Learning to Segment Inputs for NMT Favors Character-Level Processing

Julia Kreutzer[1]* and **Artem Sokolov**[1,2]
[1]Computational Linguistics, Heidelberg University, Germany
[2]Amazon Research, Germany
kreutzer@cl.uni-heidelberg, artemsok@amazon.com

arXiv:1810.01480v3 [cs.CL] 5 Nov 2018

Abstract

Most modern neural machine translation (NMT) systems rely on presegmented inputs. Segmentation granularity importantly determines the input and output sequence lengths, hence the modeling depth, and source and target vocabularies, which in turn determine model size, computational costs of softmax normalization, and handling of out-of-vocabulary words. However, the current practice is to use static, heuristic-based segmentations that are fixed before NMT training. This begs the question whether the chosen segmentation is optimal for the translation task. To overcome suboptimal segmentation choices, we present an algorithm for dynamic segmentation based on (Graves, 2016), that is trainable end-to-end and driven by the NMT objective. In an evaluation on four translation tasks we found that, given the freedom to navigate between different segmentation levels, the model prefers to operate on (almost) character level, providing support for purely character-level NMT models from a novel angle.

1 Introduction

Segmentation of input sequences is an essential preprocessing step for neural machine translation (NMT) and has been found to have a high positive impact on translation quality in recent WMT shared task evaluations (Bojar et al., 2016, 2017). This success can be explained statistically, since shorter segments are beneficial for reducing sparsity: They lower the type-to-token ratio, decrease the number of out-of-vocabulary (OOV) tokens and singletons, which in turn improves the coverage of unseen inputs. Two subword segmentation methods are presently the state-of-the-art in NMT: the *byte-pair encoding* (BPE), that starts with a dictionary of single characters and iteratively creates a new entry from the two currently most frequent entries (Gage, 1994; Sennrich et al., 2016), and a similar, likelihood-based, *wordpiece* (WP) model by Schuster and Nakajima (2012).

While being empirically more successful than word-based NMT, both BPE and WP are preprocessing heuristics, they do not account for the translation task or the language pairs at hand (unless applied to both sides jointly), and require additional preprocessing for languages that lack explicit word separation in writing. Being used in a pipeline fashion, they make it impossible for an NMT system to resegment an unfavorably presplit input and require consistent application of the same segmentation model during testing, which adds an integration overhead and contributes to the 'pipeline jungles' in production environments (Sculley et al., 2015).

On the other extreme from word-based NMT models lie purely character models. Their advantages are smaller vocabularies, thus smaller embedding and output layers, allowing for more learning iterations within a training time budget to improve generalization (Hoffer et al., 2017), and no preprocessing requirements. At the same time, longer input sequences aggravate known optimization problems with very large depths of time-unrolled RNNs (Hochreiter et al., 2001) and may require additional memory for tracking gradients along the unrolling steps.

In this work, we pose the following question: **what would the input segmentations look like if the NMT model could decide on them dynamically?** Instead of heuristically committing to a fixed (sub)word- or character-segmentation level prior to NMT training, this would allow segmentation for each input to be driven by the training objective and avoid solving the trade-offs of different levels by trial and error. To answer this question, we endow an NMT model with the capacity of adaptive segmentation by replacing the conventional

*Work was done while interning at Amazon, Berlin.

lookup embedding layer with a 'smart embedding' layer that sequentially reads input characters and dynamically decides to group a block of them into an output embedding vector, feeding it to the upstream NMT encoder before continuing with the next block (with an optional reverse process on the target side). To signal that a block of characters, encoded as an embedding vector, is ready to be fed upstream, we use accumulated values of a scalar halting unit (Graves, 2016), which learns when to output this block's embedding. It simultaneously affects weighting probabilities of intermediate output vectors that compose the output embedding. Thanks to this weighting, our model is fully differentiable and can be trained end-to-end. Similarly to BPE, it has a hyper-parameter that influences segmentation granularity, but in contrast to BPE this hyper-parameter does not affect the model size. While we evaluate our on-the-fly segmentation algorithm on RNN-based NMT systems, it is transferable to other NMT architectures like CNN-based (Gehring et al., 2017) or Transformer models (Vaswani et al., 2017), since it only replaces the input embedding layer.

Empirically, we find a strong preference of such NMT models to operate on segments that are only one to a few characters long. This turns out to be a reasonable choice, as in our experiments character-level NMT systems of smaller or comparable size were able to outperform word- and subword-based systems, which corroborates results of Chung et al. (2016, 2017). Given this finding and the unique advantages of character-level processing (no pipelining, no tokenization, no additional hyperparameters, tiny vocabulary and memory, and robustness to spelling errors (Lee et al., 2017)), we hope that character-level NMT, and in general character-level sequence-to-sequence learning, will receive more attention from researchers.

Note that, although our character-based models outperform (sub)word-based ones with similar architectures on some datasets, we are not seeking to establish a new state-of-the-art in NMT with our model. Our goal is to isolate the effects of segmentation on quality by introducing a flexibility-enhancing research tool. Therefore, in the comparisons between (sub)word- and character-based models we purposely avoided introducing changes to our baseline RNN NMT architecture beyond upgrading the embedding layer.

2 Related Work

To tackle the OOV problem in word-level models, Luong and Manning (2016) proposed a hybrid model that composes unknown words from characters both on encoder and decoder side. While their approach relies on given word boundaries, they report a purely character-based baseline performing as well as a word-based model with unknown word replacement, but taking three months to train, which seems to have cooled off the NMT community in investigating fully character-based models as an alternative to (sub)word-based ones. Unlike (Luong and Manning, 2016), we found that despite the training speed being slower than for (sub)word vocabularies, it is possible to train reasonable character-level models within a few weeks.

To combine the best of both worlds, Zhao and Zhang (2016) proposed hierarchical en-/decoders that receive inputs on both word- and character-level. The encoder learns a weighted recurrent representation of each word's characters and the decoder receives the previous target word and predicts characters until a delimiter is produced. Similar to our work, they find improvements over BPE models. The idea to learn composite representations of blocks of characters is similar to ours, but their approach requires given word boundaries, which our model learns on-the-fly.

Chung et al. (2016) combined a standard subword-level encoder with a two-layer, hierarchical character-level decoder. The decoder has gating units that regulate the influence of the lower-level layer to the higher-level one, hence fulfilling a similar purpose as our halting unit. This model outperforms a subword-level NMT system, and achieves state-of-the-art on a subset of WMT evaluation tasks. While not requiring explicit segmentation on the target side, the model still relies on given source segmentations.

Finally, Lee et al. (2017) proposed a fully character-level NMT model. They mainly address training speed, which Luong and Manning (2016) identified as a problem, and introduce a low-level convolutional layer over character embeddings to extract information from variable-length character n-grams for higher-level processing with standard RNN layers. In this way, overlapping segments are modelled with a length depending on the convolutional filters.

Perhaps closest to our work is (Chung et al., 2017), where each layer of a hierarchical RNN

encoder is updated at different rates, with the first layer modelling character-level structures, the following modelling sub(word)-level structures. They introduce a binary boundary detector, similar to our halting unit, that triggers feeding of a representation to the next level, so that latent hierarchical structures without explicit boundary information are learnt. Unlike our fully-differentiable model, such discrete decisions of the boundary detector prohibit end-to-end differentiability, forcing a recourse to the biased straight-through estimator (Bengio et al., 2013). On the other hand, while our model relies on a to-be-tuned computation time penalty, Chung et al. (2017) do not impose constraints on the number of boundaries.

