Robust QA on out of domain dataset over pretraining and fine tuning

Stanford CS224N Default Project

Yueheng Li
Department of Computer Science
Stanford University
yueheng@stanford.edu

Abstract

We have seen tremendous progress on natural language understanding problems over the last few years. Meanwhile, we face issues that models learnt from a specific domain couldn't be easily generalized to a different domain. I explored different models to build robust question answering system that can be applied to out-of-domain datasets. Models explored are baseline with and without fine tuning, adding dataset prefix in question with and without fine tuning, switching question and context in question answering system with and without fine tuning, and shorter question and context in model input with and without fine tuning. Different fine tuning techniques like changing epochs, batch size and Adam optimization learning rate were explored to find the best model performance. The best model achieved 40.367 EM and 58.467 F1.

1 Key Information to include

- Mentor: None
- External Collaborators: None
- Sharing project: None

2 Introduction

We have seen tremendous progress on natural language understanding problems over the last few years. Meanwhile, we face issues that models learnt from a specific domain couldn't be easily generalized to a different domain [1, 2, 3, 4], whereas human beings can easily generalize knowledge learnt from a book to a movie.

Very large models trained on very large dataset can help to solve the generalization problem. For example GPT-3 model [5] demonstrated astounding few-shot capabilities on myriad language understanding tasks. However, while remarkable, GPT-3 consists of 175B parameters and it makes it challenging to use in most real-world applications.

In this work, I will build a question answering system that can adapt to unseen domains with only a few training samples from the unseen domain. The question answering system will be given a paragraph and a question about that paragraph as input, the goal is to answer the question correctly. The system will not use very large models and will only use medium sized model - DistilBERT [6] as the pretrained model.

3 Related Work

T5 model [7] showed astounding results on training multiple language tasks. Two major techniques used are: first use different prefixes for input to differentiate different tasks, second sample the input

by a temperature so that smaller input can also have a relatively high chance to be sampled. I mainly experimented the first technique in this work. A future work is to experiment the second technique.

Making Pre-trained Language Models Better Few-shot Learners [8] demonstrated that different label format can have huge impact on model performances. For example, the paper showed that when the positive/negative label is expressed as "It was great/terrible", the model will perform the best, and it would be much better than label "It was terrible/great". Inspired by that, I explored if changing the question/answer format can have impact on the model performances.

Adapt Language Models to Domains and Tasks [9] showed that continue to fine tune pretrained models on new domain can greatly help to improve the model performance on new domain. So I would first pretrain models on in domain questions using the pretrained DistilBERT model, and fine tune the pretrianed model on out of domain questions.

4 Approach

Inspired by T5 model [7], Making Pre-trained Language Models Better Few-shot Learners [8], Adapt Language Models to Domains and Tasks [9], I first pretrained baseline model and models with 3 different techniques using in-domain datasets mentioned in Table 2, and then fine tuned the pretrained model using out-of-domain datasets mentioned in Table 2.

4.1 Baseline

Convert each (question, paragraph) in all datasets into multiple chunks of size 384 with a stride of 128. For example, let (q, p) be a question, paragraph pair where $q = \{q_0, q_1, ...q_{10}\}$ and $p = \{p_0, p_1, ...p_{500}\}$. They will be converted into chuncks c_1 and c_2, where $c_1 = [\text{CLS}]q[\text{SEP}]p^1[\text{SEP}]$ with $p^1 = \{p_0, p_1, ..., p_{371}\}$ and $c_2 = [\text{CLS}]q[\text{SEP}]p^2[\text{SEP}]$ with $p^2 = \{p_{128}, p_{129}, ..., p_{500}\}$. Each chunk is labeled with a start position and end position based on its offset. For chunks that do not contain the answer the start and end positions are $(0, 0)$.

The baseline model pretrains DistilBERT [6] on the training data. I'll compare the model performances of baseline model with and without furtuer finetuning on out-of-domain datasets.

The loss function is the sum of the negative log-likelihood (cross-entropy) loss for the start and end locations.

$$\text{loss} = -\log \mathbf{p}_{start}(\text{i}) - \log \mathbf{p}_{end}(\text{j})$$

where i is the gold start location and j is the gold end location.

During inference, (question, paragraph) will be preprocessed in the same way as training data preprocess, and will be splitted into multiple chunks. Each chunk will be passed to model to get corresponding start and end locations. Then select the locations with the highest sum. More specifically, choose the pair (i, j) that maximized $\mathbf{p}_{start}\mathbf{p}_{end}$ with satisfying $i \leq j$ and $j - i + 1 \leq L_{max}$ where L_{max} is a hyperparameter that sets the maximum length of a predicted answer and it's set to 15 by default.

