Robust Question Answering via In-domain Adversarial Training and Out-domain Data Augmentation

Stanford CS224N Default Project (Robust QA Track)

Zecheng Zhang
Department of Computer Science
Stanford University
zecheng@stanford.edu

Kun Guo
Department of Engineering
Stanford University
kunguo@stanford.edu

Yiyun Gu
Department of Engineering
Stanford University
guyiyun7@stanford.edu

Abstract

To explore the effectiveness of domain-related information on QA model robustness, we leverage potential domain information, both domain-specific and domain-invariant, from the text data. During training on the *in-domain* training set, we experiment adversarial training with three adversarial functions, the Kullback-Leibler divergence, Jensen-Shannon divergence and creating fake labels. At this in-domain pre-training stage, we conjecture that the QA model can learn domain-invariant feature representations from the *in-domain* training set through adversarial training. In addition to domain-invariant learning from *in-domain* training, we propose a data augmentation method that can retain high-level domain information by using named entity recognition and synonyms replacement. This augmentation method is applied to the *oo-domain* training set and we hypothesize that the model can better learn domain-specific information from augmented out-of-domain datasets and thus improve performance on out-of-domain tasks. We designed and conducted several experiments to test the effectiveness of our proposed adversarial training and augmentation methods. The experiment results, analysis and learning are provided in this report.

1 Introduction

Question Answering (QA) models trained on specific datasets and perform well in corresponding domains often perform poorly handling out-of-domain tasks. Additional training and finetuning with large amounts of out-of-domain data can be the ideal solution. However, such data is often not readily available or insufficient. This cross-domain learning challenge has been addressed by many effort, including with pretrained language models, domain generalization, and adversarial training. Previous works demonstrate various degree of successes, which suggests that domain-invariant features can be learned and utilized for cross-domain language tasks.

As illustrated in Figure 1, we approach this cross-domain learning challenge by combining *in-domain* adversarial training with *oo-domain* data augmentation in order to learn both *in-domain* domain-invariant and *out-domain* information. During in-domain training, an adversarial domain discriminator classifies domain labels from hidden representations from the QA model using DistilBERT. By doing adversarial training, the QA model fools the discriminator by making hidden representations indistinguishable so as to learn parameters that invariant to domain-specific inputs. In addition, we leverage out-of-domain information by augmenting limited amounts of out-of-domain datasets which might help to improve QA model robustness. We find that different out-of-domain datasets usually have different domain focuses. One way to differentiate their domains is to use named entity recognition. When augmenting the datasets, the entity level domain information is retained so that domain-specific feature representations can be better utilized.

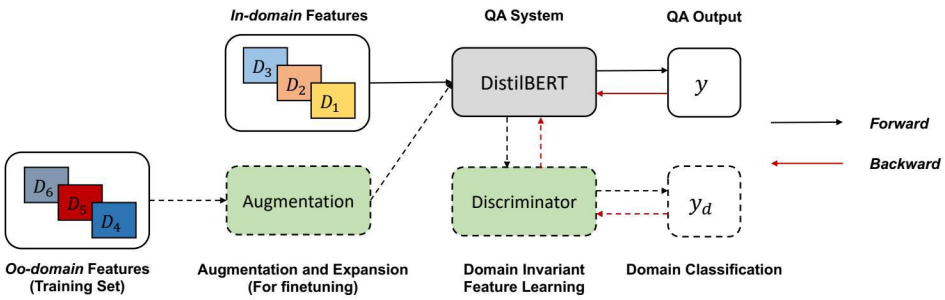

Figure 1: Overview of proposed in-domain adversarial training and out-of-domain data augmentation approach for robust QA model training.

2 Related Work

2.1 Question Answering

Given a paragraph and a question about the paragraph as input, question answering model aims to extract the correct answer from the paragraph, usually by predicting the start and end positions. There are a variety of models leveraging pretrained BERT that performs well within the same domain such as DeBERTa[1], RoBERTa[2] and question answering using DistilBERT [3]. The DistilBERT QA is the baseline in this project and our approaches are built upon the the DistilBERT QA model.

2.2 Adversarial Training

Adversarial training is first proposed by Generative Adversarial Network (GAN) with a generator and a discriminator [4] to generate images using Jensen–Shannon divergence loss. The concept of GAN is then used for text generation by modeling the data generator as a stochastic policy and directly performing gradient policy update to overcome the problems of discrete tokens[5].

