An Information-Theoretic Parameter-Free Bayesian Framework for Probing Labeled Dependency Trees from Attention Scores

Anonymous ACL submission

Abstract

Figuring out how neural language models comprehend syntax acts as a key to revealing how they understand languages. We systematically analyzed methods of extracting syntax from models, namely *probing*, and found five limitations yet widely exist in previous probing practice. We proposed a method that can directly extract labeled dependency trees from attention scores without training any network, while being able to calculate the mutual information (MI) in a mathematical-rigorous way. Compared with previous approaches, our method has a much *simpler* model, while being able to probe more *complex* dependency trees, providing much more fine-grained information about model *explanation* at the same time. We demonstrated our method's effectiveness by systematically comparing it with a great many competitive baselines, and gained informative conclusions, shedding light on our method's explanation potential. Our code is anonymously released at https://anonymous.4open.science/r/IPBP-96BC.

1 Introduction

Recent advancements in Large Language Models (LLMs) have left the world with deep impressions. This process is accompanied by confusion, since LLMs are largely black-box and usually trained on simple next-token-prediction LM tasks. While recent interpretability community assign great importance on mechanistic (circuit-based) model explaining (Elhage et al., 2021; Wang et al., 2022; Ferrando and Voita, 2024), traditional *probing* methods, which aim at extracting syntax structures from model states, are still somehow worth working on for two reasons: 1. They can provide dataset-wide conclusions, while most circuit-based methods tend to be sample-wise (except for Elhage et al., (2021), which is a model-level analysis). 2. Syntax structures are among the most complicated and all-round concept about languages, and are essential to humans' language comprehension, verified by recent brain studies (Lopopolo et al., 2020; Dotan et al., 2022; Fallon et al., 2024).

A common practice of probing is to train a supervised classifier network on top of model states (Hewitt and Manning, 2019; Pimentel et al., 2020; Müller-Eberstein et al., 2022) to predict dependency syntax trees, or directly take some model states as evidence for syntax (Htut et al., 2019). Despite the insights they gave, it is obvious that previous probing methods are **explaining by unexplainibility**: Most of them are introducing external trainable networks to extract syntax, ranging from simple linear mappings (Liu et al., 2019) to deep MLPs (Hewitt and Liang, 2019; Voita and Titov, 2020; Pimentel et al., 2020a) or pseudo attention heads (Pimentel et al., 2022). This is causing a trade-off: Linear mappings are simple and explainable, but have limited expressivity. Deeper networks can fit any co-relationships, but a deep probing network is unexplainable itself, so it's natural to raise the doubt on whether the extracted syntax structures really come from the probed LM, or just the strong probes have learned to unconditionally predict them. Moreover, since modern LLMs have larger hidden dimensions compared with pretrained models, the trainable networks have to be even larger to fit in the dimensionality, inevitably making them more unexplainable.

If we dive deeper, we might find clues about this bitter tradeoff: previous methods are putting their attention mainly on contextualized hidden states. Since vector-based hidden states have completely different modalities compared with dependency trees, a trainable mapping network is necessary.

Using hidden states is also a primary cause for the aforementioned concern of *did the probe learn the task*: Contextualized embeddings embed abundant semantics, so even though the probed LM knows nothing about syntactics, it's still possible that the deep probing model learns it. As an example, if we see two words, *eat* and *breakfast*, even without any context, there're still good reasons for us to believe that *breakfast* acts as the object of the verb *eat*. This is exactly the case of this concern.

If hidden states are not yet good enough, what's the better choice? Maybe we should *put attention on attention*. Attention is the only component that involves inter-token relationships (MLP and add/norm are applied token-wise), while dependency syntactics is exactly inter-word relationships. Moreover, attention maps are topologically consistent with syntax trees: Attention scores are matrices, while dependency trees can also be described as adjacency matrices (See Section 3.1 for details).

Unfortunately, despite those nice consistencies, to our best knowledge, there're only few methods focusing on probing attention (Clark et al., 2019; Htut et al., 2019; Vig and Belinkov, 2019; Ravishankar et al., 2021), only being able to extract inferior or even incomplete dependency trees. There seems to be a contradiction. Why is it? Our opinion is that, due to those consistencies, researchers are **over-trusting attention scores**, which is another limitation. Since attention scores are softmax-normalized, constituting a probability distribution across tokens, they tend to directly use attention scores as probabilities of a dependency relationship between two words. However, attention scores definitely do not only have this single functionality of syntax, so filtering out highly syntactical attention heads together with transformations on attention scores is necessary.

Based on our analysis, we proposed our method of **I**nformation-theoretic **P**arameter-free **B**ayesian **P**robing (IPBP): Instead of training supervised networks on hidden states, we chose to directly estimate the probabilistic distributions between attention scores and dependencies. Since it's parameter-free and attention-based, our method denies the possibilities of *letting complex probes learn the task*, as was mentioned before. With those distributions, we're able to estimate mutual information (MI) in a mathematically rigorous way, obtaining a good metric for each head's individual importance for each dependency label. We further designed a novel decoding algorithm incorporating the estimated MI and Bayesian posteriors, being able to efficiently reconstruct *labeled* dependency trees, while preventing us from dropping into the trap of directly using attention scores as dependency probabilities. We systematically compared with a series of strong baselines and achieved SOTA head importance estimation and tree-constructing performance. We further derived informative conclusions on the estimated MI and distributions, including one obsessing probing researchers for a long time: *are tree layers reflected by model layers*? (A similar question once discussed by Tenny et al., (2019)). In a word, our method is addressing the two limitations in an elegant way, while offering vast possibilities for the upcoming conclusion-intensive research thanks to its fine-grained MI and probabilities functions.

2 Related Work

Just after the birth of deep contextualize embeddings (Peters et al., 2018) and transformer-based pre-trained models (Devlin et al., 2019), researchers have started to investigate whether or not linguistics properties are embedded in these models (Conneau et al., 2018; Liu et al., 2019; Tenny et al., 2019; Hewitt and Manning, 2019). Then arguments began in this area. The frontline of these arguments is about *what probing model we can use to prevent it from learning the task itself*. While early practices and preliminary methods suggested strictly-linear probes (Alain and Bengio, 2017; Hewitt and Manning, 2019; Liu et al., 2019), Hewitt and Liang, (2019) proposed control tasks that penalizes models being ability to learn the task itself, and had attempts on several Deep MLPs. Furthermore, Pimentel et al., (2020b) admitted this trade-off and took probing as an accuracy-complexity two-goal optimizing problem, and most radically, Pimentel et al. (2020a) insisted that probes should be as deep and complex as possible since they used them as estimations of $\mathcal{V}$-Information (Xu et al., 2020). Apart from disputes, there are also alternative theories proposed by the researchers, like the code-description-length theory by Voita and Titov, (2020) and the architectural bottleneck principle by Pimentel et al., (2022). These theories can be seen as patches under the supervised probing context since they're also addressing the complexity vs. accuracy tradeoff.

