Towards a Deep and Unified Understanding of Deep Neural Models in NLP

Chaoyu Guan [*2] **Xiting Wang** [*2] **Quanshi Zhang** [1] **Runjin Chen** [1] **Di He** [3] **Xing Xie** [2]

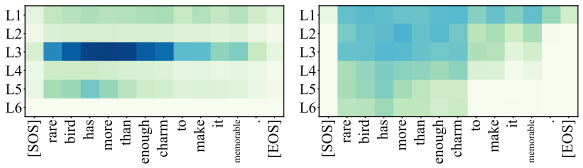
Abstract

We define a unified information-based measure to provide quantitative explanations on how intermediate layers of deep Natural Language Processing (NLP) models leverage information of input words. Our method advances existing explanation methods by addressing issues in coherency and generality. Explanations generated by using our method are consistent and faithful across different timestamps, layers, and models. We show how our method can be applied to four widely used models in NLP and explain their performances on three real-world benchmark datasets.

1. Introduction

Deep neural networks have demonstrated significant improvements over traditional approaches in many tasks (Socher et al., 2012). Their high prediction accuracy stems from their ability to learn discriminative feature representations. However, in contrast to the high discrimination power, the interpretability of DNNs has been considered an Achilles' heel for decades. The black-box representation hampers end-user trust (Ribeiro et al., 2016) and results in problems such as the time-consuming trail-and-error optimization process (Bengio et al., 2013; Liu et al., 2017), hindering further development and application of deep learning.

Recently, quantitatively explaining intermediate layers of a DNN has attracted increasing attention, especially in computer vision (Bau et al., 2017; Zhang et al., 2018a;d; 2019). A key task in this direction is to associate latent representations with the interpretable input units (e.g., image pixels or words) by measuring the contribution or saliency of the inputs. Existing methods can be grouped into three major categories: gradient-based (Li et al., 2015; Fong & Vedaldi, 2017; Sundararajan et al., 2017), inversion-based (Du et al., 2018), and methods that utilize layer-wise relevance propagation (LRP) (Arras et al., 2016). These methods have demonstrated that quantitative explanations for intermediate layers can enrich our understanding about the inner working mechanism of a model, such as, the roles of neurons.



(a) Gradient-based method (b) Ours

Figure 1. Illustration of coherency: (a) The gradient-based method highlights the third layer only because the parameters of this layer have larger absolute values; (b) Our method shows how the network gradually processes input words through layers.

Methods	Coherency			Generality
	Neuron	Layer	Model	
Gradient-based	✓	×	×	×
Inversion-based	✓	×	×	×
LRP	×	×	×	×
Ours	✓	✓	✓	✓

Table 1. Comparison of different methods in terms of coherency and generality. Our unified information-based measure can be defined with minimum assumptions (generality) and provides coherent results across neurons (timestamps in NLP), layers, and models.

The major issue of aforementioned methods is that their measures of saliency are usually defined based on heuristic assumptions. This leads to problems with respect to coherency and generality (Table 1):

Coherency requires that a method generates consistent explanations across different neurons, layers, and models. Existing measures usually fail to meet this criterion because of their biased assumptions. For example, gradient-based methods assume that saliency can be measured by absolute values of derivatives. Fig. 1(a) shows gradient-based explanations. Each line in this figure represents a layer. According to this figure, the input words contribute most to the third layer (darkest color in L3). However, the third layer stands out only because the absolute values of their parameters are large. A desirable measure should quantify word contributions without bias and reveal how the network structure

[*]Equal contribution [1]John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, Shanghai, China [2]Microsoft Research Asia, Beijing, China [3]Peking University, Beijing, China. Correspondence to: Quanshi Zhang <zqs1022@sjtu.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

gradually processes inputs through layers (Fig. 1(b)).

Generality refers to the problem that existing measures are usually defined under certain restrictions on model architectures or tasks. For example, gradient-based methods can only be defined for models whose neural activations are differentiable or smooth (Ding et al., 2017). Inversion-based methods are typical methods for explaining vision models and assume that the feature maps can be inverted to a reconstructed image by using functions such as upsampling (Du et al., 2018). This limits their application in NLP models.

In this paper, we aim to provide quantitative explanations based on a measure that satisfies coherency and generality. **Coherency** corresponds to the notion of *equitability*, which requires that the measure quantifies associations between inputs and latent representations without bias with respect to relationships of a specific form. Recently, (Kinney & Atwal, 2014) have mathematically formalized equitability and proven that mutual information satisfies this criterion. Moreover, as a fundamental quantity in information theory, mutual information can be mathematically defined without much restrictions on model architectures or tasks (**generality**). Based on these observations, we explain intermediate layers based on mutual information. Specifically, this study aims to answer the following research questions:

RQ1. How does one use mutual information to quantitatively explain intermediate layers of DNNs?

RQ2. Can we leverage measures based on information as a tool to analyze and compare existing explanation methods theoretically?

RQ3. How can the information-based measure enrich our capability of explaining DNNs and provide insights?

By examining these issues, we move towards a deep (aware of intermediate layers) and unified (coherent) understanding of neural models. We use models in NLP as guiding examples to show the effectiveness of information-based measures. In particular, we make the following contributions.

First, we **define a unified information-based measure** to quantify how much information of an input word is encoded in an intermediate layer of a deep NLP model (RQ1)[1]. We show that our measure advances existing measures in terms of coherency and generality. This measure can be efficiently estimated by perturbation-based approximation and can be used for fine-grained analysis on word attributes.

Second, we **show how the information-based measure can be used as a tool for comparing different explanation methods** (RQ2). We demonstrate that our method can be regarded as a combination of maximum entropy optimization and maximum likelihood estimation.

Third, we **demonstrate how the information-based measure enriches the capability of explaining DNNs** by conducting experiments in one synthetic and three real-world benchmark datasets (RQ3). We explain four widely used models in NLP, including BERT (Devlin et al., 2018), Transformer (Vaswani et al., 2017), LSTM (Hochreiter & Schmidhuber, 1997), and CNN (Kim, 2014).

[1]Codes available at https://aka.ms/nlp/explainability

2. Related Works

Our work is related to various methods for explaining deep neural networks and learning interpretable features.

Explaining deep vision models. Many approaches have been proposed to diagnose deep models in computer vision. Most of them focus on understanding CNNs. Among all methods, the visualization of filters in a CNN is the most intuitive way for exploring appearance patterns inside the filters (Simonyan et al., 2013; Zeiler & Fergus, 2014; Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016; Olah et al., 2017). Besides network visualization, methods are developed to show image regions that are responsible for prediction. (Bau et al., 2017) use spatial masks on images to determine the related image regions. (Kindermans et al., 2017) extract the related pixels by adding noises to input images. (Fong & Vedaldi, 2017; Selvaraju et al., 2017) compute gradients of the output with respect to the input image.

Other methods (Zhang et al., 2018b;a; 2017; Vaughan et al., 2018; Sabour et al., 2017) learn interpretable representations for neural networks. Adversarial diagnosis of neural networks (Koh & Liang, 2017) investigates network representation flaws using adversarial samples of a CNN. (Zhang et al., 2018c) discovers representation flaws in neural networks caused by potential bias in data collection.

Explaining neural models in NLP. Model-agnostic methods that explain a black-box model by probing into its input and/or output layers can be used for explaining any model, including neural models in NLP (Ribeiro et al., 2016; Lundberg & Lee, 2017; Koh & Liang, 2017; Peake & Wang, 2018; Tenney et al., 2019). These methods are successful in helping understand the overall behavior of a model. However, they fail to explain the inner working mechanism of a model as the informative intermediate layers are ignored (Du et al., 2018). For example, they cannot explain the role of each layer or how information flows through the network.