Besides text generation, adversarial training is applied to domain adaption. DANN[6] proposes augmenting feed-forward models with a few standard layers and a gradient reversal layer. For classification problems such as image classification and document sentiment analysis, in addition to a classifier for the main learning task, there is also a domain classifier that discriminates between the source and target domains during training. Based on DANN which learns domain-invariant feature representations through a domain classifier, shared LSTMs and domain-specific LSTMs through gated connection[7] are added to learn both domain-specific and domain-agnostic feature representations. Domain classifiers can also be applied in semi-supervised learning. AdaMRC[8] generates pseudo questions for unlabeled passages in the target domain and incorporates a domain classifier to distinguish the source domains from the target domains. For domain-agnostic question answering tasks, [9] uses the adversarial training with KL divergence loss and achieves promising results. In this project, we adopt parts of the adversarial training method from [9] and expand on it.

2.3 Data Augmentation

Data augmentation has been explored in many NLP tasks. Synonym replacement is first introduced in text classification [10] which replaces words or phrases with their synonyms based on geometric distribution. It is effective to combine synonym replacement, random insertion, random swap, and random deletion[11]. For reading comprehension, back-translation from a neural machine translation model[12] is proposed to paraphrase. In neural machine translation, low-frequency words are emphasized by altering existing sentences in the parallel corpus[13]. For question answering, This method [14] augments questions and answers following symmetry and transitivity logical rules. With limited training passages, questions and answers for out-of-domain datasets, the data augmentation can be useful for *oo-domain* finetuning.

3 Domain Invariant Learning

Previous works suggest that the domain-invariant feature representation can be learned and utilized for cross-domain NLP tasks. We hypothesize that the QA model robustness correlates with how well the domain-invariant features can be extracted and learned. In this section, we will describe the question answering model and different adversarial methods to learn domain-invariant feature representations.

3.1 Question Answering Model

The QA model is trained (finetuned) on the DistilBERT [3] question answering model and the *in-domain* question, context and answer triplets $\{\mathbf{q}, \mathbf{c}, \mathbf{y}\}$. During *in-domain* training, the model is trained to minimize the cross entropy loss $\mathcal{L}_{QA}$ (Equation 1) of the start and end positions between answer $\mathbf{y}^{(i)}$ and predicted probabilities $\mathbf{p}^{(i)}$ of all *in-domain* training samples.

$$\mathcal{L}_{QA} = -\frac{1}{2N}\sum_{i=1}^{N}\left(\mathbf{y}_{\text{start}}^{(i)}\log \mathbf{p}_{\text{start}}^{(i)} + \mathbf{y}_{\text{end}}^{(i)}\log \mathbf{p}_{\text{end}}^{(i)}\right) \tag{1}$$

Notice that here we average the losses of *start* and *end* positions for each samples. This model is the baseline model that adopted from the given code.

3.2 Domain Discriminator

As proposed in [9], one way to let the QA model learn domain-invariant features is by using a domain discriminator. The main purpose of the discriminator is used to discriminate the domains for each training samples using cross entropy loss.

For K *in-domain* domains, the discriminator is trained with the $\mathcal{L}_D$ loss as shown in Equation 2, where $\mathbf{p}_D^{(i)}$ is the predicted probability for the true domain label of the ith sample by forwarding the last layer *CLS* embeddings to an MLP and $\mathbf{y}_D^{(i)}$ is the domain label for ith sample.

$$\mathcal{L}_D = -\frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_D^{(i)}\log(\mathbf{p}_D^{(i)}) \tag{2}$$

3.3 Domain Invariant Loss

To make the model learn domain invariant features, the model needs to be trained with an additional loss $\mathcal{L}_I$. Here we introduce three approaches of calculating $\mathcal{L}_I$, including the the Kullback-Leibler divergence, Jensen-Shannon divergence and the fake label.

3.3.1 Kullback-Leibler Divergence

Kullback-Leibler (KL) divergence measures how probability distribution of predicted domain q diverge from expected probability of domain distribution. This is the loss used in [9] that shows promising results on domain-invariant feature learning and implementation is available.

