interpretable Model Architectures

Aida Mostafazadeh Davani
mostafaz@usc.edu

1 A general framework for self explaining neural networks

Alvarez-Melis and Jaakkola (2018) propose three desiderata for model explanations in general - explicitness, faithfulness, and stability - and show that estimating *a posteriori* explanations for previously trained models around specific predictions do not satisfy them. Their method is a generalized form of linear models, $f(x) = \sum_i^n \theta_i x_i + \theta_0$. Allowing the θs to depend on input x, results in more general model, $f(x) = \theta(x)^T x$, where θ is chosen from a complex model class Θ (can be any type of NN).

In order for the model to be interpretable, the θ for two inputs x and x', that are close to each other, should not change dramatically (θ should be tractable). For example, the model can be regularized in a manner that for all x's in the neighborhood of x_0, $\nabla_x f(x) \approx \theta(x_0)$. The individual values $\theta(x_0)_i$ act as and are interpretable as coefficients of a linear model with respect to the final prediction, but adapt dynamically to the input, albeit varying slower than x.

They also introduce *interpretable basis concepts* and use them instead of words or pixels by defining a function $h(x) : X \to Z \subset \mathbb{R}^k$ where Z is some space of interpretable atoms. So the generalized model becomes:

$$f(x) = \theta(x)^T h(x) = \sum_i^k \theta(x)_i h(x)_i \tag{1}$$

A more generalized aggregation can substitute the summatoin in (1) if it is permutation invariant, can isolate $h(x)_i$s and preserve the sign and relative magnitude of the impact of relevance $\theta(x)_i$s.

Therefore a general definition of the model can be expressed as: let $x \in X \subset R^n$ and $Y \subset R^m$ be the input and output spaces. We say that $f : X \to Y$ is a self-explaining prediction model if it has the form:

$$f(x) = g(\theta_1(x)h_1(x), ..., \theta_k(x)h_k(x) \tag{2}$$

In that case, for a given input x, we define the explanation of $f(x)$ to be the set $\xi f(x) \equiv \{(h_i(x), \theta_i(x))\}_{i=1}^k$ of basis concepts and their influence scores.

$\theta(.)$ (and potentially $h(.)$) are realized by architectures with large modeling capacity, such as deep neural networks. When $\theta(x)$ is realized with a neural network, we refer to f as a self-explaining neural network (SENN). If g depends on its arguments in a continuous way, f can be trained end-to-end with back-propagation.

1.1 Model properties

Properties we wish to impose on θ in order for it to act as coefficients of a linear model on the basis concepts $h(x)$ are:

1. g is monotone and completely additively separable
2. For every $z_i := \theta_i(x)h_i(x)$, g satisfies $\frac{\partial g}{\partial z_i} \geq 0$
3. θ is locally difference bounded by h
4. $h_i(x)$ is an interpretable representation of x

5. k is small

Definition: $f : X \subset R^n \rightarrow R^m$ is *locally difference bounded* by $h : X \subset R^n \rightarrow R^k$ if for every x_0 there exist $\delta > 0$ and $L \in R$ such that $||x - x_0|| \leq \delta$ implies $||f(x) - f(x_0)|| \leq L||h(x) - h(x_0)||$.

The first two conditions depend entirely on the choice of aggregating function g and the last two are application-dependent: what and how many basis concepts are adequate should be informed by the problem and goal at hand.

For the third condition let us consider what f would look like if the θ_i's were indeed (constant) parameters. Looking at f as a function of $h(x)$, i.e. $f(x) = g(h(x))$, let $z = h(x)$. Using the chain rule we get $\nabla_x f = \nabla_z f . J_x^h$, where J_x^h denotes the Jacobian of h (with respect to x). At a given point x_0, we want $\theta(x_0)$ to behave as the derivative of f with respect to the concept vector $h(x)$ around x_0, i.e., we seek $\theta(x_0) \approx \nabla_z f$. Since this is hard to enforce directly, we can instead plug this *ansatz* in $\nabla_x f = \nabla_z f \Delta J_x^h$ to obtain a proxy condition:

$$L_\theta(f(x)) := ||\nabla_x f(x) - \theta(x)^\top J_x^h(x)|| \approx 0 \tag{3}$$

All three terms in $L_\theta(f)$ can be computed when using differentiable architectures $h(.)$ and $\theta(.)$, we obtain gradients with respect to (3) through automatic differentiation and thus use it as a regularization term in the optimization objective. With this, we obtain a gradient-regularized objective of the form $L_y(f(x), y) + \lambda L_\theta(f)$, where the first term is a classification loss and λ a parameter that trades off performance against stability and therefore, interpretability of $\theta(x)$.