Recently, explaining the inner mechanism of deep NLP models has started to attract attention. Pioneer works on this direction can be divided into two categories. The first category learns an interpretable structure (e.g., Finite State Automaton) from RNN and use the interpretable structure as an explanation (Hou & Zhou, 2018). Works in the second category visualize neural networks to help understand their meaning composition. These works either leverage dimension reduction methods such as t-SNE to plot the latent

representation (Li et al., 2015) or compute the contribution of a word to predictions or hidden states by using first-derivative saliency (Li et al., 2015) or layer-wise relevance propagation (LRP) (Arras et al., 2016; Ding et al., 2017).

Compared with the aforementioned methods, our unified information-based method can provide consistent and interpretable results across different timestamps, layers, and models (coherency), can be defined with minimum assumptions (generality), and is able to analyze word attributes.

3. Methods

In this section, we first introduce the objective of interpreting deep NLP neural networks. Then, we define the word information in hidden states and analyze fine-grained attribute information within each word.

3.1. Problem Introduction

A deep NLP neural network can be represented as a function $f(\mathbf{x})$ of the input sentence $\mathbf{x}$. Let $\mathbf{X}$ denote a set of input sentences. Each sentence is given as a concatenation of the vectorized embedding of each word $\mathbf{x} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_n^T]^T \in \mathbf{X}$, where $\mathbf{x}_i \in \mathbb{R}^K$ denotes the embedding of the i-th word.

Suppose the neural network f contains L intermediate layers. f can be constructed by layers of RNNs, self-attention layers like that in Transformer, or other types of layers. Given an input sentence $\mathbf{x}$, the output of each intermediate layer is a series of hidden states. The goal of our research is to explain hidden states of intermediate layers by quantifying the information of the word $\mathbf{x}_i$ that is contained by the hidden states. More specifically, we explain hidden states from the following two perspectives.

- **Word information quantification:** Quantifying contributions of individual input units is a fundamental task in explainable AI (Ding et al., 2017). Given $\mathbf{x}_i$ and a hidden state $\mathbf{s} = \Phi(\mathbf{x})$, where $\Phi(\cdot)$ denotes the function of the corresponding intermediate layer, we quantify the amount of information in $\mathbf{x}_i$ that is encoded in $\mathbf{s}$. The measure of word information provides the foundation for explaining intermediate layers.
- **Fine-grained analysis of word attributes:** We analyze the fine-grained reason why a neural network uses the information of a word. More specifically, when the neural network pays attention to a word $\mathbf{x}_i$ (e.g., *tragic*), we disentangle the information representing its attributes (e.g., *negative adjective* or *emotional adjective*) away from the specific information of the word.

3.2. Word Information Quantification

In this section, we quantify the information of word $\mathbf{x}_i$ that is encoded in the hidden states of the intermediate layers. To this end, we first define information at the coarsest level (*i.e.* corpus-level), and then gradually decompose the information to fine-grained levels (*i.e.* sentence-level and word-level). Next, we show how the information can be efficiently estimated via perturbation-based approximation.

3.2.1. Multi-level Quantification

Corpus-level. We provide a global explanation of the intermediate layer considering the entire sentence space. Let random variable $\mathbf{S}$ denotes a hidden state, the information of $\mathbf{X}$ encoded by $\mathbf{S}$ can be measured by

$$MI(\mathbf{X}; \mathbf{S}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{S}), \tag{1}$$

where $MI(\cdot;\cdot)$ represents the mutual information, $H(\cdot)$ represents the entropy. $H(\mathbf{X})$ is a constant, and $H(\mathbf{X}|\mathbf{S})$ denotes the amount of information that is discarded by the hidden states. We can calculate $H(\mathbf{X}|\mathbf{S})$ by decomposing it into the sentence level:

$$H(\mathbf{X}|\mathbf{S}) = \int_{\mathbf{s}\in\mathbf{S}} p(\mathbf{s}) H(\mathbf{X}|\mathbf{s}) d\mathbf{s}. \tag{2}$$

Sentence-level. Let $\mathbf{x}$ and $\mathbf{s} = \Phi(\mathbf{x})$ denote the input sentence and its corresponding hidden state of an intermediate layer. The information that $\mathbf{s}$ discards can be measured as the conditional entropy of input sentences given $\mathbf{s}$:

$$H(\mathbf{X}|\mathbf{s}) = -\int_{\mathbf{x}'\in\mathbf{X}} p(\mathbf{x}'|\mathbf{s}) \log p(\mathbf{x}'|\mathbf{s}) d\mathbf{x}'. \tag{3}$$

$H(\mathbf{X}|\mathbf{s} = \Phi(\mathbf{x}))$ reflects how much information from sentence $\mathbf{x}$ is discarded by $\mathbf{s}$ during the forward propagation. The entropy $H(\mathbf{X}|\mathbf{s})$ reaches the minimum value if and only if $p(\mathbf{x}'|\Phi(\mathbf{x})) \ll p(\mathbf{x}|\Phi(\mathbf{x}))$, $\forall \mathbf{x}' \neq \mathbf{x}$. This indicates that $\Phi(\mathbf{x}') \neq \Phi(\mathbf{x})$, $\forall \mathbf{x}' \neq \mathbf{x}$, which means that all information of $\mathbf{x}$ is leveraged. If only a small fraction of information of $\mathbf{x}$ is leveraged, then we expect $p(\mathbf{x}'|\mathbf{s})$ to be more evenly distributed, resulting in a larger entropy $H(\mathbf{X}|\mathbf{s})$.

Word-level. To further disentangle information components of individual words from the sentence, we follow the assumption of independence between input words, which has been widely used in studies of disentangling linear word attributions (Ribeiro et al., 2016; Lundberg & Lee, 2017). In this case, we have $H(\mathbf{X}|\mathbf{s}) = \sum_i H(\mathbf{X}_i|\mathbf{s})$ and

$$H(\mathbf{X}_i|\mathbf{s}) = -\int_{\mathbf{x}_i'\in\mathbf{X}_i} p(\mathbf{x}_i'|\mathbf{s}) \log p(\mathbf{x}_i'|\mathbf{s}) d\mathbf{x}_i', \tag{4}$$

where $\mathbf{X}_i$ is the random variable of the i-th input word.

Comparisons with word attribution/importance: The quantification of word information is different from previous studies of estimating word importance/attribution with respect to the prediction output (Ribeiro et al., 2016; Lundberg & Lee, 2017). Our research aims to quantify the amount

of information of a word that is used to compute hidden states in intermediate layers. In contrast, previous studies estimate a word's numerical contribution to the final output without considering how much information in the word is used by the network. Generally speaking, from the perspective of word importance/attribution estimation (Ribeiro et al., 2016; Lundberg & Lee, 2017), our word information can be regarded as the confidence of the use of each input word.

3.2.2. Perturbation-based Approximation

Approximating $H(\mathbf{X}_i|\mathbf{s})$ by perturbation: The core of calculating $H(\mathbf{X}_i|\mathbf{s})$ is to estimate $p(\mathbf{x}_i|\mathbf{s})$ in Eq. (4). However, the relationship between $\mathbf{x}_i$ and $\mathbf{s}$ is very complex (modeled by the deep neural network) , which makes calculating the distribution of $\mathbf{X}_i$ directly from $\mathbf{s}$ intractable.

