Interpreting Self-Attention Weights

Chris Varano
Amazon
Palo Alto, CA 94303
`cvarano@stanford.edu`
Mentor: Abigail See

Abstract

A key component of designing effective neural architectures is having an understanding of how the models and the data interact. Recent work in NLP has developed large-scale, pre-trained stacks that can be applied to downstream NLP tasks with fine-tuning, revolutionizing NLP in the same way pre-trained ImageNet revolutionized computer vision. Understanding these nascent pre-trained stacks will give us a better understanding of the limitations of these methods, enabling further improvements to the their capabilities. In this work, we analyze the behaviour of a Directional Self-Attention Network (DiSAN) model, pre-trained on the SNLI dataset, to understand the underlying linguistic structure it learns from the data. We propose a method for discovering latent structure encoded in the self-attention weights by measuring the KL divergence between attention distributions and our proposed dependency distributions. Quantitative and qualitative experiments show that DiSAN does in fact encode some degree of linguistic structure.

1 Introduction

Pre-trained models for buiding contextualized representations of language continue to push state-of-the-art in NLP benchmarks. Understanding the underlying structure that they learn and what limitations they might have is essential for designing new architectures or introducing more inductive bias into existing architectures.

State-of-the-art pre-trained stacks - such as ELMo[11], BERT[6], and DiSAN[13] - all make use of self-attention layers. It is plausible that self-attention layers in particular are prime locations for learning linguistic structure, as they handle pairwise relationships between tokens. The models are not explicitly trained to learn linguistic structure, but we would like to understand whether the self-attention layers naturally acquire the underlying structure in the same way that CNN filters learn the underlying physical structure of real-world objects.

In this work, we propose a general method for measuring the encoded structure in self-attention distributions[1] using KL divergence (KLD). Specifically, we use our proposed method to measure the existence of dependency tree information encoded in the self-attention weights. We convert dependency trees into a form homologous with self-attention weights for comparison, and measure the KLD of the weights from the dependency trees.

Our quantitative analysis shows that some degree of dependency information is in fact encoded, and our qualitative analysis examines the weights for different self-attention dimensions to understand the behaviour more concretely.

[1]We use the terms attention weights and attention distributions interchangeably throughout this work.

2 Related Work

Directional Self-Attention Network. Directional Self-Attention Network (DiSAN) is a light-weight neural net for learning sentence embeddings[13]. Our work involves extensive analysis of a DiSAN model, so here we provide a brief overview of the authors' contributions. We provide some more context in Section 3.1, but for a more thorough explanation of all the components we refer the reader to the original paper.

The authors proposed a novel attention mechanism that is directional and multi-dimensional. Multi-dimensional attention extends additive attention to produce a vector of values instead of a scalar value, achieved by replacing the final weight vector with a weight matrix. Directional self-attention (DiSA) adds a forward/backward directional mask prior to the attention softmax to zero out tokens before/after the current token in order to encode temporal information. The final step of the DiSA block is a fusion gate which learns to take a convex combination of the self-attended token representations and the original token representations.

DiSAN itself is composed of two parameter-untied DiSA blocks, one forward and one backward. The output of the DiSA blocks are concatenated and processed by another multi-dimensional self-attention layer which produces the final sentence embedding. The DiSAN authors do a simple case study of the network by visualizing the the self-attention weights, shown in Figure 1. These are the weights that will be analyzed in this work.

Probing methods. Probing methods are designed to evaluate the extent to which specific linguistic structures are encoded in a given representation. For example, probes can be designed to measure part-of-speech, morphology, sentence length, or syntax trees[4, 12, 2, 8]. A probe is a supervised model for finding precisely defined information in a representation. These methods are particularly relevant to our work as they have the same goal of discovering linguistic structure.

Interpretability methods. A variety of methods have been proposed for interpreting not only the intermediate representations of models, but also interpreting the model's "reasoning". In [7] they interpret intermediate layers of Natural Language Inference (NLI) models using saliency visualizations. In [1] they investigate the role of structural compression in interpreting CNNs. [14] uses decision tree models to interpret model decisions at the semantic level. In [9] they use a few visualization techniques to interpret the underlying semantic behaviour of CNNs in the context of NLP. Finally, Layer-wise Releveance Propagation[3] is a commonly used methodology with CNNs to visualize the contributions of individual pixels to the model predictions.

3 Measuring dependency information in self-attention weights

We propose a method of measuring the distance between the model's self-attention distributions and the distributions produce from dependency tree information. We implemented and pre-trained a DiSAN model as our self-attention model[2]. We construct dependency trees and random tree baselines for our validation dataset, and convert the trees into the form of self-attention distributions in order to make them comparable with DiSAN's token2token self-attention distributions.