4.2 Add dataset prefix

Inspired by T5 model [7], which showed astounding results on training multiple language tasks, and different tasks are distinguished by adding different prefix in input, I'll first experiment adding dataset prefix.

For each dataset, each question q will become $DatasetName$: q, and each (question, paragraph) pair will be converted to (DatasetName: question, paragraph) pair.

For example, for SQuAD [10] dataset, each question q will become $squad$: q, and each (question, paragraph) pair will be converted to (squad: question, paragraph) pair.

Then the (DatasetName: question, paragraph) pair in all datasets are converted into multiple chunks of size 384 with a stride of 128 in a similar way as shown in baseline.

For example, let $(DatasetName: q, p)$ be a question, paragraph pair where $q = \{q_0, q_1, ...q_{10}\}$ and $p = \{p_0, p_1, ...p_{500}\}$. They will be converted into chuncks c_1, c_2 and c_3, where $c_1 = [\text{CLS}]DatasetName: q[\text{SEP}]p^1[\text{SEP}]$ with $p^1 = \{p_0, p_1, ..., p_{370}\}$ and $c_2 = [\text{CLS}]DatasetName: q[\text{SEP}]p^2[\text{SEP}]$ with $p^2 = \{p_{128}, p_{129}, ..., p_{499}\}$, and $c_3 = [\text{CLS}]DatasetName: q[\text{SEP}]p^3[\text{SEP}]$ with $p^3 = \{p_{256}, p_{257}, ..., p_{500}\}$. Each chunk is labeled with a start position and end position based on its offset. For chunks that do not contain the answer the start and end positions are $(0, 0)$.

The model pretrains DistilBERT [6] on the training data. I'll compare the model performances with and without further finetuning on out-of-domain datasets.

The loss function and inference are similar to baseline model.

4.3 Switch question and paragraph

Inspired by Making Pre-trained Language Models Better Few-shot Learners [8] which demonstrated that different label format can have huge impact on model performances, I'll experiment whether switching the positions of question and paragraph can have impact on model performances.

Convert each (question, paragraph) to (paragraph, question) in all datasets and then into multiple chunks of size 384 with a stride of 128. For example, let (q, p) be a question, paragraph pair where $q = \{q_0, q_1, ...q_{10}\}$ and $p = \{p_0, p_1, ...p_{500}\}$. They will be converted into (p, q) first. Then (p, q) will be converted in to chuncks c_1 and c_2, where $c_1 = [\text{CLS}]p^1[\text{SEP}]q[\text{SEP}]$ with $p^1 = \{p_0, p_1, ..., p_{371}\}$ and $c_2 = [\text{CLS}]p^2[\text{SEP}]q[\text{SEP}]$ with $p^2 = \{p_{128}, p_{129}, ..., p_{500}\}$. Each chunk is labeled with a start position and end position based on its offset. For chunks that do not contain the answer the start and end positions are $(0, 0)$.

The model pretrains DistilBERT [6] on the training data. I'll compare the model performances with and without furtuer finetuning on out-of-domain datasets.

The loss function and inference are similar to baseline model.

4.4 Shorter chunk

Also Inspired by Making Pre-trained Language Models Better Few-shot Learners [8] which demonstrated that different label format can have huge impact on model performances, I'll experiment whether a shorter chunk can have impact on model performances.

Convert each (question, paragraph) in all datasets into multiple chunks of size 120 with a stride of 40. For example, let (q, p) be a question, paragraph pair where $q = \{q_0, q_1, ...q_{10}\}$ and $p = \{p_0, p_1, ...p_{200}\}$. They will be converted into chuncks c_1, c_2, c_3, c_4, where $c_1 = [\text{CLS}]q[\text{SEP}]p^1[\text{SEP}]$ with $p^1 = \{p_0, p_1, ..., p_{107}\}$, $c_2 = [\text{CLS}]q[\text{SEP}]p^2[\text{SEP}]$ with $p^2 = \{p_{40}, p_{41}, ..., p_{147}\}$, $c_3 = [\text{CLS}]q[\text{SEP}]p^3[\text{SEP}]$ with $p^3 = \{p_{80}, p_{81}, ..., p_{187}\}$ and $c_4 = [\text{CLS}]q[\text{SEP}]p^4[\text{SEP}]$ with $p^4 = \{p_{120}, p_{81}, ..., p_{200}\}$. Each chunk is labeled with a start position and end position based on its offset. For chunks that do not contain the answer the start and end positions are $(0, 0)$.

