Speeding Up Entmax

Anonymous ACL submission

Abstract

Softmax is the de facto standard for normalizing logits in modern neural networks for language processing. However, by producing a dense probability distribution each token in the vocabulary has a nonzero chance of being selected at each generation step, leading to a variety of reported problems in text generation. α-entmax of Peters et al. (2019) solves this problem, but is unfortunately slower than softmax.

In this paper, we propose an alternative to α-entmax, which keeps its virtuous characteristics, but is as fast as optimized softmax and achieves on par or better performance in machine translation task.

1 Introduction

Sparseness of vector representations is a desirable trait in neural network models for natural language processing (NLP): words (subwords) are discrete objects by their nature, and, accordingly, are encoded by one-hot embeddings at the input and output of neural networks. However, to predict a categorical response in neural models, softmax is most often used, which produces a *dense* probability distribution, i.e. every category (word/subword) receives a non-zero probability.

Recent studies suggest that it is this output density that poses problems when the trained NLP model is used for inference. For example, in the case of text generation, unconstrained sampling from a trained language model results in poor quality of the resulting text (Holtzman et al., 2020). In neural machine translation (NMT), exact decoding from a trained model often results in empty text (Stahlberg and Byrne, 2019).[1] To get around these problems, constrained decoding techniques have been proposed, most of which artificially impose sparsity on softmax prediction. For example, Fan et al. (2018) propose to sample from the top-k probable words, and Holtzman et al. (2020) propose to sample from the most probable words, which comprise the cumulative probability p. While these methods are effective, they are ad-hoc solutions that lead to a mismatch between how the model is trained and how it is used at inference.

In this regard, the works on sparse alternatives to softmax stand apart since they allow us to make inference from the model in the same way than it was trained. Some of the most successful and elegant solutions are sparsemax (Martins and Astudillo, 2016) and its generalization α-entmax (Peters et al., 2019). When coupled with suitable losses, these transformations are not inferior to softmax, and sometimes even surpass it as measured with final performance metrics on a number of tasks. A problem with these transformations however is that they are significantly slower than softmax when the number of categories (vocabulary size) is tens of thousands, as in the case of text generation. This is because α-entmax transformation—in its original formulation—requires sorting over the logits.[2]

In this work, we ask the question: *is it possible to obtain a sparse output like that of α-entmax, but without its degradation in computational speed?* Our answer is affirmative—we propose a sparse output transformation that

- is on par or superior to softmax and α-entmax in the NMT tasks,
- works as fast as softmax during training and at inference,
- gives the same training dynamics as α-entmax (in training steps).

The most surprising thing is that such a transformation is simply a shifted ReLU raised to power $\frac{1}{\alpha-1}$, which we call $\boldsymbol{\alpha}$**-ReLU**.

[1] The authors called this phenomenon the *cat got your tongue* problem.

[2] We also compare against an approximate version which only performs sorting on the highest values of the logits.

The rest of the paper is organised as follows. In Sect. 2 we motivate the choice of α-ReLU as the output transformation, and also select an appropriate loss function. In Sect. 3 we experimentally confirm our claims about performance and output speed of α-ReLU in the NMT task. Sect. 4 is devoted to a comparative analysis of α-ReLU and α-entmax in terms of sparsity, ability to solve the empty translation problem, and training dynamics.

2 α-ReLU at Output

Our departure point is the α-entmax transformation of Peters et al. (2019) which can be defined for $\mathbf{z} \in \mathbb{R}^d$ as

$$\alpha\text{-entmax}_i(\mathbf{z}) = [(\alpha-1)z_i - \tau(\mathbf{z})]_+^{\frac{1}{\alpha-1}},$$

where $[x]_+ := \max\{x, 0\}$, and $\tau : \mathbb{R}^d \to \mathbb{R}$ is the (unique) function that satisfies $\sum_j [(\alpha-1)z_j - \tau(\mathbf{z})]_+^{\frac{1}{\alpha-1}} = 1$ for any $\mathbf{z}$. It is this threshold τ that makes the computation of α-entmax slow, because one needs to sort the components of $\mathbf{z}$ to find τ (Peters et al., 2019, Alg. 2).

As we can see, the threshold τ is only needed to ensure that α-entmax$(\mathbf{z})$ is a *probability* distribution. We loosen this constraint, and only require *non-negative* weights, which is sufficient for most uses. Consider then a transformation

$$\alpha\text{-ReLU}_i(\mathbf{z}) := [(\alpha-1)z_i - \tau]_+^{\frac{1}{\alpha-1}}, \quad (1)$$

where τ is a *constant* that does not depend on $\mathbf{z}$. In order to force α-ReLU$(\mathbf{z})$—applied to the logits $\mathbf{z}$—to converge to the one-hot vector $\mathbf{e}_y$ of the gold label y we need to adjust the corresponding loss. This can easily be done by feeding the logits $\mathbf{z}$ and the output α-ReLU$(\mathbf{z})$ into the following loss, which we call **α-ReLU loss**.

$$\ell(\mathbf{z}, y) = (\alpha\text{-ReLU}(\mathbf{z}) - \mathbf{e}_y)^\top \left(\mathbf{z} - \tfrac{\tau}{\alpha-1}\mathbf{1}\right) + \mathrm{H}_\alpha[\alpha\text{-ReLU}(\mathbf{z})], \quad (2)$$

where $\mathrm{H}_\alpha[\mathbf{p}] := \frac{1}{\alpha(\alpha-1)}\left(1 - \sum_j p_j^\alpha\right)$, $\alpha \neq 1$, is the Tsallis α-entropy (Tsallis, 1988). The rationale for coupling α-ReLU with the loss (2) is the following

Lemma 1. *For any $\tau \in \mathbb{R}$, the gradient of the α-ReLU loss (2) is given by*

$$\nabla_{\mathbf{z}} \ell(\mathbf{z}, y) = \alpha\text{-ReLU}(\mathbf{z}) - \mathbf{e}_y.$$

Proof. The proof is in Appendix B.1. □

By Lemma 1, gradient-based minimization of ℓ indeed forces α-ReLU$(\mathbf{z}) \to \mathbf{e}_y$. Notice that this is similar to what happens when the softmax normalization is coupled with the cross-entropy loss or when α-entmax is coupled with the entmax loss. In both cases differentiating the loss with respect to logits gives $\mathbf{p} - \mathbf{e}_y$, where $\mathbf{p}$ is either softmax$(\mathbf{z})$ or α-entmax$(\mathbf{z})$ (Martins and Astudillo, 2016; Peters et al., 2019).

Remark. Recall that α-entmax is a generalization of sparsemax. For example, 2-entmax is essentially sparsemax, and for $\alpha \in (1, 2)$ we get a smoothed version of sparsemax. Similarly, α-ReLU is a kind of generalization of ReLU. So, the standard ReLU is 2-ReLU (with $\tau = 0$), and for $\alpha \in (1, 2)$ we get a smoothed ReLU (see Fig. 1).