Therefore, in this subsection, we propose a perturbation-based method to approximate $H(\mathbf{X}_i|\mathbf{s})$. Let $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \boldsymbol{\epsilon}_i$ denote an input with a certain noise $\boldsymbol{\epsilon}_i$. We assume that the noise term is a random variable that follows a Gaussian distribution, $\boldsymbol{\epsilon}_i \in \mathbb{R}^K$ and $\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_i = \sigma_i^2\mathbf{I})$. In order to approximate $H(\mathbf{X}_i|\mathbf{s})$, we first learn an optimal distribution of $\boldsymbol{\epsilon} = [\boldsymbol{\epsilon}_1^T, \boldsymbol{\epsilon}_2^T, ..., \boldsymbol{\epsilon}_n^T]^T$ with respect to the hidden state $\mathbf{s}$ with the following loss.

$$L(\boldsymbol{\sigma}) = \mathbb{E}_{\boldsymbol{\epsilon}}\|\Phi(\tilde{\mathbf{x}}) - \mathbf{s}\|^2 - \lambda \sum_{i=1}^{n} H(\tilde{\mathbf{X}}_i|\mathbf{s})|_{\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma_i^2\mathbf{I})}, \quad (5)$$

where $\lambda > 0$ is a hyper-parameter, $\boldsymbol{\sigma} = [\sigma_1, ..., \sigma_n]$, and $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$. The first term on the left corresponds to the maximum likelihood estimation (MLE) of the distribution of $\tilde{\mathbf{x}}_i$ that maximizes $\sum_i \sum_{\tilde{\mathbf{x}}_i} \log p(\tilde{\mathbf{x}}_i|\mathbf{s})$, if we consider $\sum_i \log p(\tilde{\mathbf{x}}_i|\mathbf{s}) \propto -\|\Phi(\tilde{\mathbf{x}}) - \mathbf{s}\|^2$. In other words, the first term learns a distribution that generates all potential inputs corresponding to the hidden state $\mathbf{s}$. The second term on the right encourages a high conditional entropy $H(\tilde{\mathbf{X}}_i|\mathbf{s})$, which corresponds to the maximum entropy principle. In other words, the noise $\boldsymbol{\epsilon}$ needs to enumerate all perturbation directions to reach the representation limit of $\mathbf{s}$. Generally speaking, $\boldsymbol{\sigma}$ depicts the range that the inputs can change to obtain the hidden state $\mathbf{s}$. Large $\boldsymbol{\sigma}$ means that a large amount of input information has been discarded. We provide an intuitive example to illustrate this in the supplement.

Since we use the MLE loss as constraints to approximate the conditional distribution of $\mathbf{x}_i$ given $\mathbf{s}$, we can use $H(\tilde{\mathbf{X}}_i|\mathbf{s})$ to approximate $H(\mathbf{X}_i|\mathbf{s})$. In this way, we have

$$p(\tilde{\mathbf{x}}_i|\mathbf{s}) = p(\boldsymbol{\epsilon}_i) \;\Rightarrow\; H(\tilde{\mathbf{X}}_i|\mathbf{s}) = \frac{K}{2}\log(2\pi e) + K\log\sigma_i \quad (6)$$

Therefore, the objective can be rewritten as the minimization of the following loss.

$$L(\boldsymbol{\sigma}) = \sum_{i=1}^{n}(-\log\sigma_i) + \frac{1}{K\lambda}\mathbb{E}_{\tilde{\mathbf{x}}_i:\boldsymbol{\epsilon}_i \sim \mathcal{N}(\mathbf{0}, \sigma_i^2\mathbf{I})}\frac{\|\Phi(\tilde{\mathbf{x}}) - \mathbf{s}\|^2}{\sigma_{\mathbf{S}}^2}. \quad (7)$$

Here, $\sigma_{\mathbf{S}}^2$ denotes the variance of $\mathbf{S}$ for normalization, which can be computed using sampling.

Relationship with the existing perturbation method: Our perturbation method is similar to the one in (Du et al., 2018). While our method enumerates all possible perturbing directions in the embedding space to learn an optimal noise distribution, (Du et al., 2018) perturb inputs towards one heuristically designed direction that may not be optimal.

3.3. Fine-Grained Analysis of Word Attributes

In this subsection, we analyze the fine-grained attribute information inside each input word that is used by the intermediate layers of the neural network.

Given a word $\mathbf{x}_i$ (e.g., *tragic*) in sentence $\mathbf{x}$, we assume that each of its attribute corresponds to a concept $\mathbf{c}$ (e.g., *negative adjective* or *emotional adjective*). Here, concept $\mathbf{c}$ (e.g., *emotional adjective*) is represented by the set of words belonging to this concept (e.g., {*happy*, *sorrowful*, *sad*, ...}). The concepts can be mined by using knowledge bases such as DBpedia (Lehmann et al., 2015) and Microsoft Concept Graph (Wu et al., 2012; Wang et al., 2015).

When the neural network uses a word $\mathbf{x}_i$, we disentangle the information of a common concept $\mathbf{c}$ away from all the information of the target word. The major idea is to calculate the relative confidence of $\mathbf{s}$ encoding certain words with respect to random words:

$$A_i = \log p(\mathbf{x}_i|\mathbf{s}) - \mathbb{E}_{\mathbf{x}_i' \in \mathbf{X}_i}\log p(\mathbf{x}_i'|\mathbf{s}) \quad (8)$$

$$A_{\mathbf{c}} = \mathbb{E}_{\mathbf{x}_i' \in \mathbf{X}_{\mathbf{c}}}\log p(\mathbf{x}_i'|\mathbf{s}) - \mathbb{E}_{\mathbf{x}_i' \in \mathbf{X}_i}\log p(\mathbf{x}_i'|\mathbf{s}). \quad (9)$$

Here, $\mathbf{X}_{\mathbf{c}}$ is the word embeddings corresponding to $\mathbf{c}$ and $\mathbb{E}_{\mathbf{x}_i' \in \mathbf{X}_i}\log p(\mathbf{x}_i'|\mathbf{s})$ indicates the baseline log-likelihood of all random words. We use A_i (or $A_{\mathbf{c}}$) to approximate the relative confidence of $\mathbf{s}$ encoding $\mathbf{x}_i$ (or words in $\mathbf{c}$) with respect to random words. The intuition is that larger $\log p(\mathbf{x}_i'|\mathbf{s})$ corresponds to larger confidence that $\mathbf{s}$ encodes the information in $\mathbf{x}_i'$.

Based on Eqs. (8)(9), we use $r_{i,c} = A_i - A_{\mathbf{c}}$ to investigate the remaining information of the word $\mathbf{x}_i$ when we remove the information of the common attribute $\mathbf{c}$ from the word.

4. Comparative Study

In this study, we compare our methods with three baselines in terms of their explanation capability. In particular, we study whether the methods can give faithful and coherent explanations when used for comparing different **timestamps** (Sec. 4.1), **layers** (Sec. 4.2), and **models** (Sec. 4.3). Results indicate that our method gives the most faithful explanations and may be used as a guidance for selecting models or tuning model parameters. The **baselines** we use include:

- **Perturbation** (Fong & Vedaldi, 2017) is a method for explaining computer vision models. We migrate this

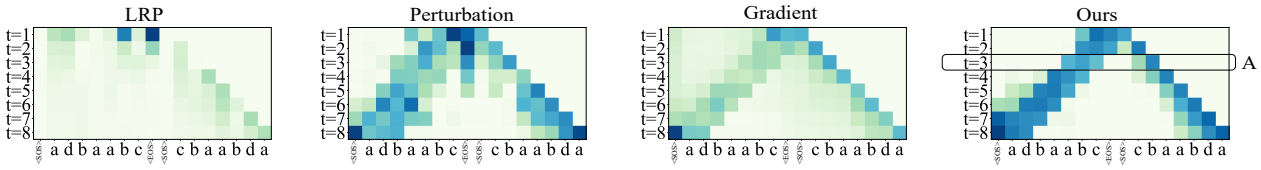


Figure 2. Saliency maps at different **timestamps** compared with three baselines. The model we analyze learns to reverse sequences. Our method shows a clear "reverse" pattern. Perturbation and gradient methods also reveal this pattern, although not as clear as ours.