1.2 Interpretable basis concepts

The authors also introduce a reasonable minimal set of desiderata for interpretable concepts is

- Fidelity: the representation of x in terms of concepts should preserve relevant information (solution: training h as an autoencoder)
- Diversity: inputs should be representable with few non-overlapping concepts (solution: enforcing diversity through sparsity)
- Grounding: concepts should have an immediate human understandable interpretation (solution: prototyping (e.g., by providing a small set of training examples that maximally activate each concept))

Using an autoencoder for h, the reconstruction loss is added to the objective as a penalty:

$$L_y(f(x), y) + \lambda L_\theta(f) + \xi L_h(x, \hat{x}) \tag{4}$$

2 Attention as interpretation

Attention model (Bahdanau et al., 2014) have been widely used as an insight into how different features of the input data can have more impact in predicting an output in a task (Chaudhari et al., 2019). These studies have made use of attention for interpretability because it is believed to directly represent the internal working of the deep learning architectures.

However, other studies have been questioning this application and tried to test how attention is related to explanation. Here we analyze tow studies that look into simple attention weights and whether they can be interpreted as input importance.

Jain and Wallace (2019) compare attention weights with gradient-based measures of importance and leave-one-out (LOO) methods calculated in several tasks. They also generate adversarial attention distribution (counterfactual attention weights), $\hat{\alpha}$, that attend to different inputs but produce essentially identical predictions to argue that attention weights are not faithful (regarding explainability definition proposed by Ross et al. (2017)). Such counterfactual distributions imply that explaining the original prediction by highlighting attended-to tokens is misleading. To calculate the Total Variation Distance difference between two sets of predictions:

$$TVD(\hat{y_1}, \hat{y_2}) = \frac{1}{2} \sum_{i=1}^{|Y|} |\hat{y_{1i}} - \hat{y_{2i}}| \tag{5}$$

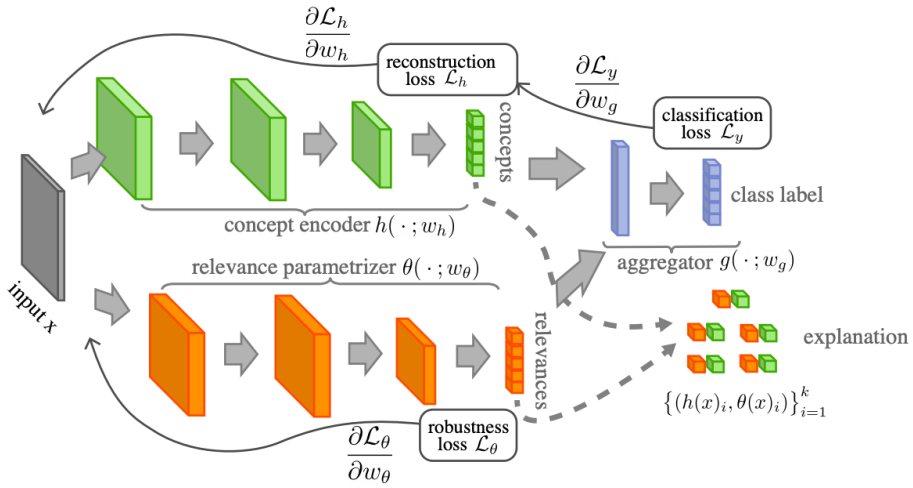


Figure 1: A SENN consists of three components: a concept encoder (green) that transforms the input into a small set of interpretable basis features; an input-dependent parametrizer (orange) that generates relevance scores; and an aggregation function that combines to produce a prediction. The robustness loss on the parametrizer encourages the full model to behave locally as a linear function on $h(x)$ with parameters $\theta(x)$, yielding immediate interpretation of both concepts and relevances.

They use the Jensen-Shannon Divergence (JSD) to quantify the difference between two attention distributions:

$$JSD(\alpha_1, \alpha_2) = \frac{1}{2} KL[\alpha_1 || \frac{\alpha_1 + \alpha_2}{2}] + KL[\alpha_2 || \frac{\alpha_1 + \alpha_2}{2}] \tag{6}$$

Three algorithms are suggested to compare attention to feature importance metrics, permuted attention and the adversarial attention weights.

