On Biasing Transformer Attention Towards Monotonicity

Annette Rios[1], **Chantal Amrhein**[1], **Noëmi Aepli**[1] and **Rico Sennrich**[1,2]
[1]Department of Computational Linguistics, University of Zurich
[2]School of Informatics, University of Edinburgh
{rios,amrhein,naepli,sennrich}@cl.uzh.ch

Abstract

Many sequence-to-sequence tasks in natural language processing are roughly monotonic in the alignment between source and target sequence, and previous work has facilitated or enforced learning of monotonic attention behavior via specialized attention functions or pretraining. In this work, we introduce a monotonicity loss function that is compatible with standard attention mechanisms and test it on several sequence-to-sequence tasks: grapheme-to-phoneme conversion, morphological inflection, transliteration, and dialect normalization. Experiments show that we can achieve largely monotonic behavior. Performance is mixed, with larger gains on top of RNN baselines. General monotonicity does not benefit transformer multihead attention, however, we see isolated improvements when only a subset of heads is biased towards monotonic behavior.

1 Introduction

Many sequence-to-sequence tasks in natural language processing are roughly monotonic in the alignment between source and target sequence, and previous work has focused on learning monotonic attention behavior either through specialized attention functions (Aharoni and Goldberg, 2017; Raffel et al., 2017; Wu and Cotterell, 2019) or pretraining (Aji et al., 2020). However, it is non-trivial to port specialized attention functions to different models, and recently, Yolchuyeva et al. (2019); Wu et al. (2021) found that a transformer model (Vaswani et al., 2017) outperforms previous work on monotone tasks such as grapheme-to-phoneme conversion, despite having no mechanism that biases the model towards monotonicity.

In the transformer, it is less straightforward to what extent individual encoder states, especially in deeper layers, still represent distinct source inputs after passing through several self-attention layers. Consequently, it is unclear whether enforcing monotonicity in the transformer is as beneficial as for recurrent neural networks (RNNs).

In this paper, we investigate the following research questions:

1. How can we incorporate a monotonicity bias into attentional sequence-to-sequence models such as the transformer?

2. To what extent does a transformer model benefit from such a bias?

Specifically, we want to incorporate a monotonicity bias in a way that is agnostic of the task and model architecture, allowing for its application to different sequence-to-sequence models and tasks. To this end, we introduce a loss function that measures and rewards monotonic behavior of the attention mechanism.[1]

We perform experiments and analysis on a variety of sequence-to-sequence tasks where we expect the alignment between source and target to be highly monotonic, such as grapheme-to-phoneme conversion, transliteration, morphological inflection, and dialect normalization and compare our results to previous work that successfully applied hard monotonic attention to recurrent sequence-to-sequence models for these tasks (Wu et al., 2018a; Wu and Cotterell, 2019).

Our results show that a monotonicity bias learned through a loss function is capable of making the soft attention between source and target highly monotonic both in RNNs and the transformer. We find that this leads to a similar improvement to previous works on hard monotonic attention for RNNs, whereas for transformer models, the results are mixed: Biasing all attention heads towards monotonicity may limit the representation power of multihead attention in a way

[1]Code and scripts available at: https://github.com/ZurichNLP/monotonicity_loss

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4474–4488
June 6–11, 2021.

that is harmful even for monotonic sequence-to-sequence tasks. However, for some tasks, we see small improvements when limiting monotonicity to only a subset of heads.

2 Related Work

Attention models (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017) are a very powerful and flexible mechanism to learn the relationship between source and target sequences, but the flexibility might come at the cost of making the relationship harder to learn. Previous work has shown that their performance can be improved by introducing inductive biases. Cohn et al. (2016) introduce various structural alignment biases into a neural machine translation model, including a positional bias. While this bias is motivated by the fact that a given token in the source often aligns with a target token at a similar relative position, it does not explicitly encourage monotonicity.

In contrast, Raffel et al. (2017) propose to modify the attention mechanism to learn hard monotonic alignments instead of computing soft attention over the whole source sequence. Several extensions have been proposed: having a pointer monotonically move over the source sequence and computing soft attention on a local window (Chiu and Raffel, 2018) or from the beginning of the sequence up to the pointer (Arivazhagan et al., 2019). For tasks like simultaneous translation and automatic speech recognition, the main benefit from hard monotonic attention is that decoding becomes faster and can be done in an online setting. However, many sequence-to-sequence tasks behave roughly monotonic and biasing the attention towards monotonicity can improve performance; especially in low-resource settings. Aharoni and Goldberg (2017) show that hard monotonic attention works well for morphological inflection if it mimics an external alignment.

Wu et al. (2018b) propose a probabilistic latent-variable model for hard but non-monotonic attention which Wu and Cotterell (2019) later extend to exact hard monotonic attention. In contrast to Aharoni and Goldberg (2017), the alignment is learned jointly with the model. Their approach outperforms several other models on grapheme-to-phoneme conversion, transliteration, and morphological inflection. Monotonic attention has also improved tasks such as summarization (Chung et al., 2020) and morphological analysis (Hwang and Lee, 2020).

