

Understanding Attention Training via Output Relevance

Anonymous EMNLP submission

Abstract

In recurrent models with attention, the learned attention weights sometimes correlate with individual token importance, even though the training objective does not explicitly reward this. To understand why, we study the training dynamics of attention for sequence classification and translation. We identify a quantity in the model, which we call the *output relevance*, and show that it drives the learning of the attention. If we ablate attention by fixing it to uniform, the output relevance still correlates with the attention of a normally trained model; but if we instead ablate output relevance, attention cannot be learned. Using output relevance, we explain why attention correlates with gradient based interpretation; and perhaps surprisingly, a Seq2Seq with attention model sometimes fails to learn a simple permutation copying task. Finally, we discuss evidence that multi-head attention improves not only expressiveness but also learning dynamics.

1 Introduction

Attention-based models underlie many recent advances in natural language processing, such as machine translation (Bahdanau et al., 2014) and pretraining (Devlin et al., 2018). Beyond its predictive performance, attention may also provide interpretability by assigning higher attention weights to more important tokens (Wang et al., 2016; Lee et al., 2017; Deng et al., 2018). However, recent work has shown that classification models can be trained to attend to irrelevant tokens without harming performance (Wiegrefe and Pinter, 2019; Pruthi et al., 2019), even though attention weights do correlate weakly with token importance (Ebrahimi et al., 2017) on classification and translation tasks if the model is normally trained (Jain and Wallace, 2019; Serrano and Smith, 2019; Wiegrefe and Pinter, 2019; Vashishth et al., 2019).

Why does a normally trained model attend to tokens that are likely to be important, even though the training objective does not explicitly reward this? Explaining this discrepancy requires understanding the training dynamics. Section 2 defines a quantity β_l called the *output relevance*, which is how much the hidden state h_l is indicative of the correct label. Higher output relevance attracts a model’s attention. We corroborate this by showing that even if the attention is fixed uniform, we can still use output-relevance to predict where a normally trained model is likely to attend (Section 3.4); furthermore, Section 3.3 shows that for translation tasks and a synthetic classification task, the model learns output relevance before learning attention.

Output relevance explains and predicts phenomena in classification and translation. For classification, output relevance increases for all hidden positions as training progresses, but does so *faster* at important token positions (Section 4.1). Intuitively, we expect this to hold for LSTM and other sequence models, since hidden states see direct gradient updates from their corresponding token but only indirect updates from other tokens. This explains why model attention is attracted to important positions.

For translation, we predict cases where learning fails. Specifically, attention is approximately uniform when training starts and so the model only uses word co-occurrence information to learn output relevance; this is evocative of the first step in the early IBM alignment models (Brown et al., 1993). Our theory thus predicts that if word co-occurrence information is removed, the model cannot learn output relevance and hence fails to learn. We verify this in Section 4.2 by designing a simple permutation-copying task. Although constructing a model that completes this task is trivial, the seq2seq models often fail to train. We also find that multi-head attention can alleviate this problem

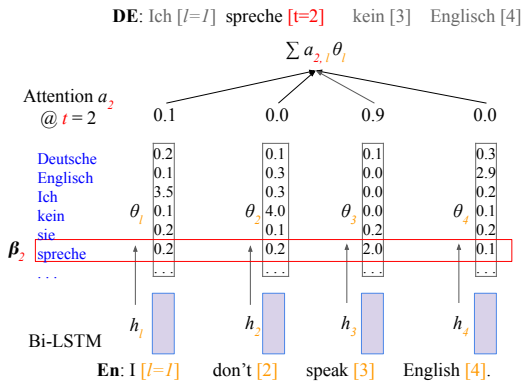


Figure 1: Each column represents θ_l values, and β_2 is defined as the row that corresponds to the second output token “spreche” (after normalizing by rows). When generating the t^{th} token, the model generates an attention distribution a_t over $[L]$ and use it to aggregate θ_l .

by effectively acting as multiple random initializations. Multi-head attention improves not only expressiveness but also learning dynamics.

2 Model and Output Relevance

We consider a recurrent Seq2Seq model with attention. A classification model is a special case of a Seq2Seq model with output length 1. Below we use $[N]$ to denote the set $\{1, \dots, N\}$ and Δ_N to denote the probability simplex.