Apart from supervised probes, there're also naïve parameter-free probes, mainly based on extracting dependency trees (or partial dependency arcs) from attention scores (Clark et al. 2019, Vig and Belinkov, 2019; Ravishankar et al., 2021), yielding not yet good enough probing performances. If we take a broader view, we'll also find parameter-free explaination methods for more general-purpose concepts in deep learning research (Mu and Andreas, 2020; Antverg and Belinkov, 2021). Together with some supervised methods (Radford et al., 2019; Lakretz et al., 2019; Dalvi et al., 2019), these methods, also called neuron analysis methods, were systematically evaluated by a recent work (Fan et al., 2024). In section 4, we'll systematically compare our methods with the principles of these strong baselines.

In a broader point of view, there are also methods extracting syntax rules from sentences without annotations, ranging from probability-based methods (Klein and Manning, 2003) to neural-network approaches (Shen et al., 2018). While achieving different tasks, they are essentially another side of a coin within the field of computational syntactics.

3 IPBP Methodology

To foster understanding, we'll first break our method into key points in the first section, and then introduce the details.

3.1 Key Aspects Analysis

Given a sentence $X = x_1 x_2 \dots x_n$ and an arbitrary token pair (x_i, x_j), we define $l^{[i][j]}$ as the variable (also an element in the dependency tree adjacency matrix) for which kind of dependency exists from x_i to x_j. $l^{[i][j]}$ can be a specific dependency type like `nsubj`, or ϕ if there's no dependency. If the sentence is feed into a transformer LM, there will be a series of attention scores matrices. An element in a specific matrix is in the form of $a_{b,h}^{[i][j]}$, which stands for the attention score of the h-th attention head from the b-th transformer block, from the i-th token to the j-th token. If we gather the observations $l^{[i][j]}$ and $a_{b,h}^{[i][j]}$ for each token pair in the dataset, we'll get two co-occurring dataset-wide variables, L and $A_{b,h}$, standing for the dependency and head (b, h)'s score at *any* token pair. Therefore, the goal of our probing can be divided into two:

- **MI Estimation**: estimating mutual information (MI) between L and $A_{b,h}$ for $\forall b, h$: $\mathrm{MI}(L; A_{b,h})$.
- **Tree Reconstruction**: A method of deriving a full dependency tree based on attention scores $A_{b,h}$

Specifically, since L is a discrete variable and $A_{b,h}$ is continuous, the joint distribution is a mixture distribution, the formula of MI is as follows, slightly different from classical definition:

$$\mathrm{MI}(L; A_{b,h}) = \sum_{l \in \mathcal{L} \cup \{\phi\}} \int f(l, a) \log \frac{f(l,a)}{P(l) f(a)} \mathrm{d}a \qquad (1)$$

Where $\mathcal{L}$ stands for the set of all dependency relationships {`nsubj`, `dobj`, …} and $f(l, a)$, $P(l)$, $f(a)$ is short for the density value of joint distribution $f(L, A_{b,h})$ at $L = l, A_{b,h} = a$, density of marginal distribution $f(A_{b,h})$ at $A_{b,h} = a$, and scalar probability $P(L = l)$.

Moreover, the second goal can be regarded as a Bayesian inference process taking $A_{b,h}$ as evidence and L as hypothesis. The posterior distributions ($f(L = l | A_{b,h} = a)$) are required for tree reconstruction. Therefore, the key to achieving these two goals are those probabilistic distributions.

In the following sections we'll dive deep into how we can infer these distributions from the dataset.

3.2 Getting the Distributions

Initialization. Assume there's a dataset $\mathcal{D}$ consisting of a series of <*sentence, dependency tree*> pairs, and a model with $\mathscr{b}$ blocks and $\mathscr{h}$ attention heads within each block. We first initialize a series of attention score sets $\mathcal{A}_{b,h;l}$ where $b \in \{1 \dots \mathscr{b}\}$, $h \in \{1 \dots \mathscr{h}\}$ and $L \in \mathcal{L} \cup \{\phi\}$. $\mathcal{A}_{b,h;l}$ means all possible attention scores of attention head b, h between token pairs having dependency l.

Gathering attention scores. We iterate over the dataset and for a specific sentence $X \in \mathcal{D}$, we feed $X = x_1 \dots x_n$ into the model, and for any token pair $\langle x_i, x_j \rangle$ $(i, j \in \{1 \dots n\})$, the dataset provide its dependency relationship $l^{[i][j]}$ and the model provides the attention scores $a_{b,h}^{[i][j]}$ $(\forall b, h)$. We add $a_{b,h}^{[i][j]}$ to the corresponding attention score set $\mathcal{A}_{b,h;l^{[i][j]}}$. After iteration, all attention score sets will have all possible attention scores in the dataset.

Getting distributions. After gathering attention scores, we'll estimate those required probabilities.

The most intuitive one might be $P(L=l)$, since we can take the empirical probability $\hat{P}(L=l) = \frac{|\mathcal{A}_{b,h;l}|}{\sum_{l' \in \mathcal{L} \cup \{\phi\}} |\mathcal{A}_{b,h;l'}|}$ (also number proportions) on the dataset as the approximate value. The tricky ones are the continuous probabilities. Since we already have abundant attention score samples, we'll use Kernel Density Estimation (KDE) to estimate the continuous ones. Specifically, for every possible $\mathcal{A}_{b,h;l}$, we regard it as the observation of the attention variable $A_{b,h}$ under the circumstance of $L=l$. The samples in $\mathcal{A}_{b,h;l}$ follow the conditional density of $f(A_{b,h}|L=l)$. We use the Gaussian kernel and take a specific bandwidth B (See Appendix A). Therefore, the kernel density $\hat{f}(A_{b,h}|L=l)$ can be estimated as:

$$\frac{1}{|\mathcal{A}_{b,h;l}| \cdot B} \sum_{i=1}^{|\mathcal{A}_{b,h;l}|} \frac{1}{\sqrt{2\pi \cdot \sigma_{\mathcal{A}_{b,h;l}}}} \exp\left(-\frac{x_0 - \mathcal{A}_{b,h;l}^{(i)}}{B}\right)^2 \tag{2},$$

where $\sigma_{\mathcal{A}_{b,h;l}}$ is the standard deviation of $\mathcal{A}_{b,h;l}$, and $\mathcal{A}_{b,h;l}^{(i)}$ means the i-th value of $\mathcal{A}_{b,h;l}$.

