A Mixture Of Experts for Out-Of-Domain Expertise

Stanford CS224N Default Project RobustQA Track

Mia Sarojini Kynadi
Stanford Center For Professional Development
Stanford University
kynadi@stanford.edu

Abstract

Out of Domain question answering is still an open problem for NLP systems. In this project, I explore the use of a Mixture-of-Experts (MoE) model in order to better tackle out of domain questions, as explained in the paper by Jacobs et al [1]. The MoE model is trained on 3 in-domain datasets, finetuned and validated on a small subset of out of domain datasets, and finally tested on out of domain datasets. I have used Cantor Pairing as a means of feeding the start and end locations of the answer span into the gating mechanism of the Mixture of Experts model [1]. I have also used a custom loss function based on the mixture of Gaussians approach described in the Jacobs et al paper.

1 Key Information to include

- Mentor: Kamil Ali
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Although current large pretrained models perform very well on NLP tasks, one of the open issues is how to make a model generalize to datasets it hasn't seen during training. This problem is especially relevant because NLP models deployed in the real world will see a high number of queries that are from a distribution different from it's training dataset distribution.

Machine learning models are limited by the data that they see during training. The mixture of experts model attempts to improve out of domain performance by using separate models or experts which are trained on specific different datasets and generate an output by combining the models with a gating mechanism. The expert models form an associative network, and the gating mechanism can be viewed as a stochastic generator of output vectors that matches the distribution of the input vectors. To this end, for the specific problem of out of domain question answering, I use Cantor Pairing as a means of generating a single unique input from each expert to the gating mechanism corresponding to the start and end locations predicted by the expert. I also use a custom loss function that is described in the Jacobs paper [1] that encourages the gating mechanism to prioritise one expert for any given input question.

3 Related Work

The mixture of experts approach is used to improve generalisation to out of domain datasets for many tasks [2]. The original paper [1] describes the use of expert models trained on specific datasets, and combined using a gating mechanism. For the problem of robust question answering, I trained a DistilBERT model on each of the three in-domain datasets, and used an MLP classifier to gate or select the inputs from the classifier. The loss function I used encourages the classifier to assign a large probability (in effect, select) one of the experts to generate the answer to each query.

One challenge was generating a single input to the classifier from the start and end logits generated by the DistilBERT models, so that the classifier can generate a probability distribution over the 3 experts. To do this, I used Cantor Pairing of start and end locations, which generates one unique natural number for any unique pair of natural numbers. The MLP uses this as input and generates a probability distribution that selects the output from one of the expert models.

4 Approach

The MoE model I used is a combination of 3 models or 'experts'. Each expert is a pretrained DistilBERTforQuestionAnswering model as specified in the RobustQA default project handout. The three experts were trained as follows:

- One DistilBERT model was trained on the SQuAD dataset only [3].
- One DistilBERT model was trained on the NewsQA dataset only [4].
- One DistilBERT model was trained on the Natural Questions dataset only [5].

The Mixture-Of-Experts model is a PyTorch neural network that includes the 3 expert models. The output of these models is passed through a classifier (multi layer perceptron). The classifier consists of 3 linear layers, 2 ReLU layers, dropout with probability 0.01, and a SoftMax layer.

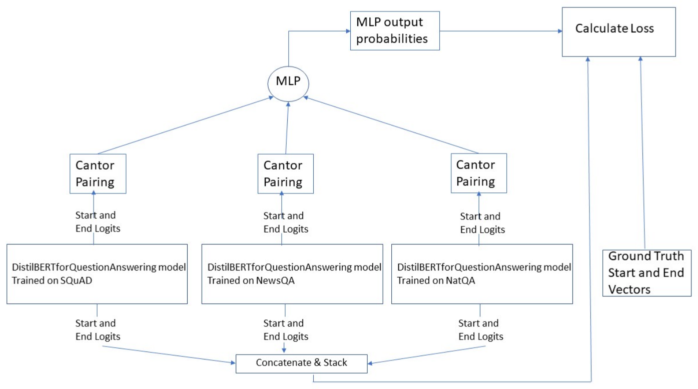


Figure 1: Model Architecture with 3 expert models and a gating mechanism to select output from a particular expert.