As we want the QA model to learn domain-invariant feature representations, the expected probability of domain distribution p should be a discrete uniform distribution with respect to number of domains K that each domain should have a PMF of $\frac{1}{K}$.

Therefore, the KL divergence can be described as

$$\mathcal{L}_I = \sum_{i=1}^{N} D_{KL}(p^{(i)}||q^{(i)}) = \sum_{i=1}^{N}\sum_{x\in K} p^{(i)}(x)\log(\frac{p^{(i)}(x)}{q^{(i)}(x)}) \tag{3}$$

where i denotes the ith sample.

3.3.2 Jensen-Shannon Divergence

Jensen-Shannon (JS) divergence is based on KL divergence and offers a symmetric and smoother approach of calculating divergence between two probability distributions. It is often used in computer vision and but seems have not been used in NLP QA related tasks. Using the aforementioned settings, the JS divergence in domain QA can be calculated as shown in the Equation 4.

$$\mathcal{L}_I = \sum_{i=1}^{N} D_{JS}(p^{(i)}||q^{(i)}) = \frac{1}{2}\sum_{i=1}^{N} D_{KL}(p^{(i)}||\frac{p^{(i)}+q^{(i)}}{2}) + D_{KL}(q^{(i)}||\frac{p^{(i)}+q^{(i)}}{2}) \tag{4}$$

3.3.3 Fake Label

The last approach is called the fake label, which is also often used in computer vision but not used in the NLP QA. Instead of measuring the distance between two distributions, we can create a fake label D' for each samples. The model learns to predict all domain labels on the fake label D'. The additional loss is

$$\mathcal{L}_I = \mathcal{L}_{D'} = -\frac{1}{N}\sum_{i=1}^{N} \mathbf{y}_{D'}^{(i)} \log(\mathbf{p}_{D'}^{(i)}) \tag{5}$$

3.4 Adversarial Training

In general, the QA model is trained with the QA loss $\mathcal{L}_{QA}$ and an additional loss $\mathcal{L}_I$ with a importance weight factor λ on the second loss.

$$\mathcal{L} = \mathcal{L}_{QA} + \lambda\mathcal{L}_I \tag{6}$$

The QA model is trained in adversarial manner. The model is at first trained with loss $\mathcal{L}$ and updates the model parameters. Then the discriminator will be updated by back-propagating the loss $\mathcal{L}_D$. In another word, during the first step, the model tries to learn both QA features and domain invariant feature representations. In the second step, the discriminator tries to make domain specific predictions. In this manner, we argue that the model may learn to better separate features from *in-domain* training datasets into QA specific and domain-invariant features and then the domain invariant features can be useful to the cross-domain tasks.

4 Domain Specific Data Augmentation

Through adversarial training, the model can learn domain-invariant feature representations from *in-domain* datasets. Such domain-invariant feature representations can be further balanced and learned from the *oo-domain* training datasets. To improve the model performance on out-of-domain question answering, model finetuning with *oo-domain* samples may be important. However, finetuning on few out-of-domain samples might not be sufficient. We propose and implement by ourselves a simple but effective data augmentation technique that can retain the domain information and augment *oo-domain* samples rapidly.

4.1 Named Entity Recognition

Named entity recognition locates and maps entity words to some pre-defined categories and is a challenging natural language processing problem. Using the named entity recognition, a list of words is extracted and classified into a relatively high-level categories. For question paragraph pair $(\mathbf{q}, \mathbf{c})$, the named entity recognition can be described as following

$$\mathbf{E} = \{w|(w, e) \in s(\mathbf{q}, \mathbf{c})\} \tag{7}$$

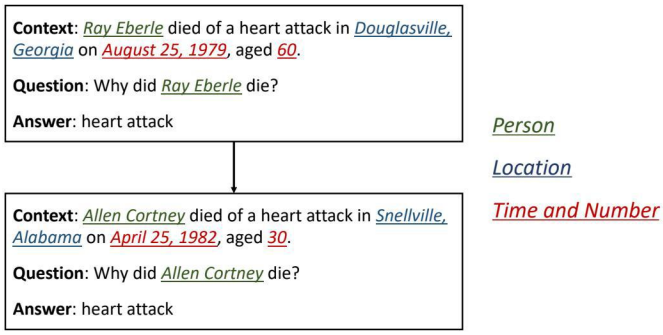

Figure 2: An example of the augmented sample using the proposed augmentation approach.