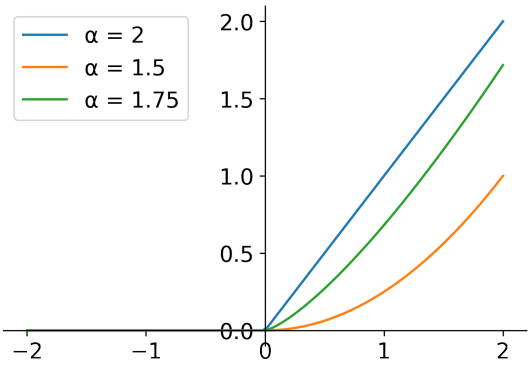

Figure 1: The graph of α-ReLU(x) for several $\alpha \in (1, 2]$, with $\tau = 0$. 2-ReLU is a standard ReLU$(x) := [x]_+$.

3 Experiments

In theory, nothing prevents α-ReLU from learning what α-entmax is learning. However, in practice we can have a different picture, because training is conditioned by many factors—the size of the dataset, the architecture of the neural network, the optimization algorithm, etc. In this section, we compare α-ReLU empirically with α-entmax (as well as with sparsemax and softmax), assuming all other factors are fixed. The goal of these experiments is to evaluate the consequences of using α-ReLU as drop-in replacement for α-entmax.

We test α-ReLU at output in a neural machine translation task (Sutskever et al., 2014), which is essentially a conditional text generation task. Compared to open-ended text generation, there is a

Output Transform	Loss	IWSLT De→En	WMT En→De	WMT En→Ru
softmax	cross-entropy	35.3	28.7	22.4
sparsemax	sparsemax loss	35.5	26.6	19.6
1.5-entmax	1.5-entmax loss	36.6	28.6	23.9
1.5-ReLU	1.5-ReLU loss	37.3	28.6	24.6
# Trainable parameters		47M	75M	75M

Table 1: NMT results: comparison of softmax, sparsemax, 1.5-Entmax and the proposed 1.5-ReLU as the output transformations in the Transformer NMT model. Reported is detokenized test BLEU.

clearer metric of the quality of the generated text—the BLEU score (Papineni et al., 2002). As in open-ended text generation, at each prediction step, the NMT system needs to make a choice from all words (subwords) of the vocabulary, the size of which can reach several tens of thousands. Therefore, the sparsity of the output distribution becomes critical in such setups, since it can explicitly prevent the occurrence of most of the words that are inappropriate in the context.

3.1 Setup

Data. We conduct experiments on three datasets of varied sizes:

- IWSLT'14 De→En (Cettolo et al.), 172K training examples,
- WMT'14 En→De (Bojar et al., 2014), 4.5M training examples,
- WMT'13 En→Ru (Bojar et al., 2013), 1.3M tranining examples.[3]

We preprocess all datasets using the byte pair encoding algorithm (Sennrich et al., 2016) with 10K merge operations on IWSLT, 40K merge operations on WMT En→De, and 60K merge operations on WMT En→Ru. We report detokenized case-sensitive BLEU with SacreBLEU (Post, 2018).[4]

Hyperparameters α and τ. In all experiments we set $\alpha = 1.5$, because this value was recommended by Peters et al. (2019); Peters and Martins (2021) as the middle ground between $\alpha = 1$ (softmax) and $\alpha = 2$ (sparsmax).

The value for τ is chosen as follows: we run the first batch through a non-trained neural network, which has 1.5-entmax at the output, in the forward direction and determine the average τ value across the batch. This value is then used to train the 1.5-ReLU network. Our preliminary experiments have shown that 1.5-ReLU convergence is sensitive to the τ value, and that having output close to the probability distribution early in the learning phase works well with the rest of hyperparameters which are set to their default values.

Training. We trained the Transformer Base (Vaswani et al., 2017) using the OpenNMT-py 2.0 toolkit (Klein et al., 2017). Optimization details are in Appendix A.

3.2 Results

The results are given in Table 1. Reported are test BLEU scores for best checkpoints which are selected based on validation BLEU. We observe that the 1.5-ReLU performs on par with 1.5-entmax or better, while sparsemax is inferior to all others.

Training Time. Fig. 2&3 show the training dynamics in training steps and in wall time on WMT'14 En→De. Despite the closeness of performance in intermediate steps and at the end of training, we see that on the larger datasets 1.5-entmax is slower in wall time than softmax and 1.5-ReLU.

To speed up the learning process, Peters et al. (2019) recommended limiting the number of sorted logits in the α-entmax to the k largest logits. We tried this on the WMT'14 En→De dataset using $k = 100$, which is the default value in the author's implementation of α-entmax.[5] The resulting training dynamics in absolute time is shown as a dashed curve in Fig. 3 (middle). As we can see, partial sorting indeed speeds up the learning process, and at the same time does not harm the quality of the translation. But in the end, learning is still slower than in the case of 1.5-ReLU. Of course, one can try to select such k that the speed of calculating

[3] We did not use the Yandex 1M Parallel Corpus because of its license restrictions.

[4] *BLEU+case.mixed+lang.en-de+numrefs.1+smooth.exp+tok.13a+version.1.5.1*

[5] https://github.com/deep-spin/entmax

the 1.5-entmax will be as close as possible to the speed of 1.5-ReLU without losing quality, but this requires additional efforts on the part of the user, and this must be done for each case separately.

In this regard, 1.5-ReLU does not require additional fine-tuning, converges as fast as softmax in absolute time and performs on par or better. Thus 1.5-ReLU combines all three desired properties: computation speed, task performance, and sparsity of output.

Inference Time. We measured inference time of translating the WMT En→Ru test data with the different strategies and with different beam sizes. The results—normalized by the smallest value—are shown in Fig. 4. As can be seen the relative difference seems independent of the beam size: softmax is almost twice faster than 1.5-entmax (with full sorting over the logits). Even though the softmax version is optimized through the softmax CUDA kernel, it performs equivalent to the 1.5-ReLU model in terms of computation speed.

4 Analysis

4.1 Empty Translations

We remind the reader that the *cat got your tongue* problem (Stahlberg and Byrne, 2019) is one of the main motivations for using sparse transformations when generating text. As Peters and Martins (2021) have shown, 1.5-entmax successfully tackles this problem by significantly lowering the proportion of cases where an empty string is more likely than the beam search hypothesis. For 1.5-ReLU, we also calculated this proportion, and compared it with the proportions for softmax and sparsemax (Table 2). As we see, 1.5-ReLU also successfully tackles the *cat got your tongue* problem.

Output Transform	IWSLT De→En	WMT En→De	WMT En→Ru
softmax	7.5%	29.8%	31.7%
sparsemax	0%	0.03%	0%
1.5-entmax	0%	0.2%	0%
1.5-ReLU	0%	0.3%	0.1%

Table 2: Percentage of development set examples for which the model assigns higher probability to the empty string than to the beam-decoded hypothesis.

4.2 Sparsity

To compare the sparsity of 1.5-ReLU and 1.5-entmax we depict in Fig. 5 the distributions of the number of zero components after applying these transformations (recall that for softmax all components are always nonzero). Since we constructed the α-ReLU in such way that it mimics the α-entmax (at least in the early stages of training), we expected that these two transformations would have similar properties, including sparsity. However, this is not the case: as we can see, the 1.5-ReLU is significantly less sparse than the 1.5-entmax. It is noteworthy that lower sparsity in this case correlates with a better performance in the translation task (see Table 1).