Concurrently to our work, Cherry et al. (2018) adapt hierarchical multi-scale RNNs (Chung et al., 2017) to NMT and compare them to several compression algorithms for character-based NMT. Similar to our work, they focus on the encoder and come to the same conclusion: deep recurrent models at character level work surprisingly well.

3 Jointly Learning to Segment and Translate

Instead of committing to a single segmentation before NMT model training, we propose to learn the segmentation-governing parameters along with the usual network parameters in a end-to-end differentiable manner. With this approach, we get rid of pipelining and pre-/postprocessing, and can adaptively segment arbitrary inputs we encounter during training or testing. Our segmentations are context-dependent, i.e. the same substring can be segmented into different parts in different contexts. Being able to smoothly interpolate between word-based and character-based models we allow the model to find a sweet spot in between.

We extend the *Adaptive Computation Time* (ACT) paradigm introduced by Graves (2016), where a general RNN model is augmented with a scalar halting unit that decides how many recurrent computations are spent on each input. For segmentation, we use the halting unit to decide how many inputs (characters) a segment consists of. The output of the ACT module can thus be thought of as an 'embedding' vector for a segment that replaces the classic lookup embedding for (sub)words in standard NMT models. While our model can in principle use larger units as elementary inputs, we will focus on character inputs

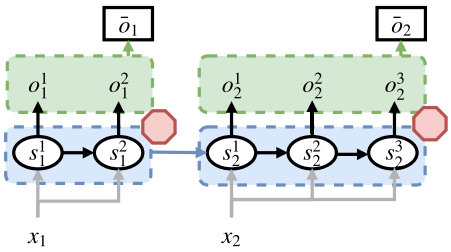


Figure 1: Graves (2016)'s ACT. Each input is repeatedly fed (gray arrows) into the recurrent functions (ellipses) that produce intermediate states and outputs for each input step. When the halting unit halts (red stop sign), intermediate states are summarized (weighted mean, blue), as well as outputs (green). These summaries, respectively, form the input to the following cell's state or are output from the network.

to be able to model the composition of arbitrary segments. That means that we only add a small amount of parameters to a basic character-based model, but explicitly model higher-level merges of characters into subwords.

3.1 ACT for Dynamic Depth

Here we summarize the ACT model (Graves, 2016), depicted in Figure 1. It is applicable to any recurrent architecture that transforms an input sequence $\mathbf{x} = (x_1, \ldots, x_T)$ into outputs $\mathbf{o} = (\bar{o}_1, \ldots, \bar{o}_T)$ via computing a sequence of states $\mathbf{s} = (s_1, \ldots, s_T)$ through a state transition function $\mathcal{S}$ on an embedded input Ex_t and a linear output projection defined by matrix W_o and bias b_o:

$$s_t = \mathcal{S}(s_{t-1}, Ex_t), \qquad o_t = W_o s_t + b_o \quad (1)$$

Instead of stacking multiple RNN layers in $\mathcal{S}$ to achieve increased complexity of an RNN network, the ACT model dynamically decides on the number of necessary recurrent steps (layers) for every input x_t. This saves computation on easy inputs, while still being able to use all of the processing power on hard inputs before emitting outputs. Concretely, an ACT cell performs an arbitrary number of internal recurrent applications of $\mathcal{S}$ for each input x_t:[1]

$$s_t^n = \begin{cases} \mathcal{S}(\bar{s}_{t-1}, Ex_t), & \text{if } n = 1 \\ \mathcal{S}(s_t^{n-1}, Ex_t), & \text{otherwise} \end{cases} \quad (2)$$

[1] In (Graves, 2016) repeated inputs are augmented with a binary flag, which we ignore here for the sake of simplicity.

The total number of internal steps is $N(t) = \min\{n' : \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\}$, where $\epsilon \ll 1$ and h_t^n is the scalar output of sigmoid halting unit,

$$h_t^n = \sigma(W_h s_t^n + b_h). \tag{3}$$

Once halted, the final output $\bar{o}_t$ and state $\bar{s}_t$ (which is fed to the next ACT step in (2)) are computed as weighted means of intermediate outputs and states:

$$\bar{s}_t = \sum_{n=1}^{N(t)} p_t^n s_t^n, \qquad \bar{o}_t = \sum_{n=1}^{N(t)} p_t^n o_t^n \tag{4}$$

where probabilities p_t^n are defined as

$$p_t^n = \begin{cases} R(t), \text{ if } n = N(t) \\ h_t^n, \text{ otherwise} \end{cases} \tag{5}$$

and remainders $R(t) = 1 - \sum_{n'=1}^{N(t)-1} h_t^{n'}$. Finally, to prevent the network from pondering on an input for too long, the remainder $R(t)$ is added as a penalty to the RNN training loss (usually cross-entropy (XENT)) with a weight τ:

$$L_{\text{ACT}} = L_{\text{XENT}} + \tau R(t). \tag{6}$$

Thanks to (4), the model is deterministic and differentiable.

3.2 ACT for Dynamic Segmentation

Dynamic segmentation can either be applied on the source side or the target side or on both. We focus on an ACT-encoder (ACT-ENC) with dynamic segmentation for the source side, and describe an ACT-decoder model for the target side (ACT-DEC), which dynamically segments outputs by compounding output characters, in Appendix A.

Segmenting Encoder. We now describe how to use the ACT paradigm to enhance an encoder for dynamic segmentation on the source side (ACT-ENC). We reuse the idea of halting units, mean field updates and τ-penalized training objective, but instead of learning how much computation is needed for each atomic input, we learn how much computation to allow for an aggregation of atomic inputs, i.e. one segment.

The input to an ACT-ENC cell is a sequence of one-hot-encoded characters $\mathbf{x} = (x_1, \ldots, x_{T_\mathbf{x}})$. The ACT-ENC, depicted in Figure 2, receives one input x_t at a time and decides whether to halt or not. In the case of no halting, the cell proceeds reading more inputs; if it halts, it produces an output 'embedding' $\bar{o}$ of a block of characters read so far, and the cell resets for reading the next block. The sequence of the output embeddings $\mathbf{o} = (\bar{o}_1, \ldots, \bar{o}_{T_\mathbf{o}})$ is then fed to upstream standard (possibly bidirectional) NMT encoder layers, replacing the usual, one-hot encoded, (sub)word lookup embeddings. The length of $\mathbf{o}$ is variable: The more frequently ACT-ENC halts, the more embeddings are generated. In extreme cases, it can generate one embedding per input ($T_\mathbf{o} = T_\mathbf{x}$) or just one embedding for the full sequence of inputs ($T_\mathbf{o} = 1$).

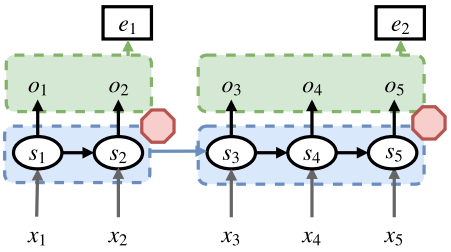


Figure 2: Diagram of the ACT-ENC encoder. Note the differences to the original ACT model: An input is here read on *every* internal recurrent iteration (gray arrows) and the halting unit (red stop sign) is repurposed to trigger feeding of an encoded embedding vector of a block of characters to the upstream NMT layers.