For the MLP to work as a gating mechanism, ideally we would need one input from each of the experts to the MLP. The DistilBERT model output is either the start and end logits of dimension 768, or the DistilBERT hidden states each of which have dimension 768. In order to generate a single input from the DistilBERT output, I used the start and end logits to get the most probable start and end locations of the answer span and then used cantor pairing to generate a single unique number from this specific start and end location pair. The cantor paired numbers from the three experts is fed to the MLP.

Cantor pairing is a primitive recursive pairing function

$$\pi : \mathbb{N} \to \mathbb{N}$$

defined by

$$\pi(x, y) = \frac{1}{2}(x + y)(x + y + 1) + y$$

The cantor pairing function is invertible, although I did not use the inversion in my model.

The output of the MLP is a probablity distribution over the expert models. These probabilities are used to compute the loss. The particular loss function used in this model is adopted from the Jacobs et al paper [1]. The loss for any given example c is defined as:

$$E^c = -log \sum_i p_i^c e^{-\frac{1}{2}||d^c - o_i^c||^2}$$

where E^c is the loss for the given example, p_i^c is the probability assigned by the MLP to the ith expert for this example, d^c is the desired target vector computed from the ground truth start and end locations of the answer span and o_i^c is the output vector of the ith expert for this example. The start and end logits output from each of the expert models is passed through a SoftMax layer and concatenated to get o_i^c. The ground truth start and end locations are converted to one-hot vectors of the same dimension as start and end logits output from the DistilBERT model and concatenated to get d^c. SoftMax is applied to the start and end logits from the expert models so that the output logits are in the same scale as the ground truth one-hot vectors.

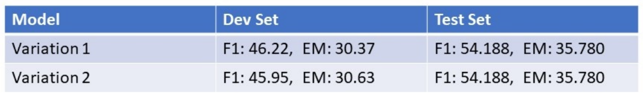
With this loss function, each expert is required to produce the whole of the output vector rather than a residual. The output of each expert model represents the mean of a multidimensional gaussian distribution. This error function is the negative log probability of generating the desired output log vector under the assumption that you first pick a hidden unit (expert model) and then pick an output vector from the gaussian distribution determined by the weight vector of the chosen hidden unit.

In order to verify that this loss function does indeed work as a gating mechanism and assign most of the probability on one of the experts for any given input, I ran the training and evaluation on two variations of the model:

Variation1: From the output of the MLP, the start and end logits of the expert assigned the highest probability is taken for evaluating F1 and EM scores.

Variation2: The output probabilities of the MLP is matrix multiplied with the start and end logits of all the experts and summed, to generate a weighted sum of the logits of all the experts, weighted by the respective probability assigned by the MLP output to each expert.

Note that this variation is only in the way the logits are passed to the evaluation function and does not affect the calculation of the loss function. These two variations of the same MoE model were trained on the 3 in-domain datasets and finetuned on the 3 out-of-domain training datasets. Further, they were evaluated on the Dev set and Test set.



Model	Dev Set	Test Set
Variation 1	F1: 46.22, EM: 30.37	F1: 54.188, EM: 35.780
Variation 2	F1: 45.95, EM: 30.63	F1: 54.188, EM: 35.780

Figure 2: Performance of two evaluation methods to check MLP classifier performance with custom loss function.

The near identical performance of the two models shows that the weighted sum of logits and choosing the most probable logits give nearly identical values, verifying that this loss function does encourage the MLP classifier to work as a gating mechanism.

In the MoE model, only the classifier parameters are updated with gradient descent. The layers of the DistilBERT models are frozen, so that the classifier can learn the right parameters to combine the outputs from the 3 expert models. The MoE model is first trained on all three in-domain datasets SQuAD, Natural Questions and NewsQA. This trained model is then finetuned on a small subset of out-of-domain training data. The out-of-domain datasets used in this project are DuoRC, RACE and RelationExtraction datasets. A small subset of out-of-domain datasets is used for finetuning the trained MoE model. The model is finally validated on a small out-of-domain validation set and then tested on the out-of-domain test set.

5 Experiments

5.1 Data

As described in the robustQA project handout, I used 3 datasets for in-domain training. The datasets are:

- SQuAD [3]: The Stanford Question Answering Dataset which has a corpus of 107,785 crowd-sourced question-answer pairs based on 536 wikipedia articles. Out of SQuAD, for this project, 50000 examples are used for training and 10,507 in the dev set.
- Natural questions dataset [5] which consists of real queries issued to the google search engine. From the natural questions dataset, 50000 examples are used for the training dataset and 12836 for the dev set.
- NewsQA dataset [4] which contains over 100,000 human generated question answer pairs based on around 10,000 CNN news articles. From the NewsQA dataset, 50000 examples are used for the training dataset and 4212 examples for the dev set.