Their results suggest that while attention modules consistently yield improved performance on NLP tasks, their ability to provide transparency or meaningful explanations for model predictions is, at

Algorithm 1 Feature Importance Computations

$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x})$, $\hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \alpha)$
$g_t \leftarrow |\sum_{w=1}^{|V|} \mathbb{1}[\mathbf{x}_{tw} = 1] \frac{\partial y}{\partial \mathbf{x}_{tw}}|\,, \forall t \in [1, T]$
$\tau_g \leftarrow \text{Kendall-}\tau(\alpha, g)$
$\Delta\hat{y}_t \leftarrow \text{TVD}(\hat{y}(\mathbf{x}_{-t}), \hat{y}(\mathbf{x}))\,, \forall t \in [1, T]$
$\tau_{loo} \leftarrow \text{Kendall-}\tau(\alpha, \Delta\hat{y})$

Algorithm 2 Permuting attention weights

$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x})$, $\hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \hat{\alpha})$
for $p \leftarrow 1$ to 100 **do**
  $\alpha^p \leftarrow \text{Permute}(\hat{\alpha})$
  $\hat{y}^p \leftarrow \text{Dec}(\mathbf{h}, \alpha^p)$ ▷ Note : $\mathbf{h}$ is not changed
  $\Delta\hat{y}^p \leftarrow \text{TVD}[\hat{y}^p, \hat{y}]$
end for
$\Delta\hat{y}^{med} \leftarrow \text{Median}_p(\Delta\hat{y}^p)$

Algorithm 3 Finding adversarial attention weights

$\mathbf{h} \leftarrow \text{Enc}(\mathbf{x})$, $\hat{\alpha} \leftarrow \text{softmax}(\phi(\mathbf{h}, \mathbf{Q}))$
$\hat{y} \leftarrow \text{Dec}(\mathbf{h}, \hat{\alpha})$
$\alpha^{(1)}, ..., \alpha^{(k)} \leftarrow$ Optimize Eq 1
for $i \leftarrow 1$ to k **do**
  $\hat{y}^{(i)} \leftarrow \text{Dec}(\mathbf{h}, \alpha^{(i)})$ ▷ $\mathbf{h}$ is not changed
  $\Delta\hat{y}^{(i)} \leftarrow \text{TVD}[\hat{y}, \hat{y}^{(i)}]$
  $\Delta\alpha^{(i)} \leftarrow \text{JSD}[\hat{\alpha}, \alpha^{(i)}]$
end for
ϵ-max JSD $\leftarrow \max_i \mathbb{1}[\Delta\hat{y}^{(i)} \leq \epsilon]\Delta\alpha^{(i)}$

best, questionable – especially when a complex encoder is used, which may entangle inputs in the hidden space.

Wiegreffe and Pinter (2019) evaluate the adversary method for testing attention conducted by Jain and Wallace (2019), mainly due to the per-instance nature of the demonstration and the fact that model parameters have not been learned or manipulated directly.

Wiegreffe and Pinter (2019) propose a model-consistent training protocol for finding adversarial attention distributions through a coherent parameterization, which holds across all training instances. Given the base model M_b, we train a model M_a whose explicit goal is to provide similar prediction scores for each instance, while distancing its attention distributions from those of M_b. Formally, we train the adversarial model using stochastic gradient updates based on the following loss formula (summed over instances in the minibatch):

$$L(M_a, M_b)^i = TVD(\hat{y}_a^i, \hat{y}_a^b) - \lambda KL(\alpha_a^i || \alpha_b^i \tag{7}$$

We can plot the two terms to evaluates the hypothesis that attention is interpretation based on comparative variances of each component regarding different λ'a.

3 Rationale based explanation models

3.1 Rationalizing neural predictions (Lei et al., 2016)

Lei et al. (2016) introduce a model that learn to extract pieces of input text as justifications – rationales – that are tailored to be short and coherent yet sufficient for making the same prediction.

The approach combines two modular components, generator and encoder, which are trained to operate well together. The generator specifies a distribution over text fragments as candidate rationales and these are passed through the encoder for prediction.

Target rationales are never provided in the training set The learning task is to minimize a cost function that favors short, concise rationales while enforcing that the rationales alone suffice for accurate prediction.