4.3 Impact of τ

The selection of τ was described in Section 3.1. However, the question arises: does the described approach lead to the choice of the *optimal* τ? To find out, we trained the α-ReLU models for $\tau \in \{0, 0.1, 0.2, ..., 0.9, 1, 2, 5, 10\}$ on the IWSLT data. Note that all of these τ's have led to almost the same result at the end of the training (as predicted by Lemma 1). In Fig. 6, we present the dynamics of early training only for $\tau \in \{0, 0.1, 0.2, 0.3, 5, 10\}$, since the curves for $\tau \in \{0.4, ..., 0.9, 1, 2\}$ practically coincided with the optimal curve corresponding to $\tau = 0.3$. Note that our τ selection method gave a value of 0.33, thus we have no evidence against the adequacy of our method.

4.4 Estimation of τ without data

On closer inspection, we noticed that the pre-entmax logits in the *untrained* Transformer model are distributed according to the normal law, regardless of what data is supplied to the input, Shapiro-Wilk test, p-value > 0.15. This allows us, using asymptotic theory, to estimate τ as

$$\hat{\tau} = \sqrt{\frac{d_{\text{model}}}{2(d_{\text{model}} + d_{\text{vocab}})}} \cdot \Phi^{-1}(1 - p^*), \quad (3)$$

where d_{model} is the size of hidden representations, d_{vocab} is the vocabulary size for a target language, $\Phi^{-1}(\cdot)$ is the probit function and p^* is the solution of a non-linear equation that involves functions related to the standard normal distribution (see Appendix B.2 for details). Table 3 compares the $\tilde{\tau}$ calculated by running data through an untrained model with the estimate $\hat{\tau}$ obtained from (3). As we can see, $\hat{\tau}$ practically coincides with $\tilde{\tau}$ with an

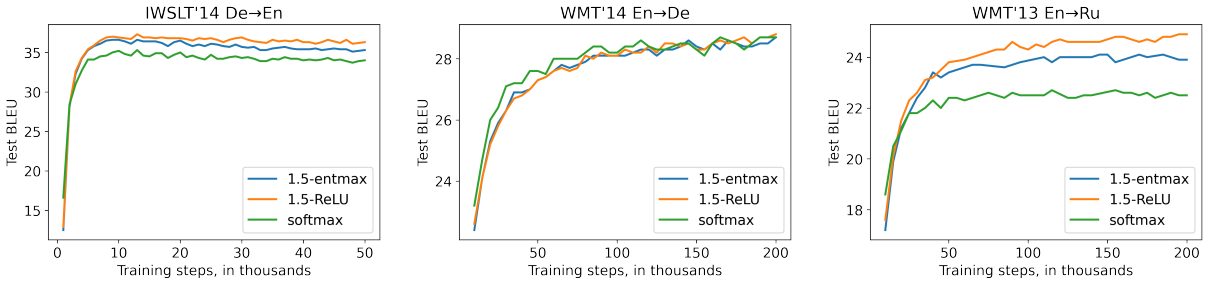

Figure 2: Training dynamics in training steps.

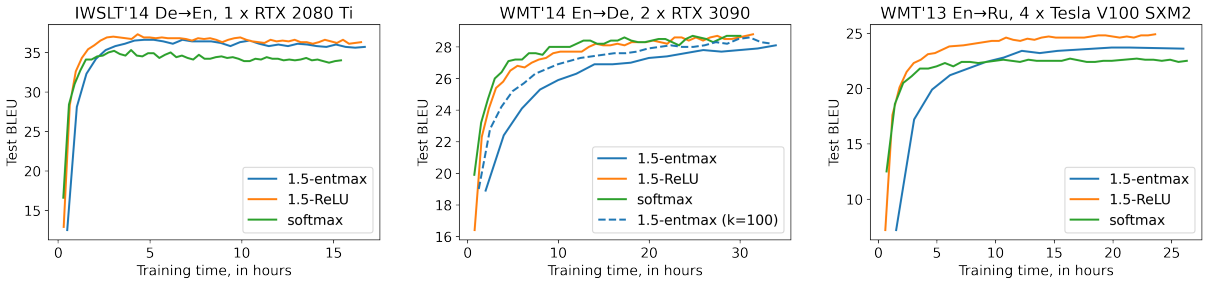

Figure 3: Training dynamics in absolute time. *1.5-entmax (k=100)* is a variant of 1.5-entmax in which sorting is performed only for the largest $k = 100$ logits.

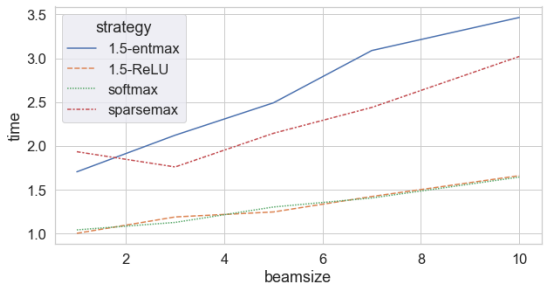

Figure 4: Normalized inference for WMT En→Ru with different beam sizes.

	IWSLT'14 De→En	WMT'14 En→De	WMT'13 En→Ru
d_{model}	512	512	512
d_{vocab}	10,000	40,000	60,000
p^*	.0184	.0171	.0169
$\tilde{\tau}$	.33	.17	.14
$\hat{\tau}$	.33	.17	.14

Table 3: Estimating threshold of 1.5-entmax: $\tilde{\tau}$ is a value obtained by running a data through an untrained model; $\hat{\tau}$ is an estimate based on asymptotic theory, i.e. without running the data through the model.

accuracy of two decimal places. Unfortunately, the formula (3) is not universal: it is only true for the Transformer architecture.

4.5 Self-normalization

The attentive reader may have noticed that the output of α-ReLU is not normalized, i.e. the components of $\alpha\text{-ReLU}(\mathbf{z})$ do not have to sum up to 1. Accordingly, the question arises: how correct is it to compare translation scores at different steps of the beam-search decoding if the conditional probabilities are not normalized? However, the comparison is possible if the $\alpha\text{-ReLU}(\mathbf{z})$ components add up to approximately the same number, i.e. if the model is self-normalizing. To check this, we ran the trained α-ReLU model on the IWSLT and WMT'14 test sets, and looked at the distribution of $\sum_i \alpha\text{-ReLU}_i(\mathbf{z})$ at each decoding step. The results are shown in Fig. 7. As we can see, the sum of the $\alpha\text{-ReLU}(\mathbf{z})$ components concentrates well around its mean ≈ 1.24 (IWSLT) and 1.09 (WMT'14), which might indicate that the model indeed has a self-normalization property.

4.6 Training Dynamics

As we noted in Sect. 3.2, the training dynamics are similar in all three cases (softmax, 1.5-entmax, 1.5-ReLU) when time is measured in training steps. Here we attempt to explain this phe-

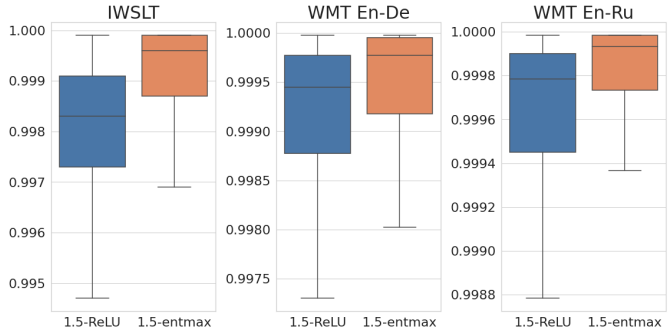

Figure 5: Sparsity as proportion of zero components after applying 1.5-ReLU and 1.5-entmax.