Suppose the input and output vocabulary have sizes I and O respectively. We denote an input token sequence (sentence) of length L as $[i_1 \dots i_L]$, where each $i_l \in [I]$. The output $[o_1, \dots, o_T]$ is a sequence of tokens in $[O]$ with length T . The model embeds the input sequence with an LSTM to obtain a hidden state h_l for each position $l \in [L]$.

To produce the predictive distribution for the t^{th} output token, the model generates an attention score α_{tl} for each input position l , then applies softmax to obtain an attention distribution $a_t \in \Delta_L$. The model also applies a learnable linear transformation to map each hidden state to a vector $\theta_l \in \mathbb{R}^O$ of output logits.¹ Finally, the model averages θ_l weighted by the attention distribution a_t , and applies softmax to obtain a probability distribution over outputs. Concretely, the predictive distribution p_t is

$$p_t = S(\gamma_t), \text{ where } \gamma_t := \sum_{l=1}^L S(a_t)_l \theta_l \quad (1)$$

¹This model is slightly different from the standard, where a non-linearity follows the weighted average of hidden states. However, it does not hurt performance in our experiments and achieves BLEU score 33.3 on IWSLT’14 DE-EN.

and S is the softmax function. Figure 1 illustrates this. We now define the output relevance β_{tl}

$$\beta_{tl} := \log \frac{e^{\theta_{lo_t}}}{\sum_{o \neq o_t} e^{\theta_{lo}}} \quad (2)$$

We interpret β_{tl} as “how strongly the l^{th} hidden state predicts the true output token o_t ”, normalized by other wrong outputs. Notice that this quantity also needs to be learned by the model, and it may or may not correspond to the correct word alignment. Additionally, β is defined based on the hidden state h_l (which can contain contextual information) rather than the token i_l . Intuitively, larger output relevance attracts model’s attention: if the model can only attend to one position, attending to maximum output relevance minimizes the loss.

3 Output Relevance Drives Attention

We next study the dynamics of the attention weights α throughout learning. Through several translation and classification tasks, we show that the output relevance β drives the learning of α . Specifically, even if the attention is fixed uniform, β is indicative of where a normally trained model would attend to; in contrast, α cannot be learned if β is fixed.²

3.1 Datasets

We consider 6 classification and 2 translation datasets, detailed in Appendix A.1. For classification, we use IMDB, AG News, 20 Newsgroups (NewsG), Stanford Sentiment Treebank (SST), YELP, and Multi-Domain Sentiment (Amzn), and for translation we use Multi30K (M30K) and IWSLT’14 DE-EN (IW14). We also define a synthetic classification task, “SynClf”, such that important tokens are clearly defined. Each input is a length-40 sequence of tokens sampled from $\{1, \dots, 40\}$ uniformly at random; a sequence is labeled positive if and only if the token “1” appears in the sequence. We report accuracy for classification tasks and token accuracy for translation tasks.

3.2 Correlation Metric

For each output position t ($t = 1$ for classification), α_t and β_t are both vectors of length L , with each value associated with an input position. We quantify correlation between α_t and β_t via two metrics. The first is exact match of the modes:

$$\text{acc}(\alpha_t, \beta_t) = \mathbb{1}[\arg \max_l \alpha_{tl} = \arg \max_l \beta_{tl}] \quad (3)$$

²All model details can be seen in the Appendix Section A.2

The second is similar, but gives credit as long as the mode l^* of α_t is highly ranked (hr) in β_t :

$$\text{hr}(\alpha_t, \beta_t) = \mathbb{1}[(\text{rank of } \beta_{tl^*}) \text{ in } \beta_t > 0.95L]$$

where $l^* = \arg \max_l \alpha_{tl}$ (4)

These ranking-based metrics account for the fact that attention vectors are typically sparse. In our experiments we measure correlation by reporting the average of this metric over all sequences and output tokens/labels in the test set. A random baseline for this metric is approximately 5% and at most 10% for all datasets we use.