The out-of-domain datasets used for the project are:

- DuoRC [6]: A collection of 186,089 unique question answer pairs created from a collection of 7680 movie plot pairs. From this dataset, there 2 are 127 training examples, 126 dev examples used for validating the trained model, and 1248 test examples.
- RACE [7]: which contains nearly 100000 reading comprehension questions and answers from English exams for middle and high school chinese students. From this set, there 127 examples in the training set, 128 examples in the dev set and 419 examples in the test set.
- The RelationExtraction dataset [8] which has over 30 million question answer pairs which are natural language queries generated from slot-filling examples. From the RelationExtraction dataset, there are 127 training examples, 128 dev examples and 2693 test examples.

5.2 Evaluation method

I used the F1 and EM evaluation metrics to compare the MoE model against the baseline DistilBERT model given with the project handout. To get a baseline performance metric, a single DistilBERT-forQuestionAnswering model is first trained on all three in-domain datasets. Then this trained model is finetuned on the out-of-domain training set. The finetuned DistilBERT model is used to evaluate performance on the out-of-domain dev set. On running this DistilBERT baseline model on the out-of-domain dev set, the performance was EM: 30.628 and F1: 47.716.

5.3 Experimental details

The experiments were run with the following configuration: The DistilBERT expert models were pretrained with batch size=16, learning rate 3e-5 for 3 epochs. The MOE model was trained on the in-domain datasets for 3 epochs with learning rate 1e-3, batch size=13. Batch size was chosen as 13 to avoid issues with the GPU running out of memory. The MLP configuration is as follows: Linear layer (4, 12), ReLU(), Linear layer(12, 12), Dropout(p=0.01), Linear layer(12, 4), Softmax(). With this configuration, one training run of the MOE model on the three in-domain datasets takes 8 hours.

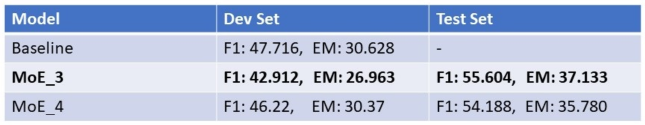
5.4 Results

Here I report Dev set and Test set results for the following 3 models.

Baseline: DistilBERTforQuestionAnswering model trained on all 3 in-domain training datasets and finetuned on the three out-of-domain training datasets.

MoE_3: The Mixture of Experts model described in approach above, with 3 experts.

MoE_4: A Mixture of Experts model where a 4th expert is added. This 4th expert is the baseline model described above, which is trained on all 3 in-domain datasets and finetuned on all 3 out-of-domain training datasets.



Model	Dev Set	Test Set
Baseline	F1: 47.716, EM: 30.628	-
MoE_3	**F1: 42.912, EM: 26.963**	**F1: 55.604, EM: 37.133**
MoE_4	F1: 46.22, EM: 30.37	F1: 54.188, EM: 35.780

Figure 3: Comparison of F1 and EM scores of 3 models.

Only Dev set F1 and EM are available for the baseline model. The MoE_3 model's performance on the dev set is lower than the baseline model's performance on the same set. In an attempt to improve generalisation of the MoE model, a fourth expert was added to the MoE model. The fourth expert is the same as the baseline model. This configuration MoE_4 performed significantly better that the MoE_3 model on the Dev set and was a promising contender for the best MoE model out of the two. However, surprisingly it performed slightly worse than the MoE_3 model on the test set. Therefore, for this final report, the MoE_3 model was chosen, based on test set performance.

The final performance on the test set is not as good as I had expected. Conceptually and intuitively, I expected the MoE model to perform better than the baseline model on the Dev and Test sets. I think the approach is valid, but further fine tuning of the model parameters, specifically the MLP layer, how it is trained, the loss function are required to get to the expected out-of-domain performance.

