Informal QA for "Detecting Local Insights from Global Labels: Supervised & Zero-Shot Sequence Labeling via a Convolutional Decomposition"

Allen Schmaltz
Department of Epidemiology
Harvard University
aschmaltz@hsph.harvard.edu

Informal QA. Last updated November 6, 2021.

1. If we have sufficient compute, should we add an explicit contrastive loss when training the K-NN model approximation (or for matching for auditing)?

The contrast, as it were, with contrastive losses is instructive for understanding how this particular model approximation fits into the larger universe of neural similarity models and dense matching.

For the particular use cases examined in this paper, using the K-NN to approximate the model, I am inclined to say no, you should not train with a contrastive loss for the *K-NN approximation*, ceteris paribus, for the following reasons. First, empirically the matches are already quite good as reflected in the model approximation being more likely to match the output of the original model (and in fact, also the ground truth) when the distance to the first match is comparatively low, and the magnitude of the K-NN output is high. Importantly, when actually looking at the matches as an end-user, they are reasonable when the distances are low and the K-NN output is high—to the extent that this mechanism is likely to be quite useful for analyzing datasets and having some qualitative sense of the model output. Obviously, there is some subjectivity there, but again note the direction of the quantitative signals which can be used to constrain the predictions.

The operative question then is what would the contrastive loss do in the ideal scenario (i.e., setting aside difficulties of training, such as mode collapse, for which in any case we now have some reasonably effective approaches)? Here, I am assuming this is unsupervised matching with respect to token predictions, possibly constrained by the document-level label (I will comment on the supervised case shortly), and that you are fine-tuning one or both of the exemplar representations and/or other layers of the network. Well, it is not going to help in the cases that are already good matches (i.e., the sign matches the original model and matches the ground truth). It may, however, flip some of the cases that were not good matches. It is possible that that could improve overall accuracy against the ground truth in some cases[1], which obviously could be

1 There is also a question to what extent converting a standard classification task into the retrieval-classification setting yields some benefits for OOD detection. (There are some other factors to consider, as well, such as whether the matching would be supervised, the overall effectiveness of the

useful for a given application, but the punchline is that there is a sense in which you would then have a different type of model than that trained with the original loss using the document- or token-level labels. In effect, you would have a bi-encoder. In such a case, you should train a model approximation over the bi-encoder. What I found in subsequent work (for the supervised case, actually with a joint bi- and cross-encoder used for traversing a search graph, but I suspect the behavior is similar for the unsupervised case in the basic setting) is that you can form the exemplar vector for the matches as the difference between the two applicable exemplar vectors, with which the auditing approach or K-NN approximation follows as before over those difference vectors. (However, importantly note that the unit of analysis then shifts to comparing to similar paired matches of features, rather than similar features.) In other words, the K-NN approximation is really only intended to reproduce the sign of the original model, and is intended to be a very lightweight model over the representations of the model. The intention is not that it would be used to greatly improve *in-domain* effectiveness against the ground truth (over whatever type of encoder/model it is approximating), although importantly in the cases I have looked at, it is consistently at least as strong as the original model, which allows us to use it as a proxy of the original model.

2. Can we use the CNN decomposition (i.e., the method for class-conditional feature detection) to back out alignments within one or more sequences?

Interestingly, no. We need a different inductive bias for alignment. We might wonder, for example, if we could just greedily minimize the distances of the exemplar vectors associated with each token against each other within a sequence. I looked into that, but found that the resulting alignments amounted to noise and did not yield clear patterns. Similarly, it also does not seem that it is possible to build up hierarchical structures by greedily minimizing the distances, for example.

In the case of fact verification (an NLI task), I found in subsequent work that we can train with a contrastive loss to minimize the difference between the max-pooled vectors from kernel-width-1 CNN's over a Transformer (as a bi-encoder and/or cross-encoder), and then just back-out alignments by tracing back the applicable tokens associated with each aligned filter. See that paper for additional details, but the point is a different inductive bias is needed for alignment. (It is also worth noting that there are indeed some real-world use cases where alignment is the end-goal, but for tasks such as NLI, the operative unit-of-analysis is really at the sequence level, rather than the word level. Thus with NLI, we are primarily interested in performing the exemplar auditing at the level of sequence matches. Nonetheless, alignment may be useful in those cases for debugging.)

