Data Augmentation for Robust QA System

Stanford CS224N Default Project

Pengfei Xing
Department of Computer Science
Stanford University
pfxing@stanford.edu

Abstract

A lot of research works show that modern Neural Networks learn brittle correlations between words or phrases which will affect the out-of-domain performance of QA System. One solution to prevent model from learning these corrections is to use data augmentation method to decrease the proportion of correlations from training data. In this project, we aim to evaluate several popular data augmentation methods and see how they can improve the out-of-domain performance of QA System. Beyond that, we also find generating even number of in-domain and out-of-domain training data could hardly help to improve the performance. Instead, we need to rebalance the distribution of in-domain data and out-of-domain data to force the model to focus on out-of-domain data without bringing significant effect to the in-domain QA performance. Finally, we also evaluate the performance of several different rebalance options and find we need to carefully choose the proportion value between in-domain data size and out-of-domain data size in order to optimize the performance of the out-of-domain QA system.

1 Introduction

A lot of research works show that generalizing QA system remains a pretty challenging task [1, 2], especially generalizing for a target domain without sufficient training data for that domain. There are multiple directions in the world to improve the performance of the Robust QA system. In this project, we explore how we can improve the performance using data augmentation techniques. Even though there are a lot of data augmentation tools existing now [3], adopting data augmentation to out-of-domain QA system is still a tough task. Some research works [4] even shows some back-translation augmentation yields no noticeable improvement to the Robust QA system. We think the negative results in those works are caused by not having a good data augmentation strategies, such as how to sampling in-domain and out-of-domain datasets and how to choose different data augmentation techniques to in-domain and out-of-domain datasets separately. We believe data augmentation has great potential in out-of-domain QA tasks as long as we develop a good strategy for data augmentation.

In this project, we find rebalancing the distribution of in-domain datasets and out-of-domain datasets could significantly improve the performance of the Robust QA System. We sample the in-domain datasets and increase the out-of-domain dataset size by 100 times. What's more, we also find we'd better apply augmentators which could generate more diverge data to the in-domain datasets and apply augmentators which produces similar data to out-of-domain datasets. All these techniques could boost the performance of the Robust QA system. In the validation leaderboard, our model ranks 17 at the end of the project due date.

2 Related Work

Considering one of the biggest challenges of Robust QA System is lacking sufficient training data of the target domain, there are a lot of work exploring how to use data augmentation techniques to better leverage the available supervised data. In [4], authors augmented the in-domain datasets using back-translation techniques. They train an 8-layer seq2seq model and use German as the pivot language to do back-translation. For the augmented datasets, they use string matching to find the new answer span and for those that fail they employ the same heuristic described in [5] to get the new estimated answer. The experiments show that back-translation augmentation yields no noticeable improvement to the Robust QA system. During our investigation, we think the reason for the negative result might be the back-translation model trained by the authors might not be able to generate divergent datasets. Instead, the augmented datasets may look pretty similar to the original ones and they can not help the model to learn the unseen data.

3 Approach

3.1 Baselines

The baseline of this project is DistilBERT [6] model trained using the provided datasets. According to the instructions, the datasets are borrowed from [7]. In my experiment, the performance of the baseline model is EM: 33.25, F1: 48.43.

3.2 Approaches

In this project, we aim to explore 4 data augmentaion methods and evaluate their effect to the performance of out-of-domain QA System. Let's say we have a sentence: "I am very busy these days because the workload of NLP is pretty heavy.". We will show the augmented sentences using different augmentation methods as below:

> **back-translation Augmenter** is a method to apply data augmentation by translating the sentence into a pivot language and then translating back to the original language. In this project, we will use French as the pivot language. The augmented sentence is "I am very busy nowadays because the workload of the NLP is quite high."
>
> **Synonym Augmenter** is an augmentation method to substitute word with similar words according to WordNet/PPDB synonym. The augmented text is "I am rattling busy these clarence day because the workload of NLP is pretty heavy."
>
> **Word Embedding(WordEmb) Augmenter** is a method to apply data augmentation by using GloVe, word2vec or fasttext. The augmented text is "I am very busy these days because the workload of NLP is pretty heavy. would is 've and got ' " the (to from 't the has ." This method would probably introduce some meaningless words or phrases into the sentences. As we can see in the augmented sentence, the later part is meaningless.
>
> **Spelling Augmenter** is a method to apply data augmentation by substitue words according to the spelling mistake dictionary. The augmented sentence is: "I'm Am very busy tese days because the workload of NLP is prety heavy."