Here s is a named entity recognition model and $\mathbf{E}$ is the set of extracted words that each word has a recognized entity e. Therefore, the extracted words in $\mathbf{E}$ can be treated as the "keys" for the question paragraph pair $(\mathbf{q}, \mathbf{c})$.

4.2 Named Entity for Different Domain QA

Different question answering datasets usually have different domain focuses. The distinct focuses can lead to different number of named entities for each entity category. We argue that the extracted named entities contains rich domain information. For example, in the DuoRC dataset [15] from movie reviews, the entities are usually people, location and time. Whereas in the RACE dataset [16], there are not many people entities as that corpus is mostly from examinations. The different number entities in person entity category exhibits some degree of different domain focuses.

4.3 Word Replacement with Named Entity Recognition

Word replacement, including the synonym replacement, is frequently used as data augmentation method in natural language processing. Given the extracted named entity word set $\mathbf{E}$, instead of randomly choosing words to replace, we perform the word replacement on the words that appear in $\mathbf{E}$. The replaced word is described as following equation

$$w = \underset{v}{\operatorname{argmax}}\left(\frac{\mathbf{h}_w \cdot \mathbf{h}_v}{||\mathbf{h}_w|| ||\mathbf{h}_v||}\right), w \in \mathbf{E} \text{ and } w \in (\mathbf{q}, \mathbf{c}) \tag{8}$$

For each word w in the extracted set $\mathbf{E}$ and in the question paragraph pair, we will find the most similar word v by searching through some low dimension embedding space and replace the word w by v. After replacing all words in the extracted set $\mathbf{E}$, a new sample $(\mathbf{q}', \mathbf{c}')$ is generated. In this way, the training set is augmented and guided by named entities, and still retain a high level domain information. Figure 2 is an example of generated sample using our proposed augmentation approach.

5 Experiments

We design and conduct several experiments with the given *in-domain* and *oo-domain* datasets and show the final performance by using the baseline and our approaches on the *oo-domain* test set.

5.1 Data

We use *in-domain* training set for all model training and finetuning on the *oo-domain* training set. For data augmentation, we augment (double) the *oo-domain* dataset using proposed augmentatin method.

5.2 Evaluation method

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

Model	EM	F1 Score
Baseline	**42.362**	**60.24**
Baseline + Freeze finetune	42.294	60.049
JS + Augmentation + Finetune	40.229	57.547
JS + Augmentation + Freeze finetune	39.541	58.126

Table 1: Experiment results on *oo-domain* test set.

5.3 Experimental Details

During *in-domain* training the best model is chosen by the best performance on *in-domain* validation set and during finetuing the best model is chosen by the best performance on *oo-domain* validation set. For our proposed augmentation method, we use the NER method from spaCy [17] and use GloVe [18] embeddings. All the models, including baseline and models with adversarial methods, are trained (finetuned) on the *in-domain* training set with the default parameter. For finetuning, we implement in two ways. The first one is to freeze all layers excluding the last linear layer in the QA system. The other is to finetune all the parameters. The training time for all models is about 20 hours and the finetuning time is about 1 hour. We set the λ value to 0.5, use *cls* embedding as the input to discriminator, and use a three-layer MLP as the discriminator. These three settings and some discriminator implementation details such as the dropout layers in discriminator are referred from the MRQA [9] implementation [1]. All models are trained with are the default hyperparameters. For models finetuned without freezing any layers, the best model is chosen from the model finetuned with learning rates $10^{-5}, 2 \cdot 10^{-5}, \cdots, 7 \cdot 10^{-5}$. For models freezing all layers excluding the last layer, the learning rates are $10^{-3}, 2 \cdot 10^{-3}, \cdots, 7 \cdot 10^{-3}$.

5.4 Results

The models performance on the test set is shown in Table 1. On test set, it does not seems that our approaches offer better performance. However as shown in Table 2, *in-domain* adversarial training and *oo-domain* data augmentation generally improve the performance on *oo-domain* validation set. The two JS divergence adversarial models with augmentation achieve higher F1 and EM on *oo-domain* validation set. The discrepancy between validation and test are not what we expected.