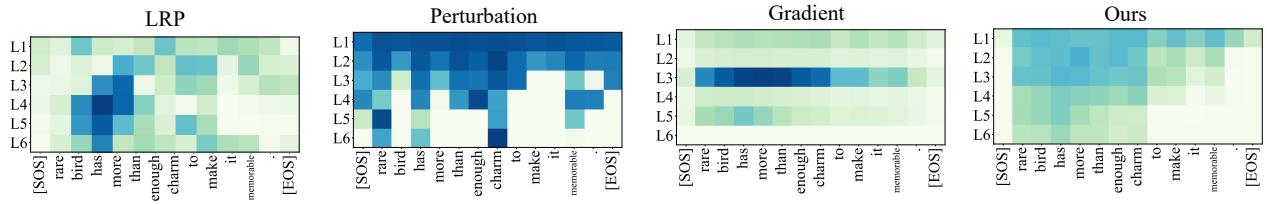


Figure 3. Saliency maps of different **layers** comparing with three baselines. Our method shows how information decreases through layers.

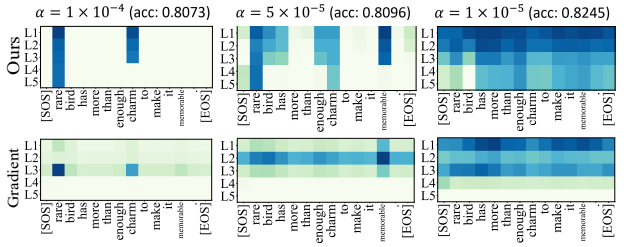


Figure 4. Saliency maps for **models** with different hyperparameters. Here, α refers to the weight of the regularization term.

method directly to NLP by treating the input sentence $\mathbf{x}$ as an image.

- **LRP** (Bach et al., 2015) is a method that can measure the relevance score of any two neurons. Following (Ding et al., 2017), we visualize the absolute values of the relevance scores between a certain hidden state and input word embeddings.
- **Gradient** (Li et al., 2015) is a method that uses the absolute value of first-derivative to represent the saliency of each input words. We use the average saliency value of all dimensions of word embedding to represent the its word-level saliency value.

The baselines are the most representative methods in each category. Other more advanced methods (Sundararajan et al., 2017) share similar issues with the selected baselines and their results are presented in the supplement.

4.1. Across Timestamp Analysis

In this experiment, we compare our methods with the baselines in terms of their ability in giving faithful and coherent explanations across timestamps in the last hidden layer.

Model. We train a two-layer LSTM model (with attention) that learns to reverse sequences. The model is trained by using a synthetic dataset that contains only four words: a, b, c, and d. The input sentences are generated by randomly sampling tokens and the output sentence is computed by reversing the input sentence. The test accuracy is 81.21%.

Result. Fig. 2 shows saliency maps computed by different explanation methods. Each line in the map represents a timestamp and each column represents an input word. For our method, we visualize σ_i calculated by optimizing Eq. (7). The saliency maps show how the hidden state in the last hidden layer changes as different words are fed into the network. For example, the line shown in Fig. 2A means that after the 3rd word b is fed into the decoder (t=3), the hidden state of the last hidden layer mainly encodes five input words: a, b, c from the encoder and c, b from the decoder. Note that all words before <EOS> are inputs to the encoder and all words after the second <SOS> are inputs to the decoder.

As shown in the figure, our method shows a very clear "reverse" pattern, which means that the last hidden layer mainly encodes two parts of information. The first part contains information about the last words fed into the decoder (e.g. c, b in Fig. 2A). Used as a query in the attention layer, this part is used to retrieve the second part of information, which are related input words in the encoder (e.g., a, b, c in Fig. 2A). By comparing the two parts, the model obtains information about the next output word (e.g., a). The gradient method and the perturbation method also reveal this pattern, although their patterns are not as clear as ours. Compared with others, LRP fails to display a clear pattern.

4.2. Across Layer Analysis

In this subsection, we compare our method and the baselines in terms of their ability in providing faithful and coherent explanations across different layers. For each layer, we con-

catenate its hidden states at different timestamps as one vector. Then, we compute the associations between the concatenated vector and the input words.

Dataset. The dataset we use is **SST-2**, which stands for Stanford Sentiment Treebank (Socher et al., 2013). It is a real-world benchmark for sentence sentiment classification.

Model. We train a LSTM model that contains four 768D (per direction) Bidirectional LSTM layers, a max-pooling layer, and a fully connected layer. The inputs word embeddings are 768D randomly initialized vectors.

Result. Fig. 3 shows the saliency maps at different layers. Our method clearly shows that the information contained in each layer gradually decreases. This indicates that the model gradually focuses on the most important parts of the sentence. Although the perturbation method shows a similar pattern, its result is much more noisy. LRP and gradient methods fail to generate coherent pattern across layers because of their heuristic assumptions about word saliency.

4.3. Across Model Analysis

In this experiment, we study how different choices of hyperparameters affect the hidden states learned by the models. Comparison of different model architectures will be presented in Sec. 5. Here, we use the encoder from Transformer (Vaswani et al., 2017) as an example. The encoder consists of 3 multi-head self-attention layers (head number is 4, hidden state size is 256, and feed-forward output size is 1024), a first-pooling layer and a fully connected layer. The input word embedding is randomly initialized 256D vectors. The dataset we use is SST-2.

Fig. 4 visualizes the saliency maps of models trained with different L2 normalization penalty value α. Our method shows that the information encoded in a model decreases with increasing regularization weight α. For example, the model with the largest α ($\alpha = 1 \times 10^{-4}$) only encodes the word *rare* in the last hidden layer. By decreasing α to 5×10^{-5}, the Transformer encodes one more word: *charm*. Although these models tend to encode the most important words, some other important words (e.g., *memorable*) are ignored because of their large α. This may be a reason why these models have lower accuracy. By using our information-based measure, we can quickly identify that 1) the models with large α ($\alpha \geq 5 \times 10^{-5}$) contain too little information and that 2) we should decrease α to improve the performance. In comparison, it is very difficult for the gradient method to provide similar guidance on hyperparameter tuning.

5. Understanding Neural Models in NLP

A variety of deep neural models have blossomed in NLP. The goal of this study is to understand these models by addressing three questions: 1) what information is leveraged by the models for prediction, 2) how does the information flow through layers in different models, and 3) how do different models evolve during training? In particular, we study four widely used models: BERT (Devlin et al., 2018), Transformer (Vaswani et al., 2017), LSTM (Hochreiter & Schmidhuber, 1997), and CNN (Kim, 2014).

Table 2. Summary of model performance on different datasets. Best results are highlighted in bold. Here, Acc stands for accuracy and MCC is the Matthews correlation coefficient.