Beyond that, we also notice that the out-of-domain QA system performs worse using all the above data augmentation method. After investigation, we find the two culprits for it. First reason is these data augmentation methods don't change the original sentence too much and the augmented sentences are still very similar to the original ones. Thus, using data augmentation can't help the QA system to learn the unseen data. Second reason is the data size of in-domain datasets is much larger than the out-of-domain datasets. The model overfits to the in-domain datasets. In order to avoid overfit, we rebalance the distribution of the training set by generating much more new data for the out-of-domain datasets while sampling a small amount of the in-domain datasets.

3.3 Implementation

In this project, we will use the augmentation API provided in nlpaug [3]. However, the API has several limitations such as unable to process long sentences. Therefore, we need to preprocess the input sentences a little bit. For example, we need to split the context into single sentence list before back-translation augmentation. Also, we implement several util functions to help sampling the in-domain datasets and increase the size of out-of-domain datasets.

3.4 Original Work

In this project, data augmentation is pretty straightforward. The most challenging part is how to tag the new answer in the augmented data. Considering we don't have enough resources for both people and time, we decided to go with string matching method and just ignore the data that couldn't be resolved using string matching. Beyond that, we also spent lots of efforts creating filter logic to identify the false positive data created by string match. For example, "Category" contains "Cat" but we don't want to tag the start index of "Category" as the answer index. We create a bunch filter logic to avoid these noises.

Another work needs to be highlighted is rebalancing datasets. In the first phase of this project, we find the QA system performs even worse by merging augmented datasets into the original or fine tune the pretrained model using the augmented datasets. After investigation and several experiments, we notice increasing the data size of augmented out-of-domain datasets and also decreasing size for in-domain datasets could significantly improve the performance of out-of-domain QA system.

4 Experiments

4.1 Data

In this project, we have 5 version of datasets in total as below:

- The first version datasets are provided by the guidebook. They consist of 3 in-domain datasets and 3 out-of-domain datasets. For all datasets, we have a Train set and Dev set. Beyond that, we have a Test set for out-of-domain dataset. Here are the list of the provided datasets:

Original Datasets					
Dataset	Question Source	Passage Source	Train	Dev	Test
SQuAD [8]	Crowdsourced	Wikipedia	18,885	48	-
NewsQA [9]	Crowdsourced	News articles	11,428	638	-
Natural Questions [10]	Search logs	Wikipedia	104,071	12,836	-
DuoRC [11]	Crowdsourced	Movie reviews	12	18	172
RACE [12]	Teachers	Examinations	109	113	391
RelationExtraction [13]	Synthetic	Wikipedia	127	128	2,693

- The second dataset is the augmented data of in-domain datasets using different techniques. The goal for this datasets is to evaluate whether we can use data augmentation technique to avoid the model overfit to the in-domain datasets by adding some noise to the in-domain train set.

Augmented In-domain Datasets				
Dataset	Spelling	Synonym	WordEmb	back-translation
SQuAD [8]	9,174	9,605	9,134	8,083
NewsQA [9]	3,873	4,023	3,831	2,702
Natural Questions [10]	83,934	85,736	83,822	14,821

- The third datasets are the augmented data of out-of-domain datasets using different techniques. The goal for creating it is to evaluate whether we can fine tune the pretrained baseline model using augmented out-of-domain datasets.

Augmented Out-of-domain Datasets Without Rebalance				
Dataset	Spelling_Train	Spelling_Dev	Synonym_Train	Synonym_Dev
DuoRC [11]	16	18	16	18
RACE [12]	200	113	200	113
RelationExtraction [13]	213	128	217	128

Augmented Out-of-domain Datasets Without Rebalance				
Dataset	WordEmb_Train	WordEmb_Dev	Backtrans_Train	Backtrans_Dev
DuoRC [11]	15	18	12	18
RACE [12]	201	113	109	113
RelationExtraction [13]	217	128	127	128

- The fourth datasets are the augmented data of both in-domain and out-of-domain datasets. However, before we do data augmentation, we sample the in-domain dataset a little bit to avoid one domain data is dominant in terms of the data size. After comparing the data size, we decide to decrease the number of Natural Questions datasets from 104,071 to 18,885. Beyond that, we also want to remove the size gap between in-domain datasets and out-of-domain datasets. Therefore, during data augmentation, we rebalance the distribution of the datasets by iterating the augmentaion process 100 times for out-of-domain datasets. For the data augmentation library we used, Spelling Augmentator and Synonym Augmentator will generate random results while Backtranlation Augmentator and Word Embedding Augmentator will generate the same results in different runs. Therefore, for this datasets, we only use Spelling Augmentator and Synonym Augmentator to do data augmentation.