3. Does this mean we get dense retrieval for free without actually training with some type of contrastive loss?

Unfortunately, no, at least not in the sense of retrieval you likely mean. This is sometimes a point of confusion. The approach in this paper is very effective at matching sim-

model on the task, compute considerations, etc. beyond the scope of this discussion.) See my subsequent retrieval-classification paper ("Coarse-to-Fine Memory Matching for Joint Retrieval and Classification") for an example, but the key point for the discussion here is that the exemplar auditing in that case is then taken over representations of the joint bi/cross-encoder, as opposed to learning those *matches of matches* with another contrastive loss.

ilar features in similar contexts. However, typically when we think of dense retrieval, it is in the context of some type of NLI-style task, such as fact verification, whereby we have some statement that we want to match to supporting evidence. As with alignment, we need a different inductive bias for that setting.

4. Are the discernible patterns associated with each *filter* of the CNN over the deep network?

Tell you what, I looked into that fairly extensively early on, but came to the conclusion that it was the wrong unit of analysis. With the grammatical error detection datasets, for example, there were some noisy patterns of a small number of filters tending to be associated with misspellings, for example. However, there was a long tail of filter activations that were difficult to make sense of in isolation. Further, manipulating individual filters turns out to not be particularly amenable as a lever for updatability. Even with a kernel-width-1 CNN and a linear layer, it turns out to be non-trivial to reason through how turning off a single filter will impact the predictions, since it will have a global impact. On the other hand, it is fairly straightforward for an end-user to make sense of other input examples, and if necessary, modify the labels associated with those examples. New architectures and methods may come along, but my general sense is that manually manipulating the actual activations of the deep networks is mostly a lost cause, since it is very difficult to reason about the side-effects. In stark contrast, *manipulation via examples (with dense matching as the bridge) is rather straightforward, and opens up some use cases that are not possible with simpler models*.

5. Instead of training the K-NN approximation to reproduce the sign of the original model, did you consider training against the true token-level labels?

First, note in the zero-shot sequence labeling setting we do not actually have access to token-level labels. Next, recall our use case here in which we are aiming to approximate the original model via a simple model over the representations and labels. If the goal is instead to significantly improve the in-domain effectiveness via matching (e.g., by essentially treating the support set as an external datastore with the hope of offloading some of the model capacity to the datastore), my general recommendation would be to consider converting the task into a search problem to which you can apply a bi+cross-encoder, as I have shown in subsequent work. See also Q. 1.

6. What if my fully-supervised sequence-labeling task does not correspond to "sparse" feature detection?

If the task involves predicting a multi-class label for every token, and especially if the sequences are also long, then you may want to drop the max-pool. In this scenario, the number of filters will typically need to be at least as long as the sequence (and probably some multiple thereof), otherwise non-selected (by the max-pool) tokens will have predictions only determined by the bias values of the linear layer. If it is computationally feasible (or you can switch to approximate search), you can just keep things as-is, but otherwise, if it is a fully-supervised setting, you may want to drop the max-pool to reduce the number of filters and by extension the search costs. I found in a recent use-case with very, very long sequences, for which we did not need sparse feature detection (since the task was fully-supervised), that the K-NN approximation behavior was as in this paper when dropping the max-pool.

7. What is the intuition for training the approximation with the BCE loss and then masking the loss for some epochs?

There is signal in the magnitude of the K-NN output, so we aim to optimize in the direction of minimizing the residuals. However, at the end of the day, we want to minimize the number of sign flips, so the masking is to avoid over-fitting to the magnitude of the outliers. Note that this differs from minimizing the squared residuals, for example, and ultimately is tied to the peculiar (and very non-homogenous) error distribution.