In more detail, ACT-ENC implements the pseudocode given in Algorithm 1. Let $\mathcal{S}(s_{t-1}, i_t)$ be any recursive computation function (in this work we use GRUs) of an RNN that receives a hidden state s_{t-1} and an input vector i_t at time step t and computes the new hidden state s_t. In line 7 this function is computed on the regular previous state or, if there was a halt in the previous step (line 16), on the mean state vector $\bar{s}$ that summarizes the states of the previous segment (line 18, cf. (4), 1st eq.). Per-step outputs o_t are computed from the hidden states s_t with a feed-forward layer (line 8, cf. (1), 2nd eq.). A sigmoid halting unit computes a halting score in each step (line 9, cf. (3)). The halting probability for step t is either the halting score h_t or the current value of remainder $1 - H$ to ensure that all halting probabilities within one segment form a distribution (line 11, cf. (5)). ϵ is set to a small number to allow halting after a single step. Whenever the model decides to halt, an output embedding $\bar{o}$ is computed

Algorithm 1 ACT-ENC

Input: Weights W_o, b_o, W_h, b_h, transition function $\mathcal{S}$, embeddings E_{src}, inputs $\mathbf{x} = (x_1, \ldots, x_{T_\mathbf{x}})$

Output: Outputs $\mathbf{o} = (\bar{o}_1, \ldots, \bar{o}_{T_\mathbf{o}})$, remainder R

1: $\mathbf{o} = [\,]$ ▷ empty sequence
2: $R = 0$ ▷ init remainder
3: $\bar{s} = \mathbf{0}$, $\bar{o} = \mathbf{0}$ ▷ init mean state and output
4: $H = 0$ ▷ init halting sum
5: $s_0 = \mathbf{0}$ ▷ init state
6: **for** $t = 1 \ldots T_\mathbf{x}$ **do** ▷ loop over inputs
7: $s_t = \mathcal{S}(s_{t-1}, E_{src}\, x_t)$ ▷ new state
8: $o_t = W_o s_t + b_o$ ▷ new output
9: $h_t = \sigma(W_h s_t + b_h)$ ▷ halting score
10: $f = [\![H + h_t \geq 1 - \epsilon]\!]$ ▷ halting flag
11: $p_t = (1 - f)\, h_t + f\, (1 - H)$ ▷ halting probability
12: $H = H + h_t$ ▷ update halting sum
13: $\bar{s} = \bar{s} + p_t s_t$ ▷ mean state
14: $\bar{o} = \bar{o} + p_t o_t$ ▷ mean output
15: $R = R + (1 - f)\, h_t$ ▷ increment remainder
16: **if** f **then**
17: $\mathbf{o} = \mathbf{o}^\frown[\bar{o}]$ ▷ append output
18: $s_t = \bar{s}$ ▷ overwrite for next step
19: $\bar{s} = \mathbf{0}$, $\bar{o} = \mathbf{0}$, $H = 0$
20: $R = \frac{1-R}{t}$ ▷ normalize remainder

as a weighted mean of the intermediate outputs of the current segment (line 17, cf. (4), 2nd eq.). The weighted mean on the one hand serves the purpose of circumventing stochastic sampling, on the other hand can be interpreted as a type of intra-attention summarizing the intermediate states and outputs of the segment. The halting scores from each step are accumulated (line 15) to penalize computation time as in (6). The hyperparameter τ here controls the segment length: The higher its value, the more preference will be given to smaller remainders, i.e. shorter segments. We introduce an additional normalization by input length (line 20), such that longer sequences will be allowed more segments than shorter sequences. This implementation exploits the fact that ACT-ENC outputs are weighted means over time steps and updates them incrementally. The algorithm allows efficient minibatch processing by maintaining a halting counter that indicates which embedding each current intermediate output in the batch contributes to. Incremental updates of embeddings and states are achieved with masks depending on the halting position.

Segmenting Decoder. We also implemented a similar segmenting decoder, ACT-DEC (see Appendix A), that 'transcribes' vectors emitted by an NMT decoder into a variable number of characters. Our preliminary experiments with both adaptive input *and* output segmentation capabilities did not improve over using only ACT-ENC with a standard character-level NMT decoder, so in this paper we report only results of the latter configuration.

Comparison to the Original ACT. While our ACT-ENC reuses the ideas of halting units, mean field updates and τ-penalized training objective, it has the following differences to the original ACT: First of all, our model has a different purpose and addresses segmentation, not the alignment of pondering time to input complexity. Instead of learning how much computation is needed for each atomic input, we learn how much computation to allow for an aggregation of atomic inputs. Second, it has a different halting behavior: ACT-ENC allows multiple halts per sequence, not only one per character ($N(t)$ is no longer a function of t). More similar to ACT, our segmenting decoder ACT-DEC (Appendix A) has one halt per input element, but can generate arbitrarily many output characters per input.

4 Experiments

We reimplemented the Groundhog RNN encoder-decoder model with attention by Bahdanau et al. (2015) in `MxNet Gluon` to allow for dynamic computation graphs. To cover a wide range of linguistic diversity, we report results on four language directions and domains, for word-, subword-, character-level and ACT-ENC segmentation: German-to-English TED talks, Chinese-to-English web pages, Japanese-to-English scientific abstracts and French-to-English news. Table 1 gives an overview of the datasets.

Data	Domain	Lang	Train	Dev	Test
IWSLT	TED talks	de-en	153,352	6,970	6,750
CASIA	web	zh-en	1,045,000	2,500	2,500
ASPEC	sci. abstracts	ja-en	2,000,000	1,790	1,812
WMT	news	fr-en	12,075,604	6,003	3,003

Table 1: Domain, language pairs and number of parallel sentences per split for the used datasets.

Preprocessing and Evaluation. The IWSLT data is split and processed as in Bahdanau

et al. (2017); since it comes pretokenized and lowercased[2], models are evaluated with tokenized, lowercased BLEU (using `sacrebleu`[3]) and chrF-score on character bigrams (Popovic, 2015). For WMT, we used the 2014 dataset prepared by Bahdanau et al. (2015)[4], additionally filtering the training data to include only sequences of a lengths 1 to 60, and models are evaluated with cased BLEU and chrF (`sacrebleu`, with the "13a" tokenizer). The CASIA and ASPEC data are, respectively, from the 2015 China Workshop on MT (CWMT), used without additional preprocessing, and from the WAT 2017 SmallNMT shared task, pretokenized with WP. Both datasets have BPE and WP vocabularies of around 16k for each side, and we report cased BLEU and chrF on them.