Again, if we take the view of Bayesian inference, with $A_{b,h}$ as evidence and L as hypothesis, then the estimated $\hat{f}(A_{b,h}|L)$ is essentially the *likelihood* density. Applying the Bayesian theorem, we'll get the following equation:

$$f(L|A_{b,h}) = \frac{f(A_{b,h}|L)P(L)}{f(A_{b,h})} = \frac{f(A_{b,h}, L)}{f(A_{b,h})} \tag{3}$$

This gives us inspirations: given the likelihood $\hat{f}(A_{b,h}|L)$ and the prior $\hat{P}(L)$, we can multiply $\hat{f}(A_{b,h}|L)$ with $\hat{P}(L)$ to get the joint densities $\hat{f}(A_{b,h}, L)$. Moreover, by summing over all possible Ls, we can estimate marginal density $\hat{f}(A_{b,h})$, and then the posterior probability $\hat{f}(L|A_{b,h})$ is computable. By now, all these required probabilities mentioned in Section 3.1 are all set.

3.3 Estimating MI

With all these distributions, we're able to proceed to our two main goals: MI estimation and Tree Reconstruction. However, if we reexamine the MI formulation in Equation 1, we'll find that the $\mathrm{MI}(L; A_{b,h})$ in Equation 1 measures how much shared information head $\langle b,h \rangle$ has about *every* possible dependencies. However, a more probable case is that head $\langle b,h \rangle$ is only responsible for *certain* dependencies. This kind of *specialist head* is also the assumption of preliminary attention-analysis work like (Htut et al., 2019). Even though such versatile heads exist, an MI corresponding to all dependencies is still too coarse-grained. Therefore, we should tweak Equation 1 to make the MI formulation fit this *specialist* assumption. The new formulation is as follows:

$$\mathrm{MI}_{\text{binary}}(l; A_{b,h}) = \int f(l,a) \log \frac{f(l,a)}{P(l)f(a)} \mathrm{d}a + \int f(\neg l, a) \log \frac{f(\neg l, a)}{P(\neg l) f(a)} \mathrm{d}a \tag{4}$$

In that equation, $f(\neg l, a)$ is short for the density value of $f(\neg l, A_{b,h})$ at $A_{b,h} = a$, where $f(\neg l, A_{b,h})$ stands for the joint density between all dependencies other than l and $A_{b,h}$. In practice, it can be gained by marginalizing $\hat{f}(A_{b,h}, L)$ over all possible Ls except for l. $P(\neg l)$ stands for the possibility of dependencies other than l, which can be estimated using $1 - \hat{P}(l)$.

3.4 Getting Highly Syntactical Heads

By now, having posterior distributions $\hat{f}(L|A_{b,h})$ and $\mathrm{MI}_{\text{binary}}$ feasible for estimating the independent importance of each dependency type, it's possible to reconstruct the trees. The basic idea of our tree reconstruction is: for every dependency relationship l, we pick out attention heads highly responsible for l, constituting the head set $\mathcal{H}_l$. We then infer the possibilities of dependency arcs of l based on the posteriors of heads from $\mathcal{H}_l$, and use $\mathrm{MI}_{\text{binary}}$ to balance between posteriors of each head from $\mathcal{H}_l$, forming the overall possibility for a dependency arc with relation l. Finally, we use a decoding algorithm to build the dependency tree based on these overall possibilities.

To form $\mathcal{H}_l$, it's natural to set a threshold on $\mathrm{MI}_{\text{binary}}(l; A_{b,h})$. However different dependency relationships might have different $\mathrm{MI}_{\text{binary}}$ magnitudes, an adaptive threshold conditioning on specific relations is necessary. Remind the fact that mutual information is upper-bounded by the individual entropies of each random variable, in our case, $H(\mathbf{1}_{\{l\}}(L))$ and $H(A_{b,h})$ (where $\mathbf{1}_{\{l\}}(L)$ denotes L equals to l or not). Since $A_{b,h}$ is continuous, and the entropy analogs of continuous variables (variational entropies) are known as inferior analogs, possibly non-positive, making it unable to act as an upper bound, we choose to estimate $H(\mathbf{1}_{\{l\}}(L))$ as:

$$\hat{H}\left(\mathbf{1}_{\{l\}}(L)\right) = \hat{P}(L)\log\hat{P}(L) + \hat{P}(\neg L)\log\hat{P}(\neg L) \quad (5)$$

If MI is divided by the entropy, the resulting proportions $\frac{\mathrm{MI}_{\mathrm{binary}}(l;A_{b,h})}{\hat{H}(\mathbf{1}_{\{l\}}(L))}$ will be in a uniform $[0,1]$ scale, which can act as the *adaptive threshold*.

3.5 Tree-Reconstruction Algorithm

After getting $\mathcal{H}_l$ s, another problem occurs: As mentioned before, previous probing practices mainly aim at building unlabeled trees. Even those supervised dependency parsing methods (Dozat and Manning, 2017; Tian et al., 2022) rely on separate networks for predicting arc labels. Therefore, these methods are operating on a simple probability space with only probabilities on *the existence of dependency arcs*. What's more, in their methods, there's only one network responsible for predicting probabilities, our method, on the other hand, has a bunch of posterior probabilities. Therefore, it's necessary that we design a decoding algorithm that not only *balances each posterior* but also constitutes *a homogeneous probability space*.

We first make an assumption that the overall possibility of dependency arcs is independently conditioned on each head in $\mathcal{H}_l$ (otherwise the problem might be too complex). Theoretically, to balance each posterior is to treat the prediction of dependency arcs as a *voting problem*: for dependency l, each head $\langle b_i, h_i\rangle \in \mathcal{H}_l$ can be seen as a participant with weight $\mathrm{MI}_{\mathrm{binary}}\left(l; A_{b_i,h_j}\right)$. The probability of a dependency arc of l can be seen as the probability of a series of heads with total weights larger than a proportion (like half or two-thirds) voting *the arc belongs to* l. However, due to the non-discrete weights, the problem cannot be efficiently dynamically programmed, resulting in a search space of $O(2^{|\mathcal{H}_l|})$, which will be rather inefficient during inference. Instead, we relax this voting problem to an easy-computing yet rational form: We take the geometric mean of the posteriors. Specifically, let $\mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l)$ be the geometrically-averaged probability of an arc of l between tokens x_i and x_j conditioned on heads in $\mathcal{H}_l$. In logarithm space, the geometric mean is:

$$\log \mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l) = \frac{\sum_{\langle b_k,h_k\rangle} \mathrm{MI}_{\mathrm{binary}}(l;A_{b_k,h_k})\cdot \hat{f}(L=l;A_{b_k,h_k})}{\sum_{\langle b_m,h_m\rangle\in\mathcal{H}_l} \mathrm{MI}_{\mathrm{binary}}(l;A_{b_m,h_m})} \quad (6)$$

