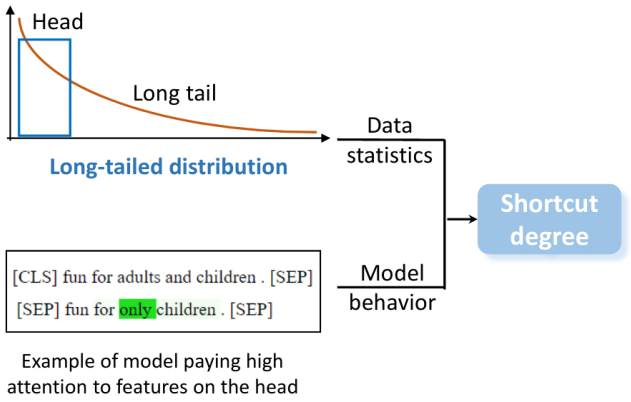
Towards Interpreting and Mitigating Shortcut Learning Behavior of NLU Models

arXiv: https://arxiv.org/abs/2103.06922

The standard training procedures cause models to utilize the simple features that reduce the training loss the most (simplicity bias), resulting in a low generalization of NLU models.

In NLU tasks, and given an input sentence pair x, the goal is to learn a mapping f(x) to predict the semantic relationship label y. In the training set, some words within x co-occur more frequently with one label than the others. Since all partitions are i.i.d, the model can achieve reasonable performance on the training, validation, and test splits, but with o.o.d data, the learned shortcuts are not present.



(a) long-tailed observation

Measurements

How can we measure the presence of such shortcut, we can define two types of measurement:

- Dataset statistics: based local mutual information (LMI) for each word w and label y, ie, LMI(w, y), to get describe the amount of information each work has. This results in a low-tailed distribution where the head of the distribution contains functional words with low information, while the tail contains words with high information but less frequent ones.

  $LMI(w, y) = p(w, y) \log(p(y|w)/p(y))$

  - $p(w, y)$ = count(w, y) / size(D),
  - $p(y|w)$ = count(w, y) / count(w),
  - $count(w, y)$ = the co-occurrence of word w with label y,
  - $|D|$ = the number of unique words in training set,
  - $count(w)$ = total number of words in the training set.

- Model behavior: to get the importance of each input work, an other method consists of using integrated gradients. For an input text composed of T works where each word has an embedding of d dimensions.

  $g\left(x_i\right)=\left(x_i-x_{\text{base}}\right) \cdot \sum_{k=1}^m \frac{\partial f_y\left(x_{\text{base}}+\frac{k}{m}\left(x_i-x_{\text{base}}\right)\right.}{\partial x_i} \cdot \frac{1}{m}$

  1- compute the gradients of the prediction with respect to individual word embeddings,

  2- using L2 norm, reduce the gradient over all embedding into a single value,

  3- using a baseline embedding x_{base} of zeros, apply this m steps resulting in a feature importance of length T for each input work

Observations

By using LMI and integrated gradients, we see that the NLU classifier are very strong superficial learners, using the head features of the long-tailed distribution while ignoring the rest.

To measure such degree of shortcut: compute the ration dataset examples where the most important words for a given prediction found using the integrated gradients, that are found in the head of the distribution (head are first 5% of the words)

	MNLI BERT-base			FEVER BERT-base		
#Words	Top 1	Top 2	Top 3	Top 1	Top 2	Top 3
Ratio	25.3%	51.3%	66.0%	10.8%	26.9%	31.44%

Table 1: The ratio of samples where top integrated gradient words locates on the head of the long-tailed distribution. We define the head as the 5% of all features. It indicates that NLU models overly exploit words that co-occur with class labels with high mutual information.

	MNLI BERT-base			FEVER BERT-base		
Subset	Entail	Contradiction	Neural	Support	Refute	Not_enough
Ratio	75.8%	94.6%	96.3%	99.4%	99.9%	83.8%

Table 2: The high ratio of samples where the word with the largest integrated gradient value is within the *hypothesis* branch of MNLI or the *claim* branch of FEVER, both of which are labelled by annotators and there are abundant of annotation artifacts.