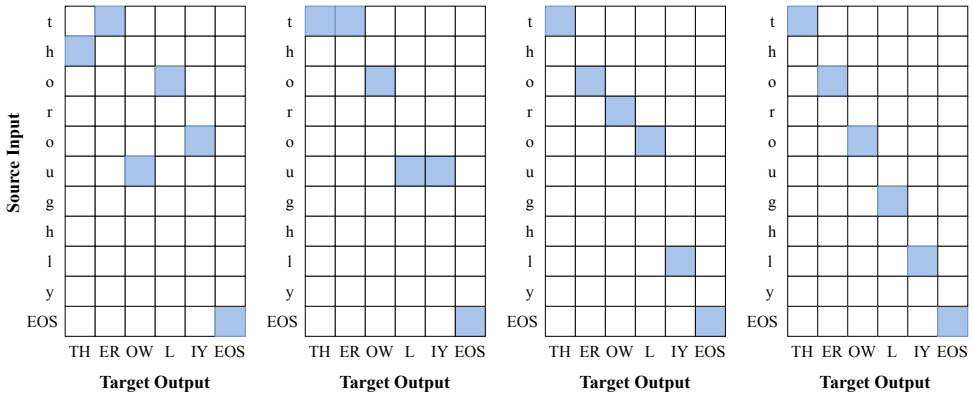
Recently, the transformer architecture (Vaswani et al., 2017) has outperformed RNNs in low-resource settings for character-level transduction tasks (Yolchuyeva et al., 2019; Wu et al., 2021) and neural machine translation (Araabi and Monz, 2020). While there has been some work on extending the methods of Raffel et al. (2017); Chiu and Raffel (2018); Arivazhagan et al. (2019) to multi-head attention (Ma et al., 2020; Liu et al., 2020), we are not aware of any work that studied monotonicity in transformers for monotonic tasks, such as grapheme-to-phoneme conversion, transliteration, or morphological inflection.

To this end, we propose a model-agnostic monotonicity loss that can seamlessly be integrated into RNNs as well as the transformer. Our monotonicity loss captures how monotone the soft attention behaves during training, while two hyperparameters allow us to control how much monotonicity is enforced. By encouraging monotonicity through a loss instead of a modification of the attention mechanism, our implementation still brings all the benefits of soft attention to tasks where fast, online inference is not paramount and allows us to explore various trade-offs between unconstrained and fully monotonic attention.

3 Monotonicity Loss

We now introduce our monotonicity loss function. The loss function is differentiable and compatible with standard soft attention mechanisms and is thus easy to integrate into popular encoder-decoder architectures such as the transformer. On a high level, we compare the attention distribution between decoder time steps in a pairwise fashion and measure whether the mean attended position increases for each pair.

Let us denote the input sequence as $X = (x_1, ..., x_{|X|})$, and the output sequence as $Y = (y_1, ..., y_{|Y|})$. The interface between the encoder and decoder is one or several attention mechanisms. In its general form, the attention mechanism computes some energy e_{ij} between a decoder state at time step i and an encoder state j. While this energy function varies, with popular choices being a feedforward network (Bahdanau et al., 2015) or (scaled) dot-product (Luong et al., 2015; Vaswani et al., 2017), they are typically normalized to a vector of attention weights α using the softmax



margin:	loss:	margin:	loss:	margin:	loss:	margin:	loss:
0	0.364	0	0.000	0	0.000	0	0.000
0.5	0.530	0.5	0.167	0.5	0.000	0.5	0.000
1	0.697	1	0.333	1	0.152	1	0.000

Figure 1: Average attention positions between target output characters and source input characters and the corresponding monotonicity loss for different attention distributions, and with different margins δ. The average attention positions were rounded to integers for visualization purposes.

function:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{|X|} \exp(e_{ik})} \quad (1)$$

These attention weights are then applied to obtain a weighted average c_i of a vector of value states V:

$$c_i = \sum_{j=1}^{|x|} \alpha_{ij} \cdot v_j \quad (2)$$

For our monotonicity loss, we also compute the mean attended position $\bar{a}_i$:

$$\bar{a}_i = \sum_{j=1}^{|x|} \alpha_{ij} \cdot j \quad (3)$$

We can then define the monotonicity loss in a pairwise fashion, comparing the mean attended position at time steps i and $i+1$:

$$L_{\text{mono}} = \sum_{i=1}^{|Y|-1} \max\left(\frac{\bar{a}_i - \bar{a}_{i+1} + \delta\frac{|X|}{|Y|}}{|X|}, 0\right) \quad (4)$$

δ is a hyperparameter that controls how deviations from the main diagonal are penalized. Let us first consider the case with $\delta = 0$: if $\bar{a}_{i+1} \geq \bar{a}_i$ for all positions i, i.e. if the mean attended position is weakly increasing[2], then the loss is 0. Any decrease in the mean attended position will incur a cost that is proportional to the amount of decrease, relative to the source sequence length;[3] this allows differentiation of the loss, and will also serve as a measure of the degree of monotonicity in the analysis.

We might want to bias the model towards strictly monotonic behavior, penalizing it if $\bar{a}$ remains unchanged over several time steps. We can achieve this by incurring a loss if $\bar{a}$ does not increase by some margin, controlled by δ. At the most extreme, with $\delta = 1$, the loss is minimized if the mean attended position follows the main diagonal of the alignment matrix, increasing by $\frac{|X|}{|Y|}$ at each time step. Figure 1 shows how the margin δ can influence the monotonicity loss with some examples.

In equation 4, costs are later summed over the target sequence. In practice, we normalize the cost by the number of tokens in a batch for training stability, as is typically done for the cross-entropy loss. If a model has multiple attention mechanisms, e.g. attention in multiple layers, or multihead attention, we separately compute the loss for each attention mechanism, then average the losses. We can also just apply the loss to a subset of attention mechanisms, allowing different attention heads to learn specialized behavior (Voita et al., 2019).