	SST-2 (Acc)	CoLA (MCC)	QQP (Acc)
BERT	**0.9323**	**0.6110**	**0.9129**
Transformer	0.8245	0.1560	0.7637
LSTM	0.8486	0.1296	0.8658
CNN	0.8200	0.0985	0.8099

We train the four models on three public accessible datasets from different domains:

- **SST-2** (Socher et al., 2013) is the sentiment analysis benchmark we introduce in Sec. 4.2.
- **CoLA** (Warstadt et al., 2018) stands for the Corpus of Linguistic Acceptability. It consists of English sentences and binary labels about whether the sentences are linguistically acceptable or not.
- **QQP** (Iyer et al., 2018) is the Quora Question Pairs dataset. Each sample in the dataset contains two questions asked on Quora and a binary label about whether the two questions are semantically equivalent.

Table 2 summarizes how the models perform on different datasets. We can see that BERT consistently outperforms the other three models on different datasets.

5.1. What Information is Leveraged for Prediction?

To analyze what information the models use for prediction, we consider $\mathbf{s}$ as the hidden state used by the output layer (the input to the final softmax function). For BERT, $\mathbf{s}$ is the [CLS] token in the last hidden layer. The results are shown in Fig. 5. We make the following observations.

Pre-trained v.s. not pre-trained. Fig. 5 shows that the pre-trained model (BERT) can easily discriminate stopwords from important words in all datasets. We further verify this observation by sampling 100 sentences in each dataset and calculating the frequent words used for prediction. Fig. 6 shows the results on the SST-2 dataset. We can see that BERT learns to focus on meaningful words (e.g., *film*, *little*, and *comedy*) while other models usually focus on stopwords (e.g., *and*, *the*, *a*). The capability to discriminate stopwords is useful for various tasks. This may be one reason why BERT achieves state-of-the-art performance on 11 tasks (Devlin et al., 2018).

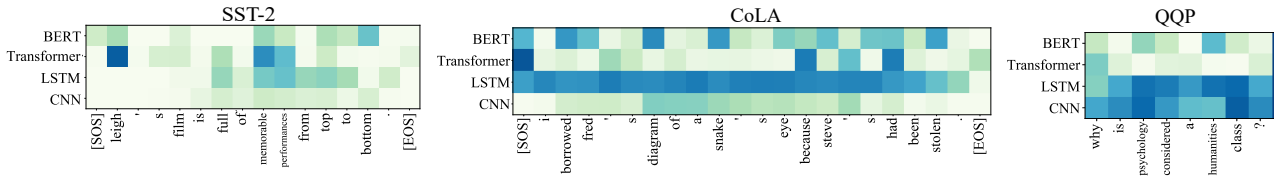


Figure 5. Words that different models use for prediction. For QQP, we only show the first question from the question pair.

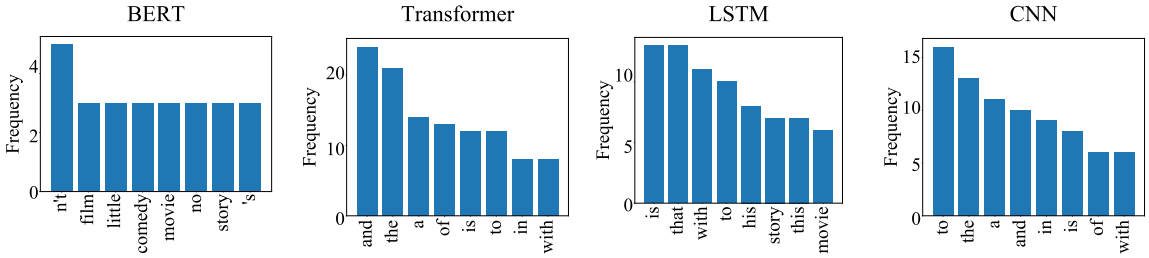


Figure 6. Words leveraged by each model for prediction (100 sampled sentences in SST-2 dataset).

Effects of different model architectures. Fig. 5 shows that LSTM and CNN tend to use sub-sequences of consecutive words for prediction, while models based on self-attention layers (BERT and Transformer) tend to use multiple word segments. For LSTM and CNN, their smooth nature may be a reason for not performing well. For example, when predicting whether a sentence is linguistically acceptable (CoLA), LSTM focus almost on the whole sentence. A potential reason is that its recurrent structure limits its ability to filter several noisy words from the whole sequence. Although Transformer does not have similar constraints in structure, it appears that it only uses information of few words. BERT is able to resolve this problem because it is pre-trained on tasks such as language modeling.

5.2. How Does the Information Flow Through Layers?

We investigate how information flows through layers from two perspectives. First, we study how much information of a word is leveraged by different layers of a model (Sec. 5.2.1). Next, we perform fine-grained analysis on which word attributes are used (Sec. 5.2.2). For each layer, we concatenate its hidden states at different timestamps as one vector and consider the concatenated vector as $\mathbf{s}$.

5.2.1. Word Information

Fig. 7 shows how different models process words through layers. Here, we show an example sentence from the SST-2 dataset. For all models other than CNN, the information gradually decreases through layers. **BERT** tends to discard information about meaningless words first (e.g., *to*, *it*). At the last layers, it discard information about words that are less related with the task (e.g., *enough*). Most words that are important for deciding the sentiment of the sentence are remained (e.g., *charm*, *memorable*). Compared with BERT, **Transformer** is less reasonable. It fails to discriminate meaningless words such as *to* with meaningful words such as *bird*. It seems that Transformer achieves reasonably good accuracy by focusing more on task related words (e.g., *memorable*). However, the information considered by Transformer is much more noisy compared with that in BERT. This again demonstrates the usefulness of the pre-training process. **LSTM** gradually focuses on the first part of sentence (i.e., *rare bird has more than enough charm*). This is reasonable, as the first part of the sentence is useful for sentiment prediction. However, an important word in the second part of the sentence (*memorable*) is ignored because of the smooth nature of LSTM. **CNN** has the most distinct behavior because four of its layers are independent with each other. The four layers (K1, K3, K5, K7) correspond to kernels with different widths. These layers detect important sub-sequences of different lengths. We can see that although Transformer, LSTM and CNN have similar accuracy, the word information they leverage and their inner work mechanisms are quite different.

5.2.2. Word Attributes

In this part, we provide a fine-grained analysis of word attributes on different models in the **SST-2** dataset. The word we use is *unhappiness* from sentence *domestic tension and unhappiness*. Fig. 8 shows $r_{i,c}$ calculated from Eq. (8) and Eq. (9) in every layer. The attributes are collected from Microsoft Concept Graph (Wu et al., 2012) and manually refined to eliminate errors. The figure shows that for all the models, $r_{i,c}$ decreases when layer number increases, which means that the hidden states in last layers utilize the concept attribute of certain words more. However, the attributes of *unhappiness* that is leveraged by different models are different. Among four models, BERT use concept attributes more and distinguish attributes the best. All models except for

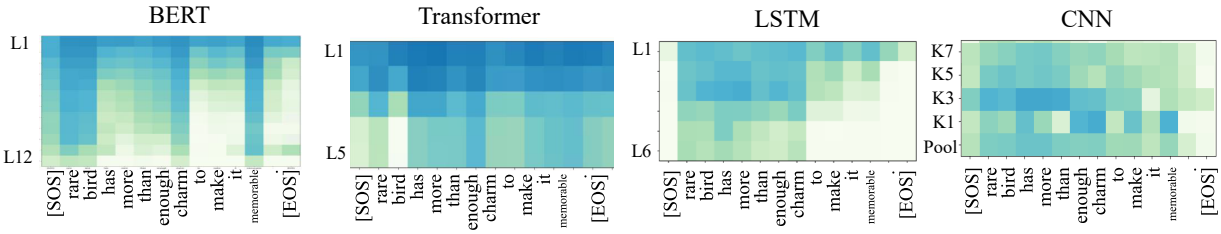


Figure 7. Layerwise analysis of word information. For all models other than CNN, the information gradually decreases through layers.