Augmented Datasets With Rebalance				
Dataset	Spelling_Train	Spelling_Dev	Synonym_Train	Synonym_Dev
SQuAD [8]	9,140	5	9,602	7
NewsQA [9]	3,858	146	4,079	148
Natural Questions [10]	15,292	370	15,605	392
DuoRC [11]	355	1,515	369	1,509
RACE [12]	9,259	9,485	9,364	9,595
RelationExtraction [13]	9,013	8,937	9,449	9,219

- The fifth datasets are similar to the fourth one but only differs in terms of the iteration number of the out-of-domain datasets. In fourth datasets, we increase the size of out-of-domain datasets by 100 times. In this datasets, we use Synonym Data Augmentation to evaluate how much effect the distribution will have to the QA performance. We increase the out-of-domain dataset size to 150 and 200 times separately.

Augmented Out-of-domain Datasets Using Synonym Augmentator With Rebalance				
Dataset	x150_Train	x150_Dev	x200_Train	x200_Dev
DuoRC [11]	560	2,270	735	2,976
RACE [12]	14,161	14,460	18,563	18,957
RelationExtraction [13]	14,309	13,765	17,925	17,890

4.2 Evaluation method

In the evaluation, we used the provided out-of-domain validation datasets and EM/F1 metrics to evaluate the performance.

4.3 Experimental details

In this project, we ran several types of experiments as below:

- **Data augmentation to avoid overfit:** In this experiment, we merge the second datasets with the original datasets and train the model from the beginning using the merged data. The purpose of this experiment is to evaluate whether we can avoid the model overfit to the in-domain dataset by introducing some noise to the training data. In this experiment, all the model configurations are the same as the default ones provided by the guidebook. The running time is a little bit longer and it takes me more than 20 hours to finish each experiment.
- **Fine tune pretrained baseline model using augmented out-of-domain datasets:** In this experiment, we use the third datasets to fine tune the pretrained baseline model. The purpose of this experiment is to evaluate whether we can fine tune the pretrained baseline model and achieve a decent performance in out-of-domain datasets only using a small amount of out-of-domain datasets. In this experiment, all the model configurations are the same as the default ones provided by the guidebook except the train/dev datasets. In this experiment, the

train/dev sets are the out-of-domain datasets. Since the data size is much smaller than the in-domain datasets, I can even run this experiment on local machine and the running time is very small which is less than 30 minutes with the Nvidia 2080 Ti GPU.

- **Fine tune pretrained model using rebalanced augmented datasets:** In this experiment, we use the fourth datasets to fine tune the pretrained baseline model. The purpose of this experiment is to evaluate whether we can improve the performance of out-of-domain QA by increasing the proportion of the out-of-domain datasets. Most of the model configurations are the same as the guidebook exception we change the train/dev set to the superset of in-domain and out-of-domain datasets. Since we sample the in-domain datasets, we are still able to run this experiment on local machine. The running time is around 3 hours.

- **Fine tune pretrained model using different rebalanced augmented datasets:** This experiment is pretty similar to the last one and only differs in the datasets. In this experiment, we used the fifth datasets and all other configurations are exactly same as the last experiment.

4.4 Results

- **Data augmentation to avoid overfit:** From the table below we can see the out-of-domain QA system performs worse using all the data augmentation methods. This result is worse than we expected. We think merging some augmented data into the original datasets can help to add some noise to the train set which can help to avoid overfit to the in-domain datasets. However, the result tells us we didn't make it. The reason might be our data augmentation techniques are not good enough to generate more diverge dataset. Thus, the new datasets are still very similar to the original one which will make the model overfit more to the in-domain datasets.