[2] We can swap $\bar{a}_i$ and $\bar{a}_{i+1}$ in equation 4 to bias the model towards monotonically decreasing attention.

[3] Making the cost relative to the source sequence length ensures that the worst-case cost per timestep is independent of source sequence length.

4 Experiments

4.1 Models and Data

We implement the loss function in sockeye (Hieber et al., 2018), and experiment with RNN and transformer models. We list the specific baseline settings for each task in Appendix A.2.

The monotonic loss function is controlled by a hyperparameter for the margin (δ), and an additional scaling factor for the loss itself (λ). Preliminary experiments have shown that the monotonicity loss has an undesirable interaction with attention dropout, which is commonly used in transformer models. Randomly dropping attention connections during training makes it harder to reliably avoid a decrease in the mean attended position, favoring a degenerate local optimum where attention resides constantly on the first (or last) encoder state. To avoid this problem, we use DropHead (Zhou et al., 2020) instead, which has a similar regularizing effect as attention dropout, but does not interact with the monotonicity loss. In addition to the standard evaluation metrics used in each task, we provide the monotonicity loss on the test set and the percentage of target tokens for which the average source attention position has increased (by some margin).

We perform experiments on three word-level and one sentence-level sequence-to-sequence tasks:

Grapheme-to-Phoneme Conversion

For grapheme-to-phoneme conversion, we use NETtalk (Sejnowski and Rosenberg, 1987)[4] and CMUdict,[5] two datasets for English, with the same data split as Wu and Cotterell (2019). For experiments with RNN models, we follow the settings in Wu et al. (2018b) (large configuration).[6]

For experiments with transformer models, we follow the settings suggested in Wu et al. (2021), however, we use dropout rates of 0.3 (NETtalk) and 0.2 (CMUdict) instead of 0.1 and 0.3. Furthermore, we use a smaller feed-forward dimension for the NETtalk models (512 instead of 1024), since this a relatively small dataset (∼14k samples).

For both RNN and transformer models, we use early stopping with phoneme error rate, as opposed to a minimum learning rate value as in Wu et al. (2018b) and Wu et al. (2021). We evaluate our models with word error rate (WER) and phoneme error rate (PER).

`vanilla`

V	SG	3	PRS	<sep>	u	s	e
0	1	2	3	4	5	6	7

`separator-centered`

V	SG	3	PRS	<sep>	u	s	e
-4	-3	-2	-1	0	1	2	3

Figure 2: Vanilla and our proposed separator-centered positional encoding for the input "use V;SG;3;PRS" in the morphological inflection task.

Morphological Inflection

For morphological inflection, we use the CoNLL-SIGMORPHON 2017 shared task dataset.[7] We choose all 51 languages from the high-resource setting where the training data for each language consists of 10,000 morphological tags + lemma and inflected form pairs (except for Bengali and Haida which have 4,243 and 6,840 pairs respectively) and from the medium-resource setting with 1,000 training examples per language. Our baselines performed very poorly on the low-resource setting with only 100 training examples and we decided to focus on the other two tasks instead.

We preprocess the data to insert a separator token between the morphological tags and the input lemma. The monotonicity loss is then only computed on the positions to the right of the separator token's position. We follow Wu et al. (2021) and use special positional encodings for the morphological tags in the transformer. Unlike their approach, where the position for all tags was set to 0, we set the position of the separator token to 0 and sequentially decrease the positions of the morphological tags to the left (Figure 2). This serves to stabilize the positional encodings of the lemma tokens, while still accounting for the fixed order of morphological tags in the dataset. In preliminary experiments, we observed an improvement of 0.63% in accuracy over vanilla positional encodings.

We train models on character-level for morphological inflection following the previously recommended settings for RNNs in Wu et al. (2018b)

[4] https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+(Nettalk+Corpus)

[5] https://github.com/cmusphinx/cmudict

[6] Even though we follow the settings in Wu et al. (2018b), our RNN models are smaller than theirs (4.5M vs. 8.6M parameters).

[7] https://github.com/sigmorphon/conll2017

and for transformers in Wu et al. (2021) (except for reducing the feed-forward dimension to 512 instead of 1024). For the high resource datasets, we use a batch size of 400, for the medium resource datasets 200. Early stopping is done in the same way as for grapheme-to-phoneme conversion. We use the official evaluation script to compute word-level accuracy (ACC) and character-level edit distance (LEV).

Transliteration

For transliteration, we experiment on the NEWS2015 shared task data (Zhang et al., 2015) and use the same subset of 11 script pairs that Wu and Cotterell (2019) used in their experiments: AR-EN, EN-BA, EN-HI, EN-JA, EN-KA, EN-KO, EN-PE, EN-TA, EN-TH, JN-JK, and TH-EN. Total training dataset sizes range from 6,761 source names for EN-KO up to 27,789 source names for EN-TH. For certain script pairs, multiple transliterations per source name are acceptable. We add all possible pairs to our training data, which only has a large effect on EN-AR, where there are on average 10 acceptable transliterations per source name. Since the references of the official shared task test sets were not released, we follow Wu and Cotterell (2019) and use the development set as our test set. We randomly sample 1,000 names from the training sets as our development sets for script pairs with more than 20,000 training examples and 100 for script pairs with fewer training examples.