This is approximately equivalent to the Logarithmic Opinion Pooling (Heskes, 1997) technique widely adopted in Bayesian inference, thus acting as a reasonable approximation when the number of experts (in our case, heads in $\mathcal{H}_l$) is relatively large. However, the problem of Logarithmic Pooling is that, if we sum over all probabilities of each possible dependency relationship (in our $\mathrm{MI}_{\mathrm{binary}}$ case, l and $\neg l$), it is not guaranteed to be 1, recalling the second problem of *homogeneous probability space*. To resolve this, we build a larger multivariate probability space of $\{0,1\}^{|\mathcal{L}|+1}$. We take the voting process of the dependency between x_i and x_j as $|\mathcal{L}|+1$ independent votes, the ℓ-th ballot votes for the existence of the ℓ-th dependency from $|\mathcal{L}|$, using the $\mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l)$ in Equation 6 as the probability of *existence*, and $1-\mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l)$ as the probability of *non-existence*. The overall probability $P(x_i,x_j;l)$, meaning the probability of an arc of l between tokens x_i and x_j conditioned on *all* highly responsible heads $\mathcal{H}_1 \cup \ldots \cup \mathcal{H}_{|\mathcal{L}|} \cup \mathcal{H}_\phi$, is calculated as follows:

$$P(x_i,x_j;l) =$$

$$\mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l)\cdot \prod_{l'\in|\mathcal{L}|+\{\phi\}-\{l\}}\left[1-\mathrm{GP}_{\mathcal{H}_l}(x_i,x_j;l)\right] \quad (7)$$

While the probability of no arc between x_i and x_j is $1-\sum_{l\in|\mathcal{L}|}P(x_i,x_j;l)$, thus resulting in a valid probability space. By now, the two problems introduced by *multi-head* and *multi-label* are both solved. We're just one step towards building the tree, that is, the decoding algorithm utilizing the overall probabilities. Specifically, following previous supervised dependency parsing methods, we're using the Eisner dynamic programming algorithm (Eisner, 1996) as the decoding algorithm. Readers might refer to Appendix A for implementation details of our methods, like hyperparameters and our GPU-optimized KDE and integral methods.

3.6 The Novelty of IPBP

Since MI estimation is a small hot topic in statistics, in case of re-inventing wheels, we've done research on related methods. We found two methods sharing (minor) princix`ples with our method: The first one (Moon et al., 1995) is a method estimating MI between two observations within a time series using KDE. They're doing three individual KDEs, with one multivariate one. While it's a known issue that KDE quickly becomes inferior when variables become more than one, known as the *dimensionality curse*, their method is inevitably introducing errors (and also unapplicable to our attention-dependency mixed-joint distribution setting). We, instead, dexterously circumvented the curse and made the least number of estimations possible (limited to 1) by exploiting mixed-joint distribution and Bayesian theorems.

Another one is also focusing on mixed joint distribution (Gao et al., 2017). However, they use a kNN-like algorithm to estimate point-wise mutual information (PMI) and average it over the dataset. Their method didn't provide any valid probability distributions, thus offering no possibility of tree reconstruction, and also providing less chance for dataset-level or visualization-based explanations.

4 Experiments

4.1 Baselines

In this section, we're going to systematically compare our method with a series of probing as well as neuron analysis baselines. Corresponding to the two sub-tasks introduced in Section 3.1, we first introduce a series of head-selection baselines, where we replace the estimated MI with other criteria, and keep the tree-construction algorithm unchanged. We'll also compare the tree-construction algorithm with common practices of previous attention-based methods. This is better for illustrating the individual contributions of each corresponding submodule.

For head-selection baselines, we'll start from several strong neuron analysis methods evaluated by a recent paper (Fan et al., 2024):

Probeless (Antverg and Belinkov, 2021): This is a parameter-free method, which gets the correlation scores by calculating mean values with respect to different concepts alongside the dataset. In our situation, we use the following instead of $\mathrm{MI}_{\text{binary}}$:

$$\mathrm{PL}\big(l; A_{b,h}\big) = \sum_{l' \in \mathcal{L}+\{\phi\}-\{l\}} \big|\bar{\mathcal{A}}_{b,h;l} - \bar{\mathcal{A}}_{b,h;l'}\big| \tag{8}$$

Where $\bar{\mathcal{A}}_{\cdot,\cdot;\cdot}$ denotes the mean value of a specific attention score set. Note that despite its simplicity, this method is evaluated as the method that is most consistent with others by Fan et al., 2024, thus most robust.

IoU (Mu and Andreas, 2020): This method uses Jaccard Similarity as a correlation criterion. In our implementation, we use the following form:

$$\mathrm{IoU}\big(l; A_{b,h}\big) = \frac{\big|\mathcal{A}_{b,h,l} \cap [\tau, +\infty)\big|}{\big|\mathcal{A}_{b,h,l}\big| + \sum_{l' \in \mathcal{L}+\{\phi\}-\{l\}} \big|\mathcal{A}_{b,h,l'} \cap [\tau, +\infty)\big|} \tag{9}$$

Where τ is a threshold serving as selecting a salient score. Following the original authors, we set it to the top 99.5% value among values in $\mathcal{A}_{b,h,l}$.

The Linear Feedforward Family: This method refers to a series of methods performing correlation ranking by training a supervised linear network W_θ. Specifically, the equation below gives a uniform formulation of these methods:

$$W_\theta = \underset{W_\theta \in \Theta}{\operatorname{argmin}} \Big[\sum_{X \in \mathcal{D}} \sum_{x_i, x_j \in X \times X} \log P_\theta \Big(l = l^{[i][j]} \Big| a^{[i][j]}_{1,1\ldots \mathscr{b},\mathscr{h}}\Big) + \\ + \lambda_1 \|\theta\|_1 + \lambda_2 \|\theta\|_2\Big] \tag{10}$$

Where W_θ is a matrix of shape $\mathscr{b}\mathscr{h} \times (|\mathcal{L}| + 1)$, and $a^{[i][j]}_{1,1\ldots \mathscr{b},\mathscr{h}}$ denotes the concatenation of attention scores between x_i and x_j for all attention heads, and $P_\theta\Big(l^{[i][j]} \Big| a^{[i][j]}_{1,1\ldots \mathscr{b},\mathscr{h}}\Big)$ stands for the probability of the ground-truth label estimated by the network. When $\lambda_1 = 1, \lambda_2 = 0$, this equation becomes Lasso (Radford et al., 2019), when $\lambda_1 = 0, \lambda_2 = 1$, it becomes Ridge (Lakretz et al., 2019), and $\lambda_1, \lambda_2 = 1$ corresponds to ElasticNet (Dalvi et al., 2019). We use ElasticNet as a representative. After gaining the trained W_θ, we use the weight entry mapping attention score of head $\langle b, h \rangle$, to the probability of relation l as the correlation value, $\mathrm{LFF}(l; b, h)$.