We first try to update all the parameters during finetuning but it might make the model easily forget feature representations learned during pretraining and also easily results in overfitting. Then we freeze all the layers except for the `qa_outputs` layer to only update the parameters of `qa_outputs` layer, so as to utilize what the model has learned from *in-domain* training. For example, the Jensen-Shannon divergence adversarial model with augmentation and BERT layer freezing has similar performance with the model without freezing on the test set. That's probably because the *oo-domain* training set is so small and is not representative enough. The model fails to learn sufficient feature representations across different domains. Also, the validation set is also small and might not fully represent the data distribution of the test set. In addition, the test *oo-domain* datasets are not as evenly distributed as the validation set. The dataset issue might cause the model with last layer finetuned on *oo-domain* has even worse performance, shown in the table. These are the possible reasons causing the discrepancy between the model performance on the validation set and test set.

6 Analysis

We analyze our approach on the robustness of different adversarial training methods and different augmentation approaches. Although the experiment results on the test set are not expected, our analysis suggest the proposed methods can still be promising in improving the model robustness.

6.1 Robustness of Adversarial Methods

To measure the robustness of different adversarial methods, we at first train (finetune on *in-domain* training set) one model for each adversarial loss functions, including the baseline model, using the

[1] https://github.com/seanie12/mrqa

F1 Score	EM	Adversarial	Augmentation	Freeze
50.27	35.60	-	-	-
50.57	36.39	-	✓	-
50.50	36.39	KL	-	-
51.56	**36.13**	KL	✓	-
50.08	35.60	JS	-	-
51.16	37.17	JS	✓	-
47.47	31.94	-	-	✓
47.01	30.89	-	✓	✓
46.40	29.84	KL	-	✓
46.68	30.37	KL	✓	✓
47.80	31.68	JS	-	✓
49.04	32.46	JS	✓	✓

Table 2: Experiment results on *oo-domain* validation set.

EM	F1	Synonym Replacement	Word Drop	Entity
35.60	49.74	✓	✓	-
36.39	49.85	✓	-	-
36.69	**50.57**	-	-	✓

Table 3: Summary of model performance on different augmentation methods.

default hyperparameters. In this step, four models are trained. Then, we finetune each model on the *oo-domain* training set with 7 different learning rates and thus we have 28 finetuned models. Notice that here we finetune on all parameters as we want to check the performance and learning process of the whole model parameters on unmet domain (*oo-domain*) data. Each of the 28 models during the finetuning is evaluated on the *oo-domain* validation set after finetuned with every 160 samples. All models are finetuned with 20 epochs. Figure 3 and Figure 4 show the averaged F1 score and EM for the baseline model and models with different loss functions.

6.1.1 F1 score

As shown in Figure 3, adversarial methods seem to be more robust that they have higher F1 scores than that of the baseline model during the full parameters finetuning and evaluated on the validation set. The robustness is more apparent when the model is finetuned to be relatively overfitted, during the finetuning steps from 8000 to 15000. The model using Kullback–Leibler divergence shows the best performance and strongest robustness. This might make sense because the discriminator tends to predict one label for each sample, but KL divergence try to balance on the domain label distribution. The Jensen–Shannon divergence is much smoother and thus the general balance effect is less apparent than that of the KL approach. The fake label approach, tries to predict all samples to just one additional fake label, might add more noise to the model and thus have worse performance.

6.1.2 EM

As shown in Figure 4, the adversarial methods might not improve the model performance on exact match and there is no apparent robustness when the models are finetuned on the *oo-domain* datasets. It seems that the learned domain invariant representation only makes the models achieve relatively high EM faster and the model using the JS divergence has relatively good result. We conjecture that compared to other adversarial methods which shift the model relatively a lot and make the exact match performance a little bit hard. But because the Jensen–Shannon is a relatively smoother one that it does not shift the model a lot and thus it can balance relatively well on exact match.

