that of function words. These words often signal structural and grammatical relationships that, while important to build up a representation of the sentence, can be easily compressed. In contrast, pruning content words, which express richer lexical meaning, is more difficult. The sharp increase of content word sparsity after the overall sparsity rate of 0.5 in Figure 5 correlates with a sharp drop in translation quality (see Figure 4(a)). We also observe that there is a large difference between BPEO and BPEH, albeit both from the same word. $\mathcal{L}_0$DROP favours to prune the encoding of BPEO, indicating that the model learns to use word-initial representations (BPEH) to represent whole words.

For character-level models, we observe that $\mathcal{L}_0$DROP identifies the inter-word structure in character sequences, and removes 89.2% encodings of the space symbol. We didn't find any other character-specific pruning patterns, but Figure 6 reveals that the encodings of characters near the ending of a word are more likely to be pruned compared to those at the beginning positions. This result resonates with our above observation on subword pruning, where $\mathcal{L}_0$DROP prefers to drop the encoding of BPEO rather than BPEH.

What's the effect of $\mathcal{L}_0$DROP on Transformer? Transformer can lose the access to around 40% source encodings while largely retaining the same performance. We try to figure out what has changed inside Transformer in order to support $\mathcal{L}_0$DROP,

[9]Function words include *CC, IN, RP, TO, UH, DT* and *WP*. Content words include *MD, JJ, NN, RB* and *VB*. Others include other POS tags except for punctuation and BPEO/BPEH, such as *CD, EX, FW* and *SYM*.

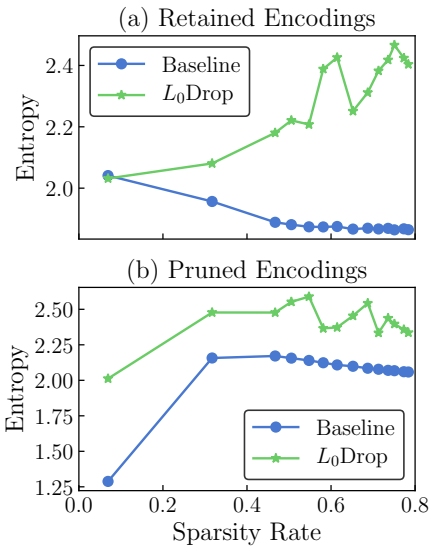


Figure 8: Entropy of the retained source encodings (top) and the pruned ones (bottom) versus the sparsity rate on the WMT14 En-De test set. We use the sparsity variable $\hat{\mathbf{g}}$ learned by $\mathcal{L}_0$DROP to classify the encodings of our baseline Transformer. Higher entropy indicates that the distribution tends to be uniform. With fewer retained encodings, Transformer tends to spread its attention weights to include more source-side information.

and analyze the attention weights (i.e. $\mathbf{A}$ in Eq. (1)) of all encoder-decoder attention sublayers and the last encoder self-attention sublayer; these sublayers are directly connected with $\mathcal{L}_0$DROP in the computation graph. We test on WMT14 En-De.

We visualize the distribution of the encoder-decoder attention weight per source word for Transformer with a sparsity rate of 47% (BLEU 27.06). Compared to the vanilla Transformer (Figure 1), distributions in Figure 7 show that the average attention weight obtained by each source word has increased (+0.77, 1.03→1.80), and the proportion of source words receiving attention weights of less than 0.6 is substantially reduced, by a factor of 10 (49.7%→4.5%). This indicates that $\mathcal{L}_0$DROP forces Transformer to distribute its attention more evenly among retained source encodings.

Apart from the encoder-decoder attention, we also inspect the self-attention in the last encoder layer. We average the self-attention weights over 8 different heads, and compare the attention entropy of the retained source encodings ($\hat{g}_i \neq 0$) and the pruned ones ($\hat{g}_i = 0$). We report average entropy values over the whole test set. Figure 8 shows how increasing sparsity affects the entropy. Although $\mathcal{L}_0$DROP selects to drop uninformative encodings, the increase in the entropy of the retained encodings (Figure 8 (a)), when compared to the baseline, suggests that the encoder actually encodes more context information into these representations, confirming that the model learns to compress context information when sparsity is enforced. Another observation is that the entropy curve of $\mathcal{L}_0$DROP for the pruned encodings is in line with that of the baseline, albeit on a larger scale (Figure 8 (b)). This signifies that $\mathcal{L}_0$DROP enforces Transformer to adapt its attentions to better coordinate with source context representations, which ensures its effectiveness on generation.

Can we prune encodings earlier in the encoder? Rather than stacking $\mathcal{L}_0$DROP on top of the encoder outputs, we insert $\mathcal{L}_0$DROP in-between every adjacent pair of encoder layers. We work on WMT14 En-De and finetune with $\lambda = 0.2$. We get a sparsity rate of 0.0%, 0.0%, 8.6%, 8.6%, 8.7% and 34.0% for the first to the last $\mathcal{L}_0$DROP layer, respectively, with a BLEU score of 26.74. This result suggests that Transformer does not gain much benefit from pruning encodings earlier. The model tends to retain encodings at shallow levels (0.0%/8.6% $<$ 34.0%), and loses 0.3 BLEU compared to its $\mathcal{L}_0$DROP baseline ($\lambda = 0.2$, sparsity rate 31.7%, BLEU 27.04). We believe that the encoder relies on low-level information (including the words) to fully 'understand' the sentence, though part of the final encodings is discardable.