$\mathcal{V}$-Information: Xu et al., 2020 proposed to use a trainable network as an approximation of conditional probabilities, and use the mean logarithm probabilities as approximations of conditional entropies based on the law of large number. This is the state-of-the-art entropy estimation algorithm, used by previous methods also taking information-theoretic perspectives (Pimentel et al., 2020a, Pimentel et al., 2022). Specifically, in our case, we use $\max H_{\mathcal{V}}(l|A_{\cdot,\cdot}) - H_{\mathcal{V}}(l|A_{b,h})$ as replacement of $MI_{\text{binary}}(l; A_{b,h})$, and as the equation shows:

$$H_{\mathcal{V}}\big(l \big| A_{b,h}\big) = \\ \frac{1}{\sum_{X \in \mathcal{D}} |X|^2} \sum_{X \in \mathcal{D}} \sum_{\langle x_i, x_j \rangle \in X} \Big[\mathbf{1}_{\{l\}}\big(L^{[i][j]}\big) \log \mathrm{MLP}_{b,h;l}\Big(a^{[i][j]}_{b,h}\Big) + \\ \mathbf{1}_{\mathcal{L}+\{\phi\}-\{l\}}\big(L^{[i][j]}\big) \log\Big(1 - \mathrm{MLP}_{b,h;l}\Big(a^{[i][j]}_{b,h,}\Big)\Big)\Big] \tag{11}$$

Where $\mathrm{MLP}_{b,h;l}(\cdot)$ are deep MLPs individually trained using head $\langle b, h \rangle$ to predict label l.

Under each head-selection setting, for fair comparison, we set a limit of the total number of syntactical heads $\sum_{l \in \mathcal{L} \cup \{\phi\}} |\mathcal{H}_l|$ of 2000.

For the tree-construction alternative, we use **Raw attention score**: Under this setting, we're still using the estimated MI as head importance criteria, while for a specific head $\langle b, h \rangle \in \mathcal{H}_l$, we use the attention score $a^{[\cdot][\cdot]}_{b,h}$ instead of the posterior $\hat{f}(A_{b,h}, L)$ in the reconstruction algorithm. This simple intuitive is the common underlying principle of previous methods focusing on attention

(Clark et al., 2019; Vig and Belinkov, 2019; Ravishankar et al., 2021). We found that due to the absence of our estimated posteriors, if $\sum_{l\in\mathcal{L}\cup\{\phi\}}|\mathcal{H}_l|$ reaches 2000, the scores of all heads will be rather noisy. Therefore, we choose to select top-k heads based on MI for each label. We did a grid search and found the top-8 settings have ideal performance.

4.2 Model, Dataset and Metrics

We're using `open_llama_7b` [1] as our probed model. `open_llama_7b` is a decoder-based LLM consisting of 32 layers and 32 attention heads within each layer. Compared with pre-trained language models like BERT (Devlin et al., 2019), `open_llama_7b` might consist of attention heads with rather varied functionalities, offering more insights under the contemporary LLM research context.

Specifically, `open_llama_7b` is a decoder-based model having triangular-masked attention scores. In implementation, we cache the Key Values of each attention head and use them to re-calculate the unmasked attention scores. While our reconstruction is inevitably introducing "useless" attention scores, we think that it is still necessary for two reasons: 1. Making compromises to the decoder structure will hinder our method from applying to non-decoder models (Chung et al., 2024; Zeng et al., 2024), thus less universal. 2. As sentences become longer, the softmax-normalized scores will be *diluted*. This is more serious for triangular attention since it has rows of varying lengths. While softmax is not bijective, using cached QK to reconstruct the unnormalized scores is inevitable.

Following previous supervised dependency parsing methods (Tian et al., 2022), we use Universal Dependencies (UD) 2.9 (Zeman et al., 2021), as dataset, with 39832 sentences in the training set and 1700 sentences in the validation set. UD 2.9 is an English treebank covering texts from multiple sources like literature, news articles, spoken languages, *etc.*, with diverse morphological and grammatical features. We also use labeled attachment scores (LAS), and unlabeled attachment scores (UAS) as metrics.

4.3 IPBP Structural Alternatives

Apart from comparing with previous methods, we're also curious about our model's designs. Therefore, we propose two alternative structures:

Positive MI: We noticed that the attention score samples exhibit a long-tail characteristic: most samples come from $\mathcal{A}_\phi$, since most pairs of words don't have dependency arc in between. $\mathcal{A}_\phi$ might be noisy, consisting of various non-syntactic inter-token relationships, and MI estimations based on samples in $\mathcal{A}_\phi$ might be affected by this long tail noisy distribution. Other score sets $\mathcal{A}_1, \ldots \mathcal{A}_{|\mathcal{L}|}$ are having approximately the same magnitudes and their corresponding token pairs are guaranteed to have any dependency relationship. Therefore, we also calculated a more syntactical MI, namely $\mathrm{MI}_{\mathrm{pos}}$, with the following formulation:

$$\mathrm{MI}_{\mathrm{pos}}(L; A_{b,h}) = \sum_{l\in\mathcal{L}} \int f_{\mathrm{pos}}(l,a)\log\frac{f_{\mathrm{pos}}(l,a)}{P_{\mathrm{pos}}(l)f_{\mathrm{pos}}(a)}\mathrm{d}a \tag{12}$$

In that equation, $P_{\mathrm{pos}}(\cdot)$, $f_{\mathrm{pos}}(\cdot,\cdot)$ actually stand for conditional possibilities when $l \neq \phi$, estimated by $\hat{P}_{\mathrm{pos}}(L=l) = \frac{\mathcal{A}_{b,h;l}}{\sum_{l'\in\mathcal{L}}\mathcal{A}_{b,h;l'}}$ and $\hat{f}_{\mathrm{pos}}(L, A_{b,h}) = \hat{f}(A_{b,h}|L)\hat{P}_{\mathrm{pos}}(L)$. During implementation, we'll use a balance factor α and calculate the mixed MI $\mathrm{MI}_{\mathrm{mix}}(\cdot;\cdot) = \alpha\mathrm{MI}_{\mathrm{binary}}(\cdot;\cdot) + (1-\alpha)\mathrm{MI}_{\mathrm{pos}}(\cdot;\cdot)$

Arc First: Unlike previous methods, we're directly obtaining labeled dependency trees, bypassing the process of dependency arc predicting. We're curious about whether it's a good choice. Under this setting, instead of estimating $\hat{f}(A_{b,h}|L=l)$, we'll directly estimate the unlabeled likelihoods $\hat{f}(A_{b,h}|L\in\mathcal{L})$ and $\hat{f}(A_{b,h}|L=\phi)$, and calculate the corresponding multivariate probabilities together with corresponding MI values. We'll compare UAS to check the quality of reconstructed unlabeled trees.

