Decoding Layer Saliency in Language Transformers

Elizabeth M. Hou, Gregory Castanon

Email: elizabeth.hou@str.us **Web:** https://www.lizardintelligence.net/

This material is based upon work supported by the United States Air Force under Contract No. FA8650-19-C-6038 and FA8650-21-C-1168. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Air Force or Department of Defense.

BLUF

- **Problem:** Saliency (importance of words for a task) in natural language isn't local and transformer stack language models capture a lot of "other" information (language structure, syntax, etc)
- **Goal:** Provide better explainability through more meaningful, clear, decision-oriented representations from the information encoded in the hidden layers of an encoder based transformer stack
- **Method:** Assign explanatory power to tokens in an input sequence from layer specific saliency scores calculated using information only downstream from that layer
- **Algorithm:** Computationally efficient projection of a layer's saliency score using a pre-trained language model head as the mapping function

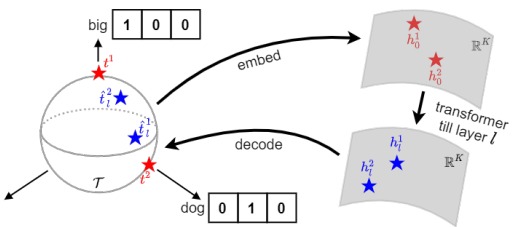
Interpreting Saliency for Intermediate Layers

- Let α_l^c be a $n \times K$ score matrix for layer l indicating the contribution of each element of h_l^k (K features of output of l-th transformer block) to the final classification decision c
- Need to project scores for layers $l > 0$ back into a space where elements of projected vector correspond directly to locations of input sequence tokens
- Language model head $f^{lm}(\cdot)$ minimizes loss between output of transformer block and its closest possible token space representation

$$\arg\min_f \mathcal{L}(t, f(h_l))$$

  where $\mathcal{L}(\cdot)$ is a cross-entropy function, t is the original input sequence, and layer $l = L$ is the final transformer block in the stack
- Conjecture: $f^{lm}(\cdot)$ is estimating the map between $\mathbb{R}^K$ and $\mathcal{T}$ *in general* because the pre-training tasks are performed over an enormous corpus

Toy Example for Decoding Layer Outputs



1. Input sequence “big dog” tokenized into t^1 and t^2 that lie on unit hypersphere $\mathcal{T}$
2. Tokens are embedded into having K continuous features (h_0^1 and h_0^2)
3. Series of l transformer blocks applied to the embedded input sequence to produce h_l^1 and h_l^2
4. Use $f^{lm}(\cdot)$ to decode onto $\mathcal{T}$
5. Outputs ($\hat{t}_l^1$ and $\hat{t}_l^2$) can be interpreted as weighted combinations of original tokens t^1 and t^2

Calculating Layer Saliency Scores

(Grad-CAM as example score metric)

$$\alpha_l^c = \left[\ldots, \frac{\partial y^c}{\partial h_l^k} h_l^k, \ldots \right] \; \forall k = 1, \ldots, K$$

$$s_l^c = g(\hat{D}_l \alpha_l^c) = ReLU(\sum_{k=1}^{K} \hat{D}_l \alpha_l^{c,k})$$

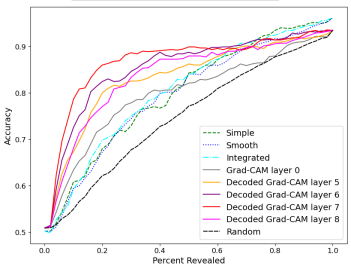
- y^c are predictions for class c
- $\hat{D}_l$ is $T \times n$ matrix: rows are columns of $\hat{P}_l = f^{lm}(h_l)$ corresponding to tokens in the input sequence
- s_l^c are saliency scores aggregated with function $g(\cdot)$ over all features for layer l

- By using $\hat{D}_l$ to project α_l^c back onto $\mathcal{T}$, elements of s_l^c correspond directly to contributions of each token in original input sequence to LM's final decision
- s_l^c capture *only* information in the model that is downstream from a specific layer l (controls amount of model information used)

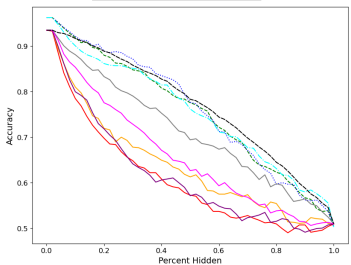
Experiments: SST-2 Dataset

Binary classification task - each sample is a sentence from a movie review labeled as either Negative or Positive sentiment

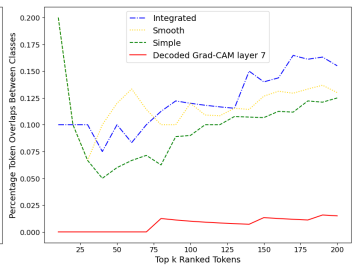
Revealing Game



Hiding Game



Token Overlap

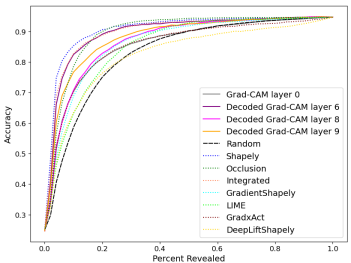


Experiments: SST-2 Dataset

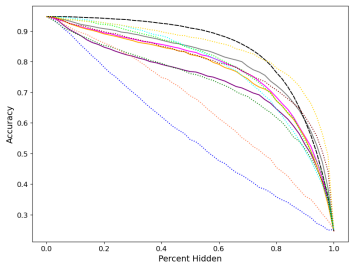
Experiments: AG News Dataset

Multiclass classification task - each sample is a sentence from a news article labeled as belonging to World, Sports, Business, or Sci/Tech topics

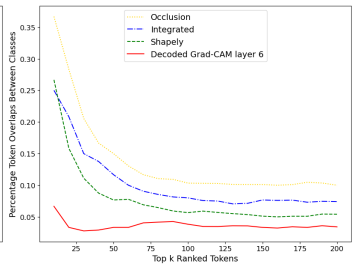
Revealing Game



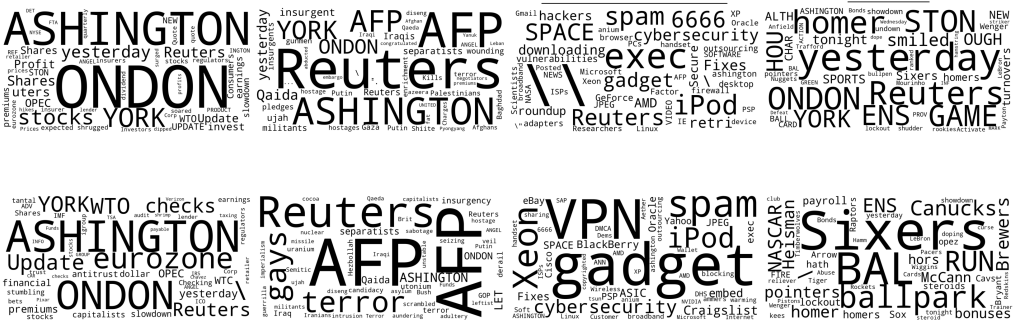
Hiding Game



Token Overlap



Experiments: AG News Dataset



Thanks!