3.3 Learning β Before α

We begin by studying how β and α evolve early in training. At iteration s , let α^s and β^s denote the attention and output relevance, respectively, and let s^* be the iteration at which the model has maximum test accuracy. To account for randomness in training, we define $\hat{\alpha}$, $\hat{\beta}$ the same as α , β , except with a different random seed. Figure 2 plots $\text{acc}(\alpha^{s^*}, \hat{\alpha}^s)$ and $\text{acc}(\alpha^{s^*}, \hat{\beta}^s)$ on the SynClf classification and Multi30K DE-EN translation tasks. These two quantities track how well $\hat{\alpha}^s$ (red) and $\hat{\beta}^s$ (blue) correlate with the model’s converged attention α^{s^*} during training. The attention of SynClf will converge exclusively to positions with the important token “1”, and that of Multi30K will converge approximately to word alignment.

For both tasks, $\text{acc}(\alpha^{s^*}, \hat{\alpha}^s)$ and $\text{acc}(\alpha^{s^*}, \hat{\beta}^s)$ start at random uniform baselines when training starts. During early iterations, the latter increases faster than the former: β is learned first, which subsequently attracts models’ attention.

Connection to IBM models. If we further assume that h_l mainly contains information about token i_l when training starts, learning output relevance under uniform attention resembles the first iteration of the IBM word alignment algorithm (Brown et al., 1993). Initially, attention is uniform and all alignments are equally likely. Under uniform attention, the model effectively translates a bag of words/hidden states from the source to a bag of words in the target, and hence use word co-occurrence statistics to learn the output relevance/vocab correspondence (e.g. “I” to “Ich” in Figure 1). The network later learns the attention from output relevance just as IBM model learns the alignment from vocab correspondence.

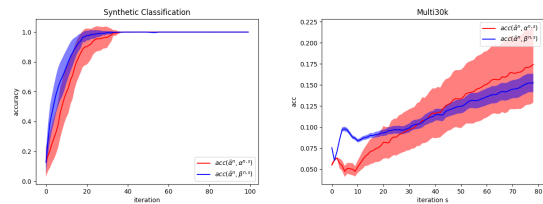


Figure 2: On SynClf and Multi30K DE-EN translation tasks, output relevance β (blue) is learned before attention α (red) is learned. We average across 50 different runs because small batch size causes noise; shaded area is 99% confidence interval of the mean.

3.4 β under Uniform Attention

We next ablate attention (by fixing attention to be uniform) and show that output relevance still correlates with the attention of a normally trained model. Let β_{unif}^s be the output relevance under uniform attention and α_{norm}^s be the attention of a normally trained model; on some data sets, the correlation between β_{unif}^s and $\alpha_{\text{norm}}^{s^*}$ first increases and then decreases, so we define s' as the iteration when $\text{hr}(\alpha_{\text{norm}}^{s^*}, \beta_{\text{unif}}^s)$ is maximized. For completeness, we plot how $\text{hr}(\alpha_{\text{norm}}^{s^*}, \beta_{\text{unif}}^s)$ changes during training in Appendix Section A.4.

Data	hr (%)	s	@s (%)	best (%)
SynClf	100	40	100	100
IMDB	18	150	84	93
AG	42	450	93	96
NewsG	36	10	61	93
SST	24	1950	82	82
Amzn	49	450	85	89
Yelp	58	150	90	96
M30K	40	1000	46	65
IW14	39	1000	33	66

Table 1: Output relevance under uniform attention $\beta_{\text{unif}}^{s^*}$ is indicative (hr) of where the model attends to if normally trained ($\alpha_{\text{norm}}^{s^*}$). s is the iteration when $\text{hr}(\alpha_{\text{norm}}^{s^*}, \hat{\alpha}_{\text{norm}}^s)$ exceeds $\text{hr}(\alpha_{\text{norm}}^{s^*}, \beta_{\text{unif}}^{s^*})$. “@s” is the accuracy of a normal attention model at iteration s ; “best” is the best performance of an attention model.