Transposed: Sometimes, we're unsure whether the attended token acts as a dependency head, or a dependant. So we'll let $l^{[i][j]}$ correspond to $a_{b,h}^{[j][i]}$, and repeat the whole IPBP process in this setting.

4.4 Result and Analysis

Results are shown in Table 1. We can see that our method is overperforming all competitive baselines, including the state-of-the-art conditional entropy

[1] https://github.com/openlm-research/open_llama

Method	UAS	LAS
Probeless	34.8	20.9
IoU	38.3	26.6
ElasticNet	41.9	31.3
V-Information	41.3	20.9
Raw Score	32.3	16.6
IPBP	49.1	30.6
IPBP (transposed)	42.6	28.0
IPBP + $\mathrm{MI}_{\mathrm{pos}}$	**49.9**	**34.8**
IPBP (arc only)	36.5	N/A

Table 1: Results of our IPBP and different baselines.

estimation method, $\mathcal{V}$-Information, which shares principles with our method while requiring a much larger computational budget. In fact, our implemented $\mathcal{V}$-Information MLP is optimized using several tricks (see Appendix B), while during its training, we still find that the trained MLPs are relatively good at detecting arcs while having poor performances on labeling. This aligns with its low LAS in tree-construction results. An insight can be drawn that supervised methods may still fall behind statistical ones, especially when the data is long-tailed or low-dimensional. Moreover, even though the head-selection settings for the raw-score method is specifically tuned, and the method is already selecting attention heads based on our estimated MI, it still has a great performance gap with our posterior-based method, further justifying the necessity of our posterior-based algorithm.

For structure alternatives, we notice that incorporating $\mathrm{MI}_{\mathrm{pos}}$ will give performance benefits, shedding light on potential improvements to our methods. The transposed setting will still capture a relatively smaller portion of dependencies. Last but not least, by comparing with our arc-based baseline, we'll find we're actually at a triangular balance, we probed for more *accurate*, also *labeled* trees, while choosing a more *straightforward* method, with no need for individual arc probing.

4.5 Further Analysis

Like previous probing methods, we'll also do fine-grained analysis of our reconstructed trees and estimated MI values. Instead of listing up MI values and doing trivial analyses, we decide to provide two intriguing and informative conclusions, giving inspirations to upcoming research.

The first conclusion is that **decoder models adaptively capture look-back/ahead dependencies**: Since the masked decoder attention can only *look back*, it's reasonable that dependencies pointing to front words can be well captured. What makes it more intriguing is that dependencies *looking ahead* might also be captured in a look-back manner. We draw this conclusion by comparing the top-10 most well-reconstructed labels between original IPBP and the transposed alternative. We find that there're more look-ahead dependencies (5 of 10) under the transposed setting compared with the original setting (3 of 10), with further statistical tests supporting it (See Appendix C).

The second conclusion is that **model layers correspond to tree layers** to some extent: lower layers are for local/phrasal dependencies, while higher layers are for global/sentence-wide dependencies. The corresponds to intuitions but was never systematically justified before due to the lack of MI-like criteria and overfocusing on unlabeled trees. Thanks to the fine-grained MI, similar to the "Center-of-Gravity" introduced by Tenny et al., (2019), we can calculate the MI-weighted layer indices for each label, where smaller weighted indices indicate dependencies having more lower-layer heads responsible for it. Among these top-10 labels, we calculated the Pearson correlation coefficient ρ between the weighted layer index and average depth (maximum distance to leaf nodes) of each dependency label, getting a result of 0.69 with p=0.03 for a null hypothesis of no correlation.

5 Conclusion

We proposed a method that can estimate MI and reconstruct labeled dependency trees without introducing any trainable networks. Indeed, our method is achieving an "impossible triangle": it has *simpler* architectures requiring negligible computation budgets, while producing more *complex* and *high-quality* trees, and also *transparent for explanation*, meaning that researchers can get fine-grained head-level MI estimation, and a bunch of intuitive probability functions, without worrying about *did my network furtively learnt the task?* Through comparing with a series of competitive baselines, we ensured its effectiveness, and then made two informative conclusions based on our estimated MI and reconstructed trees. The number of conclusions is limited due to content limit, and since our method is providing an analytical backbone, we strongly appeal to future research for fine-grained analysis on those estimated MI values and distributions.

Limitations

Despite its efficiency, our method still has several shortages: The most important one is that, to prevent the problem from being too complicated and bounded by the curse of dimensionality, our method does not consider the multivariate case, taking an assumption that all attention heads are independent. Moreover, as mentioned in Section 4.2, the introduction of reconstructed attention scores is also noticeable, meaning that the density estimations might be partially based on "pseudo-scores" that are not actually used by the models during inference. In terms of potentials of transferring to other tasks, our method is only applicable to discrete-continuous mixtures, where all probed concepts are discrete labels, but not applicable to multivariate continuous joint distributions. In terms of model and dataset variety, we have only tested on decoder-only LLMs and English datasets, which might indicate the main and fine-grained results warrant further multi-language model-independent verifications.

Ethical Considerations

Since our method is an explanation method, readers might exploit our method to perform syntactical attacks, like getting poorly captured dependency labels and designing specific prompts to confuse models. For models that are put into use in production environments, this might cause unexpectable effects.

References

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In The 5th International Conference on Learning Representations.

Omer Antverg and Yonatan Belinkov. 2021. On the Pitfalls of Analyzing Individual Neurons in Language Models. In *International Conference on Learning Representations*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le and Jason Wei. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, *25*(70), 1-53.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 33, No. 01, pp. 6309-6317).

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dror Dotan and Nadin Brutmann. 2022. Syntactic chunking reveals a core syntactic representation of multi-digit numbers, which is generative and automatic. *Cognitive research: principles and implications*, *7*(1), 58.

Timothy Dozat and Christopher D. Manning. Deep Biaffine Attention for Neural Dependency Parsing. 2017. In *International Conference on Learning Representations*. 2017.

Jason M. Eisner. 1996. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell et al. 2021. A Mathematical Framework for Transformer Circuits. *Transformer Circuits Thread*, *1*(1), 12.