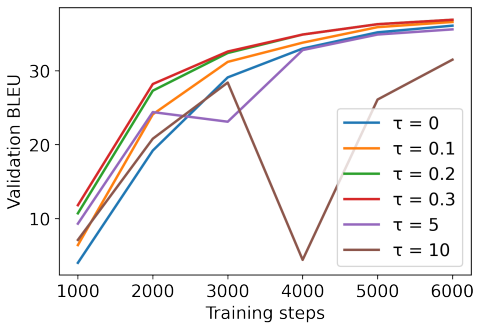

Figure 6: Impact of τ on training dynamics, IWSLT'14 En→De.

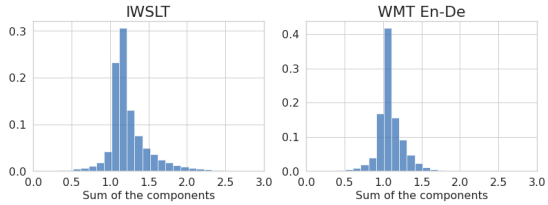

Figure 7: Distribution of the sum of α-ReLU($\mathbf{z}$) components across the IWSLT'14 and WMT'14 test sets: α-ReLU self-normalizes.

Model	Avg. Score	Std. Dev.
Reference	3.9	0.30
Softmax	3.3	0.75
1.5-entmax	3.2	0.74
1.5-ReLU	3.3	0.74

Table 4: Results of Human Evaluation across 270 random examples (with repetitions) from WMT'13 En→Ru test split. Scores are on a 4-point scale.

nomenon through the recently proposed Neural Tangent Kernel (NTK) approach of Jacot et al. (2018). Roughly speaking, the NTK theory suggests that a sufficiently wide neural network trains like a kernel regression. We use this theory to show (in Appendix B.3) that in all three cases the logits $\mathbf{z}(x, t)$ for a training instance x at a training step t evolve (approximately) according to the same differential equation

$$\frac{d\mathbf{z}}{dt} = -\mathbb{E}_{(x',y')}[\mathbf{K}_\sigma(x, x') \cdot (\sigma(\mathbf{z}') - \mathbf{e}_{y'})], \quad (4)$$

where expectation is over training examples (x', y'), $\sigma(\cdot)$ is one of the transformations considered (softmax, α-entmax, or α-ReLU), and $\mathbf{K}_\sigma(x, x') \in \mathbb{R}^{d\times d}$ is a positive semi-definite matrix that depends on σ. The Equation (4) is a non-linear matrix differential equation which in general cannot be solved analytically. However, it has an equilibrium point $\mathbf{z}(x, t)$ such that $\mathbb{E}_{(x',y')}[\mathbf{K}_\sigma(x, x') \cdot (\sigma(\mathbf{z}') - \mathbf{e}_{y'})] = \mathbf{0}$, thus its solution converges to this point as $t \to \infty$. This similarity in the evolution of $\sigma(\mathbf{z})$ implies the similarity in the evolution of the perfomance metric—such as BLEU—accross all three transformations.

4.7 Human Evaluation

Although the BLEU metric (Papineni et al., 2002) has stood the test of time, it is still an automated assessment of translation quality. To double-check the reliability of the results from Table 1, we decided to manually evaluate the translations from the WMT'13 En→Ru test split. To do this, we followed the human evaluation setup from (Berard et al., 2019). We formed two random samples of 135 instances each and gave them to two annotators. 45 instances were shared across two samples in order to calculate inter-annotator agreement. Each instance consists of an original sentence in English and 4 candidate translations into Russian (reference, softmax, entmax, α-ReLU). The annotators were to rate each translation on a 4-point scale. For annotation instructions, see Appendix C.

The order of candidate translations was shuffled for each instance, so the annotators did not know which sentence is from which model. Nevertheless, the annotator always had a good chance of guessing which translation was the reference one, due to the large difference in quality between human and machine translation.

The results of human evaluation are shown in Table 4. Cohen's $\kappa = 0.56$, indicating moderate agreement between annotators. As we can see, all

three models give approximately the same translation quality, and all three are significantly inferior to the reference translation. This is generally consistent with the results of 1.5-ReLU and 1.5-entmax in Table 1, but at the same time casts doubt on the softmax lag behind 1.5-ReLU and 1.5-entmax as the BLEU metric suggests.

In Appendix D we give a few examples where 1.5-ReLU translates better than 1.5-entmax and vice versa.

5 Related Work

Sparse seq2seq models. Our proposed α-ReLU transformation is based on the α-entmax transformation of Peters et al. (2019), which in turn is a generalization of the sparsemax transformation (Martins and Astudillo, 2016). In our work, we study sparseness at the output of a neural network. Nevertheless, there are a number of works aimed at sparsification within a neural network. For example, Malaviya et al. (2018); Peters et al. (2019); Correia et al. (2019) show that sparsemax and α-entmax can replace softmax in the attention mechanism with some success. A recent work of Zhang et al. (2021) attempted to replace softmax with a component-wise ReLU in the attention mechanism. Unfortunately, in its pure form, this replacement leads to the inability of the model to learn at all, since its loss function does not decrease during optimization. The authors solve this problem by adding a normalizing layer on top of the attention layer.

These and other works (Zhang et al., 2019) state that sparsity in the weights of attention produces more interpretable patterns. However, Meister et al. (2021) questioned this claim and were unable to find clear evidence to support it. Therefore, in this work, we focused on the application of α-ReLU to the output of the transformer model, and not to the mechanism of attention, but at the same time we do not deny the possibility of studying the latter.

Self-normalization. Self-normalizing training aims to bypass the need of normalization during inference time. This is done by tweaking the learning mechanism so that the sum of all predictions sums (approximately) to a constant value. Theoretical work on why this works is poorly understood (Andreas et al., 2015) but early work in neural machine translation has shown its empirical value. Vaswani et al. (2013) achieves that by using noise-contrastive estimation (the neural model is used to re-rank the output of a hierarchical phrase-based machine translation system). Noise-contrastive estimation is also the standard training mechanism for word2vec (more popular than the alternative hierarchical softmax), which also eschews any expensive normalization. Differently, Devlin et al. (2014) changes the training loss to include a factor that encourages the normalizing factor to be 1. At inference time, this is just assumed and decoding time is reported to achieve a 15x speed-up.

6 Limitations and Risks

We believe that the main limitations of our work are as follows:

- α-ReLU's output is still not a probability distribution, as required by the classical formulation of a probabilistic classification model.
- τ evaluation requires either running the data through an untrained model with α-entmax at the output, or deriving a formula similar to (3) for each individual architecture.
- Our approach only works for the case when α-ReLU is used at the output of the model, but it is not clear how to use it as an alternative to softmax/α-entmax in the attention layer.

The last mentioned limitation leads to the potential risk of inability to learn if α-ReLU is misused in the intermediate layers of the neural network such as attention layers. The experiments of Zhang et al. (2021) using vanilla ReLU (2-ReLU with $\tau = 0$ in our notation) instead of softmax to produce attention weights lead to a divergence of the loss function of the Transformer model. This translates into a waste of energy, especially when training large models on large datasets. Therefore, we believe that in the future, a preliminary mathematical analysis and/or experiments with small models on small datasets should be carried out as to why the unnormalized distribution of attention weights leads to the inability of the model to learn.