Again, we follow Wu et al. (2018b) for hyperparameters in RNNs and Wu et al. (2021) in transformers (smaller feed-forward dimensions of 512). We early stop training as for grapheme-to-phoneme conversion. We evaluate our models following Zhang et al. (2015) and compute word-level accuracy (ACC) and character-level mean F-score (MFS). The formula for MFS is in Appendix A.1.

Dialect Normalization

For this work, we consider dialect normalization as a machine translation task from dialect to standard. We work with the dataset described in Aepli and Clematide (2018), which consists of 26,015 crowd-sourced German translations of 6,197 original Swiss German sentences. We use three documents (10%) as test sets and randomly split the rest in development and training set (10% and 80% respectively). The alignment between Swiss German and the German translations is highly monotonic, but there are occasional word order differences, as illustrated in Figure 3.

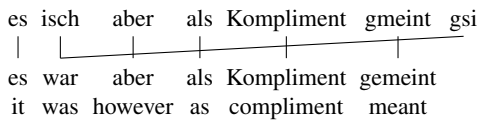


Figure 3: Swiss-German to German dialect normalization example with verb reordering.

The models are trained on subwords obtained via BPE (Sennrich et al., 2016), created with subword-nmt computing 2000 merges. We treat this as a low-resource machine translation task, and thus follow hyperparameters by Sennrich and Zhang (2019) for the RNN models, while the transformer models are trained according to Araabi and Monz (2020). We evaluate our models with BLEU (Papineni et al., 2002).[8]

4.2 Results

In addition to task-specific evaluation metrics, we use the loss function to score the monotonicity of the attention on the test set for all models (reported as L_{MONO}). Furthermore, we report the percentage of decoding states for which the average source attention position $\bar{a}$ increases by at least $\delta\frac{|X|}{|Y|}$ as $\%mono$. In other words, this is the percentage of states for which the pairwise loss is 0.

Grapheme-to-Phoneme Conversion

We test different settings on the grapheme-to-phoneme task, see Table 1 for results with RNNs (top) and transformers (bottom). We find that models trained with the additional loss have more monotonic attention than the baselines (see $\%mono$ and L_{MONO}). We observe large differences both in terms of WER and PER across multiple runs for the baseline, especially for the small data set.[9] We therefore report the average result of three runs with standard deviations for each model.

Attention in the RNN baselines is already quite monotonic, but we observe small improvements with $\delta = 0.5$. For transformer models, on the other hand, $\delta > 0$ seems to harm the performance, therefore we only report results with $\delta = 0$. In general, multihead attention in the transformer does not seem to benefit much from enforced monotonicity.

[8] SacreBLEU (Post, 2018): BLEU+case.mixed+numrefs.1 +smooth.exp+tok.13a+version.1.3.6

[9] Standard deviation on NETtalk with RNN is >1.2 WER and >0.27 PER across baseline runs.

	Grapheme-to-Phoneme Conversion							
			$\delta = 0.0$		$\delta = 0.5$		$\delta = 1.0$	
	WER ↓	PER ↓	$\%mono$	L_{MONO}	$\%mono$	L_{MONO}	$\%mono$	L_{MONO}
RNN								
Wu et al. (2018b)	28.20	6.8						
Wu and Cotterell (2019)	28.20	6.9						
baseline ($\lambda = 0$)	28.76±0.73	7.16±0.16	84.7%	2.91e-04	84.0%	3.94e-03	26.8%	1.03e-01
$\lambda = 0.1, \delta = 0.0$	28.88±0.32	7.16±0.03	84.8%	1.24e-04				
$\lambda = 0.1, \delta = 0.5$	**28.55**±0.18	**7.13**±0.09			84.3%	1.74e-03		
$\lambda = 0.1, \delta = 1.0$	29.02±0.55	7.32±0.19					44.5%	4.05e-02
Transformer								
Wu et al. (2021)	27.63	6.9						
baseline	**27.79**±0.24	**7.00**±0.09	77.0%	7.26e-02				
$\lambda = 0.1, \delta = 0.0, h$ = all:	27.99±0.60	7.11±0.18	84.6%	5.12e-05				

Table 1: Results for grapheme-to-phoneme, monotonicity loss for transformer on all layers and heads. Average over three runs with independent seeds with $\lambda = 0.1$. Our best models are marked in bold.

	Morph. Infl. High Resource				Morph. Infl. Medium Resource			
	ACC ↑	LEV ↓	$\%mono$	L_{MONO}	ACC ↑	LEV ↓	$\%mono$	L_{MONO}
RNN								
Wu et al. (2018b)	93.60	0.128	-	-				
Wu and Cotterell (2019)	94.81	0.123	-	-				
baseline ($\lambda = 0$)	**94.97**±0.06	**0.098**±0.002	65.5%	1.17	**78.15**±0.24	**0.441**±0.005	64.6%	1.16
$\lambda = 0.1, \delta = 0.0$	94.63±0.01	0.105±0.002	83.5%	3.78e-4	74.11±0.35	0.560±0.009	84.3%	1.63e-3
Transformer								
Wu et al. (2021)	95.59	0.088	-	-				
baseline ($\lambda = 0$)	**95.05**±0.03	**0.097**±0.001	58.1%	1.34	**81.33**±0.02	**0.378**±0.001	58.1%	1.35
$\lambda = 0.1, \delta = 0.1, h$ = all	94.98±0.07	0.099±0.002	87.5%	4.49e-4	81.02±0.17	0.383±0.001	85.7%	1.45e-3

Table 2: Results for morphological inflection, monotonicity loss for transformer on all layers and heads. Average over three runs with independent seeds with $\lambda = 0.1$. Our best models are marked in bold.