Jacqueline Fallon and Liina Pylkkänen. 2024. Language at a glance: How our brains grasp linguistic structure from parallel visual input. *Science Advances*, *10*(43), eadr9951.

Yimin Fan, Fahim Dalvi, Nadir Durrani, and Hassan Sajjad. 2024. Evaluating neuron interpretation methods of nlp models. *Advances in Neural Information Processing Systems*, *36*.

Javier Ferrando and Elena Voita. 2024. Information Flow Routes: Automatically Interpreting Language Models at Scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17432–17445, Miami, Florida, USA. Association for Computational Linguistics.

Weihao Gao, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. 2017. Estimating mutual information for discrete-continuous mixtures. *Advances in neural information processing systems*, *30*.

John Hewitt and Percy Liang. 2019. Designing and Interpreting Probes with Control Tasks. 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)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. 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 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in BERT track syntactic dependencies?. *arXiv preprint arXiv:1911.12246*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2002. A Generative Constituent-Context Model for Improved Grammar Induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic Knowledge and Transferability of Contextual Representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Alessandro Lopopolo, Antal Van den Bosch, Karl-Magnus Petersson, and Roel M. Willems. 2020. Distinguishing Syntactic Operations in the Brain: Dependency and Phrase-Structure Parsing. *Neurobiology of Language, 2*, 152 - 175.

Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. 1995. Estimation of mutual information using kernel density estimators. *Physical Review E*, 52(3), 2318.

Jesse Mu and Jacob Andreas. 2020. Compositional explanations of neurons. *Advances in Neural Information Processing Systems*, *33*, 17153-17163.

Max Müller-Eberstein, Rob van der Goot, and Barbara Plank. 2022. Probing for Labeled Dependency Trees. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7711–7726, Dublin, Ireland. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Rowan Hall Maudslay, Ran Zmigrod, Adina Williams, and Ryan Cotterell. 2020a. Information-Theoretic Probing for Linguistic Structure. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4609–4622, Online. Association for Computational Linguistics.

Tiago Pimentel, Naomi Saphra, Adina Williams, and Ryan Cotterell. 2020b. Pareto Probing: Trading Off Accuracy for Complexity. In *Proceedings of the 2020 Conference on Empirical Methods in Natural

Language Processing (EMNLP), pages 3138–3153, Online. Association for Computational Linguistics.

Tiago Pimentel, Josef Valvoda, Niklas Stoehr, and Ryan Cotterell. 2022. Attentional Probe: Estimating a Module's Functional Potential. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11459–11472, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. "Language models are unsupervised multitask learners." *OpenAI blog* 1, no. 8 (2019): 9.

Vinit Ravishankar, Artur Kulmizev, Mostafa Abdou, Anders Søgaard, and Joakim Nivre. 2021. Attention Can Reflect Syntactic Structure (If You Let It). In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3031–3045, Online. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin-wei Huang, and Aaron Courville. 2018. Neural Language Modeling by Jointly Learning Syntax and Lexicon. In *International Conference on Learning Representations*.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT Rediscovers the Classical NLP Pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das and Ellie Pavlick. 2019. What do you learn from context? Probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Yuanhe Tian, Yan Song, and Fei Xia. 2022. Enhancing Structure-aware Encoder with Extremely Limited Data for Graph-based Dependency Parsing. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 5438–5449, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Tom Heskes. 1997. Selecting weighting factors in logarithmic opinion pools. In *Advances in neural information processing systems*, 10.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the Structure of Attention in a Transformer Language Model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, Florence, Italy. Association for Computational Linguistics.

Elena Voita and Ivan Titov. 2020. Information-Theoretic Probing with Minimum Description Length. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 183–196, Online. Association for Computational Linguistics.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the wild: a circuit for indirect object identification in gpt-2 small. *arXiv preprint arXiv:2211.00593*.

Yilun Xu, Shengjia Zhao, Jiaming Song, Russell Stewart, and Stefano Ermon. 2020. A Theory of Usable Information under Computational Constraints. In *International Conference on Learning Representations*.

Daniel Zeman, Nivre Joakim, Abrams Mitchell, Ackermann Elia, and others. 2021. Universal Dependencies 2.9. *LINDAT/CLARIAH-CZ digital library* at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Jingyu Sun, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, Zihan Wang. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

A Implementation Details of IPBP

In this section, we'll briefly introduce the implementation details, like the hyperparameters and key algorithms we use to implement IPBP.

As shown by the source code, we use the PyTorch framework to implement the whole IPBP process. We're not relying on off-the-shelf packages that have KDE functionalities like SciPy and Scikit-Learn, since their KDE implementation is CPU-based and thus too inefficient under our experiment settings.

Specifically, we take samples in $\mathcal{A}_{b,h;l}$ as a whole long tensor $\boldsymbol{a}_{b,h,l} \in \mathbb{R}^{|\mathcal{A}_{b,h;l}|}$. We calculate the minimum and maximum values of $\mathcal{A}_{b,h;l}$, and build a tensor of real numbers $\boldsymbol{X} = \{x_1, x_2, \ldots x_{n_x}\}$, ensuring that $x_1 < \min \mathcal{A}_{b,h;l}$, $x_{n_x} > \max \mathcal{A}_{b,h;l}$, and $x_1 < x_2 < \cdots < x_{n_x}$. These discrete x values serve as the points to calculate densities. Next, we calculate the mutual differences between each point in $\boldsymbol{X}$ and each element in $\boldsymbol{a}_{b,h,l}$, by repeating $\boldsymbol{X}^T$ for n_x times, getting a matrix $\overbrace{[\boldsymbol{X}^T, \ldots \boldsymbol{X}^T]}^{|\mathcal{A}_{b,h;l}| \text{ times}}$ of shape $n_x \times |\mathcal{A}_{b,h;l}|$, and repeating $\boldsymbol{a}_{b,h,l}$ for n_x times, also getting a matrix $\overbrace{[\boldsymbol{a}_{b,h,l}^T, \ldots \boldsymbol{a}_{b,h,l}^T]^T}^{n_x \text{ times}}$ of shape $n_x \times |\mathcal{A}_{b,h;l}|$. The absolute differences of the two matrices $\left|[\boldsymbol{X}^T, \ldots \boldsymbol{X}^T] - [\boldsymbol{a}_{b,h,l}^T, \ldots \boldsymbol{a}_{b,h,l}^T]\right|$, are the mutual differences, let's say $D(\boldsymbol{X}^T, \boldsymbol{a}_{b,h,l}) \in \mathbb{R}^{n_x \times |\mathcal{A}_{b,h;l}|}$. Then we calculate the standard deviation of $\mathcal{A}_{b,h;l}$, i.e., $\sigma_{\mathcal{A}_{b,h;l}}$, and take a rule-of-thumb value $\left(\frac{1}{\sum_{i=1}^{|\mathcal{A}_{b,h;l}|} w_i^2}\right)^{-\frac{1}{5}}$ for the bandwidth B, with all weight w_is equal to 1. We then calculate element-wise, following the following equation:

$$\frac{1}{B\sqrt{2\pi \cdot \sigma_{\mathcal{A}_{b,h;l}}}} \exp\left\{-\left(\frac{D(\boldsymbol{X}^T, \boldsymbol{a}_{b,h,l})}{B}\right)^2\right\} \tag{13}$$