The model pretrains DistilBERT [6] on the training data. I'll compare the model performances with and without furtuer finetuning on out-of-domain datasets.

The loss function and inference are similar to baseline model.

5 Experiments

5.1 Data

As shown in Table 2, datasets used are divided into in-domain datasets and out-of-domain datasets. In-domain datasets contain SQuAD [10], NewsQA [11] and Natural Questions [12], and they will be used for training and dev, and generate the pretrained model. Out-of-domain datasets contain DuoRC [13], RACE [14] and RelationExtraction [15], and they will be used for training, dev and test, and fine tune the pretrained model. The final model will be evaluated with out-of-domain test dataset.

Dataset	Question Source	Pasage Source	Train	Dev	Test
in-domain datasets					
SQuAD [10]	Crowdsources	Wikipedia	500000	10,507	-
NewsQA [11]	Crowdsources	Wikipedia	500000	4,212	-
Natural Questions [12]	Search logs	Wikipedia	500000	12,846	-
oo-domain datasets					
DuoRC [13]	Crowdsources	Movie reviews	127	126	1248
RACE [14]	Teachers	Examinations	127	126	419
RelationExtraction [15]	Synthetic	Wikipedia	127	128	2693

Table 1: Statistics for datasets used for building the QA system for this project. Question Source and Passage Source refer to data sources from which the questions and passages were obtained. Table borrowed from [16]

5.2 Evaluation method

Model performance is measured via two metrics: **Exact Match (EM)** score and **F1** score.

For example, if the system answered a question with "York" but the ground truth answer was "New York".

EM is a binary measure of whether the system output matches the ground truth answer exactly. For the example mentioned above $EM = 0$

F1 is the harmonic mean of precision and recall. For the example mentioned above, the precision is 100% (answer is a subset of the ground truth) and 50% recall (answer only included 50% of the words in the ground truth), and the F1 score is $2 * (precision * recall)/(precision + recall) = 2 * (100 * 50)/(100 + 50) = 66.67\%$

5.3 Experimental details

Most model configurations: Batch size: 16, Epochs: 3, Optimization algorithm: Adam, Learning rate: 3e-5

I'll mention the difference if the experiment model use different configurations

5.4 Results

I'm on the RobustQA track, and the scores obrained on the test leaderboard are:

EM: 40.367, F1: 58.467

The scores are as expected. Because with all the experiments shown in Table2, models performances are improving because of different pretraining and different fine tuning. However, I believe there is still a lot of room to further improve the scores by using different approaches.

6 Analysis

6.1 With fine tuning vs without fine tuning

As shown in Table 2, on out-of-domain validation test, baseline model without fine tuning obtained 31.41 EM and 46.71 F1 respectively, and baseline model with fine tuning obtained 31.91 EM and 47.23 F1 respectively. We can see baseline model with fine tuning performs better than baseline model without fine tuning. It's expected because fine tuning would provide more information about the out-of-domain dataset. The same results are shown for all models including adding dataset prefix, switching question and context, and shorter question and context.

6.2 Fine tuning models

When we compare baseline with fine tuning, switching question and context with fine tuning, shorter question and context with fine tuning, we can see that both switching question and context and shorter

Experiment	EM	F1
Baseline		
Baseline no fine tuning	31.41	46.71
Baseline with fine tuning	31.94	47.23
Switch question and context		
Switch question/context no fine tuning	19.37	29.41
Switch question/context with fine tuning	20.16	30.14
Shorter question and context		
Shorter question context no fine tuning	31.41	46.71
Shorter question context with fine tuning	31.94	45.86
Add dataset prefix		
Add prefix no fine tuning	31.68	47.63
Add prefix with fine tuning	31.94	47.60
Add dataset prefix - more fine tuning		
Add prefix with fine tuning 20 epochs	31.94	47.60
Add prefix with fine tuning 4 batch size	31.94	48.02
Add prefix with fine tuning 1 batch size	32.20	48.29
Add prefix with fine tuning 1 batch size and 5e-5 adam	32.46	48.66
Add prefix with fine tuning 1 batch size and 5.5e-5 adam	**32.46**	**49.01**

Table 2: Model performance on out-of-domain validation set

question and context won't help to improve model performance. Switching question and context with fine tuning EM and F1 are 20.16 and 30.14, and shorter question and context with fine tuning EM and F1 are 31.94 and 45.86, whereas baseline model with fine EM and F1 are 31.91 and 47.23 respectively. That is because neither switching question and context nor shorter question and context provides extra information for out-of-domain dataset compared with baseline model, and as shown in "Making Pre-trained Language Models Better Few-shot Learners [8]" paper, the prompt format can have big impact on model performance, and switching question and context and shorter question and context would change the data format and those changes happen to have negative impact.