7 Conclusion

It seems that the sparsity of the output is natural for (sub)word prediction models. Nevertheless, sparsity does not have to come with slowdown of computations, as our work shows. The proposed transformation, α-ReLU, gives a sparse output, shows competitive performance, and is as fast as softmax. The reduced dependency on the vocabulary size

seems particularly important in translation, where neural models are moving more and more towards multi-lingual ones, which in general have a much higher vocabulary size in order to accommodate enough tokens for all languages.

A natural extension of this work will be the evaluation of α-ReLU in the problem of open-ended text generation, as well as a replacement for softmax in the attention layers of Transformer models.

References

Jacob Andreas, Maxim Rabinovich, Michael I Jordan, and Dan Klein. 2015. On the accuracy of self-normalized log-linear models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1783–1791.

Barry C. Arnold, N. Balakrishnan, and H. N. Nagaraja. 2008. *A First Course in Order Statistics (Classics in Applied Mathematics)*. Society for Industrial and Applied Mathematics, USA.

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

Alexandre Berard, Ioan Calapodescu, Marc Dymetman, Claude Roux, Jean-Luc Meunier, and Vassilina Nikoulina. 2019. Machine translation of restaurant reviews: New corpus for domain adaptation and robustness. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 168–176, Hong Kong. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria. Association for Computational Linguistics.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, WMT@ACL 2014, June 26-27, 2014, Baltimore, Maryland, USA*, pages 12–58. The Association for Computer Linguistics.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2174–2184. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann N. Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 889–898. Association for Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Arthur Jacot, Clément Hongler, and Franck Gabriel. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 8580–8589.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Chaitanya Malaviya, Pedro Ferreira, and André F. T. Martins. 2018. Sparse and constrained attention for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 370–376. Association for Computational Linguistics.

André F. T. Martins and Ramón Fernandez Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1614–1623. JMLR.org.

Clara Meister, Stefan Lazov, Isabelle Augenstein, and Ryan Cotterell. 2021. Is sparse attention more interpretable? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, pages 122–129. Association for Computational Linguistics.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Ben Peters and André F. T. Martins. 2021. Smoothing and shrinking the sparse seq2seq search space. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 2642–2654. Association for Computational Linguistics.

Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse sequence-to-sequence models. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1504–1519. Association for Computational Linguistics.

Matt Post. 2018. A call for clarity in reporting bleu scores. *arXiv preprint arXiv:1804.08771*.

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, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3354–3360. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Constantino Tsallis. 1988. Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52(1):479–487.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392, Seattle, Washington, USA. Association for Computational Linguistics.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2021. Sparse attention with linear units. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6507–6520. Association for Computational Linguistics.

Jiajun Zhang, Yang Zhao, Haoran Li, and Chengqing Zong. 2019. Attention with sparsity regularization for neural machine translation and summarization. *IEEE ACM Trans. Audio Speech Lang. Process.*, 27(3):507–518.

A Optimization

IWSLT'14 De→En

- Architecture: Transformer, embedding size 512, 6 layers, 8 heads, hidden size 1024, shared vocabulary.
- Batch size: 4096 tokens (with gradient accumulation for 8 steps).
- Optimizer: *ADAM*, $\beta_1 = 0.9$, $\beta_2 = 0.998$, noam decay, learning rate 2.0, 4000 warmup steps.
- Dropout: 0.3
- No label smoothing.

WMT'14 En→De

- Architecture: Transformer, embedding size 512, 6 layers, 8 heads, hidden size 2048, shared vocabulary of 40K tokens, shared embeddings and decoder embeddings.
- Batch size: 4096 tokens (with gradient accumulation for 4 steps).
- Optimizer: *ADAM*, $\beta_1 = 0.9$, $\beta_2 = 0.998$, noam decay, learning rate 2.0, 8000 warmup steps, average decay 0.0005.
- Dropout: 0.1.
- Attention dropout: 0.1.
- No label smoothing.

WMT'13 En→Ru Same as in WMT'14 En→De, except that Dropout is 0.3.

A.1 GPU Power Consumption

Dataset	IWSLT'14 En→De	WMT'14 De→En	WMT'13 En→Ru
GPU(s)	1 × RTX 2080 Ti	2 × RTX 3090	4 × Tesla V100 SXM2
Power consumption, W	250	2×320	4×300
		Training time, hours	
softmax	15.41	30.06	28.43
sparsemax	24.43	73.33	58.82
1.5-entmax	26.11	79.89	61.20
1.5-ReLU	16.50	31.44	24.21
TOTAL hours	82.44	214.72	172.67
TOTAL kW-hours	20.61	137.42	207.20
GRAND TOTAL kW-hours		**365.23**	

Table 5: Power consumed by GPUs for training.

We do not report CO_2 consumption, as experiments were run in different countries, making aggregate statistics difficult to compute. The largest experiment (on WMT'13), were run in [MASKED], which benefits from a very low CO_2 emission intensity in its electrical mix.

B Proofs

Notation. We let $\mathbb{R}$ denote the real numbers. Bold-faced lowercase letters ($\mathbf{x}$) denote vectors in Euclidean space, bold-faced uppercase letters ($\mathbf{A}$) denote matrices, plain-faced lowercase letters (x) denote scalars, $\|\cdot\|$ denotes the Euclidean norm: $\|\mathbf{x}\| := \sqrt{\mathbf{x}^\top \mathbf{x}}$. The gradient of $f : \mathbb{R}^d \to \mathbb{R}$ is denoted by ∇f. The Jacobian of $\mathbf{z} \mapsto g(\mathbf{z})$ is denoted by $\mathbf{J}_g(\mathbf{z})$. Also, we denote $\mathrm{ReLU}(x) := [x]_+ := \max\{x, 0\}$, $[d] := \{1, \ldots, d\}$, $\Delta^{d-1} := \{\mathbf{p} \in \mathbb{R}^d \mid \sum_i p_i = 1,\, p_i \geq 0\}$, $\mathbf{e}_y := (0, \ldots, 0, 1, 0, \ldots, 0)$ where 1 is at y^{th} position.

B.1 Proof of Lemma 1.

First, let us calculate the Jacobian of the mapping $\mathbf{z} \mapsto \alpha\text{-ReLU}(\mathbf{z})$. Recall that

$$\alpha\text{-ReLU}_i(\mathbf{z}) := [(\alpha-1)z_i - \tau]_+^{\frac{1}{\alpha-1}}.$$

Therefore, the partial derivatives are given by

$$\begin{aligned}
\frac{\partial[\alpha\text{-ReLU}_i(\mathbf{z})]}{\partial z_i} &= \frac{1}{\alpha-1}\cdot[(\alpha-1)z_i-\tau]_+^{\frac{1}{\alpha-1}-1}\cdot(\alpha-1) = [(\alpha-1)z_i-\tau]_+^{\frac{2-\alpha}{\alpha-1}}, \\
&= [\alpha\text{-ReLU}_i(\mathbf{z})]^{2-\alpha} \\
\frac{\partial[\alpha\text{-ReLU}_i(\mathbf{z})]}{\partial z_j} &= 0. \qquad i \neq j
\end{aligned}$$

Thus, the Jacobian can be written concisely as

$$\mathbf{J}_{\alpha\text{-ReLU}}(\mathbf{z}) = \operatorname{diag}\{[\alpha\text{-ReLU}(\mathbf{z})]^{2-\alpha}\}, \tag{5}$$

where raising to power is done component-wise (i.e. $\mathbf{x}^\beta = [x_1^\beta, \ldots, x_d^\beta]$), and $\operatorname{diag}[\mathbf{x}]$ is a diagonal matrix with $\mathbf{x}$ on its diagonal.

