Appunti sul paper *Self-Explaining Structures Improve NLP Models*

A.A. 2023/2024

Rosso Carlo

Contents

1 Introduction

Existing approaches to explaining deep learning models in NLP usually suffer from two major drawbacks:

- the main model and the explaining model are decoupled, thus existing explaining tools are not self-explainable;

- the probing model is only able to explain a model's predictions by operating on low-level features;

We propose a simple yet general and effective self-explianing framework fro deep learning models in NLP. The key idea is to put an additional layer, *interpretation layer*, on top of any existing NLP model. This layer aggregates the information for each text span, which is then associated with a specific weight, and their weighted combination is fed to the softmax function for the final prediction.
The proposed model comes with the following merits:

- span weights make the model self-explainable;

- the proposed model is general;

- the weight associated with each span provides direct importance scores.

We, for the first time, show that interpretability does not come at the cost of performance.

A long term criticism against deep learning models is the lack of interpretability. The black-box nature of neural models limits the scope of applications of deep learning models, and also hinders model behavior analysis and error analysis.
Therefore we rise the following question: what makes a good interpretation method for NLP?

- the method should be self-explainable;

- the model should offer precise and clear saliency scores for any level of text units;

- the self-explainable nature does not trade off performances.

The key point is to put an additional layer on top of any existing NLP model and this layer aggregates the information for all (i.e. $O(n^2)$) text spans. Therefore, no additional information is incorporated, nor any information is lost at the interpretation layer.

2 Related Work

In NLP, approaches to interpret naural models include extracting pieces of input text called "rationales" as justifications to model predictions.
Using attentions as a tool of model interpretation, someone visualized attention heatmaps to understand how natural language inference models build interactions between two sentences. However, some works show the highly inconsistency between attentions and predictors, and suggest that attentions should not be treated as justification for a decision.

Saliency methods are widely used in computer vision and NLP for model interpretation. The key idea is to find the salient features responsible for a model's prediction.

In the context of NLP, somone used saliency maps to identify and extract task-specific salient sentences from documents to maximally preserve document topics and semantics. Someone crafted white-box adversarial examples to fine the most salinet text-editing operations to trick models by computing derivatives.

The proposed framework does not require a surrogate model.

3 Model

3.1 Notation

Given an input sequence $x = \{x_1, \ldots, x_n\}$, where n denotes the length of x. Let $x(i, j) = \{x_i, x_{i+1}, \ldots, x_{j-1}, x_j\}$. We wish to predict the label y for x based on $p(y|x)$. Let v denote vocabulary size, and word representations are stored in $W \in \mathbb{R}^{v \times d}$. The input x is associated with the label $y \in Y$.

3.2 Architecture

Input Layer

The input layer for the proposed model consists of the stack of word representations for words in the input sequence, where x_t is represented as a d-dimensional vector $W_{[x_t,:]}$.

Intermediate Layer

On the top of the input layer are the k encoder stacks, where each stack consists of multi-head attentions, layer normalization and residual connections. The representation for i-th layer at position t is denoted by $h_t^i \in \mathbb{R}^{1 \times d}$.

Span Infor Collecting Layer (SIC)

For an arbitrary test span $x(i, j)$, we first obrain a dense representation $h(i, j)$ of the span:

$$h(i, j) = F(h_i^k, h_j^k) \tag{3.1}$$

where F denotes the mapping function, which we will detail soon. The strategy of using the first and last representations to represent the whole span is inspired by some recent works. The SIC layer iterates over all text spans, and collects $h(i, j)$ for all $i, j \in [1, n]$, with the complexity being $O(n^2)$.

Interpretation Layer

The interpretation layer aggregates information from all text spans $h(i, j)$:

$$o(i,j) = \hat{h}^T h(i,j)$$
$$\alpha(i,j) = \frac{\exp(o(i,j))}{\sum_{i\in[1,n],j\in[i,n]} \exp(o(i,j))}$$
$$\tilde{h} = \sum_{i\in[1,n],j\in[i,n]} \alpha(i,j) h(i,j)$$

where $\hat{h} \in \mathbb{R}^{1\times d}$ is a learnable parameter, and $\tilde{h} \in \mathbb{R}^{1\times d}$ is the aggregated representation of all text spans.

Output Layer

The output layer is the probability distribution over labels using the softmax function:

$$p(y|x) = \frac{u_y^T \tilde{h}}{\sum_{y\in Y} u_y^T \tilde{h}} \tag{3.2}$$

where $u_y \in \mathbb{R}^{1\times d}$ is a learnable parameter. As can be seen, $\alpha(i,j)$ is the weight associated with the span $x(i,j)$, thus indicates the importance of the text span to the final prediction.

Efficient Computing

Towards efficient computations, we propose that F takes the following form:

$$F(h_i, h_j) = \tanh[W(h_i, h_j, h_i - h_j, h_i \odot h_j)] \tag{3.3}$$

where $W = [W_1, W_2, W_3, W_4] \in \mathbb{R}^{d\times 4d}$, and W_i are learnable parameters. The complexity of computing $F(h_i, h_j)$ is $O(n^2 d)$.

3.3 Training

The training objective is the standard cross entropy loss. An additional regularization on α is needed:

$$\mathcal{L} = \log p(y|x) + \lambda \sum_{i,j} \alpha^2(i,j) \tag{3.4}$$

With the constraint $\sum_{i,j} \alpha(i,j) = 1$. This is done because we want the model to focus on a very small number of text spans, thus we wanto the distribution of α to be sharp.

4 Evaluation

We want an evaluation method that is flexible upder any task and dataset, and the extracted rationales ought to semantically influence the model's prediction for the same instance.

Intuitively, if extracted rationales can faithfully represent (be equivalent to) their corresponding text inputs with respect to model predictions the following should hold:

1. a model trained on the original inputs should perform comparably well when tested on the extracted rationales;

2. a model trained on the extracted rationales should perform comparably well when tested on the original inputs;

3. a model trained on the extracted rationales should perform comparably well when tested on other extracted rationales.

We use *full* to refer to the situation of training or testing a model on the original texts, and *span* to refer to the situation of training or testing a model on the extracted rationales. By denoting the original full dataset by $D_{\text{train}}, D_{\text{dev}}, D_{\text{test}}$, and the newly constructed rationale-based dataset by $D'_{\text{train}}, D'_{\text{dev}}, D'_{\text{test}}$, the settings are as follows:

- **FullTrain** $\rightarrow$ **SpanTest**: train a model on D_{train} and test it on D'_{test};

- **SpanTrain** $\rightarrow$ **FullTest**: train a model on D'_{train} and test it on D_{test};

- **SpanTrain** $\rightarrow$ **SpanTest**: train a model on D'_{train} and test it on D'_{test}.

Note that this system is not perfect: the system can be gamed if the extracted plan is just the same as the original span.