Hyperparameters. All models are trained with Adam (Kingma and Ba, 2015) and a learning rate of 0.0003, halved whenever the validation score (tokenized BLEU) has not increased for 3 validations. Training stopped when the learning rate has been decreased 10 times in a row. Models are validated every 8000 training instances. All models use recurrent cells of size 1000 for the decoder, with a bidirectional encoder of size 500 for each direction, input and output embedding of size 620, and the attention MLP of size 1000, all following (Bahdanau et al., 2015). When multiple encoders layers are used for character-based models, they are all bidirectional (Chen et al., 2018) with attention on the uppermost layer. The ACT layer for ACT-ENC models has size 50 for IWSLT, CASIA and ASPEC, and 25 for WMT (we picked the ACT size over $\{25, 50, 75, 100, 150\}$). The word-based models on IWSLT and WMT have a vocabulary of 30k for each side. BPE models have separate 15k vocabularies for IWSLT, and a joint 32k vocabulary for WMT. For IWSLT, CASIA and ASPEC all characters from the training data were included in the vocabularies, 117 (de) and 97 (en), 7,284 (zh) and 166 (en), and 3,212 (ja) and 233 (en), respectively. For WMT the vocabularies included the 400 most frequent characters on each side. Word- and BPE-based models are trained with minibatches of size 80, character-based models with 40. The maximum sequence length during training is 60 for word- and BPE-based models, 200 for character-based models and 150 for ACT-ENC, to fit into available memory. Graves (2016) observed that tuning τ was crucial for success of ACT. A suboptimal value of τ, that in our case influences possible segment lengths, might make it hard to achieve good performance in terms of BLEU or chrF. We therefore searched over a range of τs[5] on the dev sets, keeping other hyperparameters fixed: $\tau = 1.0$ delivered the highest BLEU score for IWSLT and CASIA, $\tau = 0.8$ for WMT and $\tau = 0.7$ for ASPEC. Following Graves (2016), we fixed $\epsilon = 0.01$ in all the experiments. During inference, we use beam search with a beam size of 5 and length-normalization parameters $\beta = 0.0, \alpha = 1.0$ (Wu et al., 2016).

Data	Model	BLEU	chrF	Param	SegLen	TrainTime
IWSLT de-en	Word	22.11	0.44	80.5M	4.66	23h
	BPE	25.38	0.49	46.5M	4.09	20h
	Char	22.63	0.46	13.4M	1.00	1d22h
	ACT-ENC	22.67	0.46	13.5M	1.88	9d21h
CASIA zh-en	BPE	10.59	0.37	49.9M	1.72	18h
	Char	12.60	0.40	21.0M	1.00	10d6h
	ACT-ENC	9.87	0.36	21.3M	1.006	3d13h
ASPEC ja-en	WP	21.05	0.53	50.0M	2.07	4d4h
	Char	22.75	0.55	15.6M	1.00	24d15h
	ACT-ENC	15.82	0.46	15.6M	1.0007	15d4h
WMT fr-en	Word	20.32	0.49	80.5M	5.19	4d9h
	BPE	27.02	0.55	86.0M	4.05	3d23h
	Char	24.25	0.53	14.1M	1.00	9d
	ACT-ENC	13.74	0.42	14.2M	1.82	13d8h

Table 2: Evaluation results on respective test sets for 1-layer models, and number of parameters and average source segment lengths on dev sets. Training time to reach stopping criterion is given in (d)ays and (h)ours.

Results. Table 2 lists the results for the most comparable, 1-layer, configuration. BPE models expectedly outperform word-based models, however word-based models are also outperformed by character-based models. The picture is similar w.r.t. the chrF with even smaller relative differences. The ACT-ENC model with one unidirectional ACT layer manages to match the 1-layer bidirectional character-based model on IWSLT. But it does not reach the results of other models on CASIA and ASPEC, which can be explained by increased complexity of doing simultaneous segmentation during training on sentences longer than the average sentence length in IWSLT. However, the main finding here is that ACT-ENC recov-

[2] https://github.com/rizar/actor-critic-public/tree/master/exp/ted

[3] https://pypi.org/project/sacrebleu

[4] http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/

[5] $\{-0.5, 0, 0.001, 0.3, 0.4, 0.45, 0.5, 0.505, 0.55, 0.49, 0.6, 0.7, 0.75, 0.8, 0.85, 0.9, 1.0, 1.5\}$

ers an almost character-level segmentation (compare the "SegLen" column in Table 2). On the IWSLT dev set, the average segment length is only 1.88 (with a maximum of 5 chars per segment). For CASIA and ASPEC domains and with the larger datasets than IWSLT, the ACT-ENC segmentations becomes more fine-grained: The average segment length is, respectively, just 1.006 and 1.0007 on the dev set, with a maximum of 2 chars per segment. Given that the character model outperforms the model with the BPE/WP segmentation, it is not surprising that ACT-ENC converged to the character segmentation.

We hypothesize that ACT-ENC could not improve over the 1-layer bidirectional character model because of complexity of identifying segments in Chinese and Japanese, unidirectionality of its initial layer, and increased hardness of optimization of character-based models with extra non-linearities (Ling et al., 2015), that causes earlier convergence to poorer minima in many runs. Similarly for WMT, failing to match the performance of the character model could be caused by harder optimization task on particularly long sentences in the WMT data, and unidirectionality of ACT-ENC. The ACT-ENC's segment length on the dev set is 1.82 (max. 6 characters per segment), again close on average to a purely character segmentation.

Inspired by the ACT-ENC's recovery of almost character segmentation and by the competitive performance of pure character-based models, we decided to verify if the advantage of character-level processing carries over to multiple layers. Since the character models are much smaller than their word-/BPE-based counterparts, one should allow multiple layers (consuming the same or less memory) to make up for the difference in number of parameters for fairer comparison. This also aimed to verify whether an increased number of non-linearities (one of ACT's benefits (Fojo et al., 2018)) plays a role.

Table 3 shows the test results after tuning the number of bidirectional encoder layers, from 1 to 6, on dev sets. First, we observe the modest parameter number of character models even with multiple layers, that allows them to take advantage of deeper cascades of non-linearities while staying well below the memory budget of (sub)word-based 1-layer models. Second, we discover that BPE/WP models are outperformed by character-based models with multiple encoder layers on two datasets, achieving gains of 2.8 BLEU points on IWSLT, 0.7 BLEU on ASPEC, and losing half a point only on WMT (with a minor decrease in chrF), despite having at least 3.5 times fewer parameters. Such ranking of character- and BPE-based models on WMT might be explained by much longer sentences in the corpus, compared to IWSLT and ASPEC, since the ability of character and ACT-based models to cover unseen input is limited by the maximum training sequence length limit (here 200 characters), which on WMT data crops 30.5% of sentences.

Data	Model	BLEU	chrF	Param	TrainTime
IWSLT de-en	Word, 4-layer	24.54	0.45	97.0M	1d8h
	BPE, 1-layer	25.38	0.49	46.5M	20h
	Char, 5-layer	**28.19**	**0.51**	26.9M	3d10h
	ACT-ENC, 3-layer	25.10	0.49	25.6M	9d7h
CASIA zh-en	BPE, 3-layer	11.01	0.38	58.9M	24h
	Char, 3-layer	**13.43**	**0.42**	30.0M	5d6h
	ACT-ENC, 2-layer	10.35	0.37	21.3M	10d
ASPEC ja-en	WP, 3-layer	22.02	**0.55**	61.4M	4d2h
	Char, 1-layer	**22.75**	**0.55**	15.6M	24d15h
	ACT-ENC, 1-layer	15.82	0.46	15.6M	15d4h
WMT fr-en	Word, 2-layer	21.04	0.48	94.0M	4d16h
	BPE, 3-layer	**27.93**	**0.56**	98.0M	5d3h
	Char, 6-layer	27.23	0.55	27.6M	18d13h
	ACT-ENC, 2-layer	14.01	0.43	21.7M	9d10h

Table 3: Results on respective test sets after tuning number of encoder layers on the dev set.