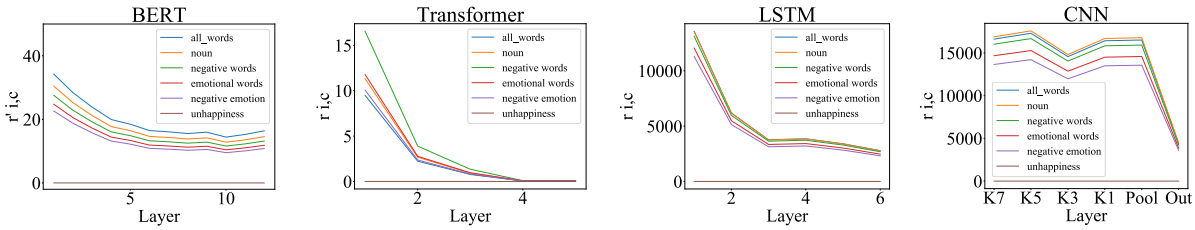


Figure 8. Layerwise analysis of word attributes. The models tend to gradually emphasize the information of word attributes through layers. The Transformer fails to learn which attribute of the word *unhappiness* is important for sentiment analysis.

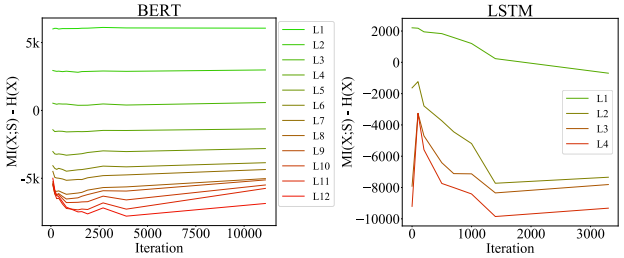


Figure 9. Mutual Information change of each layer during training process of BERT and LSTM.

Transformer leverages attribute *negative emotion* the most, and the attributes such as *noun* and *all words*, which are not relative to *unhappiness*, are relatively less likely to be leveraged by these models. LSTM and CNN appear to collapse concepts because of the scale of the vertical axis (wider ranges compared with that of BERT). They actually can distinguish concepts relatively well, with $\frac{\max(r_{i,c})}{\min(r_{i,c})} > 1.2$. Transformer, however, fails to effectively utilize the fine-grained attribute information inside *unhappiness*. That may be a reason why it performs poorly on this dataset.

5.3. How Do the Models Evolve During Training?

Fig. 9 shows how mutual information changes during the training processes on all layers of LSTM and BERT. We can see that, the mutual information in BERT is more stable than that in LSTM during training, with only some adjustments at several last layers. We also observe that LSTM experiences an information expansion state at the start of training, during which the mutual information will increase. After that, LSTM compresses its information. It can be explained that during training, LSTM will first pass as much input information as possible to last layers for prediction, and then discard unimportant input information to further boost up its performance.

5.4. Summary and Takeaways

The major takeaways of our comparative study are threefolds. In terms of **understanding**, we find that the good performance of BERT stems from its ability to discard meaningless words at first layers, reasonably utilize word attributes, and fine-tune stablely. With respect to **diagnosis**, we show that different models have different drawbacks. LSTM and CNN tend to focus on sub-sequences and are easy to use information of noisy words. Transformer tends to focus on individual words and may be too flexible to learn well. Such analysis leads to suggestions on future **refinement**. For example, to improve LSTM and CNN, we may focus on how to eliminate their inclination on noisy words (e.g., increase model flexibility). For Transformer, we may focus on pre-training, which may alleviate its over-flexibility issue.

6. Conclusion

We define a unified information-based measure to quantitatively explain intermediate layers of deep neural models in NLP. Compared with existing methods, our method can provide consistent and faithful results across timestamps, layers, and models (coherency). Moreover, it can be defined with minimum assumptions (generality). We show how our information-based measure can be used as a tool for comparing different explanation methods and demonstrate how it enriches our capability in understanding DNNs.

Acknowledgements

The corresponding author Quanshi Zhang thanks the support of Microsoft Research Asia and the Huawei-Shanghai Jiao Tong University Long-term Collaboration Fund in Intelligent Multimedia Technology.

Literatur

Arras, L., Horn, F., Montavon, G., Müller, K.-R., and Samek, W. Explaining predictions of non-linear classifiers in nlp. In *Workshop on Representation Learning for NLP*, pp. 1–7, 2016.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. *arXiv preprint arXiv:1704.05796*, 2017.

Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8): 1798–1828, 2013.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ding, Y., Liu, Y., Luan, H., and Sun, M. Visualizing and understanding neural machine translation. In *Annual Meeting of the Association for Computational Linguistics*, volume 1, pp. 1150–1159, 2017.

Dosovitskiy, A. and Brox, T. Inverting visual representations with convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4829–4837, 2016.

Du, M., Liu, N., Song, Q., and Hu, X. Towards explanation of dnn-based prediction with guided feature inversion. *arXiv preprint arXiv:1804.00506*, 2018.

Fong, R. C. and Vedaldi, A. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision*, pp. 3449–3457. IEEE, 2017.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hou, B.-J. and Zhou, Z.-H. Learning with interpretable structure from rnn. *arXiv preprint arXiv:1810.10708*, 2018.

Iyer, S., Dandekar, N., , and Csernai, K. 2018. Quora question pairs.

Kim, Y. Convolutional neural networks for sentence classification. In *Empirical Methods in Natural Language Processing*, pp. 1746–1751, 2014.

Kindermans, P.-J., Schütt, K. T., Alber, M., Müller, K.-R., Erhan, D., Kim, B., and Dähne, S. Learning how to explain neural networks: Patternnet and patternattribution. *arXiv preprint arXiv:1705.05598*, 2017.

Kinney, J. B. and Atwal, G. S. Equitability, mutual information, and the maximal information coefficient. *National Academy of Sciences*, pp. 201309933, 2014.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. *International Conference on Machine Learning*, 2017.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., Van Kleef, P., Auer, S., et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

Li, J., Chen, X., Hovy, E., and Jurafsky, D. Visualizing and understanding neural models in nlp. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2015.

Liu, M., Shi, J., Li, Z., Li, C., Zhu, J., and Liu, S. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2017.

Lundberg, S. M. and Lee, S.-I. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

Mahendran, A. and Vedaldi, A. Understanding deep image representations by inverting them. In *IEEE conference on computer vision and pattern recognition*, pp. 5188–5196, 2015.

Olah, C., Mordvintsev, A., and Schubert, L. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

Peake, G. and Wang, J. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 2060–2069, 2018.

Ribeiro, M. T., Singh, S., and Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In *ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pp. 3856–3866, 2017.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision*, pp. 618–626. IEEE, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Socher, R., Bengio, Y., and Manning, C. D. Deep learning for nlp (without magic). In *Tutorial Abstracts of ACL 2012*, pp. 5–5. Association for Computational Linguistics, 2012.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing*, pp. 1631–1642, 2013.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328, 2017.

