Ronak, Xinyu, Peter, Maya
Instructor: Noah Smith

CSE 517: Natural Language Processing
Winter 2021

1 Weight-Of-Evidence Attribution Score for Classification

Let $x = (x_1, ..., x_n)$ be a sentence with vocabulary $\mathcal{V}$ and y be a class label. Let $p_\theta(y|x)$ be a model that predicts probability of class labels given an input. Define

$$p_\theta^{[i]}(y|x) = \sum_{w \in \mathcal{V}} p(y|x_{1:i-1}, w, x_{i+1:T}) \cdot p(w \mid x_{1:i-1}, x_{i+1:T})$$

This can also be written as

$$\log p_\theta^{[i]}(y|x) = \text{LogSumExp}\left([\log p(y|x_{1:i-1}, w, x_{i+1:T}) + \log p(w \mid x_{1:i-1}, x_{i+1:T})]_{w \in \mathcal{V}}\right)$$

The probability $p(w \mid x_{1:i-1}, x_{i+1:T})$ can be given by masking token x_i, and calling the masked-language-model of BERT. The log-probability $\log p(y|x_{1:i-1}, w, x_{i+1:T})$ can be computed in parallel by generating a $|\mathcal{V}|$-sized batch by duplicating x replacing each instance of x_i with different w's. Technically, one can compute this value for all elements in the sequence by making n batches and concatenating them together. This would push through $n \cdot |\mathcal{V}|$ sequences through the model, which might be the best way to do things if we have access to more GPU compute. The log-odds ratio is given by

$$\begin{aligned}\text{logodds}\left(p_\theta(y|x)\right) &= \log \frac{p_\theta(y|x)}{\log(1 - p_\theta(y|x))} \\ &= \log p_\theta(y|x) - \log(1 - \exp \log p_\theta(y|x))\end{aligned}$$

This is exactly given by the logits of the $p_\theta(y|x)$ model indexed by y. While $\text{logodds}\left(p_\theta^{[i]}(x)\right)$, I don't think we can get this number in terms of the logits of any of the models, other than going through the whole computation. Finally, the weight-of-evidence attribution score a is given by

$$a^{[i]}(x) = \text{logodds}\left(p_\theta(x)\right) - \text{logodds}\left(p_\theta^{[i]}(x)\right)$$

1.1 Extension to Language Modeling

Assume access to $\log p_\theta(x) = g_\theta(x)$, where g_θ is GPT-2.

$$\begin{aligned}\log p_\theta(x) &= \sum_{t=1}^{T} \log p_\theta(x_t \mid x_{1:t-1}) \\ \text{logodds}\left(p_\theta(x)\right) &= \log \frac{p_\theta(x)}{\log(1 - p_\theta(x))} \\ &= \log p_\theta(x) - \log(1 - \exp \log p_\theta(x))\end{aligned}$$

We also have access to the logits of each call of the model, i.e.

$$\begin{aligned}z_t &= \text{logodds}\left(p_\theta(x_t \mid x_{1:t-1})\right) \\ &= \log p_\theta(x_t \mid x_{1:t-1}) - \log(1 - \exp \log p_\theta(x_t \mid x_{1:t-1})),\end{aligned}$$

but it's not super clear to me that that is helpful. Say we marginalize this sentence over the token at index i, and call the marginal probability $p_\theta^{[i]}(x)$. Assume $p(w|x_{1:i-1}, x_{i+1:T})$ is given be the masked language model (MLM) of BERT, whereas $p(x_{1:i-1}, w, x_{i+1:T})$ can be computed by GPT-2 (by replacing the i-th token of x with word w).

$$\begin{aligned}
p_\theta^{[i]}(x) &= \sum_{w \in \mathcal{V}} p(x_{1:i-1}, w, x_{i+1:T}) \cdot p(w \mid x_{1:i-1}, x_{i+1:T}) \\
\log p_\theta^{[i]}(x) &= \log \sum_{w \in \mathcal{V}} p(x_{1:i-1}, w, x_{i+1:T}) \cdot p(w \mid x_{1:i-1}, x_{i+1:T}) \\
&= \log \sum_{w \in \mathcal{V}} \exp \left\{ \log p(x_{1:i-1}, w, x_{i+1:T}) + \log p(w \mid x_{1:i-1}, x_{i+1:T}) \right\} \\
&= \text{LogSumExp} \left([\log p(x_{1:i-1}, w, x_{i+1:T}) + \log p(w \mid x_{1:i-1}, x_{i+1:T})]_{w \in \mathcal{V}} \right)
\end{aligned}$$

