0.1 Incremental Training

Most supervised text classifiers use a cross-entropy loss function to train. To compute the loss for a training example, a one-hot vector is used to represent the correct distribution. In some applications, categories overlap and the correct distribution may not put all of its mass on the highest probability class. Moreover, there may be some common, natural distributions in the data that don't concentrate mass on one class, but do have a consistent max index.

Our hypothesis is that if we can move model outputs incrementally so that for a given text input, model output gets closer to a natural distribution with the correct label, then we will end up with a model that is both more accurate, more robust and more generalizable that a single-shot supervised model.

Start with full training set T with n elements. Choose a number, k, of blocks = number of models to train. Train M_0 on $S_0 = n/k$ elements randomly selected from T (actually int floor of n/k). Train M_0 starting with a base transformer model and fine-tuning it in the normal way, which computes cross-entropy loss using one-hot vectors with '1' in the spot for the correct label. Then train student models $M_1, ...M_k$ using sample blocks, giving them teacher output, transforming as needed to get the probability for the correct label above a configured threshold. The intuition here is that the "correct" model output is a distribution and that distribution may have a decent amount of mass not on the correct answer. Also, certain kinds of distributions may be natural and common. Hypothesis: Pushing model output toward distributions that are common will result in more accurate and more robust models. only the mass assigned to the correct classification count As students get better, they learn to correctly generate these distributions.

Create training set T_1 as follows: Select $S_1 = n/5$ random elements from $T - S_0$. For each s in S_1, Let v_s be the full softmax array returned by M_0 when provided with $s['text']$ as text input. Instead of computing the cross-entropy loss using a one-hot array that concentrates all of the mass on the correct label, make proportional reductions in non-correct dimensions and increases in the correct one in v_s until the correct one is higher than a configurable threshold (.9 by default). (Optionally set a threshold below which a one-hot vector is used.)