Let u a measure of this behavior for a given sample x, if the most or second most important word found using integrated gradient falls in the top 5% in LMI distribution, set u to 1, 0 otherwise.

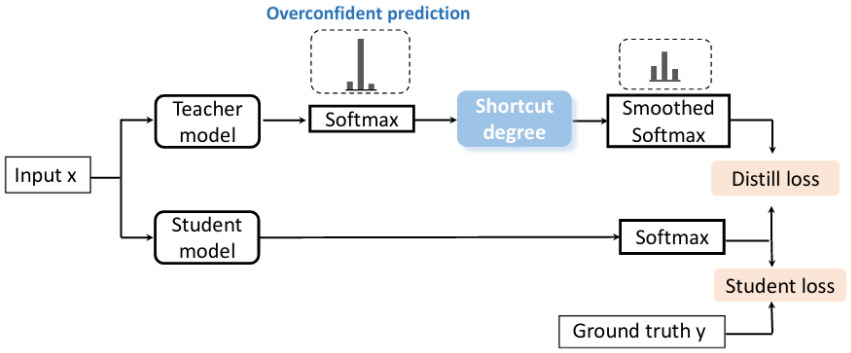
When are these shortcuts learned? what makes such shortcuts hard to deal with is that they are learned at the beginning of training, leading to a rapid drop of the loss function, while the rest of the features are gradually learned to further reduce the training loss. To measure this, compare the integrated gradients between two model span shots at different training iterations using cosine similarity, but only if the predictions at both time steps are the same, give 0 as a similarity otherwise. Let v be this similarity measure.

Finally, the shortcut measurement is $b = normalize(u + v)$ for each training examples x.

This measurement can be used to mitigate shortcut learning.

LTGR mitigation framework

First train a network using the standard setting, and then compute the shortcut measurement b for each training example. Then using the trained model as a teacher where the produced output are smoothed if the input contains a shortcut.



(b) Mitigation framework

Algorithm 1: LTGR mitigation framework.

Input: Training data $D = \{(x_i, y_i)\}_{i=1}^{N}$.

Set hyperparameters m, α.

while *first stage* **do**

  Train teacher network $f_T(x)$. Fix its parameters.

while *second stage* **do**

  Initialize the student network $f_S(x)$;

  Calculate shortcut degree b_i and softmax $\sigma(z_i^T)$ for each training sample $\{(x_i)\}_{i=1}^{N}$;

  Smoothing softmax:

  $s_{i,j} = \frac{\sigma(z_i^T)_j^{1-b_i}}{\sum_{k=1}^{K} \sigma(z_i^T)_k^{1-b_i}}$;

  Use Eq. 5 to train the student network $f_S(x)$

Output: Discard $f_T(x)$. Use $f_S(x)$ for prediction.

	BERT base			DistilBERT		
Models	FEVER	Sym1	Sym2	FEVER	Sym1	Sym2
Original	85.10	54.01	62.40	85.57	54.95	62.35
Reweighting	84.32	56.37	64.89	84.76	56.28	63.97
Product-of-expert	82.35	**58.09**	64.27	85.10	**56.82**	64.17
Order-changes	81.20	55.36	64.29	82.86	55.32	63.95
LTGR	**85.46**	57.88	**65.03**	**86.19**	56.49	**64.33**

Table 3: Generalization accuracy comparison (in percent) of LTGR with baselines for the FEVER task. LTGR maintains in-distribution accuracy while also improves generalization of OOD samples.

	BERT base			DistilBERT		
Models	MNLI	Hard	HANS	MNLI	Hard	HANS
Original	84.20	75.38	52.17	82.37	72.95	53.83
Reweighting	83.54	76.83	57.30	80.52	73.27	55.63
Product-of-expert	82.19	77.08	**58.57**	80.17	**74.37**	52.21
Order-changes	81.03	76.97	56.39	80.37	74.10	54.62
LTGR	**84.39**	**77.12**	58.03	**83.16**	73.63	**55.88**

Table 4: Generalization accuracy comparison (in percent) of our method with baselines for MNLI task. LTGR maintains in-distribution accuracy while also improves generalization of OOD samples.