The term inside the "LogSumExp" function is the vector generated by computing the term for each $w \in \mathcal{V}$. Hopefully that can be done in parallel, and without major numerical issues. Once this is computed, we can compute

$$\text{logodds}\left(p_\theta^{[i]}(x)\right) = \log p_\theta^{[i]}(x) - \log(1 - \exp \log p_\theta^{[i]}(x))$$

Finally, the attribution score a is given by

$$a^{[i]}(x) = \text{logodds}\left(p_\theta(x)\right) - \text{logodds}\left(p_\theta^{[i]}(x)\right)$$

Now, the last thing to check is if there is a clean way to compute $[\log p(x_{1:i-1}, w, x_{i+1:T})]_{w \in \mathcal{V}}$ and $[\log p(w \mid x_{1:i-1}, x_{i+1:T})]_{w \in \mathcal{V}}$. For the first term, we could in theory just evaluate the language model at all words in the vocabulary for all indices in the sequence. Because there are T scores, $|\mathcal{V}|$ words, and the model requires T calls to get the probability of a sequence, this would take $O(|\mathcal{V}|T^2)$ which is really nasty because Wikitext is a character-level dataset. If $i = 1$, then every call $\log p(x_t \mid w, x_{2:t-1})$ will depend on the w and have to be re-computed. One fact that we can take advantage of is that, for example, $\log p(x_2 \mid x_1)$ is a repeated call for all attribution scores greater than $a^{[i]}(x)$ for $i > 2$. This can be cached. Is there any benefit to caching calls such as $\log p(x_t \mid w, x_{2:t-1})$? I don't believe they would be used again. It seems we cannot beat the $O(|\mathcal{V}|T^2)$, but we can save half of the calls which could be useful.

As for the $[\log p(w \mid x_{1:i-1}, x_{i+1:T})]_{w \in \mathcal{V}}$ term, this is essentially masking each word and computing the distribution on the value of the masked word. Not much insight here either, other than the fact that the sequence can be converted into a batch of T sequences, each with a word masked, and can be pushed through in $O(1)$ (assuming the work is well-scheduled on the GPU). This becomes a little difficult when a batch of sentences are given to score originally, but we will ignore this unless it becomes a problem.

2 Deletion Curves and AUC for Classification

Let $x_{(1)}, ..., x_{(n)}$ be the tokens of sentence, ordered such that

$$a^{[(n)]}(x) \geq a^{[(n-1)]}(x) \geq \cdots \geq a^{[(1)]}(x).$$

That is, $x_{(n)}$ has the largest attribution score and $x_{(1)}$ the smallest. Let

$$x^{[i]} = x \text{ with } x_{(1)}, x_{(2)}, ..., x_{(i)} \text{ replaced with } \texttt{<PAD>}$$

Let $\hat{y} = \arg\max_{\,y} p_\theta(y|x)$. The deletion curve for model $p_\theta(\hat{y}|x^{[i]})$ against $i = 1, ..., n$. The curve is meant to drop rapidly to indicate that the attribution scores are faithful.

2.1 Extension to Language Modeling

A general theme for extending the method to language modeling has been to replace $p_\theta(y|x)$ with $p_\theta(x)$ where possible. Consequently, one easy way to generalize the curve is to define $x^{[i]}$ similarly, and plot $p_\theta(x^{[i]})$ against $i = 1, ..., n$. Because all of these probabilities will be zero, it might make more sense to plot $\log p_\theta(x^{[i]})$. This is not as interpretable. To use a score more suited for language modeling we can plot perplexity-per-word

$$\text{PPW}(x^{[i]}) = 2^{-\frac{1}{n}\log_2 p_\theta(x^{[i]})}$$

against $i = 1, ..., n$. In this case, we should see the curve rapidly *increase* if the chosen attribution score is good, as it would start low and then the model would become more and more confused as we pad more important tokens. Computationally, there's not much to say here, as this only requires evaluating the log-probability of the model.