Experiments Result		
Name	EM	F1
Baseline	33.25	48.43
Spelling	33.25	48.30
Synonym	30.63	46.21
WordEmb	29.84	47.15
back-translation	29.12	47.54

- **Fine tune pretrained baseline model using augmented out-of-domain datasets:** From the table below we can see the out-of-domain QA system performans worse using all the data augmentation methods. The result is also worse than we expected. We think fine tuning the pretrained model using the out-of-domain datasets could slightly modifies the model to fit the out-of-domain knowledge. The reason for the bad performance might be the small data size. Since the data size used to fine tuning the model is pretty small, the model can't learn the out-of-domain questions very well.

Experiments Result		
Name	EM	F1
Baseline	33.25	48.43
Spelling	31.15	44.95
Synonym	21.73	36.71
WordEmb	22.46	37.22
back-translation	31.54	46.01

- **Fine tune pretrained model using rebalanced augmented datasets:** From the table below we can see the performance of the out-of-domain QA system improves a lot. This result is the same as we expected. The consistent improvements in both Spelling Augmentation and Synonym Augmentation proves our assumption which is increasing the size of out-of-domain datasets could force the model to focus on out-of-domain questions. Also, we still need to add some sampled in-domain datasets to make the model stable e.g. still be able to perform well on in-domain datasets.

Experiments Result		
Name	EM	F1
Baseline	33.25	48.43
Spelling	34.03	49.62
Synonym	36.91	50.43

- **Fine tune pretrained model using different rebalanced augmented datasets:** From the table below, we can see choosing different proportions of the out-of-domain datasets could slightly affect the performance of QA system. However, the performance of Synonym_200 is a little bit beyond our expectation. Since we think more out-of-domain dataset could help to boost the QA performance but Synonym_200 is worse than both Synonym_100 and Synonym_150. The reason might be when iterating the data augmentation pipeline for 200 times, there might be a lot of repeated train/dev data which couldn't help the model to learn the unseen data. Instead, it may make the model to overfit to several particular questions. All of these situations will hurt the perfomrance of out-of-domain QA.

Experiments Result		
Name	EM	F1
Baseline	33.25	48.43
Synonym_x100	36.91	50.43
Synonym_x150	37.17	50.62
Synonym_x200	31.15	44.95

- **Best leaderboard results:** We noticed the final ranking in test leaderboard is not as good as the rank in validation leaderboard. The reason might be our model overfits to the validation datasets. Thus, in the future, we may need to reserve some validation datasets to evaluate the model performance before evaluating it using test datasets.

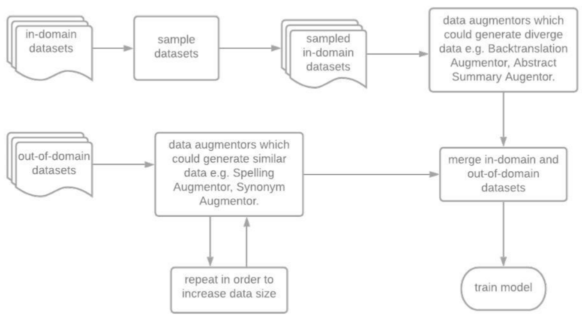
Leaderboard Result				
Name	Rank	EM	F1	Tag
Validation	17	37.173	50.624	XPF
Test	55	40.803	58.116	XPF

5 Analysis

Based on all the above experiments, we find we need to take different data augmentation and sampling strategies to in-domain and out-of-domain datasets. For in-domain datasets, We need to sample the datasets to force the model pay less attention to the in-domain knowledge and avoid overfit to the in-domain datasets. This can be proven in the third experiment. After decreasing the proportion of in-domain datasets, the EM/F1 increases from 33.25/48.43 to 36.91/50.43. What's more, we also need to use the in-domain datasets to help the model to learn the unseen data. Thus, we need adopt data augmentors which could generate more diverge datasets to in-domain datasets, such as back-translation augmentator or abstract summary augmentator. In our experiments, the performance of back-translation augmentator is 29.12/47.54 which is worse than the baseline model. The reason is the back-translation API we used is not good enough to produce diverge results. The augmented datasets are pretty similar to the original one.