3.1 DiSAN token2token multi-dimensional self-attention

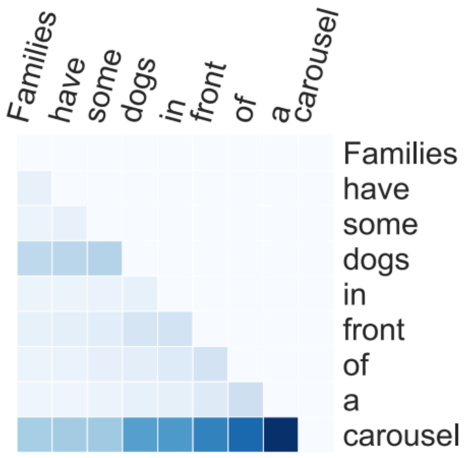
Here we present a few more relevant details of multi-dimensional "token2token" self-attention from [13], but we refer the reader to the original paper for a full explanation.

Token2token self-attention explores the dependency between x_i and x_j from the same source x, and generates context-aware coding for each element.

$$f(x_i, x_j) = W^T \sigma(W^{(1)} x_i + W^{(2)} x_j + b^{(1)}) + b$$

Computing this for all pairs of tokens results in a tensor of logits in $\mathbb{R}^{n \times n \times d_e}$, where n is the sequence length and d_e is the number of dimensions. Softmax is then applied to each column in each dimension, i.e. $n \times d_e$ softmax computations, representing the attention distributions for each x_j attending to each x_i in every dimension. These self-attention weights are what we will be analyzing in this work.

[2]**For graders**: We did the core implementation of the DiSAN models, but cross-referenced with https://github.com/taoshen58/DiSAN and https://github.com/baaesh/DiSAN-pytorch as a sanity check.



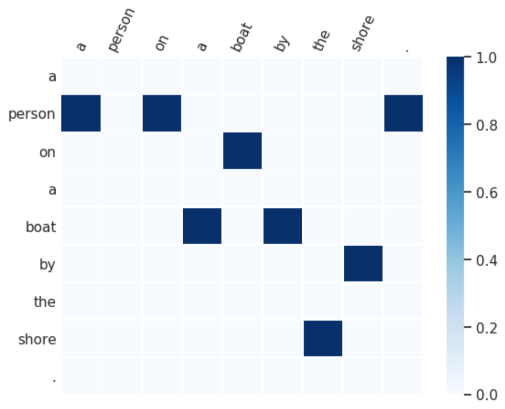


Figure 1: DiSAN case study of token2token self-attention weights (left) and dependency distributions constructed from dependency tree information (right)

3.2 Incorporating dependency tree information

Overview. From a dependency tree, we can read to which tokens a given token "attends"; that is, for a given token in the tree, we can think of it as attending uniformly to all of it's children. Thus for every token in the sentence we will have a probability distribution across all other tokens in the sentence, homologous with the self-attention weights. We henceforth refer to these constructed distributions as **dependency distributions**. It is easy to extend this method in different ways. For example, we could incorporate prior linguistic knowledge to alter the distribution from uniform to something more informed. In fact, this method is not limited to measuring dependency structures, but can measure any structure that can be formulated as self-attentive.

Our method. The specific method we use in this work is to have each child attend to its parent. Since this is a tree, each child has exactly one parent, and so the resulting distribution puts all of the probability mass on its sole parent. The ROOT is not considered part of the sentence, so the token whose parent is the ROOT does not attend to anything, and does not contribute to the final KL divergence measurement. It is worth noting that, by construction, ROOT always has exactly one child token. An example dependency distribution is shown in Figure 1.

In order to get dependency trees, we apply the Stanford CoreNLP[10] dependency parser to all sentences in our validation dataset. We use the "English SD" model with the default configuration.

3.3 Constructing random dependency trees

As a baseline comparison, we produce a random dependency tree for each of the sentences in our validation dataset. We implemented an arc-standard transition-based dependency parser that chooses an available action uniformly at random[3]. Furthermore, to mitigate any bias incurred by the sequential nature of the data, the tokens are shuffled (tupled with their original index) before being put into the buffer. The desired result is a valid dependency tree chosen uniformly at random from the space of all dependency trees containing the given set of tokens. Manual inspection of the resulting random trees, as well as some quantitative results discussed in Section 5.1, suggest that our implementation produces the desired result.