In order to get the kernel values, which are also in shape $n_x \times |\mathcal{A}_{b,h;l}|$, we then calculate the row-wise mean of the kernel values to get the final kernel density values in shape n_x. Since all operations of this process are element-wise matrix operations, this is easily parallel-optimizable by PyTorch. As a result, the computation time for extracting attention score sets ($\mathcal{A}_{\cdot,\cdot;\cdot}$) and performing all kernel density estimations is within 1 hour using a single RTX 4090 GPU. Since attention score allocating and distribution estimations are only required to be done once, our method is extremely time-saving compared to most of the supervised probing methods.

For inferring on estimated probabilities (like inferring on posteriors $\hat{f}(l|A_{b,h})$ in Section 3.5, we take the estimated posteriors as a set of n_x discrete points, and an attention score in range $[x_i, x_{i+1}]$ will get its corresponding posterior value by interpolating between x_i and x_{i+1}. The interpolation-based inferring, together with other processes mentioned in Section 3.5, like head selection and score-weighted averaging, are all parallel-optimized, resulting in being able to run inference within 5 minutes on all baseline settings on a 4090 GPU.

For calculating integrals, specifically, the MI values like $\mathrm{MI}_{\mathrm{binary}}$, $\mathrm{MI}_{\mathrm{pos}}$, we use the trapezoid method to estimate the integral value: as mentioned in the section before, the kernel densities are described by n_x points, we take the n_x points as the integral limits and for every interval between x_i and x_{i+1}, we calculate the trapezoid areas and add them up to get the integral values.

During tree reconstruction, we empirically set the total number of heads, i.e., $\sum_{l \in \mathcal{L} \cup \{\phi\}} |\mathcal{H}_l|$ to a fixed value (2000), and use binary search.

We're using Universal Dependencies 2.9 (Zeman et al., 2021) as our dataset. That dataset is publicly available, using CC BY-SA 4.0 license[2] allowing free redistributions upon notifications. The Universal Dependencies (UD) dataset is designed to provide a standardized framework of grammatical identifications for NLP researchers, so we're following its intended usage.

B Implementation Details of Baselines

For the $\mathcal{V}$-Information MLPs, we found that training on all datasets will result in a network always predicting ϕ for all possible attention scores, due to the long-tail essential discussed in Section 4.3. This will result in many infinite $\mathcal{V}$-Information values, since there will be many estimated probabilities (for label in $\mathcal{L}$ other than ϕ) rather close to zero. Therefore, we apply a sample balancing technique, truncating $\mathcal{A}_{\cdot,\cdot;\phi}$ to make their numbers of samples

[2] https://creativecommons.org/licenses/by-nc-sa/4.0/deed.en

the same as the total number of samples in other score sets $\mathcal{A}_{\cdot,\cdot;1}, \mathcal{A}_{\cdot,\cdot;2}, \ldots \mathcal{A}_{\cdot,\cdot;|\mathcal{L}|}$. What's more, we also did a search on several network sizes, and found that if $\mathrm{MLP}(\cdot)$ is $W_2(\mathrm{act}(W_1(\cdot)))$, where W_1 in shape 1×2 and a W_2 in shape 2×4 achieves better fitting. This aligns with (Pimentel et al., 2020a) to some extent. We also use PyTorch to implement the baselines. Specifically, for ElasticNet that requires additional training, we use AdamW optimizer, $1e-5$ for both λ_1 and λ_2, and use a constant learning rate of $1e-3$, training for 12 epochs. For the $\mathcal{V}$-Information MLP, since we need $\mathscr{b}\mathscr{h} \times (|\mathcal{L}|+1)$ individual networks for predicting the alternatives of binary MI, we initialize $\mathscr{b}\mathscr{h} \times (|\mathcal{L}|+1)$ sets of matrices, each constituting the weights of a specific network $W_1, W_2, \ldots W_{\mathrm{n\_layers}}$, with W_1 having a dimension of 1 and $W_{\mathrm{n\_layers}}$ having a dimension of $|\mathcal{L}|+1$. During training and inferencing, we concatenate all attention scores $a_{b,h}^{[i][j]}$ for any $b \in \{1 \ldots \mathscr{b}\}$ and $h \in \{1 \ldots \mathscr{h}\}$ into a tensor of shape $\mathscr{b}\mathscr{h}$, and use `torch.bmm` to map each element of that tensor to $\mathscr{b}\mathscr{h} \times (|\mathcal{L}|+1)$ probabilities (standing for the probabilities of each label conditioned on each attention head's attention score, estimated by the variational family). Using `torch.bmm` will avoid training $\mathscr{b}\mathscr{h} \times (|\mathcal{L}|+1)$ networks separately, which is a disaster on computation loads, and can exploit GPU's parallel processing abilities. We use leaky_relu between hidden layers and use sigmoid to form the final probabilities. We use $1e-2$ as learning rate with exponential decay (0.8 at each epoch), together with an additional warmup epoch at the beginning. The hyperparameters differ for $\mathcal{V}$-Information since otherwise the variational family network will be more poorly trained. We also trained for 12 epochs.

C Experiment Details of Further Analysis

For the term *look-ahead* and *look-back* with respect to dependency relations, we make statistics on the number of dependency arcs in the training dataset pointing to previous words / subsequent words, and we take relationships with more arcs pointing to previous words as look-back dependencies, and vice versa. To make the result more rigorous, apart from comparing the numbers of look-ahead / look-back dependencies in top-10 most well-reconstructed labels of original / transposed IPBP, we also made more fine-grained statistical tests: For each dependency relationship among the top-10-captured dependencies, we calculate the number of look-head / look-back arcs divided by total number of arcs, and made one-sided paired t-tests on between the proportions of look-head / look-back arcs with respect to normal / transposed settings. Results show a p-value of 0.048 for "greater", *i.e.*, the number proportions of look-ahead arcs from top-10 arcs under transposed setting is significantly higher than those under normal settings, confirming our conclusion in another aspect.