In terms of the out-of-domain datasets, the first priority task is to increase the proportion of data size. We can easily achieve this goal by iterating the data augmentation pipeline for multiple times. However, we need to pay attention to the number of iterations because in our experiments, iterating 150 times could help to boost the performance of iterating 100 times. The performance improves from 36.91/50.43 to 37.17/50.62. However, when we increase the iteration number, the performance drops to 31.15/44.94. The reason might be when iterating too many times, there will be high possible to generating duplicated data which will make model to overfit to several particular questions.

We summarize the analysis in the diagram below:


in-domain datasets
sample datasets
sampled in-domain datasets
data augmentors which could generate diverge data e.g. Backtranslation Augmentor, Abstract Summary Augentor.
out-of-domain datasets
data augmentors which could generate similar data e.g. Spelling Augmentor, Synonym Augmentor.
merge in-domain and out-of-domain datasets
repeat in order to increase data size
train model

6 Conclusion

In this project, we identify the trade-off between different data augmentation strategies for Robust QA System. For in-domain datasets, we need to sample the datasets first to avoid overfitting and then use more advanced data augmentation techniques, such as back-translation and abstract summary augmentation, to generate more diverge datasets in order to help the model learn the unseen data. For out-of-domain datasets, we need to use data augmentation technique that could generate similar datasets, such as spelling augmentation and synonym augmentation. Also, we need to iterate the data augmentation for multiple times in order to increase the proportion of out-of-domain datasets. The iteration number needs to be carefully designed because it may also slightly affect the final performance of the Robust QA System.

Due to the limitation of time and resources, we are not able to generate more diverge data for in-domain datasets. We test several back-translation augmentators but the augmented data is pretty similar to the original one. This will stop the model from learning unseen data. In the future, we can keep exploring more advanced data augmentation techniques to see if we can generate more diverge data and improve the performance of Robust QA System.

References

[1] Jonathan Berant Alon Talmor. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. In *Association for Computational Linguistics*, 2019.

[2] Jerome Connor Tomas Kocisky Mike Chrzanowski Lingpeng Kong Angeliki Lazaridou Wang Ling Lei Yu Chris Dyer Phil Blunsom Dani Yogatama, Cyprien de Masson d'Autume. Learning and evaluating general linguistic intelligence. 2019.

[3] Edward Ma. nlpaug. GitHub, 2020.

[4] Zhucheng Tu Chris DuBois Shayne Longpre, Yi Lu. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. In *2nd Workshop on Machine Reading for Question Answering*, 2019.

[5] Minh-Thang Luong Rui Zhao Kai Chen Mohammad Norouzi Quoc V. Le Adams Wei Yu, David Dohan. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*, 2018.

[6] Julien Chaumond-Thomas Wolf Victor Sanh, Lysandre Debut. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS*, 2020.

[7] Robin Jia-Minjoon Seo Eunsol Choi Danqi Chen Adam Fisch, Alon Talmor. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *EMNLP 2019 Workshop on Machine Reading for Question Answering*, 2019.

[8] Konstantin Lopyrev-Percy Liang Pranav Rajpurkar, Jian Zhang. Squad: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[9] Xingdi Yuan-Justin Harris Alessandro Sordoni Philip Bachman Kaheer Suleman Adam Trischler, Tong Wang. Newsqa: A machine comprehension dataset. In *Computation and Language (cs.CL); Artificial Intelligence*, 2017.

[10] Olivia Redfield-Michael Collins Ankur Parikh Chris Alberti Danielle Epstein Illia Polosukhin Matthew Kelcey Jacob Devlin Kenton Lee Kristina N. Toutanova Llion Jones Ming-Wei Chang Andrew Dai Jakob Uszkoreit QuocLe Tom Kwiatkowski, Jennimaria Palomaki and Slav Petrov. Natural questions: a benchmark for question answering research. In *Transactions of the Association of Computational Linguistics*, 2019.

[11] Mitesh M. Khapra Karthik Sankaranarayanan Amrita Saha, Rahul Aralikatte. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *ACL*, 2018.

[12] Hanxiao Liu Yiming Yang Eduard Hovy Guokun Lai, Qizhe Xie. Race: Large-scale reading comprehension dataset from examinations. In *EMNLP*, 2017.

[13] Eunsol Choi Luke Zettlemoyer Omer Levy, Minjoon Seo. Zero-shot relation extraction via reading comprehension. In *Association for Computational Linguistics*, 2017.