Recall the definition of the Tsallis α-entropy:

$$\mathrm{H}_\alpha[\mathbf{p}] := \frac{1}{\alpha(\alpha-1)}\left(1-\sum_j p_j^\alpha\right).$$

Its gradient w.r.t. $\mathbf{p}$ is

$$\nabla_{\mathbf{p}}\,\mathrm{H}_\alpha[\mathbf{p}] = -\frac{1}{\alpha-1}\mathbf{p}^{\alpha-1},$$

Combining this with (5), and using the chain rule, we have

$$\begin{aligned}
\nabla_{\mathbf{z}}\,\mathrm{H}_\alpha[\alpha\text{-ReLU}(\mathbf{z})] &= [\mathbf{J}_{\alpha\text{-ReLU}}(\mathbf{z})]^\top\cdot\left(-\frac{1}{\alpha-1}[\alpha\text{-ReLU}(\mathbf{z})]^{\alpha-1}\right) \\
&= \left[\operatorname{diag}\{[\alpha\text{-ReLU}(\mathbf{z})]^{2-\alpha}\}\right]^\top\cdot\left(-\frac{1}{\alpha-1}[\alpha\text{-ReLU}(\mathbf{z})]^{\alpha-1}\right) \\
&= -\frac{1}{\alpha-1}[\alpha\text{-ReLU}(\mathbf{z})]^{2-\alpha}\odot[\alpha\text{-ReLU}(\mathbf{z})]^{\alpha-1} \\
&= -\frac{1}{\alpha-1}\,\alpha\text{-ReLU}(\mathbf{z}),
\end{aligned} \tag{6}$$

where $\odot$ is the Hadamard product (element-wise multiplication), and we used $\operatorname{diag}[\mathbf{x}]\cdot\mathbf{y} = \mathbf{x}\odot\mathbf{y}$. Taking into account (6), the gradient of the α-ReLU loss (2) w.r.t. $\mathbf{z}$ is

$$\begin{aligned}
\nabla_{\mathbf{z}}\ell(\mathbf{z},y) &= \nabla_{\mathbf{z}}\left[(\alpha\text{-ReLU}(\mathbf{z})-\mathbf{e}_y)^\top\left(\mathbf{z}-\frac{\tau}{\alpha-1}\mathbf{1}\right)\right] + \nabla_{\mathbf{z}}\,\mathrm{H}_\alpha[\alpha\text{-ReLU}(\mathbf{z})] \\
&= (\alpha\text{-ReLU}(\mathbf{z})-\mathbf{e}_y) + \mathbf{J}_{\alpha\text{-ReLU}(\mathbf{z})}^\top\left(\mathbf{z}-\frac{\tau}{\alpha-1}\mathbf{1}\right) - \frac{1}{\alpha-1}\,\alpha\text{-ReLU}(\mathbf{z}) \\
&= (\alpha\text{-ReLU}(\mathbf{z})-\mathbf{e}_y) + \frac{1}{\alpha-1}\left[\operatorname{diag}\{[\alpha\text{-ReLU}(\mathbf{z})]^{2-\alpha}\}\right]^\top[(\alpha-1)\mathbf{z}-\tau\mathbf{1}] - \frac{1}{\alpha-1}\,\alpha\text{-ReLU}(\mathbf{z}) \\
&= (\alpha\text{-ReLU}(\mathbf{z})-\mathbf{e}_y) + \frac{1}{\alpha-1}\underbrace{[(\alpha-1)\mathbf{z}-\tau\mathbf{1}]_+^{\frac{2-\alpha}{\alpha-1}}\odot[(\alpha-1)\mathbf{z}-\tau\mathbf{1}]}_{\alpha\text{-ReLU}(\mathbf{z})} - \frac{1}{\alpha-1}\,\alpha\text{-ReLU}(\mathbf{z}) \\
&= \alpha\text{-ReLU}(\mathbf{z})-\mathbf{e}_y,
\end{aligned}$$

where in the fourth line we used $[\mathbf{x}]_+^\beta\odot\mathbf{x} = [\mathbf{x}]_+^\beta\odot[\mathbf{x}]_+ = [\mathbf{x}]_+^{\beta+1}$. This concludes the proof.

B.2 Approximation of τ for 1.5-entmax

We derive the formula (3) in two steps: first in Lemma 2, we approximate $\tau(\mathbf{z})$ of 1.5-entmax when $\mathbf{z}$ is an arbitrary random sample from the normal distribution with zero mean and variance σ^2; next in Lemma 3, we compute σ^2 for the case when $\mathbf{z}$ is the pre-softmax vector of logits in the Transformer model.

Lemma 2. *Let $z_1, \ldots, z_d$ be independent and identically distributed random variables from the normal distribution $\mathcal{N}(0, \sigma^2)$. Then the thresholding function of $1.5\text{-entmax}(\mathbf{z})$ can be approximated as*

$$\tau(\mathbf{z}) \approx \frac{\sigma}{2}\Phi^{-1}(1-p^*),$$

where $\Phi^{-1}(\cdot)$ is a probit function, and p^* is the solution of

$$\Phi^{-1}(1-p) = m(p) - \sqrt{\frac{4}{\sigma^2}\cdot\frac{\epsilon}{p} - s(p)}$$

with

$$m(p) := \frac{1}{p-\epsilon}\left[\phi(\Phi^{-1}(x)\right]_{x=\epsilon}^{x=p} \tag{7}$$

$$s(p) := \frac{1}{p-\epsilon}\left[-\phi(\Phi^{-1}(x))\cdot\Phi^{-1}(x) + x\right]_{x=\epsilon}^{x=p} - [m(p)]^2 \tag{8}$$

$$\phi(t) := \frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}}$$

$$\epsilon := \frac{1}{d}$$

Proof. Let $z_{(1)} \geq \ldots \geq z_{(d)}$ be a sorting of $z_1, \ldots, z_d$ in descending order. Peters et al. (2019) showed that

$$\tau(\mathbf{z}) = \frac{M(k)}{2} - \sqrt{\frac{1}{k} - \frac{S(k)}{4}}, \tag{9}$$

where $k \in [d]$ is any index that satisfies

$$\frac{z_{(k)}}{2} \geq \frac{M(k)}{2} - \sqrt{\frac{1}{k} - \frac{S(k)}{4}} \geq \frac{z_{(k+1)}}{2} \quad \Leftrightarrow \quad z_{(k)} \geq M(k) - \sqrt{\frac{4}{k} - S(k)} \geq z_{(k+1)} \tag{10}$$

with

$$M(k) := \frac{1}{k}\sum_{i=1}^{k} z_{(i)}, \qquad S(k) := \frac{1}{k}\sum_{i=1}^{k} z_{(i)}^2 - [M(k)]^2.$$

Approximating $z_{(i)}$ by its asymptotic mean $\sigma\Phi^{-1}\left(1-\frac{i}{d}\right)$ (Arnold et al., 2008), and denoting $\epsilon := \frac{1}{d}$, $p := \frac{k}{d}$, we have

$$M(k) \approx \frac{1}{k}\sum_{i=1}^{k}\sigma\Phi^{-1}\left(1-\frac{i}{d}\right) \approx \frac{\sigma}{p-\epsilon}\int_{\epsilon}^{p}\Phi^{-1}(1-x)dx = \frac{\sigma}{p-\epsilon}\int_{\epsilon}^{p} -\Phi^{-1}(x)dx$$

$$= \frac{\sigma}{p-\epsilon}\left[\phi(\Phi^{-1}(x))\right]_{x=\epsilon}^{x=p} = \sigma m(p),$$

where we approximated the average of finitely many numbers $\{\Phi^{-1}(1-i/d)\}_{i=1}^{k}$ by the mean value of the function $\Phi^{-1}(1-x)$, and then we used the fact that $-\phi(\Phi^{-1}(x))$ is an antiderivative for the probit function $\Phi^{-1}(x)$; and $m(p)$ is defined by (7).