We compare $\text{hr}(\alpha_{\text{norm}}^{s^*}, \beta_{\text{unif}}^{s'})$ with a floor and a ceiling. The floor is a random baseline, which is at most 10% for every dataset. The ceiling is the correlation between the attention weights of two identical models with different random seeds. According to Section 3.3, the attention is not correlated with where it will converge to when training begins, and hence $\text{hr}(\alpha_{\text{norm}}^{s^*}, \hat{\alpha}^s)$ is around the random baseline. As the model trains and converges, $\text{hr}(\alpha_{\text{norm}}^{s^*}, \hat{\alpha}^s)$

will converge to approximately 1 if the attention weights are relatively stable across training. We report the iteration s when $\alpha_{\text{norm}}^{s*}$ is more correlated with the model’s attention weights $\hat{\alpha}^s$ than output relevance $\beta_{\text{unif}}^{s'}$; i.e., when $\text{hr}(\alpha^{s*}, \hat{\alpha}^s)$ exceeds $\text{hr}(\alpha^{s*}, \beta_{\text{unif}}^{s'})$. We also report the performance of the normal attention model at iteration s .

Table 1 shows the results. On all datasets, $\text{hr}(\alpha_{\text{norm}}^{s*}, \beta_{\text{unif}}^{s'})$ outperforms the random baseline. On datasets like IMDB, AG, and Yelp, the output relevance under uniform attention is even more correlated with the normally trained attention than itself, at an iteration that yields non-trivial accuracy. In contrast, if we freeze β by only training the attention layer while fixing all other parameters at random initialization, the resulting attention is not correlated with $\alpha_{\text{norm}}^{s*}$ (appendix Table 3).

4 Empirical Predictions

We now use output relevance under uniform training to explain and predict various phenomena.

4.1 Attention Correlates With Interpretation

In all classification tasks, training with uniform attention leads to nontrivial accuracy, which implies that β must increase on average. However, β increases faster at positions that correspond to more influential tokens. We use a gradient based influence approximation method (Ebrahimi et al., 2017) to obtain the influence ξ_l for each token i_l and then calculate $\text{hr}(\xi, \beta_{\text{unif}}^{s*})$. Figure 3 shows that these quantities are correlated. This explains why attention is correlated with relative token importance on many classification tasks, even though the training objective does not explicitly reward this.

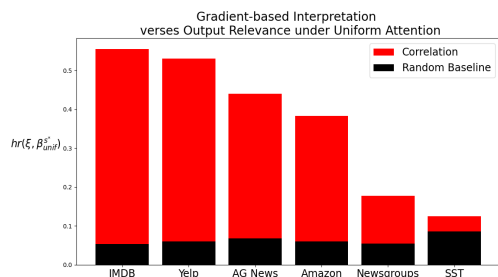


Figure 3: β_{unif}^{s*} and ξ correlates on all datasets.

Unfortunately, we cannot predict exactly where output relevance tends to increase faster under uniform attention *in general*. However, reasoning about output relevance under uniform attention is

conceptually simpler than directly studying the attention mechanism. As shown in Section 3.3, on tasks like SynClf and machine translation where individual token importance attribution is relatively clear, output relevance does increase interpretably.

4.2 Permutation Copying is Hard for Seq2Seq

In Section 3.3, we argued that models leverage word co-occurrence information to learn β early in training; however, if there is no word co-occurrence information, our theory predicts that it would be hard to learn β , which causes the attention to stay uniform and obstructs learning. To verify this prediction, we consider the following simple task: each input sequence is permutation of $\{1, \dots, 40\}$, and the output is equal to the input. If the attention is uniform, the model is approximately a bag of words model, and all input and output sequences appear as the same bag of words, thus making $\beta \approx 0$ and hence $\alpha \approx 0$; training thus gets stuck.

We verify this using a standard Seq2Seq model with single directional LSTM and hidden dimension 256. We say that a model “succeeds” if it achieves 90% token accuracy after 200 iterations and otherwise “fails”. Over 20 random initializations, 11 succeed and 9 fail. In contrast, if input tokens are instead sampled uniformly from $[1, 40]$ (so that there is variation in word frequency across examples), all 20 runs succeed.

Since some random initializations still successfully learn, we investigated whether we can learn more robustly by using several attention heads. We first produce 5 *single-head* “bad” initializations that failed to learn, and 5 “good” ones that learned successfully. We then consider all $2^{10} - 1$ ways of combining these initializations into a multi-head initialization. 22 out of 31 combinations that only contain bad initializations fail, while only 19 fail for the rest of the 992 combinations. In this example, multi-head attention benefits from having multiple attention head initializations; if one of the heads is lucky enough to learn the task successfully, then the whole model is able to learn. Although a single attention head is enough to *construct* a model that completes this task, multi-head attention increases *learnability* via over-parameterization. This observation is consistent with Voita et al. (2019) and Michel et al. (2019): most of the heads can be pruned in a transformer after training, but we cannot train a model with fewer heads from scratch.