The learning problem can be defined as mapping the input sequence $x = \{x_1, x_2, ..x_n\}$, $x_t \in \mathbb{R}^d$ to a target vector $y \in [0, 1]^m$.

we can solve the associated learning problem by estimating a complex parameterized mapping `enc(x)` from input sequences to target vectors. We call this mapping an encoder. Squared error for a prediction $\tilde{y}$ is defined as:

$$loss(x, y) = ||\tilde{y} - y||_2^2 = ||enc(x) - y||_2^2 \tag{8}$$

The goal is to select a subset of the input sequence as a rationale 1) the selected words should be interpretable and 2) they ought to suffice to reach nearly the same prediction. To this end rationale generator, `gen(x)`, selects words from input sequences to shorter sequences of words. Thus gen(x) must include only a few words and enc(gen(x)) should result in nearly the same target vector as the

original input passed through the encoder or enc(x). the generator is probabilistic and specifies a distribution over possible selections.

The rationale for a given sequence x can be equivalently defined in terms of binary variables $z_1, .., z_n$ where each $z_t \in 0, 1$ indicates whether word x_t is selected or not:

$$z \sim gen(x) \equiv p(z|x) \tag{9}$$

In a simple generator, the probability that the t^{th} word is selected can be assumed to be conditionally independent from other selections given the input x.

$$p(z|x) = \prod_{t=1}^{n} p(z_t|x) \tag{10}$$

So $p(z_t|x)$ can be produced by bi-LSTM and a fully connected layer with sigmoid activation.

$$p(z_t|x) = \sigma_z(W^z[\overrightarrow{h_t}, \overleftarrow{h_t}] + b^z) \tag{11}$$

In order to implement more conditioned selection:

$$p(z|x) = \prod_{t=1}^{n} p(z_t|x, z_1, .., z_{t-1}) \tag{12}$$

Which can also be expressed as a recurrent neural network that uses a second hidden state, s_t:

$$p(z_t|x, z_{1,t-1}) = \sigma_z(W^z[\overrightarrow{h_t}, \overleftarrow{h_t}; s_{t-1}] + b^z) \tag{13}$$

$$s_t = f_z([\overrightarrow{h_t}, \overleftarrow{h_t}; z_t], s_{t-1}) \tag{14}$$

(I think z_t should be replaced with z_{t-1})

They show the rationale $\{x_k|z_k = 1\}$ as (z,x); then:

$$L(z, x, y) = ||enc(z, x) - y||_2^2 \tag{15}$$

and to have short and consecutive rationales:

$$\Omega(z) = \lambda_1||z|| + \lambda_2 \sum_t |z_t - z_{t-1}| \tag{16}$$

where the first term penalizes the number of selections while the second one discourages transitions (encourages continuity of selections).

$$cost(z, x, y) = L(z, x, y) + \Omega(z) \tag{17}$$

$$\min_{\theta_e, \theta_g} \sum_{(x,y)\in D} \mathbb{E}_{z\sim gen(x)}[cost(z, x, y)] \tag{18}$$

where θ_e and θ_g denote the set of parameters of the encoder and generator, respectively, and D is the collection of training instances.

Minimizing the expected cost is challenging since it involves summing over all the possible choices of rationales z. They use stochastic gradient decent by deriving a sampled approximation to the gradient of the expected cost objective separately for each input text x. Consider a pair (x, y), for θ_g:

$$\frac{\partial \mathbb{E}_{z\sim gen(x)}[cost(z, x, y)]}{\partial \theta_g} \tag{19}$$

$$= \sum_z cost(z, x, y).\frac{\partial p(z|x)}{\partial \theta_g} \tag{20}$$

multiply by $\frac{p(z|x)}{p(z|x)}$ and then to $\partial logp(z|x)$

$$= \sum_z cost(z, x, y). \frac{\partial p(z|x)}{\partial \theta_g} \frac{p(z|x)}{p(z|x)} \tag{21}$$

$$= \sum_z cost(z, x, y). \frac{\partial logp(z|x)}{\partial \theta_g} p(z|x) \tag{22}$$

$$= \mathbb{E}_{z \sim gen(x)}[cost(z, x, y) \frac{\partial logp(z|x)}{\partial \theta_g}] \tag{23}$$

Same thing can be computed for θ_e The objective is intractable to compute, the lowerbound, in particular, requires marginalization of $O(2^n)$ binary sequences.

We can simply sample a few rationales z from the generator gen(x) and use the resulting average gradient in an overall stochastic gradient method. REINFORCE-style algorithm (Williams, 1992) where the gradient with respect to the parameters is estimated by sampling possible rationales.