	Transliteration				Dialect Normalization		
	ACC ↑	MFS ↑	$\%mono$	L_{MONO}	BLEU ↑	$\%mono$	L_{MONO}
RNN							
Wu et al. (2018b)	41.10	89.40	-	-			
Wu and Cotterell (2019)	41.20	89.50	-	-			
baseline ($\lambda = 0$)	39.53±0.56	89.06±0.06	74.4%	0.06	**33.41**±0.39	83.1%	0.42
$\lambda = 0.1, \delta = 0.0$	**40.03**±0.39	**89.18**±0.04	81.7%	1.4e-3	33.29±0.23	90.2%	0.10
Transformer							
Wu et al. (2021)	43.39	89.70	-	-			
baseline ($\lambda = 0$)	**42.08**±0.55	**89.63**±0.04	69.1%	0.12	**32.83**±0.20	71.7%	1.23
$\lambda = 0.1, \delta = 0.0, h$ = all	41.32±0.53	89.47±0.08	82.2%	7.1e-4	32.17±0.78	91.1%	0.05

Table 3: Results for transliteration and dialect normalization, all experiments with $\delta = 0$. Monotonicity Loss for transformer on all layers and heads. Average over three runs with independent seeds with $\lambda = 0.1$. Our best models are marked in bold.

Morphological Inflection

For morphological inflection, we show the average results over all 51 languages in Table 2. Our RNN baseline is slightly better than previous work, whereas our transformer baseline performs slightly worse. We notice that the transformer models trained with $\delta = 0$ on the morphological inflection tasks result in the model always attending to the same source position at every decoding state. We therefore set δ to 0.1 for transformer models trained on this task. For the remaining tasks, we report results with δ set to 0 and λ always set to 0.1 so as not to overfit hyperparameters on each task.

The baseline monotonicity loss for this task is higher than for grapheme-to-phoneme conversion but training with the monotonicity loss can drastically increase the monotonicity of the attention mechanisms. This can be seen both in the lower monotonicity score and the higher percentage of decoding states where the average source attention position increases from the previous state. In terms of performance, we do not see an improvement over the baselines.

Transliteration

Our results for transliteration are shown in Table 3 (average over all 11 datasets). Again, we can see that the monotonicity loss effectively biases the attention towards a more monotonic behavior, decreasing the monotonicity score and increasing the percentage of decoding states where the average source attention position increases. In terms of performance, there is a small gain for RNNs both in word-level accuracy and character-level mean F-score. Training with the monotonicity loss does not improve the performance of the transformer compared to the baseline.

Dialect Normalization

Since dialect normalization is our only sentence-level sequence-to-sequence task, it is interesting to see how the monotonicity loss works on longer sequences where more reordering is possible compared to the previous tasks. The less monotonic nature of this task is reflected in the fact that neither of our models trained towards monotonicity outperforms the non-monotonic baselines, see Table 3. Dialect normalization is also the only task where the transformer does not outperform the RNN models.

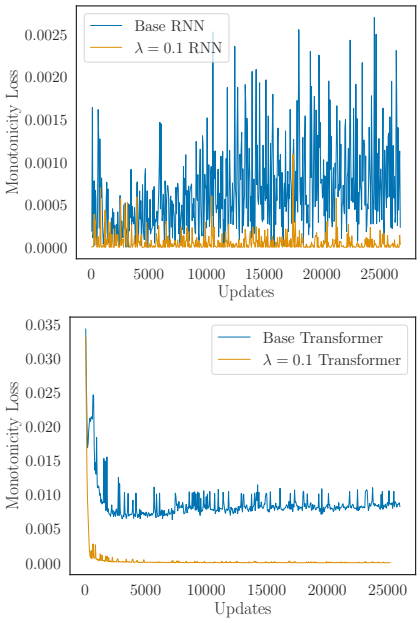


Figure 4: Monotonicity score during training on the EN-JA transliteration dataset with $\delta = 0$. Upper plot: RNN, lower plot: transformer (all heads).

5 Analysis

Overall, our results show that the proposed monotonicity loss succeeds in making attention more monotonic, but effects on quality are more positive for RNNs than for transformers. We now analyze the proposed loss function in more detail.

Monotonicity Over Time

First, we plot the monotonicity score during training and compare how fast it decreases over time. We find that the monotonicity score decreases very fast for the models trained with our loss function and then stays rather constant. The baseline models show various behaviors: for some datasets and models, the score decreases over training time - suggesting that the model does learn to attend more monotonically even without the loss. For other data sets, the score is initially lower and increases over training time, and, for some, the score stays more or less constant. What all baselines have in common, is that the monotonicity score oscillates much more than when trained with the monotonicity loss. Figure 4 shows an example plot for the EN-JA transliteration dataset.