8. Is this specific to language, or could this be applied to images, as well?

These general behaviors/methods have subsequently held-up on other text datasets I have looked at, as well as on one non-text sequence modality (forthcoming). With each dataset, a decision has to be made as to how to model the relationship between the local level and the global prediction, and we now have a few options to select from that cover many use-cases. It is in the queue, but I have not yet specifically looked at images. The one difference with images is that it is perhaps less immediately clear what the analogue to tokenizing the input is, but I suspect that one of either deterministically tiling the input, and/or having a penultimate (or lower) layer that learns the relevant patches, would work. As a first pass, you might consider just using the recent Transformer architectures applied to images; presumably in that setting the methods I have proposed can then just be applied essentially as-is.

9. Can we use such feature detection to address the issue of spurious features, or so-called short-cut features?

There is a lot of interest in addressing "spurious features", but I find the problem, as presented, to often be under-specified, or at least, not addressed in a manner that would reflect a real-world scenario. Really what people mean when they are aiming to reduce dependence on spurious features is they want robustness over domain shifts. However, then we are right back to some of the issues addressed in this paper: We now have some new methods for screening the input under the model unlike that seen in training, but we have not really gained anything in terms of prediction robustness over domain shifts, per se, other things being equal.

It is important to test any approach for reducing spurious features by analyzing its domain-shift robustness, and in particular, to not conflate improving in-domain (zero-shot) sequence labeling with reducing dependence on spurious features. Improving zero-shot sequence labeling can be useful in its own right as an end-goal, but it is not sufficient evidence for reducing the impact of spurious features (and thus, improving robustness over domain shifts). For example, in the approach I show in the paper, one means of obtaining token-level labels is to simply train with a standard cross-entropy loss and then back-out the labels via the convolutional decomposition. Alternatively, we can add a sparsity constraint. In some cases, the resulting token-level labels can be rather different between those two options. Does that mean the sparsity constraint has reduced the impact of the spurious features? No. We may have improved our in-domain labeling effectiveness by biasing the predictions to be closer to the distribution of our ground-truth labels, but I have not seen any consistently convincing evidence that that leads to significantly improved robustness over domain shifts.

10. What if I need to fine-tune the underlying deep network to get good effectiveness for my given task?

In this paper, I do not fine-tune the Transformer, since it works well frozen on the given tasks. Based on my subsequent work, it seems to work well to fine-tune the Transformer and then freeze it and train the CNN. (In the subsequent retrieval-classification work, with the contrastive loss, I use a variation on this theme where I iteratively freeze either the Transformer or the CNN, since we need to perform retrieval during training.) The intuition against updating them together is that the high-capacity lower layers of the Transformer could learn to fully describe the task, while the CNN weights could essentially become an identity function. I would not be surprised if there is a particular learning schedule/etc. in which you can just update them together to similar effect, but I found the fine-tuning-then-freezing approach to be easy to implement and works well.

11. Are these approaches robust against proactively adversarial attacks?

I have not run any specific experiments yet, but I can only assume that the answer must be no, in general, as long as there is any error headroom in the model or matching, but I do not currently have a good sense of the degree relative to not having a matching component. With closer distances to the matches and agreement with the support set (and more so as the K of the K-NN increases), we can determine a relatively reliable slice of the test predictions. Among others, an interesting question to consider is whether adversarial attacks become easier/harder/unchanged when only considering these subsets deemed reliable, and whether or not having public access to the original training set (or the support set, more generally) impacts the difficulty of an attack.

12. Is this particular model approximation dependent on the use of Transformers, or will this work with other architectures?

I am not sure, as of writing. There would seem to be some complications, or at least some modifications needed, for pyramidal encoders, which in any case I have not seen used for classification recently, or any other type of classification architecture where there is not an output hidden state associated with each of the input tokens (or whatever the lowest resolution of interest is for the given application). In other words, architectures that are otherwise pre-trainable with masked-language-model-style losses (at the applicable lowest unit of analysis to which you aim to back out labels) seem to be the most straightforward variants to consider.

In terms of concrete near-term experiments on this front, it would be interesting to see if this behavior holds for the sparsely activated architectures (mixture-of-experts and variants thereof). If it does, that would potentially open up some interesting possibilities for dense matching over very large models trained on very large datasets over multiple input modalities and tasks.