References

- 400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben-
gio. 2014. Neural machine translation by jointly
learning to align and translate. *arXiv preprint*
arXiv:1409.0473.
- Peter F. Brown, Stephen A. Della Pietra, Vincent J.
Della Pietra, and Robert L. Mercer. 1993. **The math-
ematics of statistical machine translation: Parameter
estimation**. *Computational Linguistics*, 19(2):263–
311.
- Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa
Bentivogli, and Marcello Federico. Report on the
11th iwslt evaluation campaign, iwslt 2014.
- Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and
Alexander Rush. 2018. Latent alignment and varia-
tional attention. In *Advances in Neural Information
Processing Systems*, pages 9712–9724.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and
Kristina Toutanova. 2018. Bert: Pre-training of deep
bidirectional transformers for language understand-
ing. *arXiv preprint arXiv:1810.04805*.
- Javid Ebrahimi, Anyi Rao, Daniel Lowd, and De-
jing Dou. 2017. Hotflip: White-box adversarial
examples for text classification. *arXiv preprint*
arXiv:1712.06751.
- Desmond Elliott, Stella Frank, Khalil Sima’an, and
Lucia Specia. 2016. Multi30k: Multilingual
english-german image descriptions. *arXiv preprint*
arXiv:1605.00459.
- Sarthak Jain and Byron C Wallace. 2019. Attention is
not explanation. *arXiv preprint arXiv:1902.10186*.
- Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim.
2017. Interactive visualization and manipulation
of attention-based neural machine translation. In
*Proceedings of the 2017 Conference on Empirical
Methods in Natural Language Processing: System
Demonstrations*, pages 121–126.
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham,
Dan Huang, Andrew Y. Ng, and Christopher Potts.
2011. **Learning word vectors for sentiment analy-
sis**. In *Proceedings of the 49th Annual Meeting of
the Association for Computational Linguistics: Hu-
man Language Technologies*, pages 142–150, Port-
land, Oregon, USA. Association for Computational
Linguistics.
- Paul Michel, Omer Levy, and Graham Neubig. 2019.
Are sixteen heads really better than one? In *Ad-
vances in Neural Information Processing Systems*,
pages 14014–14024.
- Danish Pruthi, Mansi Gupta, Bhuwan Dhingra, Gra-
ham Neubig, and Zachary C Lipton. 2019. Learning
to deceive with attention-based explanations. *arXiv*
preprint arXiv:1909.07913.
- Sofia Serrano and Noah A Smith. 2019. Is attention
interpretable? *arXiv preprint arXiv:1906.03731*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason
Chuang, Christopher D Manning, Andrew Y Ng,
and Christopher Potts. 2013. Recursive deep mod-
els for semantic compositionality over a sentiment
treebank. In *Proceedings of the 2013 conference on
empirical methods in natural language processing*,
pages 1631–1642.
- Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh
Tomar, and Manaal Faruqui. 2019. Attention in-
terpretability across nlp tasks. *arXiv preprint*
arXiv:1909.11218.
- Elena Voita, David Talbot, Fedor Moiseev, Rico Sen-
nrich, and Ivan Titov. 2019. Analyzing multi-
head self-attention: Specialized heads do the heavy
lifting, the rest can be pruned. *arXiv preprint*
arXiv:1905.09418.
- Yequan Wang, Minlie Huang, Xiaoyan Zhu, and
Li Zhao. 2016. Attention-based lstm for aspect-
level sentiment classification. In *Proceedings of the
2016 conference on empirical methods in natural
language processing*, pages 606–615.
- Sarah Wiegrefe and Yuval Pinter. 2019. Atten-
tion is not not explanation. *arXiv preprint*
arXiv:1908.04626.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015.
Character-level convolutional networks for text clas-
sification. In *Advances in neural information pro-
cessing systems*, pages 649–657.
- 450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499

A Appendices

A.1 Dataset Summarization

We summarize the datasets that we use for classification and machine translation.