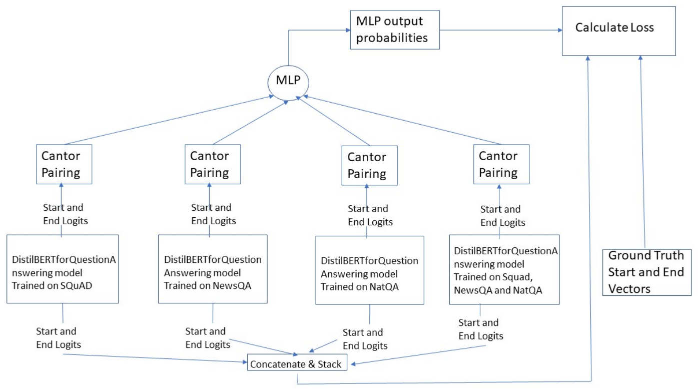


Figure 4: MoE_4 model architecture.

6 Analysis

One type of question where the model almost always makes an error, is when the question asks for the time when something was started, or completed, or ended, or when someone was born or someone died. The correct answer predicts the year, whereas when a time span is available in the context, the model erroneously predicts the time span instead of just the correct year. For example:

Question: What year did 39th government of Turkey start?

Context: The 39th government of Turkey (31 March 1975 – 21 June 1977) was a historical government of Turkey.

Answer: 1975

Prediction: 1975 – 21 June 1977

Question: What year did Province of Brabant end?

Context: The Province of Brabant was a province in Belgium from 1830 to 1995.

Answer: 1995

Prediction: 1830 to 1995

In many cases, the model tends to predict the answer with a longer context window than the provided answer (ground truth label), causing the EM score to drop, but not affecting the F1 score as much. For example,

Question: What color does Ellis color her hair?

Context: (long context, hence not included here)

Answer: blond

Prediction: dyed blond

Question: Where is Rachel hiding?

Context: (long context, hence not included here)

Answer: farm

Prediction: in a farm in the Dutch countryside

Another common mode of failure is when the answer is purely numeric (phone numbers, dates in number only, money values etc. to name a few). The model picks a numeric answer from the context, but sometimes the wrong one if the context contains multiple numbers. For example,

Question: What number can you call if you need some advice for skin care?

Context: (long context, hence not included here)

Answer: 860-868-0710

Prediction: 630-571-5466

Interestingly, I found one instance where the model's got the answer span wrong according to the provided answer, however from reading the passage, I think the model's prediction is also correct after reading the context paragraph.

Question: Who do Stubby and bunny run into?

Context: (long context, hence not included here)

Answer: Chaco

Prediction: another one of Stubby's old acquaintances,

7 Conclusion

I found that the mixture of experts model just about matches the out-of-domain performance of my baseline model in question answering. One original approach I explored is the use of cantor pairing to generate a single number input to the gating mechanism. The loss function described in the paper by Jacobs et al was implemented and found to improve performance over cross-entropy loss.

One issue to analyse is the behavior of F1 and EM scores when the MoE model is trained on the in-domain datasets. Although the training loss does decreases, the F1 and EM scores also fall during training. Analysing and remedying this would further improve the performance of the model.

One avenue for future work is dataset clustering: The three in-domain datasets available are considered as three separate classes and the expert models are trained on these classes separately. I would like to explore the possibility of running an unsupervised clustering algorithm like k-Means on three datasets to see if there is another classification of the datasets that can be found. This classification might depend on topic, question-answer length, or type of inference needed etc. The datasets could then be separated into groups based on this classification and then the experts could be trained separately on these groups to see if that makes the models perform better on out of domain QA.

References

[1] Michael Jordan et. al., Robert Jacobs. Adaptive mixtures of local experts. *Neural Computation*, 3:79 – 87, 1991.

[2] Richard Maclin David Opitz. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research 11*, 1999.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. *Association for Computational Linguistics (ACL)*, 2018.

[4] Tong Wang et. al. Adam Trischler. Newsqa: A machine comprehension dataset. *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, 2017.

[5] Jennimaria Palomaki et. al. Tom Kwiatkowski. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7, 2019.

[6] Rahul Aralikatte et. al. Amrita Saha. Duorc: Towards complex language understanding with paraphrased reading comprehension. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, page 1683–1693, 2018.

[7] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, 2017.

[8] Eunsol Choi Luke Zettlemoyer Omer Levy, Minjoon Seo. Zero-shot relation extraction via reading comprehension. *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, 2017.