13. How does the K-NN approximation compare to the recently proposed kernel-based approximations (e.g., operating in some type of gradient space/path kernel)?

It is an interesting question as to how the K-NN compares to kernel approximations and kernel machines[2], more generally, and what the theoretical and practical implications of those differences might be. At a very high-level, these ideas are similar in so far as they can be viewed as some type of explicit weighting over the training set, but obviously how they arrive at that weighting is quite different. Much more can be said on this, but here I will be brief. Keep in mind that I have already shown empirically, with the commonly used SOTA-yielding architecture (Transformer), that the K-NN approximation already has many (all?) of the practical properties (w.r.t. interpretability, uncertainty/reliability, updatability, etc.) one might reasonably hope to glean from an approximation of the original deep neural model, and it is clear now how to do this at varying resolutions of the input, importantly without harming the effectiveness against the ground-truth. Perhaps surprisingly, Euclidean space actually turns out to work quite well in this context. The approach I have proposed appears to potentially be architecture dependent, but nonetheless, is relatively straightforward to implement, all things considered. It remains to be seen if these practical characteristics will hold for the proposed kernel approximations, with otherwise similar deep networks.

14. How should we turn the signals for prediction reliability/uncertainty into a prediction bound/set/confidence interval/confidence band/etc.?

Dr. Danielle Rasooly and I examine the key factors to consider and introduce an approach in a forthcoming paper (coming soon).

15. Do Randomized Control Trials (RCTs), or observational studies analyzed with the various causal frameworks, obviate the need for such interpretability approaches?

No. By all means, run your RCTs with your neural application, but we need approaches to analyze the results to ensure certain sub-groups are not systemically (and unexpectedly) under-covered or subject to much lower model effectiveness. It is not so obvious how you do that without the approaches I have proposed. Keep in mind that when the distances are far and the magnitude of the K-NN output is low, I have observed cases in which the model effectiveness falls to chance. It will of course depend on the particular setting, but those slices of the data may not necessarily correspond to sub-groups to which we are otherwise aware. Additionally, at inference in higher-risk settings, we will want to severely constrain predictions to those similar to those actually seen in the RCT, throwing unusual cases to humans for further examination.

One last comment on this point: It will likely tend to be the case that the non-admitted predictions will be associated with clusters of the data that are less densely covered in the training sets, so some caution is needed to ensure that porting those decisions over to humans for adjudication does not itself introduce some systematic lower quality outcomes for the end-users associated with those instances.

2 See, e.g., "Every Model Learned by Gradient Descent Is Approximately a Kernel Machine" (Domingos 2020): https://arxiv.org/pdf/2012.00152.pdf.

16. How do these methods speak to the larger discussion about AI safety?

That is well beyond the scope of this paper, as well as the follow-up papers in this line of work, as there are so many factors to consider.

However, it is worth mentioning that discussions in the popular media, even with AI researchers, have not considered the implications of dense matching constraints against the data with known labels. In particular, there is now an inherent dichotomy that is likely to cause endless confusion and debate in coming years, but in fact, for practical, real-world settings, it seems likely to translate into simple empirical questions that can be tested for any given application. The dichotomy is that previously (without the approaches I have proposed) the large models did feel a bit shaky as they are brittle; we did not really have a good sense of when they would go off the rails; they can regurgitate data that was not well-curated; etc., so it was not an unreasonable concern to say maybe we should hold-off on building larger models and training over massive datasets and putting them into production. However, and it is a *big* however, once you throw dense matching into the mix, the calculus shifts. In this context, you probably want the model to be as large/expressive as possible, trained over as much data and as many tasks as possible. In this context, at inference, you will constrain to your well-curated sets. As such, you want the representations to reliably match to your well-curated sets, and to reliably match to the large space of non-curated data you aim to reject. Since we typically do not really otherwise have good parametric models of the subsets of the input space over which we would like to abstain from predicting, our best bet, given what we currently know, is likely to train with as much data as possible and place as much data as possible in the support set for matching.