[3]The possible actions are shift, left-arc, and right-arc, as covered in lectures. The parser only chooses between available actions, thus cannot, for example, shift when the buffer is empty.

4 Experimental setup

4.1 Dataset

Our DiSAN model is pre-trained on the Stanford Natural Language Inference (SNLI) dataset[5]. The NLI task is as follows: given two sentences, a premise and a hypothesis, classify whether the hypothesis is entailed by the premise, contradicts the premise, or is neutral. For example, "A soccer game with multiple males playing" entails "Some men are playing a sport", while "A black race car starts up in front of a crowd of people" is contradicted by "A man is driving down a lonely road". The SNLI dataset contains 570k human-written English sentence pairs manually labeled for balanced classification.

4.2 Evaluation

Overview. Our goal is to compute a scalar score that represents the similarity between the self-attention weights and the dependency distributions. To this end, we compute the KL divergence between them and average across all dimensions and all tokens in the sequence. Averaging these values across a batch of examples gives us our final scalar score for the batch. For qualitative analysis, we only average across the tokens in the sequence, and use the per dimension KLD values to determine dimensions of interest.

KL divergence. Given multi-dimensional self-attention weights and the constructed dependency distributions, we apply KL divergence to each token in each dimension to measure how far the dependency distributions are from the attention distributions.

$$D_{KL}(P||Q) = \sum_x P(x) \log P(x) - \sum_x P(x) \log Q(x)$$

Note that by construction of our dependency distributions described in Section 3.2, the first term becomes 0, and the D_{KL} reduces to the cross entropy[4]. Computing this on a single example produces a tensor in $\mathbb{R}_{\geq 0}^{n \times d_e}$, where n is the length of the sequence and d_e is the number of dimensions.

Hypothesis. If DiSAN learns to encode dependency tree information, we should see $D_{KL}(P_{dep}||Q_{disan})$ decrease as training progresses, while $D_{KL}(P_{rand}||Q_{disan})$ should not decrease (and ideally increase, since it should be minimized when Q_{disan} is the uniform distribution). If the results of this quantitative analysis are positive, it provides reasonable evidence that the DiSAN self-attention weights encode some degree of dependency tree information.

4.3 Model settings

We trained a DiSAN model as described in [13], to which we refer the reader. We trained an additional model with smaller embedding and hidden sizes (50 instead of 300) using GloVe 6B 50d pretrained embeddings[**glove**], henceforth referred to as DiSAN-50; no further hyperparameter tuning was done.

5 Results

5.1 Quantitative Analysis

Figure 2 shows the KLD values for every epoch during the DiSAN and DiSAN-50 training process. We see that $D_{KL}(P_{dep}||Q_{disan})$ does in fact decrease as training progresses, and $D_{KL}(P_{rand}||Q_{disan})$ does increase.

The attentive reader will notice the discrepancy at epoch 0, before training. We investigated this discrepancy and found that there is an inherent bias in the true dependency trees towards parent tokens appearing prior to their respective child tokens in the sentence. We counted the number of tokens that have a parent that appears before itself in the sequence across all examples in our validation dataset, and divided that by the total number of tokens in the dataset. We performed the same calculation for the random tree baselines. After correcting for the bias incurred by having the special ROOT token,

[4]This is not true for general dependency distributions, only our special case where we have each child attend to its sole parent.

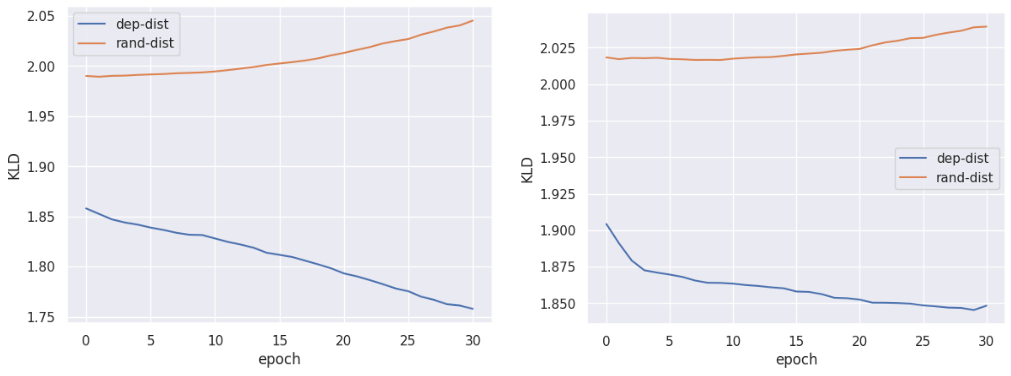


Figure 2: KLD per epoch during training. $d_e = 300$ (left) and $d_e = 50$ (right)
Blue is $D_{KL}(P_{dep}||Q_{disan})$, Orange is $D_{KL}(P_{rand}||Q_{disan})$.

our results showed that 55.36% of tokens in true dependency trees have parents that appear before them in their respective sequences, while this holds for only 49.80% in the random tree baselines.