6.2 Augmentation Performance

Similar to the settings in the last subsection, we also tested various augmentation approaches performance finetuned on the *oo-domain* training set and evaluated on the validation set. Other augmentation

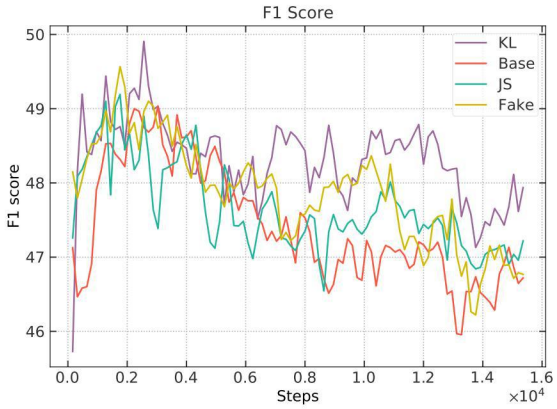

Figure 3: The average F1 score of the models during finetuning on *oo-domain* training set. The F1 score is averaged on 7 models (with different learning rate) for each of the 4 models.

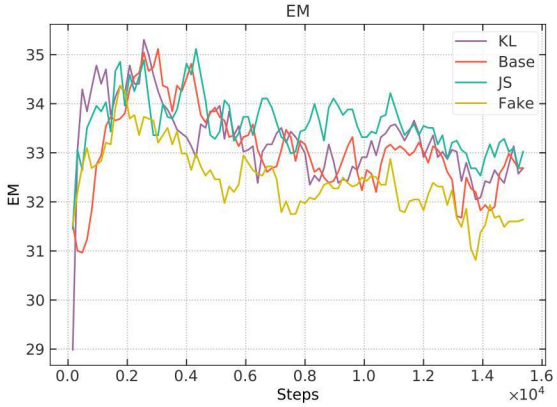

Figure 4: The average EM of the models during finetuning on *oo-domain* training set. The EM is averaged on 7 models (with different learning rate) for each of the 4 models.

approaches include synonym replacement and word dropping using existing augmentation implementation [2]. The result is shown in Table 3. This shows that our method can achieve relatively better results than compared augmentation methods when full model is finetuned on the *oo-domain* training set and evaluated on the *oo-domain* validation set.

7 Conclusion

In this project, we explore various adversarial learning approaches, the loss functions, on the out-of-domain question answering task and propose a new named entity guided data augmentation method that can retain domain information. The experiment results on the *oo-domain* test set do not match our expectations. There are several conjectures including that no hyperparameter tuning when pretraining the model due to the limitation of computing resouces, relatively small and non-representative *oo-domain* training set, and biased choice on the best model using small and non-representative *oo-domain* validation set. Our analysis suggestes that our approaches can be promising and the future work includes training by trying more model hyperparameters, choosing more advanced named entity recognition method, and mixing batches of in-domain and out-of-domain data for our augmentation method.

[2]https://github.com/searchableai/KitanaQA

References

[1] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. *arXiv e-prints*, page arXiv:2006.03654, June 2020.

[2] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv e-prints*, page arXiv:1907.11692, July 2019.

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

[4] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv e-prints*, page arXiv:1406.2661, June 2014.

[5] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv e-prints*, page arXiv:1609.05473, September 2016.

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *arXiv e-prints*, page arXiv:1505.07818, May 2015.

[7] Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. Adversarial training for cross-domain Universal Dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[8] Huazheng Wang, Zhe Gan, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, and Hongning Wang. Adversarial Domain Adaptation for Machine Reading Comprehension. *arXiv e-prints*, page arXiv:1908.09209, August 2019.

[9] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. In *Association for Computational Linguistics (ACL)*, 2019.

[10] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level Convolutional Networks for Text Classification. *arXiv e-prints*, page arXiv:1509.01626, September 2015.

[11] Jason Wei and Kai Zou. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. *arXiv e-prints*, page arXiv:1901.11196, January 2019.

[12] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. *arXiv e-prints*, page arXiv:1804.09541, April 2018.

[13] Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data Augmentation for Low-Resource Neural Machine Translation. *arXiv e-prints*, page arXiv:1705.00440, May 2017.

[14] Akari Asai and Hannaneh Hajishirzi. Logic-Guided Data Augmentation and Regularization for Consistent Question Answering. *arXiv e-prints*, page arXiv:2004.10157, April 2020.

[15] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension. *arXiv e-prints*, page arXiv:1804.07927, April 2018.

[16] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding Comprehension Dataset From Examinations. *arXiv e-prints*, page arXiv:1704.04683, April 2017.

[17] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[18] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.