Similarly, for the second empirical moment, we have

$$\frac{1}{k}\sum_{i=1}^{k} z_i^2 \approx \frac{1}{k}\sum_{i=1}^{k}\left[\sigma\Phi^{-1}\left(1-\frac{i}{d}\right)\right]^2 \approx \frac{\sigma^2}{p-\epsilon}\int_{\epsilon}^{p}[\Phi^{-1}(1-x)]^2dx = \frac{\sigma^2}{p-\epsilon}\int_{\epsilon}^{p}[\Phi^{-1}(x)]^2dx$$

$$= \frac{\sigma^2}{p-\epsilon}\left[-\phi(\Phi^{-1}(x))\cdot\Phi^{-1}(x) + x\right]_{x=\epsilon}^{x=p},$$

and thus

$$S(k) \approx \frac{\sigma^2}{p-\epsilon}\left[-\phi(\Phi^{-1}(x))\cdot\Phi^{-1}(x)+x\right]_{x=\epsilon}^{x=p} - [m(p)]^2 = \sigma^2 s(p),$$

where $s(p)$ is defined by (8). Hence, finding $k \in [d]$ that satisfies (10) is (approximately) equivalent to finding $p \in (0,1)$ that satisfies

$$\sigma\Phi^{-1}(1-p) = \sigma m(p) - \sqrt{4\cdot\frac{\epsilon}{p} - \sigma^2 s(p)} \quad \Leftrightarrow \quad \Phi^{-1}(1-p) = m(p) - \sqrt{\frac{4}{\sigma^2}\cdot\frac{\epsilon}{p} - s(p)}. \tag{11}$$

Let p^* be the solution of (11). Then, taking into account (9), we have

$$\tau(\mathbf{z}) \approx \frac{\sigma m(p^*)}{2} - \sqrt{\frac{\epsilon}{p^*} - \frac{\sigma^2 s(p^*)}{4}} = \frac{\sigma}{2}\left(m(p^*) - \sqrt{\frac{4}{\sigma^2}\cdot\frac{\epsilon}{p^*} - s(p^*)}\right) = \frac{\sigma}{2}\Phi^{-1}(1-p^*),$$

which concludes the proof. □

Lemma 3. *Let $\mathbf{z} = \mathbf{W}\mathbf{x}$ be a pre-softmax vector of logits in the OpenNMT-py (Klein et al., 2017) implementation of the Transformer model (Vaswani et al., 2017). Then for any input, in a non-trained model the logits $z_1, \ldots, z_d$ are distributed according to the normal distribution $\mathcal{N}\left(0, \frac{2\cdot d_{model}}{d_{model}+d_{vocab}}\right)$, where d_{model} is the size of hidden representations, and d_{vocab} is the vocabulary size for a target language.*

Proof. The default Transformer configuration in OpenNMT-py implies that the elements w_{ij} of $\mathbf{W}$ are initialized from a uniform distribution $\mathcal{U}[-a,a]$, where $a = \sqrt{\frac{6}{d_{\text{model}}+d_{\text{vocab}}}}$, thus

$$\mathbb{E}[w_{ij}] = 0, \qquad \mathrm{Var}[w_{ij}] = \frac{(2a)^2}{12} = \frac{a^2}{3} = \frac{2}{d_{\text{model}}+d_{\text{vocab}}} \tag{12}$$

Since $\mathbf{x}$ is the result of a layer normalization (Ba et al., 2016), we have

$$\frac{1}{d_{\text{model}}}\sum_{j=1}^{d_{\text{model}}} x_j = 0, \qquad \frac{1}{d_{\text{model}}}\sum_{j=1}^{d_{\text{model}}} x_j^2 = 1 \tag{13}$$

Therefore, from (12) and (13), we have

$$\mathbb{E}[z_i] = \mathbb{E}\left[\sum_{j=1}^{d_{\text{model}}} w_{ij}x_j\right] = \sum_{j=1}^{d_{\text{model}}} \mathbb{E}[w_{ij}]\cdot x_j = 0,$$

$$\mathrm{Var}[z_i] = \mathrm{Var}\left[\sum_{j=1}^{d_{\text{model}}} w_{ij}x_j\right] = \frac{2}{d_{\text{model}}+d_{\text{vocab}}}\sum_{j=1}^{d_{\text{model}}} x_j^2 = \frac{2\cdot d_{\text{model}}}{d_{\text{model}}+d_{\text{vocab}}}.$$

Being a sum of independent random variables, by the Central Limit Theorem, each z_i tends to normal distribution with the mean and variance above. □

B.3 Derivation of the Equation (4)

We provide derivation for the case of α-ReLU. Extension to α-entmax and softmax is done analogously. Let $\mathbf{x} \in \mathbb{R}^{n_0}$ be the input vector. We define a feedforward neural network with $L-1$ hidden layers recursively:

$$\mathbf{h}^{(0)} = \mathbf{x}$$

$$\mathbf{z}^{(k)} = \frac{1}{\sqrt{n_{k-1}}}\mathbf{W}^{(k-1)}\mathbf{h}^{(k-1)},$$

$$\mathbf{h}^{(k)} = \sigma(\mathbf{z}^{(k)}), \quad k = 1, \ldots, L-1$$

where $\mathbf{W}^{(k-1)} \in \mathbb{R}^{n_k \times n_{k-1}}$ is the weight matrix in the k^{th} hidden layer, and $\sigma(\cdot)$ is a nonlinear activation function applied element-wise. We consider the case of a multi-label classification, i.e. the output layer is a vector

$$\mathbf{z} := \mathbf{z}^{(L)} \in \mathbb{R}^d,$$

which is fed into the α-ReLU loss:

$$\ell(\mathbf{z}, y) = (\alpha\text{-ReLU}(\mathbf{z}) - \mathbf{e}_y)^\top \left(\mathbf{z} - \frac{\tau}{\alpha - 1} \mathbf{1} \right) + \mathrm{H}_\alpha[\alpha\text{-ReLU}(\mathbf{z})], \tag{14}$$

where $\mathrm{H}_\alpha[\mathbf{p}] := \frac{1}{\alpha(\alpha-1)} \sum_j (p_j - p_j^\alpha)$, $\alpha \neq 1$, is the Tsallis α-entropy (Tsallis, 1988). Given a training sample $S := \{(x, y)\}$ learning is performed by minimizing the training error

$$\mathcal{L} := \mathbb{E}_{(x,y)\sim S}[\ell(\mathbf{z}(x), y)] \tag{15}$$

with respect to the network parameters $\boldsymbol{\theta} := \mathrm{vec}\left(\{\mathbf{W}^{(k-1)}\}_{k\in[L-1]}\right)$.