Due to this inherent bias, when computing KLD on forward-masked self-attention there are more terms that contribute 0 to the final value, giving us a lower KLD by default. Backward-masked self-attention shows the opposite effect, where the KLD is higher than the random tree baseline at initialization before any training occurs. Plots that support this hypothesis are shown in Figure 3. Note that, since we still see a relative decrease/increase in the dependency/baseline KLD values, our previous conclusions still hold.

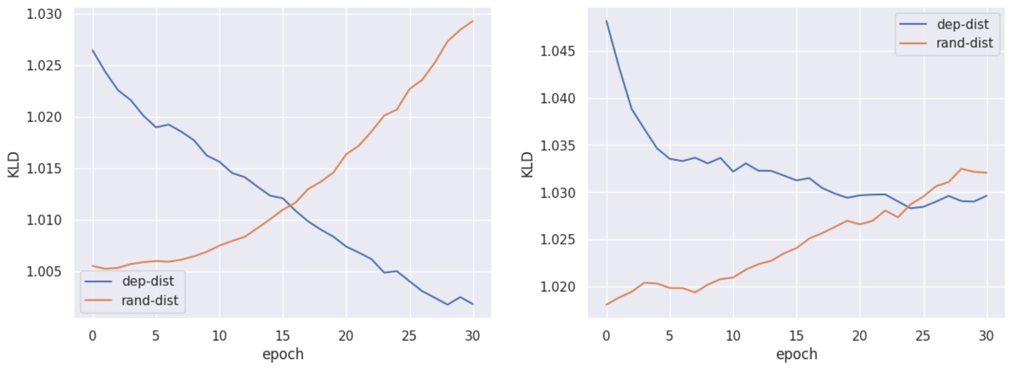


Figure 3: KLD for backward masks per epoch during training, $d_e = 300$ (left) and $d_e = 50$ (right)

5.2 Qualitative Analysis

In order to understand more concretely what the different dimensions of self-attention are learning, we plotted the attention weights for individual dimensions. We sorted the dimensions by lowest KLD for both the dependency distributions and the random tree baselines, and inspected the weights for those dimensions. Some examples are shown in Figure 4.

As expected, the dimensions of lowest KLD with respect to the random tree baselines were uniform distributions. The results for the dependency distributions were more interesting. We found that the dimensions of lowest KLD with the dependency distributions show very clear bias towards important keywords in the sentence; however, many of the lowest KLD dimensions all have their distributions focused on the same set of keywords, which is suspicious. Upon further inspection we found that, although the different dimensions focus on the same set of tokens overall, their individual distributions are actually quite different. From these results, we formulate two hypotheses: 1) The task (NLI classification) is likely too simple, and the model has no need to capture any strong linguistic

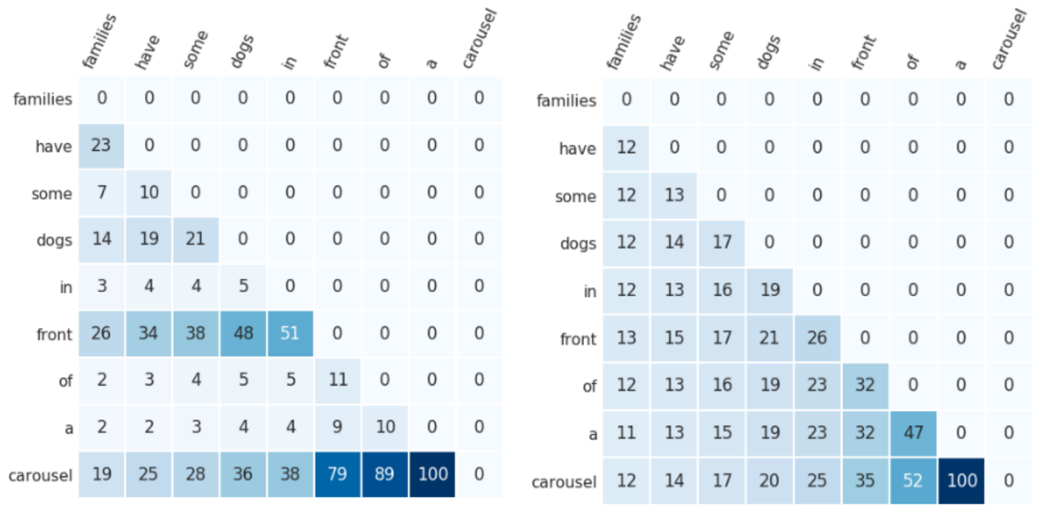


Figure 4: The dimension of lowest KLD of the multi-dimensional self-attention for dependency distributions (left) and random tree baselines (right). Annotated numbers are softmax percentages.