Especially switching question and context with fine tuning, both EM and F1 dropped around 30% compared with baseline with fine tuning, which means chunck format $[\mathrm{CLS}]q[\mathrm{SEP}]p[\mathrm{SEP}]$ used by baseline models is better than the chunk format $[\mathrm{CLS}]q[\mathrm{SEP}]p[\mathrm{SEP}]$ used by switching question and context models.

When we compare baseline with fine tuning and adding dataset prefix with fine tuning, we can see that adding dataset prefix with fine tuning improved the F1 score from 47.23 to 47.60. That is because adding dataset prefix can help to provide extra information about the domain of data.

6.3 Further fine tuning for adding dataset prefix models

Since adding dataset prefix with fine tuning showed the largest model performance improvement compared with baseline model without fine tuning, I fine tuned the adding dataset prefix without fine tuning model in a couple of different ways to furture improve the model performance.

We can see that the adding prefix with fine tuning using default model parameters obtained 31.94 EM and 47.60 F1, and increasing fine tuning epochs from 3 epochs to 20 epochs also obtained 31.94 EM and 47.60 F1. So increasing adding prefix fine tuning epochs won't help with model performance. That is because 3 epochs is already reaching the optimal model performance and increasing epochs will not further increase the model performance.

We can also see that decreasing batch size of adding prefix with fine tuning from 16 to 4 would improve F1 from 47.60 to 48.02, and decreasing the batch size to 1 would further improve EM from 31.94 to 32.20 and EM from 48.02 to 48.29. So the smaller the batch size is the better the adding

dataset prefix with fine tuning model performance is. That is because smaller batch size reduces the variability of the batch, and each batch will have a better gradient update towards the optimal.

Besides epochs and batch size, I also experimented different learning rate for Adam optimization. The default learning rate is 3e-5, and I experimented 1e-5, 1.5e-5, 2e-5, 2.5e-5, 3e-5, 3.5e-5, 4e-5, 4.5e-5, 5e-5, 5.5e-5 and 6e-5, and found the model would perform the best when Adam learning rate is 5.5e-5, and it achieved 32.46 EM and 49.01 F1.

7 Conclusion

7.1 Achievement and Main findings

I experimented different models with or without fine tuning on out-of-domain dataset to improve the EM and F1 score on out-of-domain test dataset. Different models are baseline model, switching question and context in model input, using shorter question and context as input, and adding dataset prefix in questions.

I found that models with fine tuning perform better than models without fine tuning on out-of-domain dataset. Among models with fine tuning on out-of-domain dataset, input format can have huge impact on model performance. For example, chunck format $[\mathrm{CLS}]q[\mathrm{SEP}]p[\mathrm{SEP}]$ used by baseline models is better than the chunk format $[\mathrm{CLS}]q[\mathrm{SEP}]p[\mathrm{SEP}]$ used by switching question and context models. Adding dataset prefix can help to provide extra information for dataset, thus help to improve the model performance. And various fine tuning techniques can help to forthur improve model performance, including having smaller batch size, and I found Adam optimization 5.5e-5 learning rate will help to achieve the best scores among all the models I experimented.

7.2 Limitations

The input format changes like switching question and context in model input, using shorter question and context as input are manually selected, and there should be many other input format that would change model performances. This report didn't explore more possibilities. More over, there can be some automatical way to select the input format as mentioned in "Making Pre-trained Language Models Better Few-shot Learners [8]" paper.

7.3 Future work

Explore automatic ways to select input format as mentioned in "Making Pre-trained Language Models Better Few-shot Learners [8]" paper. Explore more fine tuning techniques besides optimization learning rate, epochs and batch sizes. Explore sampling the input by a temperature so that smaller input dataset can also have a relatively high chance to be sampled.

References

[1] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328, 2017.

[2] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[3] Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy, July 2019. Association for Computational Linguistics.

[4] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online, July 2020. Association for Computational Linguistics.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[6] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[7] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

[8] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners, 2020.

[9] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks, 2020.

[10] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[11] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[12] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, March 2019.

[13] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *CoRR*, abs/1804.07927, 2018.

[14] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

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

[16] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China, November 2019. Association for Computational Linguistics.