7 Exploring Rule-based Sparse Patterns

Our analysis shows that the sparsity induced by $\mathcal{L}_0$DROP follows certain patterns, with the encodings of 'less content-bearing' words pruned first. This suggests that we may be able to define heuristic patterns manually. In this section, we explore the following three rule-based patterns according to our study on WMT14 En-De:

POS Pattern This pattern discards the source encodings of those easy-to-prune types, including function words, punctuation, BPEO and *MD, EX*, which account for 46.4% of the source-side WMT14 En-De training data.

Freq Pattern Inspired by the fact that punctuation and function words are high frequency words, we propose to filter out the source encodings corresponding to top-frequent words with a threshold of 46.3% (top 100 words). We also include an inverse version, *Inv Freq Pattern*, for comparison, which drops the encodings of most rare words; source words whose fre-

Pattern	WMT14 En-De		CNN/Daily Mail	
	Sparsity	BLEU	Sparsity	RL
Baseline	0.00%	27.59	0.00%	36.88
$\mathcal{L}_0$DROP	46.7%	27.06	47.6%	36.51
POS Pattern	46.7%	27.11	39.6%	35.57
Freq Pattern	42.1%	26.98	47.8%	35.67
Group Pattern	50.0%	26.82	50.0%	30.69
Inv Freq Pattern	44.7%	26.42	39.0%	27.89

Table 2: Sparsity and generation quality for different models on the WMT14 En-De (measured by tokenized case-sensitive BLEU) and the CNN/Daily Mail (measured by ROUGE-L or RL) test set. The sparsity rate is evaluated on test set.

quency ranks lower than 452 are removed, covering ∼40.0% of the source training data.

Group Pattern We explore a position-based pattern that only feeds the encodings at odd positions to the decoder, indicating a sparsity rate of ∼50%. This pattern is partially motivated by Child et al. (2019).

The design of these patterns follows our analysis on $\mathcal{L}_0$DROP, where we match the sparsity rate in each pattern to the optimal rate of $\mathcal{L}_0$DROP on WMT14 En-De. We examine the feasibility of these patterns on WMT14 En-De and CNN/Daily Mail.

Table 2 shows the results. On WMT14 En-De, Transformer using these rule-based patterns achieves comparable translation quality to $\mathcal{L}_0$DROP (-0.24 to +0.05 BLEU) with similar sparsity rate. One interesting observation is that Transformer also works with language- and context-agnostic sparsity patterns (Freq Pattern). The performance drop by Inv Freq Pattern (-0.64 BLEU) is in line with the information-theoretic expectation that information from frequent words is easier to compress than that of rare words.

However, note that we developed our heuristics to mimic the behaviour of $\mathcal{L}_0$DROP for WMT14 En-De task. $\mathcal{L}_0$DROP has the advantage that it is data-driven and task-agnostic so that we can easily apply $\mathcal{L}_0$DROP to summarization. By contrast, these rule-based patterns discovered on translation tasks are not optimal for other tasks, which results in deteriorated performance on CNN/Daily Mail (-5.82 to -0.84 RL). In particular, Transformer suffers from a large performance drop with the Group pattern (-5.82 RL). These results suggest that using rule-based sparse patterns to manually define the sparsity of encoder outputs is possible though the patterns lack generalization ability to different tasks.

8 Conclusion and Future Work

By introducing a L_0-regularized neural sparsity layer ($\mathcal{L}_0$DROP) in Transformer, we confirm that the encoder outputs are compressible to varying degrees. Pruning encoder outputs often results in a drop in performance, but we can get comparable results with 40–70% source encodings dropped. One benefit of pruning source encodings is to shorten encoding sequences for the decoder, which is especially beneficial for efficiency on long sequences, and accelerates the decoding speed by up to 1.65× on document summarization tasks and 1.20× on character-based machine translation. Our analysis on WMT14 En-De shows that $\mathcal{L}_0$DROP learns to drop the encodings of (relatively frequent) function words and retain encodings of (relatively rare) content words, but relies on self-attention to reroute information from these to-be-pruned positions. Based on our analysis, we define rule-based sparsity patterns, which also allow for compression without degrading translation quality much, and show that frequent tokens are more amenable to sparsification than rare tokens. However, we find that our rule-based patterns do not generalize across tasks, while $\mathcal{L}_0$DROP is data-driven and applicable across tasks. We hope that, besides practical implication, our work contributes to better understanding encoder-decoder models.

For future work, we find that the sparsity induced by $\mathcal{L}_0$DROP is highly task-dependent and hardly manipulated. We will develop novel algorithms to make the sparsity induction more controllable.