Varying Monotonicity

We can vary how much we constrain attention to be monotonic by varying the weight of the monotonicity loss function (λ). We analyze how this

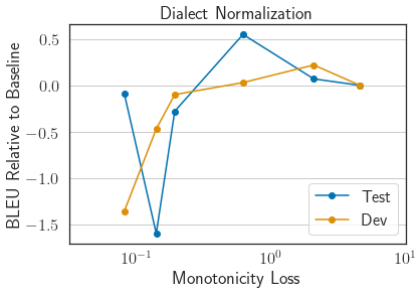


Figure 5: Relative BLEU scores as a function of the monotonicity loss for dialect normalization with transformer (all heads). Different data points obtained by varying λ. ($\lambda \in \{0.3, 0.2, 0.1, 0.01, 0.001, 0\}$).

influences the performance on dialect normalization. Figure 5 shows that non-monotonic behavior (as defined by the monotonicity loss) can be reduced by a factor of 10-20 with stable or even slightly improving performance. However, BLEU drops drastically for large λ. This highlights the advantage of our loss function over hard monotonic attention. Through λ we can regulate the degree of monotonicity in the attention mechanism, which can be beneficial for tasks where hard monotonic attention would be too strict.

Monotonicity Loss on Single Heads

Since we calculate the loss on each attention component separately, we can also limit its application to specific layers and heads (in the case of multihead attention). We test how restricting the monotonic behavior to only one head per layer influences the performance of the transformer on our chosen tasks. Results are presented in Table 4. We find that monotonicity on only one head generally improves performance compared to on all heads, except for dialect normalization. For grapheme-to-phoneme conversion and morphological inflection in the medium resource setting, we even see performance gains over the baseline.

Our results support the belief that the flexibility of multihead attention is key to the success of the transformer. If applied to all heads, the monotonicity loss reduces variability in the attention distribution of the different heads, i.e. with high λ, all heads attend to the same source position. We suspect that this severely limits the capacity of transformer models and explains why rewarding monotonicity on only one head is beneficial.

These findings are also important in the context of the work by Voita et al. (2019) who find that attention heads tend to learn specialized functions. Having one monotonic attention head could be a complementary way to encourage more diversity amongst heads, next to disagreement regularization (Li et al., 2018). Indeed, we observe that for grapheme-to-phoneme conversion and dialect normalization the remaining heads trained without the monotonicity loss tend to become less monotonic.

Attention Maps

Attention maps are particularly interesting for dialect normalization where 1) the transformer baseline has one of the highest monotonicity losses of all our models and 2) reordering of source and target tokens is possible. Figure 6 shows the attention maps for our baseline transformer and the corresponding model trained with the monotonicity loss. The bottom sentence is an example where the alignment between the source and the target is monotonic. Here, the baseline does show tentative monotonic behavior but with the monotonicity loss, the attention follows the main diagonal much more closely. The sentence on the top, on the other hand, contains a non-monotonic alignment. For a correct alignment of the past tense of “to be”, the model needs to peek at the very last token before the full stop. This is reflected in the baseline attention map where the attention at the second decoding step is highest on the third-to-last source position. However, for our model trained with the monotonicity loss, the attention follows the main diagonal and fails to mirror the correct alignment. Occasional reorderings like this may explain why the monotonicity loss did not work well for this task despite it being largely monotonic.

6 Conclusion

We propose a model-agnostic loss function that measures and rewards monotonicity and can easily be integrated into various attention mechanisms. To achieve this, we track how monotonically the average position of the attention shifts over the source sequence across time steps. We show that this loss function can be seamlessly integrated into RNNs as well as transformers. Models trained with our monotonicity loss learn largely monotonic behavior without any specific changes to the attention mechanism. While we see some performance gains in RNNs, our results show that biasing all attention heads in transformers towards monotonic behavior is undesirable. However, a bias towards monotonicity may be helpful if applied to only a subset of

	Performance		heads with L_{MONO}		heads without L_{MONO}	
G2P	WER ↓	PER ↓	$\%mono$	L_{MONO}	$\%mono$	L_{MONO}
baseline	27.79±0.24	7.00±0.09			77.0%	7.26e-02
$\lambda = 0.1, \delta = 0.0, h$ = all:	27.99±0.60	7.11±0.18	84.6%	5.12e-05		
$\lambda = 0.1, \delta = 0.0, h = 1$:	**27.70**±0.37	**6.96**±0.07	84.9%	2.49e-05	75.1%	8.26e-02
Morph. Infl. High	ACC ↑	LEV ↓				
baseline	**95.05**±0.03	**0.097**±0.001			58.1%	1.34
$\lambda = 0.1, \delta = 0.1, h$ = all:	94.98±0.07	0.099±0.002	87.5%	4.49e-4		
$\lambda = 0.1, \delta = 0.1, h = 1$:	95.00±0.03	0.098±0.000	89.3%	6.49e-5	59.6%	1.35
Morph. Infl. Medium	ACC ↑	LEV ↓				
baseline	81.33±0.02	0.378±0.001			58.1%	1.35
$\lambda = 0.1, \delta = 0.1, h$ = all:	81.02±0.17	0.383±0.001	85.7%	1.45e-3		
$\lambda = 0.1, \delta = 0.1, h = 1$:	**81.67**±0.13	**0.366**±0.003	88.6%	4.28e-4	59.0%	1.37
Transliteration	ACC ↑	MFS ↑				
baseline	**42.08**±0.55	**89.63**±0.04			69.1%	0.12
$\lambda = 0.1, \delta = 0.0, h$ = all:	41.32±0.53	89.47±0.08	82.2%	7.1e-4		
$\lambda = 0.1, \delta = 0.0, h = 1$:	41.71±0.37	89.61±0.06	80.4%	1.12e-4	69.6%	0.11
Dialect Normalization	BLEU ↑					
baseline	**32.83**±0.20				71.7%	1.23
$\lambda = 0.1, \delta = 0.0, h$ = all:	32.17±0.78		91.1%	0.05		
$\lambda = 0.1, \delta = 0.0, h = 1$:	31.55±0.71		77.9%	0.01	70.5%	1.64

Table 4: Transformer results for all tasks with monotonicity on all heads vs. only on one head. Monotonicity loss is computed on all layers. Average over three runs with independent seeds. Our best models are marked in bold.