Tenney, I., Xia, P., Chen, B., Wang, A., Poliak, A., McCoy, R. T., Kim, N., Durme, B. V., Bowman, S., Das, D., and Pavlick, E. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference for Learning Representations*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., and Nair, V. N. Explainable neural networks based on additive index models. *arXiv preprint arXiv:1806.01933*, 2018.

Wang, Z., Wang, H., Wen, J.-R., and Xiao, Y. An inference approach to basic level of categorization. In *acm international on conference on information and knowledge management*, pp. 653–662. ACM, 2015.

Warstadt, A., Singh, A., and Bowman, S. 2018. Corpus of linguistic acceptability.

Wu, W., Li, H., Wang, H., and Zhu, K. Q. Probase: A probabilistic taxonomy for text understanding. In *ACM SIGMOD International Conference on Management of Data*, pp. 481–492. ACM, 2012.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

Zhang, Q., Cao, R., Wu, Y. N., and Zhu, S.-C. Growing interpretable part graphs on convnets via multi-shot learning. In *AAAI Conference on Artificial Intelligence*, 2017.

Zhang, Q., Cao, R., Shi, F., Wu, Y. N., and Zhu, S.-C. Interpreting cnn knowledge via an explanatory graph. In *AAAI Conference on Artificial Intelligence*, 2018a.

Zhang, Q., Nian Wu, Y., and Zhu, S.-C. Interpretable convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8827–8836, 2018b.

Zhang, Q., Wang, W., and Zhu, S.-C. Examining cnn representations with respect to dataset bias. In *AAAI Conference on Artificial Intelligence*, 2018c.

Zhang, Q., Yang, Y., Liu, Y., Wu, Y. N., and Zhu, S.-C. Unsupervised learning of neural networks to explain neural networks. In *arXiv:1805.07468*, 2018d.

Zhang, Q., Yang, Y., Ma, H., and Wu, Y. N. Interpreting cnns via decision trees. In *IEEE conference on computer vision and pattern recognition*, 2019.

Towards A Deep and Unified Understanding of Deep Neural Models in NLP: Supplementary Materials

Chaoyu Guan [*2] **Xiting Wang** [*2] **Quanshi Zhang** [1] **Runjin Chen** [1] **Di He** [3] **Xing Xie** [2]

1. Proof of $H(\mathbf{X}|\mathbf{s}) = \sum_i H_i$

$$H(\mathbf{X}|\mathbf{s}) = -\mathbb{E}_{\mathbf{x}}\big[\log p(\mathbf{x}|\mathbf{s})\big] = -\mathbb{E}_{\mathbf{x}}\big[\log \prod_i p(\mathbf{x}_i|\mathbf{s})\big] = \sum_i -\mathbb{E}_{\mathbf{x}}\big[\log p(\mathbf{x}_i|\mathbf{s})\big] = \sum_i H_i$$

2. Proof of the Maximum Likelihood Estimation of $\{\sigma_1, \ldots, \sigma_n\}$

We can roughly assume $p(\tilde{\mathbf{x}}|\mathbf{s}) \approx p(\tilde{\mathbf{s}}|\mathbf{s})$ and $\tilde{\mathbf{s}}|\mathbf{s} \sim \mathcal{N}(\boldsymbol{\mu} = \mathbf{s}, \boldsymbol{\Sigma} = \sigma_{\mathbf{s}}^2\mathbf{I})$ follows a Gaussian distribution, where we define $\tilde{\mathbf{s}} = \Phi(\tilde{\mathbf{x}}) \in \mathbb{R}^d$. Thus, we get

$$p(\tilde{\mathbf{x}}|\mathbf{s}) \approx p(\tilde{\mathbf{s}}|\mathbf{s}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} \exp\big[-\frac{1}{2}(\tilde{\mathbf{s}}-\mathbf{s})^T\boldsymbol{\Sigma}^{-1}(\tilde{\mathbf{s}}-\mathbf{s})\big]$$

Therefore, we obtain

$$\begin{aligned}
&\text{argmax}_{\{\sigma_1,\ldots,\sigma_n\}} \log \prod_{\substack{\tilde{\mathbf{x}}_i=\mathbf{x}_i+\epsilon_i:\\ \epsilon_i\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_i)}} p(\tilde{\mathbf{x}}|\mathbf{s}) \\
\approx &\text{argmax}_{\{\sigma_1,\ldots,\sigma_n\}} \sum_{\substack{\tilde{\mathbf{x}}_i=\mathbf{x}_i+\epsilon_i:\\ \epsilon_i\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_i)}} \Big\{-\log(\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}) - \frac{1}{2}(\tilde{\mathbf{s}}-\mathbf{s})^T\boldsymbol{\Sigma}^{-1}(\tilde{\mathbf{s}}-\mathbf{s})\Big\} \\
= &\text{argmin}_{\{\sigma_1,\ldots,\sigma_n\}} \sum_{\substack{\tilde{\mathbf{x}}_i=\mathbf{x}_i+\epsilon_i:\\ \epsilon_i\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_i)}} \frac{\|\tilde{\mathbf{s}}-\mathbf{s}\|^2}{2\sigma_{\mathbf{s}}^{2d}} \\
= &\text{argmin}_{\{\sigma_1,\ldots,\sigma_n\}} \sum_{\substack{\tilde{\mathbf{x}}_i=\mathbf{x}_i+\epsilon_i:\\ \epsilon_i\sim\mathcal{N}(\mathbf{0},\boldsymbol{\Sigma}_i)}} \|\tilde{\mathbf{s}}-\mathbf{s}\|^2
\end{aligned}$$

In this way, we can consider the minimization of $\|\tilde{\mathbf{s}}-\mathbf{s}\|^2$ as the MLE of $\{\sigma_1, \ldots, \sigma_n\}$.

3. Intuitive Explanation about Loss function

In this section, we provide an intuitive explanation to help understand our loss function:

$$L(\boldsymbol{\sigma}) = \mathbb{E}_{\boldsymbol{\epsilon}}\|\Phi(\tilde{\mathbf{x}}) - \mathbf{s}\|^2 - \lambda\sum_{i=1}^{n} H(\tilde{\mathbf{X}}_i|\mathbf{s})|_{\boldsymbol{\epsilon}_i\sim\mathcal{N}(\mathbf{0},\sigma_i^2\mathbf{I})} \tag{1}$$

[*]Equal contribution [1]John Hopcroft Center and the MoE Key Lab of Artificial Intelligence, AI Institute, at the Shanghai Jiao Tong University, Shanghai, China [2]Microsoft Research Asia, Beijing, China [3]Peking University, Beijing, China. Correspondence to: Quanshi Zhang <zqs1022@sjtu.edu.cn>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