structure; and 2) the differences between the dimensions may suggest some entanglement between them.

Although the uniform distributions found from the random tree baselines is a nice sanity check, we question why such dimensions exist in the first place. Our primary hypothesis is that the model may have much more learning capacity than is required for the NLI classification task, and results in dimensions that are not used at all during inference. This is also supported by the results in the previous paragraph.

6 Conclusions

We studied the encoding of dependency tree structure in self-attention weights[5]. We proposed a general method for measuring the encoding of structure in self-attention distributions using KL divergence. Our results show that the DiSAN model does in fact encode some degree of dependency structure. We further analyze some qualitative examples of multi-dimensional self-attention and show that the learned self-attention distributions look mostly as expected, but also raise some questions that we would like to address in future work.

There is plenty of further work to be done on understanding what other structures are encoded in the self-attention weights. Using more complicated dependency distributions based on prior linguistic knowledge can show whether more intricate structures are encoded. For example, we could construct dependency distributions that encode pairwise distances based on distance in the dependency tree.

Our qualitative analysis suggests that perhaps the learning task, NLI classification, may not be complicated enough for the network to need to learn elaborate linguistic structures. We would like to explore multi-dimensional self-attention models in the context of a task like language modeling where more structure is required to perform the task well.

Finally, we would like to pursue work in the area of disentangling the multi-dimensional self-attention so that each dimension more concretely learns to encode very particular structures. Further extending this with stacked layers of self-attention may provide the model with the opportunity to encode progressively more abstract linguistic structures.

[5]**For graders**: I had personal goals for this project to improve my practical skills, e.g. PyTorch proficiency, and deepen my understanding of self-attention. As such, I spent a lot of time implementing the core pieces of DiSAN's self-attention mechanisms in PyTorch myself, as well as the infrastructure for running all of the experiments, which left a little less time for experimentation. Overall this was a very fun project though! :-)

References

[1] Reza Abbasi-Asl and Bin Yu. "Interpreting Convolutional Neural Networks Through Compression". In: *arXiv preprint arXiv:1711.02329* (2017).

[2] Yossi Adi et al. "Fine-grained analysis of sentence embeddings using auxiliary prediction tasks". In: *arXiv preprint arXiv:1608.04207* (2016).

[3] Sebastian Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[4] Yonatan Belinkov et al. "Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks". In: *arXiv preprint arXiv:1801.07772* (2018).

[5] Samuel R. Bowman et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2015.

[6] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[7] Reza Ghaeini, Xiaoli Z. Fern, and Prasad Tadepalli. "Interpreting Recurrent and Attention-Based Neural Models: a Case Study on Natural Language Inference". In: *CoRR* abs/1808.03894 (2018). arXiv: `1808.03894`. URL: `http://arxiv.org/abs/1808.03894`.

[8] John Hewitt and Christopher D. Manning. "A Structural Probe for Finding Syntax in Word Representations". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Minneapolis, USA: Association for Computational Linguistics, 2019.

[9] Mahnaz Koupaee and William Yang Wang. "Analyzing and Interpreting Convolutional Neural Networks in NLP". In: *arXiv preprint arXiv:1810.09312* (2018).

[10] Christopher D. Manning et al. "The Stanford CoreNLP Natural Language Processing Toolkit". In: *Association for Computational Linguistics (ACL) System Demonstrations*. 2014, pp. 55–60. URL: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[11] Matthew E Peters et al. "Deep contextualized word representations". In: *arXiv preprint arXiv:1802.05365* (2018).

[12] Matthew E Peters et al. "Dissecting contextual word embeddings: Architecture and representation". In: *arXiv preprint arXiv:1808.08949* (2018).

[13] Tao Shen et al. "Disan: Directional self-attention network for rnn/cnn-free language understanding". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[14] Quanshi Zhang et al. "Interpreting cnns via decision trees". In: *arXiv preprint arXiv:1802.00121* (2018).