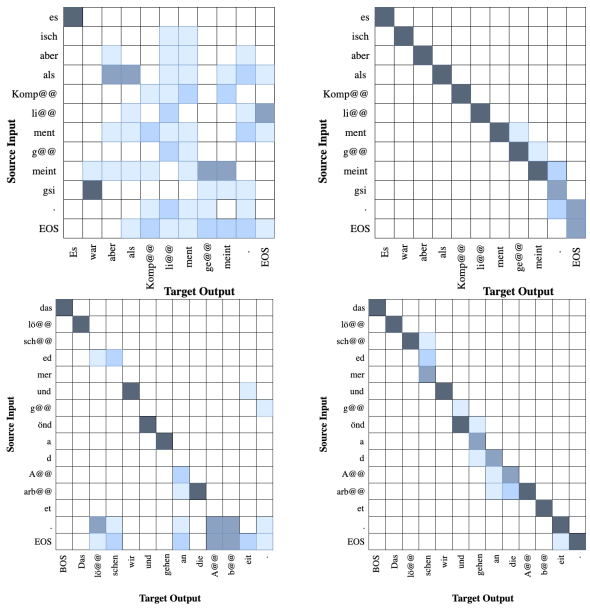


Figure 6: Transformer attention maps for the sentence shown in Figure 3; *"but it was meant as compliment"* and *"we delete this and get to work"*. Left: baseline (λ=0), right: with monotonicity loss on all heads (λ=0.1).

heads.

For the future, we are interested in more sophisticated schedules for the monotonicity loss, possibly reducing λ over the course of training. This would help to learn monotonic behavior in the early training stages but gives the model more flexibility to deviate from such an attention pattern if needed. In this context, our loss function could also be used as an additional pretraining objective for transfer to very low-resource tasks. We would also like to test our loss function on tasks where the alignment may be harder to learn, for example in multimodal models or for long sequences. Finally, using our loss function as a way to measure monotonicity could be an interesting tool for interpretability research.

Acknowledgments

We thank the anonymous reviewers for their feedback. This project has received funding from the Swiss National Science Foundation (project nos. 176727 and 191934).

References

Noëmi Aepli and Simon Clematide. 2018. Parsing approaches for swiss german. In *Proceedings of the 3rd Swiss Text Analytics Conference*, Winterthur, Switzerland.

Roee Aharoni and Yoav Goldberg. 2017. Morphological inflection generation with hard monotonic attention. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2004–2015, Vancouver, Canada. Association for Computational Linguistics.

Alham Fikri Aji, Nikolay Bogoychev, Kenneth Heafield, and Rico Sennrich. 2020. In neural machine translation, what does transfer learning transfer? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7701–7710, Online. Association for Computational Linguistics.

Ali Araabi and Christof Monz. 2020. Optimizing transformer for low-resource neural machine translation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3429–3435, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. 2019. Monotonic infinite lookback attention for simultaneous machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.

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*.

Chung-Cheng Chiu and Colin Raffel. 2018. Monotonic chunkwise attention. In *6th International Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*, Vancouver, Canada.

Tonglee Chung, Yongbin Liu, and Bin Xu. 2020. Monotonic alignments for summarization. *Knowledge-Based Systems*, 192:105363.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 876–885, San Diego, California. Association for Computational Linguistics.

Felix Hieber, Tobias Domhan, Michael Denkowski, David Vilar, Artem Sokolov, Ann Clifton, and Matt Post. 2018. The sockeye neural machine translation toolkit at AMTA 2018. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 200–207, Boston, MA. Association for Machine Translation in the Americas.

Hyunsun Hwang and Changki Lee. 2020. Linear-time korean morphological analysis using an action-based local monotonic attention mechanism. *ETRI Journal*, 42(1):101–107.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R. Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2897–2903, Brussels, Belgium. Association for Computational Linguistics.

Baiji Liu, Songjun Cao, Sining Sun, Weibin Zhang, and Long Ma. 2020. Multi-head monotonic chunkwise attention for online speech recognition. *Computing Research Repository*, arXiv:2005.00205.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Xutai Ma, Juan Miguel Pino, James Cross, Liezl Puzon, and Jiatao Gu. 2020. Monotonic multihead attention. In *International Conference on Learning Representations ICLR*, Addis Ababa, Ethiopia.

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, Brussels, Belgium. Association for Computational Linguistics.

Colin Raffel, Minh-Thang Luong, Peter J. Liu, Ron J. Weiss, and Douglas Eck. 2017. Online and linear-time attention by enforcing monotonic alignments. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2837–2846. PMLR.

Terrence J. Sejnowski and Charles R. Rosenberg. 1987. Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

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.

Rico Sennrich and Biao Zhang. 2019. Revisiting low-resource neural machine translation: A case study. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 211–221, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008.

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.

Jiewen Wu, Rafael E. Banchs, Luis Fernando D'Haro, Pavitra Krishnaswamy, and Nancy Chen. 2018a. Attention-based semantic priming for slot-filling. In *Proceedings of the Seventh Named Entities Workshop*, pages 22–26, Melbourne, Australia. Association for Computational Linguistics.