IMDB Sentiment Analysis (Maas et al., 2011)

A sentiment analysis data set with 50,000 (25,000 train and 25,000 test) IMDB movie reviews and their corresponding positive or negative sentiment.

AG News Corpus (Zhang et al., 2015) 120,000 news articles and their corresponding topic (world, sports, business, or science/tech). We classify between the world and business articles.

20 Newsgroups³ A news data set containing around 18,000 newsgroups articles split between 20 different labeled categories. We classify between baseball and hockey articles.

Stanford Sentiment Treebank (Socher et al., 2013) A data set for classifying the sentiment of movie reviews, labeled on a scale from 1 (negative) to 5 (positive). We remove all movies labeled as 3, and classify between 4 or 5 and 1 or 2.

Multi Domain Sentiment Data set⁴ Approximately 40,000 Amazon reviews from various product categories labeled with a corresponding positive or negative label. Since some of the sequences are particularly long, we only use sequences of length less than 400 words.

Yelp Open Data Set⁵ 20,000 Yelp reviews and their corresponding star rating from 1 to 5. We classify between reviews with rating ≤ 2 and ≥ 4 .

Multi-30k (Elliott et al., 2016) English to German translation. The data is from translation image captions.

IWSLT'14 (Cettolo et al.) German to English translation. The data is from translated TED talk transcriptions.

A.2 Model Architectures

Classification We use GloVe-6B pre-trained embeddings to embed the tokens and use a single layer bidirectional LSTM to produce the hidden states h_l (dimension 256) for each position l . We then produce attention from these hidden states by taking

$$a_l = S(V * ReLU(Wh_l)) \quad (5)$$

, where v and W are learn-able parameters. We then produce a weighted sum of the hidden state

³<http://qwone.com/~jason/20Newsgroups/>

⁴<https://www.cs.jhu.edu/~mdredze/datasets/sentiment/>

⁵<https://www.yelp.com/dataset>

Data	median train seq len	train #
IMDB	181	25000
AG News	40	60000
NewsG	183	1197
SST	16	5130
Amzn	71	32514
Yelp	74	88821
IWSLT14	(23 src, 24 trg)	160240
Multi-30k	(16 src, 16 trg)	29000

Table 2: statistics for each dataset. Median sequence length in the training set and train set size.

vectors:

$$\gamma_t = \sum_{t=1}^T a_t h_t \quad (6)$$

. Then we feed it to a final layer to produce the output logits, which we Softmax over for classification. Our classification model has 8,049,216 parameters.

Translation We use a a bidirectional two layer LSTM to encode the source and the use last hidden state h_L as the first hidden state of the decoder. At each output position t , let the decoder output be d_t , we create calculate the attention logits for each input position by

$$\alpha_{tl} = d_t^T W h_l \quad (7)$$

, where W is a learnable parameter. We then take the soft max to create an attention distribution a_t over the input positions. Then we aggregate the hidden states to obtain the context vector by taking a weighted average:

$$\gamma_t = \sum_l a_{tl} h_l \quad (8)$$

We add this context vector to the decoder output d_t and feed it to the final layer to produce vocab logits. This architecture achieves 33.3 BLEU score on the IWSLT'14 DE-EN task.

Permutation Copying We use single directional single layer LSTM with hidden dimension 256 for both the encoder and the decoder.

Multi-head Attention For each head x we use a separate parameter W^x to produce the attention logits. After each attention head produces the context vector γ_t^x , we apply another learn-able linear

transformation G^x after γ_t^x , and sum the results:

$$\gamma_{\text{aggregate}} = \sum_x G^x \gamma_t^x \quad (9)$$

We then feed $\gamma_{\text{aggregate}}$ to the final fully connected layer to produce vocab logits. Our translation model has 10,594,382 parameters.

A.3 Other Configurations

Classification procedure For all classification datasets we used a batch size of 32. We trained for different number of iterations for each dataset: 2000 for IMDB, 1500 for AG News, 1500 for 20 Newsgroups, 2000 for SST, 2500 for Multi-Domain Sentiment (Amazon), and 2300 for Yelp. We train on the pre-defined training set if a dataset has one, except that we pull out 400 examples to use as validation for comparing the correlation metrics. Additionally, if a dataset had a predefined test set, we randomly sample 400 examples from this test set. If a dataset did not have predefined splits, we pulled out 400 examples as validation, 400 as test, and left the rest for training.