Translation Analysis. Randomly selected translation examples from the IWSLT dev set and their segmented sources are given in Table 4 (more in Table 7 in Appendix D). In general, when encountering rare inputs, word-based models fail by producing the unknown word token (<unk>), and the BPE-model is able to translate only a more common part of German compounds (e.g. 'tiere' → 'animals'). The character-based models invent words ('altients', 'jes lag') that are similar to strings that they saw during training and the source. In a few cases they fallback to a language-modeling regime having attended to the first characters of a corresponding source word: e.g., instead of translating 'reisen' to 'journeys', the ACT-ENC model translates it to 'rows' (confusing 'reisen' to a similarly spelled German 'reihen'), or 'layering' instead of 'shift work' (confusing 'schichten' to the prefix-sharing 'schichtarbeit'). This is confirmed by attention plots in Figure 6a in Appendix C: The model frequently attends to the correct source word, but mainly to the first characters only; when adding 4 more layers, the

Ref	in social groups of animals , the juveniles always look different than the adults .
Word	in gruppen sozialer tiere sehen die jungtiere immer anders aus als die alttiere . in groups of social animals , the children are always different from the other than the <unk>.
BPE	in gruppen sozialer tiere sehen die jung@@ tiere immer anders aus als die alt@@ tiere . in groups , in groups , the juveniles are seeing the same animals as well as the animals .
ACT-ENC	in| g|ru|pp|en| s|oz|ia|le|r |ti|er|e |se|he|n |d|ie| j|un|gt|ie|re| i|m|m|er| a|nd|er|s |au|s |al|s |d|ie| a|lt|ti|er|e |.| in groups , the juvenile seems to see the different approach than the algaes .
Char	i|n| |g|r|u|p|p|e|n| |s|o|z|i|a|l|e|r| |t|i|e|r|e| |s|e|h|e|n| |d|i|e| |j|u|n|g|t|i|e|r|e| |i|m|m|e|r| |a|n|d|e|r|s| |a|u|s| |a|l|s| |d|i|e| |a|l|t|t|i|e|r|e| |.| in groups of social animals , the juveniles are still in the alite of the altients .
Ref	we 're living in a culture of jet lag , global travel , 24-hour business , shift work .
Word	wir leben in einer zivilisation mit jet-lag , weltweiten reisen , nonstop-business und schichtarbeit . we live in a civilization with <unk> , global travel , <unk> and <unk> .
BPE	wir leben in einer zivilisation mit jet@@ -@@ lag , weltweiten reisen , non@@ sto@@ p-@@ business und sch@@ icht@@ arbeit . we live in a civilization with a single , a variety of global travel , presidential labor and checking .
ACT-ENC	w|ir| l|eb|en| i|n |ei|ne|r |z|iv|il|is|at|io|n |m|it| j|et|-la|g |,| w|el|tw|ei|te|n |re|is|en| ,| n|on|st|op|-bu|si|ne|ss| u|nd| s|ch|ic|ht|ar|be|it| .| we live in a civilization with jes lag , worldwide rows , nonstop business and failing .
Char	w|i|r| |l|e|b|e|n| |i|n| |e|i|n|e|r| |z|i|v|i|l|i|s|a|t|i|o|n| |m|i|t| |j|e|t|-|l|a|g| |,| |w|e|l|t|w|e|i|t|e|n| |r|e|i|s|e|n| |,| |n|o|n|s|t|o|p|-|b|u|s|i|n|e|s|s| |u|n|d| |s|c|h|i|c|h|t|a|r|b|e|i|t| |.| we live in a civilization with jet walk , global journeys , nonstop-business and layering

Table 4: Examples from the IWSLT dev set: segmented sources and greedy translations. Word, BPE and ACT-ENC models have 1 encoder layer, and the character model has 5 layers.

character model develops a behavior to attend to the first positions of source sentence words, see Figure 6b. Note that ACT-ENC segmentations are context-dependent, e.g. occurrences of 'tiere' are segmented differently.

Segmentation Analysis. Table 5 lists the most frequent segments produced by 1-layer ACT-ENC. For IWSLT, we observe that many segments make sense statistically (frequent or rare patterns) and linguistically to some extent: Many of the frequent segments include whitespace (itself a frequent symbol); 2-gram segments amongst others include frequent word suffixes ('en', 'in', 'er'), but also frequent diphthongs ('ei' and 'ie'); 3-grams start with rare characters like 'x' and 'y' or single dashes; 4-grams combine single characters with whitespaces and double dashes; 5-grams cover numbers, in particular, years. Importantly, though, since the best test BLEU scores were obtained by a multi-layer character-based model, the ACT-ENC model has done a reasonable job in improving over the already well-performant strategy, one character per segment, despite having only a single NMT layer.

For CASIA and ASPEC, ACT-ENC converged to a segmentation even closer to pure characters and the longest segments consist of 2 characters. As shown in Table 5, the most frequent 2-grams for CASIA are punctuation marks combined with frequent pronoun 他 or preposition 的, or with the hieroglyph 明 from a common phrase '[smth.] shows, [that]' (all 4-10k in train), and parts of rare English words. For ASPEC, it is mostly the Hiragana letter 'き' that starts the segments. While this letter also occurs as singleton (183 times in the dev set, vs. 52 times as part of a learned segment), and is rather frequent in the training set (239k), it is not the most frequent letter. See Table 6 in Appendix D for translation examples for wordpiece- and character-level models.

For WMT observe the following patterns (Table 5): identified character 2-gram segments are all very frequent in the training data (8-11M occurrences) while longer segments are very rare (max. 1k occurrences) or completely absent from the training data; higher order segments include umlauts (ü, ö), which are parts of the vocabulary, but are atypical for French, except for loan words or proper names in German, which should be treated as one unit semantically. As for IWSLT, we observe that both very frequent and very rare patterns constitute segments.

Gating Behavior. To investigate the reasons for success of the deep character-based encoders and their better or on-par performance with the segmenting ACT-ENC model, we analyzed average activations of GRU gates. A GRU cell computes the next state as: $s_t = z \odot \tanh(x_t W_s + (s_{t-1} \odot r) W_g) + (1 - z) \odot s_{t-1}$, where z is the update gate and r the reset gate, both being outputs of sigmoid layers receiving

Data	Len	Segments
IWSLT	2	en; n␣; er; ␣d; ie; e␣; ei; in; ␣s; ␣w …
	3	yst; -␣d; xtr; -␣u; 100; xpe; -␣w; xis; -␣e; -ge …
	4	--␣d; --␣w; --␣s; --␣i; --␣e; --␣u; --␣g; --␣m; --␣a; --␣k …
	5	1965␣; 969␣,; 1987␣; 1938␣; 1621␣; 1994␣; 1985␣; 1979␣; 1991␣; 1990e …
CASIA	2	”。; ”，; er; ”他; --; ”的; le; 明，; li; ut; …
ASPEC	2	きる; きた; きな; きに; りん; きは; き，; きて; きの; きゅ …
WMT	2	e␣; s␣; ␣d; t␣; ␣l; es; on; ␣a; de; en …
	3	übe; Rüc; rüb; öve; ürs; Köp; üsl
	4	ümov; ölln; rüng; Jürg; ülle; Müsl; Müni; üric; üdig; ürri …
	5	iñera; Mölln; örsdo; hönha
	6	ürdo␣d; ñora B

Table 5: Up to 10 most frequent source segments for a given length for the ACT-ENC on the dev sets.