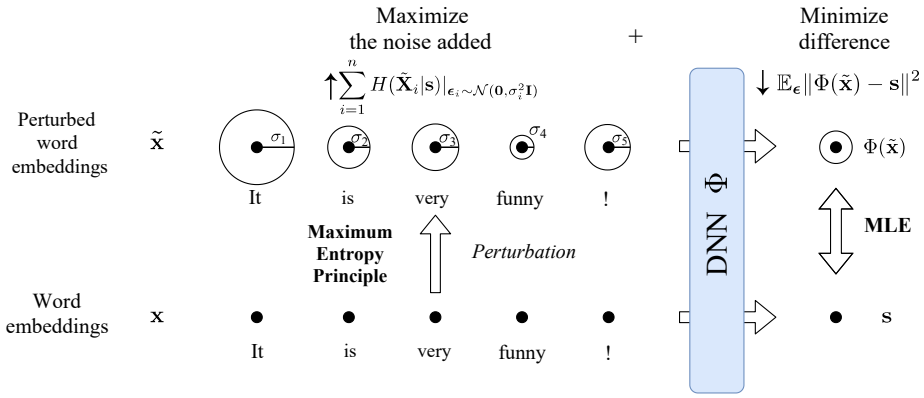


Figure 1. An example for understanding our loss function $L(\boldsymbol{\sigma})$. $\mathbf{x}$ represents the input word embeddings and $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$ denotes the perturbed word embeddings. The distribution of noise $\boldsymbol{\epsilon}$ can be characterized by its standard deviation $\boldsymbol{\sigma} = [\sigma_1, ...\sigma_5]$, where $\sigma_i \in \mathbb{R}, \forall i$. The left part, which maximizes the conditional entropy $H(\tilde{\mathbf{X}}_i|\mathbf{s}) = \frac{K}{2}\log(2\pi e) + K\log\sigma_i$, tries to add as much as noise (enlarge σ_i) to each word as possible. At the same time, the right part minimizes the difference between perturbed results $\Phi(\tilde{\mathbf{x}})$ and original hidden state $\mathbf{s}$. As a result, if an embedding of the i-th word can change a lot without impacting the hidden states, σ_i will be large (e.g., σ_1 for word *It*). If a word is important (e.g., funny), its σ_i will be small.

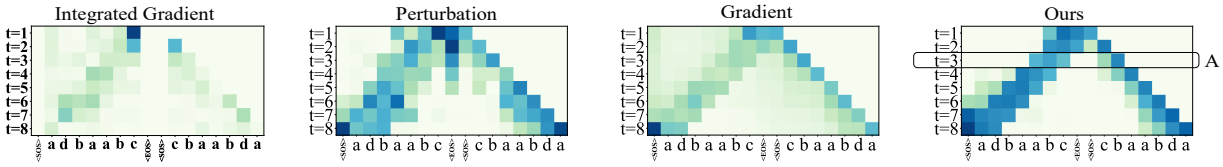


Figure 2. Saliency maps at different timestamps comparing with three baselines. The model we analyze learns to reverse sequences. Our method shows a clear reverse pattern. Other methods also reveal this pattern, although not as clear as ours.

We illustrate the two terms of our loss (Maximum Entropy and MLE) in Fig. 1. Given sentence *It is very funny!*, $\mathbf{x}$ represents its input word embeddings and $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\epsilon}$ denotes the perturbed word embeddings. The distribution of noise $\boldsymbol{\epsilon}$ can be characterized by its standard deviation $\boldsymbol{\sigma} = [\sigma_1, ...\sigma_5]$, where $\sigma_i \in \mathbb{R}, \forall i$. The second term of the loss corresponds to the left part of the figure, which aims at maximizing the conditional entropy $H(\tilde{\mathbf{X}}_i|\mathbf{s}) = \frac{K}{2}\log(2\pi e) + K\log\sigma_i$, tries to add as much as noise (enlarge σ_i) to each word as possible. The first term $\mathbb{E}_{\boldsymbol{\epsilon}}\|\Phi(\tilde{\mathbf{x}}) - \mathbf{s}\|^2$ (MLE) corresponds to the right part of the figure and minimizes the difference between perturbed results $\Phi(\tilde{\mathbf{x}})$ and original hidden state $\mathbf{s}$. As a result, if an embedding of the i-th word can change a lot without impacting the hidden states, σ_i will be large (e.g., σ_1 for word *It*). If a word is important, its σ_i will be small. In the example of Fig. 1, the information of word *funny* is largely kept (σ_4 is small), which means *funny* is important to hidden state $\mathbf{s}$. The information of other stop words like *It*, *very* is largely discarded (corresponding σ_i is large), which means hidden state $\mathbf{s}$ does not utilize much information of these words.

4. Results of Baseline (Sundararajan et al., 2017)

In this section, we show results of a more advanced method, Integrated Gradient (Sundararajan et al., 2017). It is based on gradient and LRP method, and also suffers from coherent issues because of their heuristic assumptions. Following Sec. 4, we use it for comparing different **timestamps**, **layers** and **models**. All the results bellow use the same experiment settings (models and examples) in Sec. 4.

Fig. 2 shows the comparison of hidden states in different timestamps. The Integrated Gradient method also shows a reverse pattern, but not as clear as other methods.

Fig. 3 shows the comparison of hidden states in different layers. The Integrated Gradient method fails to give coherent results in this experiment setting, just like other baselines.

Fig. 4 shows the comparison of hidden states in different models. The Integrated Gradient method fails to give clearer patterns, just like gradient method in Sec. 4.3.

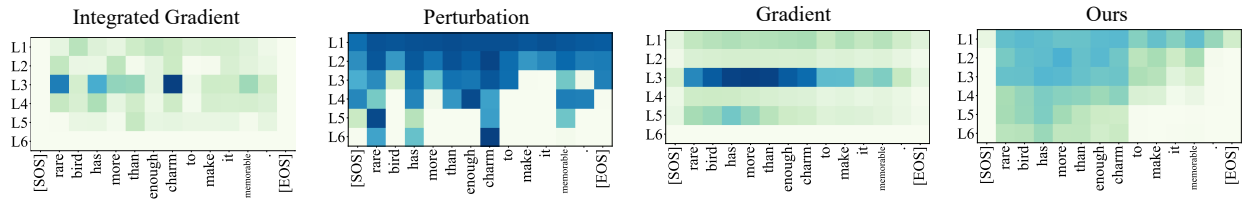


Figure 3. Saliency maps of different layers comparing with three baselines. Our method shows how information decreases through layers.

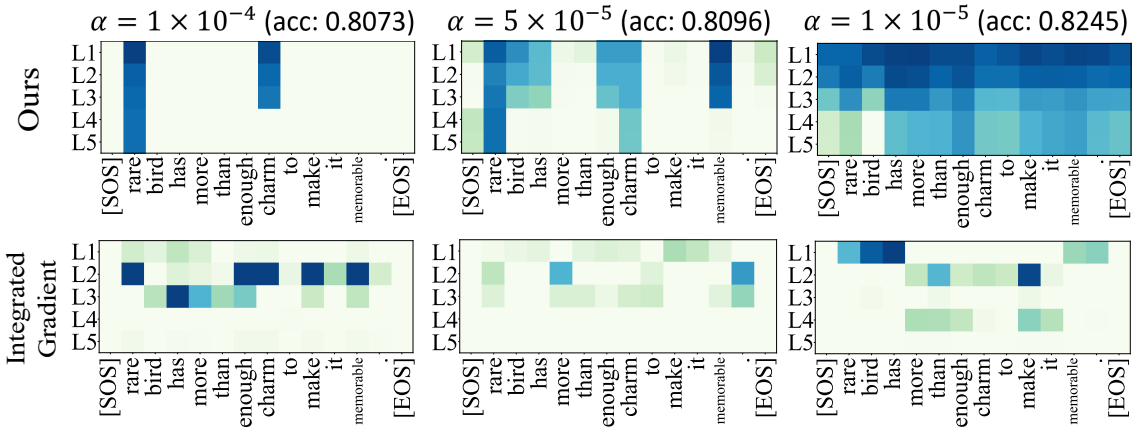


Figure 4. Saliency maps for models with different hyperparameters. Here, α refers to the weight of the regularization term.

In conclusion, Integrated Gradient method also suffers from coherent problems.

References

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3319–3328, 2017.