DEMix Layers: Disentangling Domains for Modular Language Modeling

Main Idea: DEMix, which stands for domain-expert mixture, is a type of architecture that encourages domain specialization. It looks to train multiple feedforward networks that are each specialized in a specific domain, and similarly to MoE, pick one to run during inference, depending on the input space. DEMix layers are modular, meaning they can be mixed, added, removed or used to initialize other layers after initial training. DEMix aims to achieve domain specialization in the sparse layer, while retaining generalization knowledge with shared parameters.

Motivation

Dense training consists of updating all parameters to minimize loss on all the data. This means that it assumes that the model will be able to learn/fit different domains equally. In practice, domains are skewed to domains that are more prevalent in the training data, so models have a hard time generalizing to other domains. Fine-tuning these large dense models can also be expensive and lead to catastrophic forgetting – worsening performance on pre-training domains not represented in the fine-tuning data – since all weights need to be updated. Finally, managing unwanted behavior in dense models is also a challenge.

To help with these issues, a DEMix (modular) architecture is proposed. That is, an LLM with different components that can be modified during inference.

Some Characteristics/Highlights of DEMix

- DEMix is a substitute for an FF layer in the Transformers architecture (every FF layer is replaced by a DEMix layer) and can be conditioned on the input text in cases where the domain is previously known, as well as when the input domain is not known.
- The router used for DEMix is parameter-free and thus not learned, depending on the natural segmentation of the data.
    - Parameter-free probabilistic approach to dynamically estimate a weighted mixture of domains during inference, which is used for novel domains (when it is not clear/known in advance where the input is from, or it is from a brand-new domain).
- Mixing (like using top-k > 1) experts is shown to improve performance in novel domains as well as training domains during test time (probably due to overlap between domains that the shared parameters are not enough to capture).
- The modularity of DEMix offers flexibility by enabling the removal or addition of new domains at inference, thus allowing the ability of choosing what is forgotten. Catastrophic forgetting is also not an issue since a new domain expert can be initialized or an existing one can be further specialized without modifying the model's behavior on other domains.

Data

- 8 training domains
- 8 testing domains

    - Used to test robustness of mixing experts to data distribution shifts not seen during training.

DEMix vs Traditional MoE

- While in traditional MoE the routing function is learned through training at a token-level, DEMix routing is done at the document (sequence) level and only needs to be performed once per input (all tokens in an input sequence are routed the same way).
    - Token-level routing has been shown to specialize experts in token-level areas, such as semantics and syntax. Document-level routing should enable experts to specialize in specific tasks/domains.
- Because of this characteristic in specializing in domains compared to semantics, the experts are more flexible in terms of addiction and subtraction to the network and provide an ease of interpretation that traditional MoEs don't have (they are more of a black box).

Training

- During training, each domain expert is assigned to a single GPU (similarly to how it is done in traditional MoE).
- Each mini batch sends k domain examples to each expert (a balanced load is easy to achieve since we know each input's domain for training).

- Distributed data parallelism is used (expert is replicated through the number of GPUs available for that specific expert, since there were more GPUs available than experts)
    - This is efficient because only globally shared parameters are synced through all GPUs, while domain expert parameters are only synced between the GPUs assigned to that expert.
        - Reduced communication costs due to a decrease in alltoall computations.

Evaluation

- In-domain performance
    - 4 variations used:
        - DENSE – regular dense model with no conditioning on domain.
        - DENSE (balanced) – dense model with equal amount of data used for each domain.
        - +DOMAIN-TOKEN – variation of DENSE (balanced) with a prepended token on every input sequence to indicate its domain.
            - The motivation behind this is to add info about the domain of the input to the context to try to create a dense oracle gate.
        - DEMix – DEMix architecture with known domain for each input.
            - Uses top-1 routing for in-domain experts based on the already known domain of the input.

    - Adding domain info (DENSE (balanced) and +DOMAIN-TOKEN) is shown to help the dense baseline.
        - The smaller the model, the more helpful this is.
    - Heterogeneous domains (diverse domains like WEBTEXT and REDDIT) have more overlap with other training domains, and thus don't really benefit from DEMix vs a dense baseline.
- Unknown domain performance – mixing experts at inference time.
    - Routing approach
        - In practice, the domain of an input is not always known. In this case, it makes more sense to use a soft choice for routing (top-2 routing), as it was proposed for cases where the domain was known.
        - To not increase training costs with a learned routing approach (more communication costs), a probabilistic routing score based on Bayes' Rule was used (this is parameter-free).
        - Probabilistic Routing Score:
            - The main part of this is calculating the domain posterior – the probability that the input is from a certain domain d.
            - This approach is very inefficient (the input needs to go through each expert, so the routing is useless in practice) and is improved in future work.
        - They propose 3 variations on the posterior calculation:
            - Uniform - each domain is estimated to be equally likely.

        - Updating - weighted moving average of the posteriors from the previous timesteps.
        - Cached – fixed prior estimated from the test data (100 test sequences used)
    - The estimates of posteriors for both the training and the novel domains is shown to be sparse, justifying the top-1 and top-2 routing selections (so not all experts need to be used, aka sparsity is justified).
    - Ensembling DEMix experts (mixing) using the cached approach performs better than all models analyzed.
        - Compared to DENSE, this is beneficial at smaller scales, while the dense models can catch up as the parameter count increases.
            - Perhaps more data is needed when increasing the DEMix parameters?
        - Ensembling/mixing is also shown to lead to improvements on training domains, especially more heterogeneous ones (more diverse domains).

<u>DEMix-DAPT</u>

DEMix-DAPT consists of adopting existing experts to new domains.

- Previously, experiments were made to evaluate the performance of DEMix in novel domains (domains not seen during training). DEMix-DAPT is different in the sense that it applies new domain data to existing domain experts to create a new expert.

- The new expert is initialized with the parameters of the closest existing domain expert. So, the new expert is a fine-tuned version of an existing domain expert.
    - How close each domain expert is from each other is calculated from the router's domain posterior.
- In DEMix-DAPT, only the expert parameters are trained. Shared parameters are kept frozen.
- For inference, the cached posterior approach is taken.

Results (DEMix-DAPT)

- DEMix-DAPT is compared to Dense-DAPT, which is a dense version of adapting to a new domain.
- As expected, it is shown that Dense-DAPT suffers from the issue of catastrophic forgetting. This is apparent when seeing how training a Dense-DAPT model in a novel domain leads to degraded performance on (original) training domains.
- As expected, adding experts through DEMix-DAPT significantly improves performance on those novel domains.

In this paper, it was also shown how removing an expert from an unwanted domain (for example, due to hate speech or leaking of private data), leads to similar performance on that domain compared to DEMix models not trained on that domain. This shows that expert domains can be removed from DEMix, if desirable. This also shows that most domain specialization comes from the DEMix layers.