Lemma 4. *Let the training error (15) be minimized by gradient descent with infinitesimally small learning rate. Let $\mathbf{z}(x, t) \in \mathbb{R}^d$ be the network output on any training instance x at time t, and y be the desired output. Then, as the widths of hidden layers $n_k \to \infty$, $\forall k \in [L-1]$, the output $\mathbf{z}(x, t)$ follows the following evolution*

$$\frac{d\mathbf{z}}{dt} = - \mathop{\mathbb{E}}_{(x',y')\sim S}[\mathbf{K}(x, x') \cdot (\alpha\text{-ReLU}(\mathbf{z}') - \mathbf{e}_{y'})], \tag{16}$$

where $\mathbf{K}(x, x') \in \mathbb{R}^{d\times d}$ is a positive semidefinite matrix, and $\mathbf{z}' := \mathbf{z}(x', t)$.

Proof. From (15) and Lemma 1 we have

$$\nabla_{\mathbf{z}} \mathcal{L} = \nabla_{\mathbf{z}} \mathbb{E}_{(x',y')\sim S}[\ell(\mathbf{z}', y)] = \nabla_{\mathbf{z}} \ell(\mathbf{z}, y) = \alpha\text{-ReLU}(\mathbf{z}) - \mathbf{e}_y, \tag{17}$$

where we denoted $\mathbf{z} := \mathbf{z}(x, t)$ and $\mathbf{z}' := \mathbf{z}(x', t)$ for shorthand. Now, consider the gradient descent update

$$\boldsymbol{\theta}_{t+\eta} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L} \quad \Leftrightarrow \quad \frac{\boldsymbol{\theta}_{t+\eta} - \boldsymbol{\theta}_t}{\eta} = -\nabla_{\boldsymbol{\theta}} \mathcal{L}, \tag{18}$$

where η is the learning rate. Taking the limit in (18) as $\eta \to 0$, we have:

$$\frac{d\boldsymbol{\theta}}{dt} = -\nabla_{\boldsymbol{\theta}} \mathcal{L} = - \mathbb{E}_{(x',y')\sim S}[\mathbf{J}_{\mathbf{z}'}^\top(\boldsymbol{\theta}) \cdot \nabla_{\mathbf{z}'} \mathcal{L}],$$

where the last equality is due to the chain rule. Combining this with (17), we get

$$\frac{d\boldsymbol{\theta}}{dt} = - \mathbb{E}_{(x',y')\sim S}[\mathbf{J}_{\mathbf{z}'}^\top(\boldsymbol{\theta}) \cdot (\alpha\text{-ReLU}(\mathbf{z}') - \mathbf{e}_{y'})] \tag{19}$$

Applying the chain rule again, and using (19), we have

$$\frac{d\mathbf{z}}{dt} = \mathbf{J}_{\mathbf{z}}(\boldsymbol{\theta}) \cdot \frac{d\boldsymbol{\theta}}{dt} = - \mathbb{E}_{(x',y')\sim S}[\underbrace{\mathbf{J}_{\mathbf{z}}(\boldsymbol{\theta}) \mathbf{J}_{\mathbf{z}'}^\top(\boldsymbol{\theta})}_{\mathbf{K}(x,x';\boldsymbol{\theta})} \cdot (\alpha\text{-ReLU}(\mathbf{z}') - \mathbf{e}_{y'})].$$

The quantity $\mathbf{K}(x, x'; \boldsymbol{\theta})$ was named the *Neural Tangent Kernel* by Jacot et al. (2018). They also showed (see their Theorem 1) that

$$\mathbf{K}(x, x'; \boldsymbol{\theta}) \to \mathbf{K}(x, x') \quad \text{as} \quad n_1, \ldots, n_{L-1} \to \infty,$$

where $\mathbf{K}(x, x') \in \mathbf{R}^{d\times d}$ is the deterministic kernel that does not depend on $\boldsymbol{\theta}$. This concludes the proof. □

C Instructions for Human Annotators

You are shown a reference sentence and several candidate translations. Please indicate, for each, on a 4-point scale, how much of the meaning is represented in the translation, ignoring the language quality.

Imagine you are a forgiving reader, ignoring any error that does not prevent you from getting the meaning of the text. So please ignore language oddities, typographic errors and the like. (This is difficult but key to us!)

The scale of *meaning preservation* is: 4 = Everything / 3 = Most / 2 = Little / 1 = None

As we are interested in comparing system's output, you can refine your judgement using $+$ or $-$, e.g. $3+$.

When you do not know, simply leave empty.

For instance, given the reference sentence

"This restaurant is beautiful and the staff is very friendly",

valid judgements for different translations are provided in Table 6.

Score	Sentence
4	"This restaurant is beautiful and the staff is very friendly."
4	"This restaurant is beautiful and the staff is very friendly.."
4	"Beautiful restaurant, staff is very friendly."
4−	"This restaurant is beautiful and the staff is friendly."
4−	"Beautiful restaurant, staff is friendly."
2+	"Friendly staff"
2	"This is a restaurant."
1	"Hello guys!"
1	"Bad restaurant"
1−	"Bad restaurant, bad staff"

Table 6: Evaluation example

We insist that evaluating by meaning differs from a natural intuitive evaluation. Provided the meaning is not impacted, we want to ignore the language quality, the punctuation, the casing.

D Translation Examples

Source	Flake was the central figure in *Friday's* drama.	
1.5-entmax	Flèjk byl central'noj figuroj v drame pjatnadcatogo veka.	Flake was the central figure in fifteenth century drama.
1.5-ReLU	Flèjk byl central'noj figuroj v drame pjatnicy.	Flake was the central figure in Friday's drama.
Source	There were smiles and blue skies on Saturday (September 29) as the leaders of Turkey and Germany met for *breakfast* in Berlin.	
1.5-entmax	V subbotu (29 sentjabrja) byli ulybki i goluboe nebo, poskol'ku lidery Turcii i Germanii vstretilis' dlja razvala v Berline.	There were smiles and blue skies on Saturday (September 29) as the leaders of Turkey and Germany met for a breakup in Berlin.
1.5-ReLU	V subbotu (29 sentjabrja) byli ulybki i goluboe nebo, tak kak lidery Turcii i Germanii vstretilis' dlja otdyha v Berline.	There were smiles and blue skies on Saturday (September 29) as the leaders of Turkey and Germany met for a holiday in Berlin.
Source	That Was *Really Bad* Body Language:	
1.5-entmax	Èto byl dejstvitel'no plohoj jazyk tela	That was really bad body language.
1.5-ReLU	Èto byl real'nyj jazyk tela	That was real body language.
Source	The city of Palu, which has more than 380,000 people, *was strewn with debris from collapsed buildings*	
1.5-entmax	Gorod Palu, v kotorom prozhivaet bolee 380 000 chelovek, byl razrushen zdanijami.	The city of Palu, home to over 380,000 people, was destroyed by buildings.
1.5-ReLU	Gorod Palu, u kotorogo bolee 380 000 chelovek, nahodilsja v upadke zdanija.	The city of Palu, which has over 380,000 inhabitants, was in decay building.

Table 7: Translation Examples