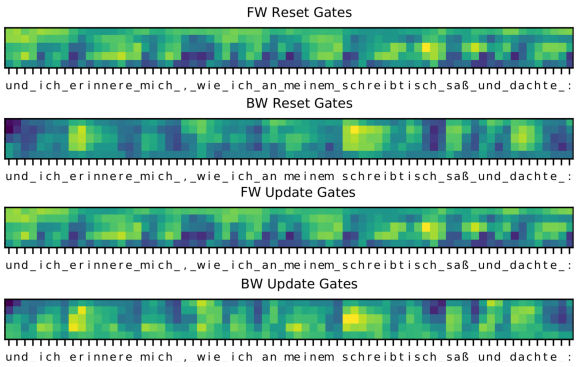


Figure 3: Mean activations for reset and update GRU gates for an IWSLT sentence and the 5-layer character model. The sentence is cropped to a maximum length of 80. Special characters: # stands for padding, | for end of sentence. Dark blue: low values close to 0, bright yellow: high values close to 1.

x_t and s_{t-1} (Cho et al., 2014). Taking a closer look at the average values of these gates, we find patterns of segmentation as depicted in Figure 3 for a 5-layer character model. Most of the time, a whitespace character triggers a visible change of gate behavior: Forward reset gates close (reset) one character after a whitespace and backward reset gates close at whitespaces and then both open at the subsequent character. The update gates show similar regularities, but here the average gate values are less extreme. For longer words all gate activations progressively decay with the length (as also observed for attention in Figure 6b in Appendix C). In addition, the block-wise processing of the compound ‘schreibtisch’ (German: ‘writing table’) that was correctly split into ‘schreib’ and ‘tisch’, points to decompounding abilities that pure character-level models possesses beyond simple whitespace tokenization. The pattern for the 1-layer character model is similar (see Figure 5 in Appendix B), compared to which here the forward update gate gets repurposed, focusing only on the first character, which relates to the attention behavior we also observe in Figure 6a in Appendix C.

Overall, this illustrates that the recurrent gates equip pure character models with the capacity to implicitly model input segmentations, which would explain why ACT-ENC could not find a radically different or advantageous segmentation.

5 Summary & Conclusion

We proposed an approach to learning (dynamic and adaptive) input and output segmentations for NMT by extending the Adaptive Computation Time paradigm by Graves (2016). Experiments on four translations tasks showed that our model prefers to operate on (almost) character level. This is echoed by the quantitative success of purely character-level models and a qualitative analysis of gating and attention mechanisms, suggesting that our adaptive model rediscovers the segmenting capacity already present in gated recurrent, pure character-based models. Given this and the absence of many development hurdles with character-based models (pipelines, tokenization, hyperparameters), their lower memory consumption and higher robustness, the presented dynamic segmentation capacity, being primarily a diagnostic research tool, does not seem to be necessary to be modelled explicitly. We hope these insights can serve as justification for intensification of research in pure character-level NMT models.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, et al. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. In *arXiv:1308.3432*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, et al. 2016. Findings of the 2016 conference on machine translation. In *WMT*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, et al. 2017. Findings of the 2017 conference on machine translation. In *WMT*.

Mia X. Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *arXiv:1804.09849*.

Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. 2018. Revisiting character-based neural machine translation with capacity and compression. In *EMNLP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, et al. 2014. Learning phrase representations using RNN encoder–decoder for Statistical Machine Translation. In *EMNLP*.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2017. Hierarchical multiscale recurrent neural networks. In *ICLR*.

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *ACL*.

Daniel Fojo, Víctor Campos, and Xavier Giró-i Nieto. 2018. Comparing fixed and adaptive computation time for recurrent neural networks. In *Workshop Track of ICLR*.

Philip Gage. 1994. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. In *ICML*.

Alex Graves. 2016. Adaptive computation time for recurrent neural networks. In *arXiv:1603.08983*.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2017. Sockeye: A toolkit for neural machine translation. In *arXiv:1712.05690*.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A field guide to dynamical recurrent neural networks*. IEEE Press.

Elad Hoffer, Itay Hubara, and Daniel Soudry. 2017. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *NIPS*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *TACL*, 5:365–378.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015. Character-based neural machine translation. In *arXiv:1511.04586*.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *ACL*.

Maja Popovic. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and Korean voice search. In *ICASSP*.

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, et al. 2015. Hidden technical debt in machine learning systems. In *NIPS*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, et al. 2016. Minimum risk training for neural machine translation. In *ACL*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al. 2017. Attention is all you need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. In *arXiv:1609.08144*.

Shenjian Zhao and Zhihua Zhang. 2016. Deep character-level neural machine translation by learning morphology. In *arXiv:1608.04738*.

A Segmenting Decoder

After having received the ACT-ENC's output sequence $\mathbf{o}$, i.e. 'embeddings' of character blocks, an NMT encoder-decoder can encode it and decode into a sequence of hidden states as usual. While on the input side the ACT-ENC and the upstream NMT layers are simply stacked onto each other, on the output side the NMT decoder's layers have to be interleaved because of the autoregressive processing and teacher-forcing. Here we describe our implementation geared towards a standard RNN-based NMT decoder, but the model can be easily adapted to other architectures, CNN or Transformer.

The ACT-DEC predicts one target character at a time, the halting unit dictates how many of them per segment. The history input for the RNN state hence consists of a summary of characters of the previous segment, and the history input for the ACT-DEC always consists of the single previous character. Figure 4 depicts the ACT-DEC decoder.

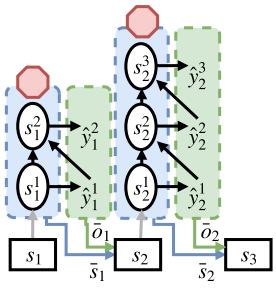


Figure 4: Diagram of the ACT-DEC decoder.

Algorithm 2 describes the ACT computations on the decoder side during inference.[6] While the elementary ACT computations, such as the computation of the ACT state, halting probabilities, remainder and penalty (lines 13, 17, 23) are the same as in ACT-ENC (cf. Algorithm 1), additional complexity is introduced by the fact that the history is computed on the fly. Usually the input to each attention RNN step is the embedded previously generated target symbol, but with ACT we generate an arbitrary number of characters in each RNN step, such that the history is instead a weighted mean over embedded generated symbols, i.e. a summary of the previously generated segment (line 27). The attention[7] is only computed on the RNN level (line 6) with the rationale that alignments are modelled between segments, such that each element within one segment attends to the same source.[8]

The greedy choice of the generated target symbol (line 15) can be replaced by sampling if the training objective requires it (e.g. scheduled sampling (Bengio et al., 2015) or minimum risk training (Shen et al., 2016)).