Classification evaluation We tested each model every 10 iterations for the first 100 iterations, and then every 50 iterations after that.

Classification Tokenization We tokenized the data at the word level. We mapped all words occurring less than 3 times in the training set to $\langle \text{unk} \rangle$. For 20 Newsgroups and AG News we mapped all non-single digit integer "words" to $\langle \text{unk} \rangle$. For 20 Newsgroups we also split words with the " " character.

Classification Training We trained all classification models on a single GPU. Some datasets took slightly longer to train than others (largely depending on average sequence length), but each train took at most 45 minutes.

Translation Hyper Parameters For translation all hidden states in the model are dimension 256. We use the sequence to sequence architecture described above. The LSTMs used dropout 0.5.

translation procedure For all translation tasks we used batch size 16 when training. We trained IWSLT'14 for 50 epochs on the provided train split, and we tested it every 1000 iterations on the provided validation split. We trained multi-30k for 20 epochs on the provided train split, and we tested every 200 iterations on the provided validation split.

translation training We trained all translation models on a single GPU. IWSLT'14 took approximately 5-6 hours to train, and multi-30k took closer to 1 hour to train.

translation tokenization We tokenized both translation datasets using the Sentence-Piece tokenizer trained on the corresponding train set.

A.4 $\text{hr}(\alpha_{\text{norm}}^{s*}, \beta_{\text{unif}}^s)$ Plots

For each classification dataset we plot the correlation between $\alpha_{\text{norm}}^{s*}$ and β_{unif}^s on the test set, as β_{unif}^s evolves over the course of training.

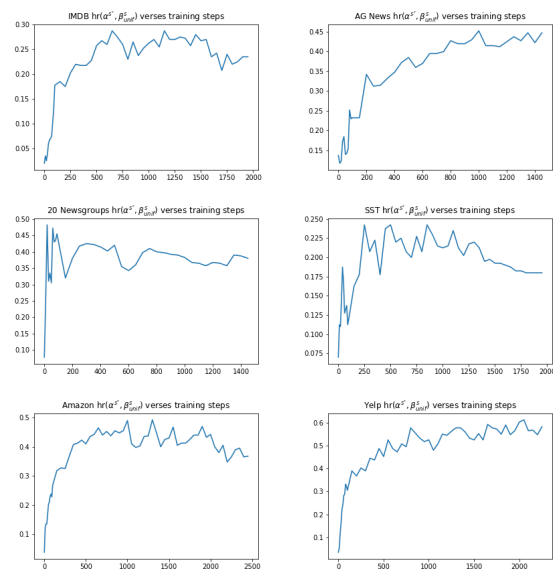


Figure 4: Correlation between $\alpha_{\text{norm}}^{s*}$ and β_{unif}^s first increases over a random baseline, which is never greater than 0.1 for all datasets, but eventually decreases later in training.

A.5 Frozen β Correlations

For each classification task we initialize a random model and freeze all parameters except for the attention layer (frozen β model). We then compute the correlation between this trained attention (defined as α_{froz}^s) and the normal attention $\alpha_{\text{norm}}^{s*}$. Table 3 reports this correlation at iteration s' , where α_{froz}^s is most correlated with $\alpha_{\text{norm}}^{s*}$ on a held out validation set. As shown in Table 3, the left column is consistently lower than the right column. This indicates that the model can learn output relevance without attention, but not vice versa.

Dataset	$hr_{\text{froz}}(\%)$	$hr_{\text{norm}}(\%)$
IMDB	9	18
AG News	17	42
Newsgroups	19	36
SST	14	24
Amazon	15	49
Yelp	8	58

Table 3: $hr(\alpha_{\text{norm}}^{s*}, \alpha_{\text{froz}}^{s'})$ is referred to as hr_{froz} . We compare it against $hr(\alpha_{\text{norm}}^{s*}, \beta_{\text{unif}}^{s'})$, the column $hr\%$ defined in Table 1