Notes: Using RCNN works best. They tried it for multi-aspect Sentiment Analysis and Similar Text Retrieval on QA Forum

3.2 Interpretable Neural Predictions with Differentiable Binary Variables (Bastings et al., 2019)

They use the same approach as Lei et al. (2016), but as we mentioned because gradients do not flow through discrete samples, the rationale extractor is optimized using REINFORCE (Williams, 1992). An L_0 regularizer is used to make sure the rationale is short. Bastings et al. (2019) propose an alternative to purely discrete selectors for which gradient estimation is possible without REINFORCE, instead relying on a reparameterization of a random variable that exhibits both continuous and discrete behavior (Louizos et al., 2017), penalizing the objective as a function of the expected proportion of selected text and also use of Lagrangian relaxation to target a specific rate of selected input text.

They propose to replace Bernoulli variables by rectified continuous random variables that exhibit both discrete and continuous behaviour. They also employ a relaxed form of L_0 regularization (the first loss that involves the length of the rationale).

The Kumaraswamy distribution (Kumaraswamy, 1980) is a two-parameters distribution over the open interval (0, 1), we denote a Kumaraswamy distributed variable by $K \sim Kuma(a, b)$, where $a \in \mathbb{R}_{>0}$ and $b \in \mathbb{R}_{>0}$ control the distribution's shape. Figure 2 illustrates the density of Kuma(0.5, 0.5). The process includes:

1. start from a distribution over the open interval $(0, 1)$ (see dashed curve in Figure 2)
2. stretch its support from $l < 0$ to $r > 1$ in order to include $\{0\}$ and $\{1\}$, $T \sim Kuma(a, b, l, r)$ (see solid curve in Figure 2)
3. collapse the probability mass over the interval $(l, 0]$ to $\{0\}$, and similarly, the probability mass over the interval $[1, r)$ to $\{1\}$, $H \sim HardKuma(a, b, l, r)$ (shaded areas in Figure 2).

Using this distribution for z values:

$$z \sim HardKuma(a, b, l, r) \tag{24}$$

where shape parameters $a, b = g(x, \phi)$ and l, r are fixed hyperparameters.

They use Lagrangian relaxation

$$\max_{\lambda \in \mathbb{R}} \min_{\phi, \theta} -\mathbb{E}(\phi, \theta) + \lambda^\top (R(\phi) - r) \tag{25}$$

where $R(\phi)$ is a vector of regularisers, e.g. expected L_0 and expected fused lasso (in the original loss function), and λ is a vector of Lagrangian multipliers λ.

Then in a simple classification task:

$$e_i = emb(x_i), h_1^n = bi - rnn(e_1^n; \phi_r), u_i \sim U(0, 1) \tag{26}$$

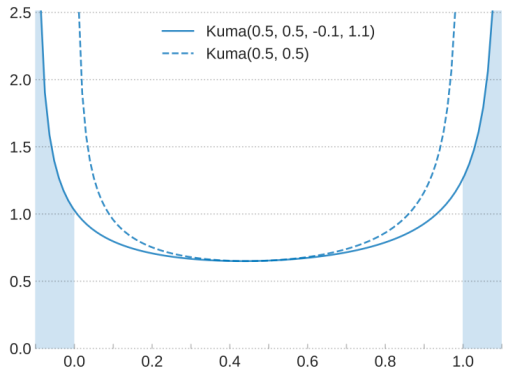


Figure 2: The HardKuma distribution: we start from a Kuma(0.5, 0.5), and stretch its support to the interval (-0.1, 1.1), finally we collapse all mass before 0 to $\{0\}$ and all mass after 1 to $\{1\}$.

$$a_i = f_a(h_i; \phi_a), b_i = f_b(h_i; \phi_b), z_i = s(u_i; a_i, b_i, l, r) \tag{27}$$

The sampled z is then used to modulate inputs to the classifier:

$$h_i = rnn(h_{i_1}, z_i e_i; \theta_h), o = f_o(h_n, \theta_o) \tag{28}$$

At test time, the model uses the most likely value for z_i, either $= 1$, $= 0$ or $0 < z_i < 1$.

4 Sequence to sequence explanation

In contrast with statistical machine translation (SMT), neural machine translators are not easily explainable because it is often difficult to extract a comprehensible explanation for the predictions of these models as information in the network is represented by real-valued vectors or matrices. It is tempting to interpret encoder-decoder attention matrices (Bahdanau et al., 2014) in neural models as (soft) alignments, but previous work has found that the attention weights in NMT are often erratic and differ significantly from traditional word alignments.