For standard cross-entropy training the target history in ACT-DEC $\hat{y}_t^n$ (lines 22 and 24) is replaced by the target symbols at the correspond-

Algorithm 2 ACT-DEC

Input: Parameters W_o, b_o, W_h, b_h, function $\mathcal{S}$, output embeddings E_{trg}, decoder RNN $\mathcal{R}$, attention mechanism $\mathcal{A}$, RNN to ACT projection $\mathcal{P}$, output layer $\mathcal{O}$, initial decoder state s_0

Output: Penalty R, output sequence $\hat{Y}$

1: $\hat{Y} = [\,]$ ▷ empty sequence
2: $R = 0$ ▷ init remainder
3: $e_1 = E_{trg}\langle\text{bos}\rangle$ ▷ initial embedded input
4: **for** $t = 1 \ldots T_{max}$ **do** ▷ RNN loop
5: **if** $\langle\text{eos}\rangle \in \hat{Y}$ **then** break
6: $c_t = \mathcal{A}(s_{t-1})$ ▷ attention vector
7: $s_t = \mathcal{R}(s_{t-1}, [e_t; c_t])$ ▷ new RNN state
8: $\bar{s} = \bar{o} = \mathbf{0}$ ▷ init state and output averages
9: $\bar{h} = 0$ ▷ init halting sum
10: $s_t^0 = \mathcal{P}(s_t)$ ▷ init ACT state
11: $i_t^0 = e_t$ ▷ init ACT input
12: **for** $n = 1 \ldots T_{max}$ **do** ▷ ACT loop
13: $s_t^n = \mathcal{S}(s_t^{n-1}, i_t^{n-1})$ ▷ new ACT state
14: $o_t^n = W_o s_t^n + b_o$ ▷ new ACT output
15: $\hat{y}_t^n = \arg\max \mathcal{O}(o_t^n)$ ▷ greedy prediction
16: $\hat{Y} = \hat{Y} \frown [\hat{y}_t^n]$ ▷ append
17: $h_t^n = \sigma(W_h s_t^n + b_h)$ ▷ halt. score
18: $f = [\![\bar{h} + h_t^n \geq 1 - \epsilon]\!]$ ▷ halting flag
19: $H = H + h_t^n$ ▷ increment halting sum
20: $p_t^n = (1 - f)\, h_t^n + f\,(1 - H)$ ▷ halt. prob.
21: $\bar{s} = \bar{s} + p_t^n s_t^n$ ▷ update averages
22: $\bar{o} = \bar{o} + p_t^n E_{trg} \hat{y}_t^n$ ▷ update output
23: $R = R + (1 - f)\, h_t^n$ ▷ increment penalty
24: $i_t^n = E_{trg} \hat{y}_t^n$ ▷ next ACT step input
25: **if** f **or** $\langle\text{eos}\rangle \in \hat{Y}$ **then** break
26: $s_t = \bar{s}$ ▷ next RNN state history
27: $e_t = \bar{o}$ ▷ next RNN input
28: $R = \frac{1-R}{t}$ ▷ compute total penalty

[6] For simplicity we use the same names for ACT parameters as in ACT-ENC. In practice they are not the same weights but can be shared if their sizes agree.

[7] The attention vector can involve a complex computation, as e.g. in (4) of (Hieber et al., 2017).

[8] In the ACT state computation, the attention vector is not fed as input (line 13) but only to the computation of the initial ACT state. It might improve the model if the attention vector was also fed in every ACT step, since connections to the encoder were shorter.

ing positions y_t^n (teacher forcing). Similar to ACT-ENC the ACT-DEC penalty is added to the training loss with coefficient τ:

$$L_{\text{ACT-ENC-DEC}} = L_{\text{ACT-ENC}} + \tau\, R. \qquad (7)$$

A larger τ will again prefer smaller R, i.e. smaller remainders with result from shorter segments.

A.0.1 Limitations.

Since in every decoder step an arbitrary number of characters can be generated the comparison of beam search hypotheses becomes hard and its implementation non-trivial. Another challenge is efficient minibatching: Due to the flexible breaking conditions in the nested loops (line 5 and 25) and therefore variable output lengths, there can be a lot of overhead in computation time for each batch. When working with minibatches, the breaks are only executed as soon as every element of the batch fulfills the condition. During training, when the reference output length is known, there are two extreme cases that can occur in the same batch: 1) full number of RNN steps required, i.e. ACT-DEC halts after every step, 2) full number of ACT-DEC steps required, i.e. ACT-DEC does not halt. In practice, however, we observe roughly coordinated halting behaviour for instances in the same batch, which gives reason to believe that the worst case scenario is rare.

B Gating Behaviour for One-Layer Character Models

We plot the average update and reset gates activations for a single layer character model in Figure 5. As for the case of 5-layer model (Figure 3) we also observe large changes of their amplitude on whitespaces and punctuation, and on German compound words.

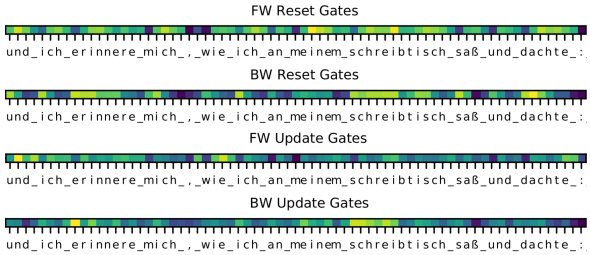
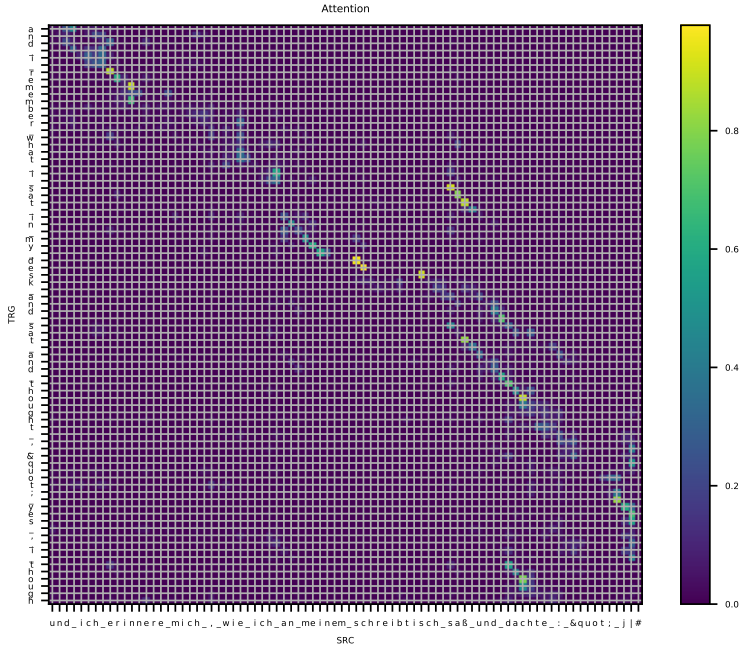
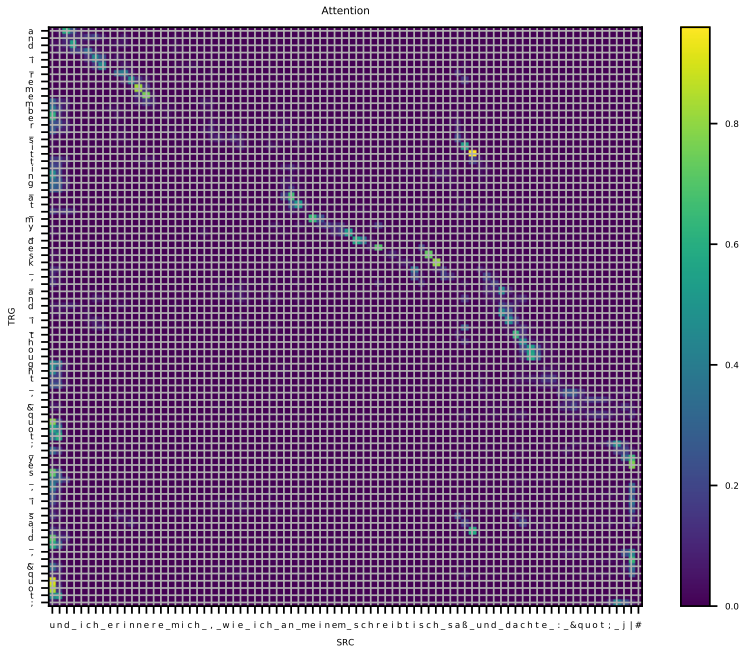