Acknowledgments

We thank the reviewers for their insightful comments. This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements 825460 (ELITR) and 825299 (GoURMET). Rico Sennrich acknowledges support of the Swiss National Science Foundation (MUTAMUR; no. 176727). Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2963–2977, Florence, Italy. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4295–4305, Brussels, Belgium. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 357–361, Berlin, Germany. Association for Computational Linguistics.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3539–3544, Hong Kong, China. Association for Computational Linguistics.

Yingqiang Gao, Nikola I. Nikolov, Yuhuang Hu, and Richard H.R. Hahnloser. 2020. Character-level translation with self-attention. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1591–1604, Online. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

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

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics*, 5:365–378.

Jindřich Libovický and Alexander Fraser. 2020. Towards reasonably-sized character-level transformer NMT by finetuning subword systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2572–2579, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Vlad Niculae and Mathieu Blondel. 2017. A regularized framework for sparse and structured neural attention. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 3338–3348. Curran Associates, Inc.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Abigail See, Minh-Thang Luong, and Christopher D. Manning. 2016. Compression of neural machine translation models via pruning. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 291–301, Berlin, Germany. Association for Computational Linguistics.

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335, Florence, Italy. Association for Computational Linguistics.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 252–259.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Mingxuan Wang, Jun Xie, Zhixing Tan, Jinsong Su, Deyi Xiong, and Lei Li. 2019. Towards linear time neural machine translation with capsule networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 803–812, Hong Kong, China. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018. Hard non-monotonic attention for character-level transduction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4425–4438, Brussels, Belgium. Association for Computational Linguistics.

J. Zhang, Y. Zhao, H. Li, and C. Zong. 2019. Attention with sparsity regularization for neural machine translation and summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):507–518.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.

A Experimental Settings

Machine Translation We train translation models on the WMT14 English-German translation task (En-De) (Bojar et al., 2014) and the WMT18 Chinese-English translation task (Zh-En) (Bojar et al., 2018). WMT14 En-De and WMT18 Zh-En contain around 4.5M and 25M training sentence pairs, respectively. We use newstest2013 as the validation set for WMT14 En-De and newstest2017 for WMT18 Zh-En. We evaluate the translation quality with BLEU metric (Papineni et al., 2002), and report tokenized BLEU on newstest2014 for WMT14 En-De and detokenized BLEU on newstest2018 for WMT18 Zh-En using *sacreBLEU* (Post, 2018). We apply the byte pair encoding (BPE) algorithm (Sennrich et al., 2016) with 32K merging operations to handle rare words for both translation tasks.

Document Summarization We train abstractive summarization models on the CNN/Daily Mail dataset (Hermann et al., 2015) and the WikiSum dataset (Liu et al., 2018) for single- and multi-document summarization task, respectively. We use the non-anonymized version of CNN/Daily Mail (Gehrmann et al., 2018). We pre-process this dataset with a BPE vocabulary of 32K and truncate each article to 400 subwords (Gehrmann et al., 2018). We use the ranked version of WikiSum (Liu and Lapata, 2019), where top-40 paragraphs are extracted for each instance paired with a summary of 121 words on average. We concatenate all these paragraphs into one source sequence following the given ranking order. CNN/Daily Mail pairs news articles (791 words on average) with multi-sentence summaries (63 words on average), and involves 287,227 training pairs, 13,368 validation pairs and 11,490 test pairs. WikiSum contains 1.58M training pairs, 38,144 validation pairs and 39,357 test pairs. We employ BPE preprocessing following Liu and Lapata (2019) and truncate each source sequence to 2048 subwords. We evaluate the summarization quality using the F_1 score of ROUGE-L (Lin, 2004). The used parameters for ROUGE-1.5.5.pl are *-m -a -n 2*.

Model Settings We formulate all the tasks as sequence-to-sequence tasks, and experiment with the base setting of Transformer (Vaswani et al., 2017): $d = 512$, the middle layer size of $\textsc{Ffn}(\cdot)$ is 2048, and the number of attention head is 8. Following Louizos et al. (2018), we set $\epsilon = -0.1$, and $\beta = 2/3$ for $\mathcal{L}_0\textsc{Drop}$. We tune the hyperparameter λ for different tasks. We augment the MLE loss with label smoothing of 0.1. We use Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.98$) (Kingma and Ba, 2015) for parameter tuning, and schedule the learning rate based on the inverse square root of running steps with a warm-up step of 4K. We apply dropout to attention weights and residual layers to avoid overfitting, with a rate of 0.1/0.1 except for CNN/Daily Mail where 0.3/0.5 is used. We train different models with varied training steps: 300K for WMT14 En-De, 500K for WMT18 Zh-En, 100K for WikiSum and 80K for CNN/Daily Mail, where sequence pairs of roughly 25K target subwords are organized into one minibatch. We average the last 5 checkpoints for evaluation where beam search is adopted for decoding with beam size of 4 and length penalty of 0.6.