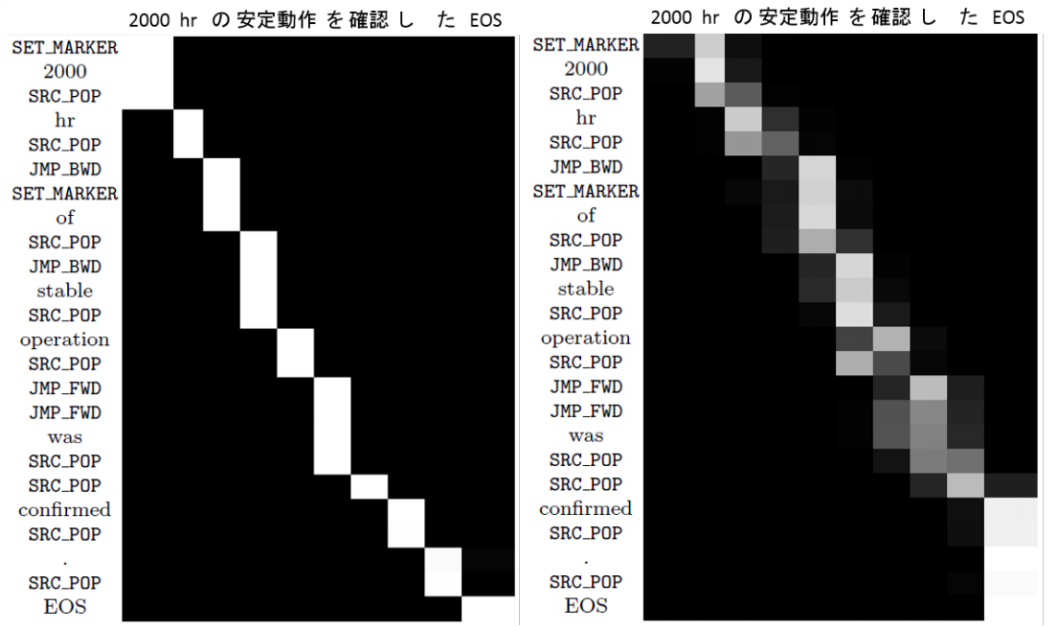
In Operation Sequence Neural Machine Translation (OSNMT; Stahlberg et al. (2018)), the neural seq2seq model learns to produce a sequence of operations by keeping track of the positions of a source-side read head and a target-side write head. The read head monotonically walks through the source sentence, whereas the position of the write head can be moved from marker to marker in the target sentence.

Operations:

- `POP SRC`: Move the read head right by one token.
- `SET MARKER`: Insert a marker symbol into the target sentence at the position of the write head.
- `JMP FWD`: Move the write head to the nearest marker right of the current head position in the target sentence.
- `JMP BWD`: Move the write head to the nearest marker left of the current head position in the target sentence.
- `INSERT(t)`: Insert a target token t into the target sentence at the position of the write head.

Word alignments can be extracted from the operation sequence in form of 1:n source-target relations.

For training the model the target sequence is not a plain sequence of subword or word tokens but a sequence of operations. The operation sequence for the target sentences is generated by a statistical word aligner and removing all alignments that violate the 1:n relation. They use an algorithm to generate operation sequences based on alignments. The neural model then learns to align and translate at the same time.



(a) Layer 4, head 1; attending to the source side read head.

(b) Layer 2, head 3; attending to the right trigram context of the read head.

Figure 3:

We have found that many of these attention matrices have strong and interpretable links to the translation process represented by the OSNMT sequence. For example, Figure 3a shows that the first head in layer 4 follows the source-side read head position very closely: at each SRC POP operation the attention shifts by one to the next source token. Other attention heads have learned to take other responsibilities. For instance, head 3 in layer 2 (Figure 3b) attends to the trigram right of the source head.

References

David Alvarez-Melis and Tommi S Jaakkola. 2018. Towards robust interpretability with self-explaining neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7786–7795. Curran Associates Inc.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Joost Bastings, Wilker Aziz, and Ivan Titov. 2019. Interpretable neural predictions with differentiable binary variables. *ACL*.

Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. 2019. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. *arXiv preprint arXiv:1902.10186*.

Ponnambalam Kumaraswamy. 1980. A generalized probability density function for double-bounded random processes. *Journal of Hydrology*, 46(1-2):79–88.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing neural predictions. *EMNLP*.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.

Felix Stahlberg, Danielle Saunders, and Bill Byrne. 2018. An operation sequence model for explainable neural machine translation. *arXiv preprint arXiv:1808.09688*.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.