Figure 5: 1-layer encoder: Mean activations for reset and update GRU gates for an IWSLT sentence. The sentence is cropped to a maximum length of 80. Special symbols: # stands for padding, | for end of sentence. Dark blue: low values close to 0, bright yellow: high values close to 1.

C Attention Plots



(a) 1 layer



(b) 5 layers

Figure 6: Attention scores for 1 and 5-layer encoder character-based models. The sentences are cropped to a maximum length of 80. Special symbols: # stands for padding, | for end of sentence.

D Translation Examples on ASPEC and WMT

Table 6 lists examples for BPE and character-level models for ASPEC. In the first example, both BPE and character-based models struggle with the first part of the translation ('second' vs. 'middle ' vs. 'medium') and the long noun-phrase at the end. In both examples the ACT-ENC system suffers from repetitions of phrases.

Table 7 presents examples from the WMT dev set for different models. The first example's source is incomplete with a missing last word and a period (they appear in the reference). The word-based models shows no hallucination behavior, while BPE- and character-based models make up additional words ('the company's second stage', 'people') or mistranslate verbs ('gone', 'gained', 'fame'). The missing character in 'orgnisés' in the second example is an interesting showcase that contributes to our advocacy of using character-based models: Both character-based and our ACT-ENC models manage to correct this typo due to their strong language-modeling abilities (that were somewhat detrimental for IWSLT in Table 4). All models have difficulties with the rare 'clou', being translated as <unk>, 'bell', 'club' or 'cloud'. While the word-based model can only output <unk>, the subword models try to find or translate a word that it similar to 'clou' (French for 'bell' is 'cloche').

Ref	With the second one, sodium hypochlorite is injected for detection of a compound and measurement of gas produced by reaction with it.
WP	▁中 者 では ， 次 亜 塩素 酸 ナ ト リウム を注入し ， 生成 成分 の検出 とその 反応 に伴う ガス の測定 を行う 。 In the middle part, sodium hypochlorite is injected, and the detection of the generated component and the gas with the reaction are carried out.
ACT-ENC	中|者|で|は|，|次|亜|塩|素|酸|ナ|ト|リ|ウ|ム|を|注|入|し|，|生|成|成|分|の|検|出|と|そ|の|反|応|に|伴|う|ガ|ス|の|測|定|を|行|う|。| In the medium with the sodium chloride, the detection of generation and the reaction with the reaction was measured with the detection of the formatio
Char	中|者|で|は|，|次|亜|塩|素|酸|ナ|ト|リ|ウ|ム|を|注|入|し|，|生|成|成|分|の|検|出|と|そ|の|反|応|に|伴|う|ガ|ス|の|測|定|を|行|う|。| In the middle of the method, sodium hypochlorite is injected and the detection of the production component and the measurement of gas with the reaction are carried out.
Ref	In case of the beam subjected to axial tensile force, the shear crack position and its angle can be changed by the size of axial force.
WP	▁軸方向 引張 力 を受ける 梁 の場合 ， せん断 ひび割れ 位置 及びその 角度 は 軸力 の大きさ によって 変化する 。 In the case of beam subjected to axial tensile force, the crack opening position and its angle changes with the axial force.
ACT-ENC	軸|方|向|引|張|力|を|受|け|る|梁|の|場|合|，|せ|ん|断|ひ|び|割|れ|位|置|及|び|そ|の|角|度|は|軸|力|の|大|き|さ|に|よ|っ|て|変|化|す|る|。| In the case of beam case of axial tension, the shear strain convection position and the angle of the axial force changes with the size of the axial forc
Char	軸|方|向|引|張|力|を|受|け|る|梁|の|場|合|，|せ|ん|断|ひ|び|割|れ|位|置|及|び|そ|の|角|度|は|軸|力|の|大|き|さ|に|よ|っ|て|変|化|す|る|。| In the case of a beam which received axial tension, the shear crack position and their angle varies with the size of the axial force.

Table 6: Examples from the ASPEC dev set: segmented sources and greedy translations. Word, BPE and ACT-ENC models have 1 encoder layer, and the character model has 2 layers.

Ref	In Montenegro they won 1:0 and celebrate a 200 million windfall.
Word	Ils ont gagné au Monténégro 1 : 0 et fêtent la qualification pour 200 millions They won in Montenegro 1 : 0 and <unk> the qualification for 200 million .
BPE	Ils ont gagné au Monténégro 1 : 0 et f@@ ê@@ tent la qualification pour 200 millions They have gained in Montenegro 1 : 0 and fame the qualification for the 200 million people .
ACT-ENC	Il|s |on|t |ga|gn|é |au| M|on|té|né|gr|o |1 |:| 0| e|t |f|ê|t|en|t |la| q|ua|li|f|ic|at| io|n |p|ou|r |20|0 |mi|ll|io|ns| They have gone to Montenegro 1 : 0 and celebrating the qualification for 200 million
Char	I|l|s| |o|n|t| |g|a|g|n|é| |a|u| |M|o|n|t|é|n|é|g|r|o| |1| |:| |0| |e|t| |f|ê|t|e|n|t |l|a| |q|u|a|l|i|f|i|c|a|t|i|o|n| |p|o|u|r| |2|0|0| |m|i|l|l|i|o|n|s| They won the company 's second stage in Montenegro 1 : 0 and celebrate the qualification for 200 million
Ref	The main focus of the festival is on two concerts taking place on November 17.
Word	Le clou du festival est formé de deux concerts orgnisés le 17 novembre . The <unk> of the festival is composed of two concerts on 17 November .
BPE	Le clo@@ u du festival est formé de deux concerts or@@ gn@@ isés le 17 novembre . The festival 's bell is composed of two concerts , on 17 November .
ACT-ENC	L|e c|lo|u |du| f|es|ti|v|al| e|st| f|or|mé| d|e |de|ux| c|on|c|er|ts| o|rg|ni|sé|s |le| 1|7 |no|v|em|b|re| .| The festival club is the form of two concerts organized on 17 November .

Table 7: Examples from the WMT dev set: segmented sources and greedy translations. Word, BPE and ACT-ENC models have 1 encoder layer, and the character model has 6 layers.