Shijie Wu and Ryan Cotterell. 2019. Exact hard monotonic attention for character-level transduction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1530–1537, Florence, Italy. Association for Computational Linguistics.

Shijie Wu, Ryan Cotterell, and Mans Hulden. 2021. Applying the transformer to character-level transduction. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, Online. Association for Computational Linguistics.

Shijie Wu, Pamela Shapiro, and Ryan Cotterell. 2018b. 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.

Sevinj Yolchuyeva, Géza Németh, and Bálint Gyires-Tóth. 2019. Transformer based grapheme-to-phoneme conversion. In *Proceedings Interspeech 2019*, pages 2095–2099, Graz, Austria. ISCA.

Min Zhang, Haizhou Li, Rafael E. Banchs, and A Kumaran. 2015. Whitepaper of NEWS 2015 shared task on machine transliteration. In *Proceedings of the Fifth Named Entity Workshop*, pages 1–9, Beijing, China. Association for Computational Linguistics.

Wangchunshu Zhou, Tao Ge, Furu Wei, Ming Zhou, and Ke Xu. 2020. Scheduled DropHead: A regularization method for transformer models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1971–1980, Online. Association for Computational Linguistics.

A Appendix

A.1 Character-level Mean F-score (MFS)

$$LCS(c_i, r_i) = \frac{1}{2}(|c_i| + |r_i| - ED(c_i, r_i))$$

$$R_i = \frac{LCS(c_i, r_i)}{|r_i|}$$

$$P_i = \frac{LCS(c_i, r_i)}{|c_i|}$$

$$F_i = \frac{2 * R_i * P_i}{R_i + P_i}$$

$$MFS = \frac{1}{N} \sum_{i=1}^{N} F_i$$

Where c_i is the i-th candidate and r_i is the corresponding reference transliteration with the smallest edit distance (ED).

A.2 Hyperparameters

Training Hyperparameters Transformer				
	G2P	MI	TR	DN
training settings:				
batch type	sentence			word
batch size	400	400/200	400	4096
max-seq-len	20:20	85:85	85:85	200:200
word-min-count	1:1			
seed	1, 2, 3			
model settings:				
encoder	transformer			
decoder	transformer			
transformer-positional-embedding-type	fixed			
transformer-preprocess	n			
transformer-postprocess	dr			
num-layers	4:4			5:5
transformer-model-size	256			512
transformer-attention-heads	4			2
num-embed	256:256			512:512
weight-tying-type	trg_softmax			src_trg
transformer-feed-forward-num-hidden	512/1024	512	512	512
optimization settings:				
optimizer	adam			
optimizer-params	beta2:0.98			
checkpoint interval	400			
max-num-checkpoint-not-improved	10			
gradient-clipping-threshold	none			
learning-rate-scheduler-type	fixed-rate-inv-sqrt-t			plateau-reduce
optimized-metric	PER			bleu
label-smoothing	0.1			0.6
initial-learning-rate	0.001			0.0001
learning-rate-warmup	4000			0
initialization settings:				
weight-init	xavier			
weight-init-scale	3.0			
weight-init-xavier-factor-type	avg			
dropout settings:				
transformer-dropout-attention	0			
embed-dropout	0.3/0.2	0.3	0.3	0.1
transformer-drophead-attention	0.3/0.2	0.3	0.3	0.0/0.1
transformer-dropout-act	0.3/0.2	0.3	0.3	0.3
transformer-dropout-prepost	0.3/0.2	0.3	0.3	0.3

Table 5: Sockeye hyperparameters for transformer models (values with ':' = encoder:decoder)

Training Hyperparameters RNN				
	G2P	MI	TR	DN
training settings:				
batch type		sentence		word
batch size	20	20	50	1000
max-seq-len	20:20	85:85	85:85	200:200
word-min-count		1:1		
seeds		1, 2, 3		
model settings:				
encoder		rnn		
decoder		rnn		
rnn-cell-type		lstm		
num-layers		2:1		1:1
num-embed		200:200		512:512
rnn-num-hidden		400		1024
optimization settings:				
learning-rate-scheduler-type		plateau-reduce		
learning-rate-warmup		0		
optimizer		adam		
optimized-metric		PER		bleu
checkpoint interval		4000		400
max-num-checkpoint-not-improved		7		10
label-smoothing		0.0		0.2
gradient-clipping-threshold		5		–
initial-learning-rate		0.001		0.0005
learning-rate-reduce-num-not-improved		1		8
learning-rate-reduce-factor		0.5		0.7
initialization settings:				
weight-init		xavier		
weight-init-scale		3.0		
weight-init-xavier-factor-type		avg		
dropout settings:				
embed-dropout		0.4		0.5
rnn-decoder-hidden-dropout		0.4		0.5

Table 6: Sockeye hyperparameters for RNN models (values with ':' = encoder:decoder)

A.3 Model Size

	RNN Models
G2P	4.5M
MI	4.5M
TR	4.5M
DN	25.1M
	Transformer Models
G2P	
CMUdict (ff = 1024):	7.3M
NETtalk (ff = 512):	5.3M
MI	5.3M
TR	5.3M
DN	23.2M

Table 7: Approximate model size in number of parameters for the different tasks (exact numbers can vary slightly due to variable vocabulary sizes with different data sets. G2P, MI and TR numbers correspond to the "large" configuration in Wu et al. (2018a).