Domain-Adversarial Training For Robust Question-Answering

Stanford CS224N Default Project

Aditi Talati
Department of Computer Science
Stanford University
atalati@stanford.edu

Abstract

In this project, we are building question-answering systems that can be robustly extended to answer questions on domains outside the training question-answering datasets. We used domain-adversarial training methods to improve upon the baseline of a pretrained DistilBERT system by creating a domain classifier that attempts to classify the domain of the input based on the last hidden layer of the DistilBERT model. This accomplished some improvement on the baseline model, with a score of EM = **41.353** and F1 = **59.803** on the test set, and a score of EM = **31.675** and F1 = **48.836** on the validation set.

1 Key Information

- Mentor: Mandy Lu

2 Introduction

The question of robust learning is an important one, especially with the increase of large pretrained models such as BERT. It is very easy for such models to overfit to any given training dataset, allowing it to perform very highly on test data from the given dataset, but then adapt much more poorly to new out-of-domain datasets. The goal of making robust AI allows for tools that adapt very quickly to new situations, and this is the goal we are trying to accomplish with this paper.

One common method to extend a learning process to be more robust is the process of adversarial learning, where there is a second model responding to the data in some way that is adversarial to the model we are training, so that our training model is more fit for a robust variety of situations. In our method, we are trying to make our model domain-independent by simultaneously trying to train a discriminator, which takes the final hidden layer of the model as input and tries to predict which domain the input came from. Then, as our model is trained to minimize its loss and maximize the loss of the domain discriminator, it is forced to prioritize features that are domain-invariant, so that they don't clue the discriminator into learning what domain the hidden layer came from.

By building this structure around a DistilBERT pretrained model, we were able to get an F1 score of 48.806 on the validation set, which improved upon the baseline score by 0.404.

3 Related Work

There are many different approaches to improving robustness in the field of question-answering. Each of these approaches can be built upon DistilBERT, which is a transformer-based pretrained model to be trained on specific tasks, such as the reading comprehension task we are using here. DistilBERT is a smaller version of BERT, which is is optimized so that it performs only slightly worse on given tasks using much less computing power than the larger model BERT [1].

Some such approaches are focusing on fine-tuning the hyperparameters of the model itself, such as the learning rate and the number of epochs the model fine-tunes for [2]. Because the size of the fine-tuning data sets are often very small, modifying these hyperparameters can cause large changes in the quality of the results given [2]. Another such approach is meta-learning, where instead of training the model specifically on the given input, you train a model that takes in the new domain input and gives as an output a modified version of the question-answering model, which is specialized to deal with the input at hand [3].

The approach we used is adversarial learning. Adversarial learning in general is the process of training on adversarial examples, or the sorts of examples that the model tends to fail on during testing time. The adversarial learning we are using in this paper takes a slightly different approach, where both a question-answering model and discriminator are trained simultaneously, and the goal of the question-answering model also becomes to maximize the loss of the discriminator. This was discussed in a 2017 paper that built a cross-domain dependency parser using adversarial training, wherein each input was passed to both a domain-specific LSTM and a domain-agnostic LSTM [4]. The output from these were passed through one shared gate to get an output prediction for the dependency tree of the dataset and another shared gate to reach the domain classifier LSTM [4]. The point of this paper was to prioritize features that are shared among all domains while still preserving important domain-specific features in the smaller domain-specific LSTM [4]. However, this was an older paper and still used LSTM's instead of transformers, and it is not exactly specific to our task, as the goal was to train dependency trees rather than answer reading comprehension questions.

The paper that this project was based upon was a 2019 paper for the MRQA shared task. The group for this paper accomplished the shared task of what they called domain-agnostic question answering by building on top of BERT a model that is divided into two parts, so that the final hidden layer outputted from the BERT question-answering model is passed through the discriminator, or domain classifier, in order to predict both the answer to the question and the domain the question came from. Then, the question-answering model optimized to minimize the loss from the question-answering prediction but maximize the loss for the domain classifier, so that the question-answering model would prioritize domain-invariant features [5]. This model was built upon BERT, and the question we are answering is whether such a model would perform similarly well when built upon a smaller pretrained model such as DistilBERT.

4 Approach

For this paper, we adapted the adversarial training model from the 2019 paper to work for DistilBERT, and then ran experiments changing a few of the hyperparameters for the adversarial training.

4.1 Math

The math section for this paper is adapted from the math section for the Domain-Agnostic Question Answering with Adversarial Training paper, since we are using the model from this paper [5].

We can describe the problem as a set of K in-domain datasets (in our case, $K = 3$) $\mathcal{D}_i$, where each dataset is a set of N_i question-context-answer triples $\mathbf{q}_k, \mathbf{c}_k, \mathbf{y}_k$. Then, the model learning from $\{\mathcal{D}_i\}_{i=1}^{K}$ predicts the answer $\mathbf{y}_l$ from the question-context pair $\mathbf{q}_l$, $\mathbf{c}_l$ for each question-context pair in the set $\left\{\mathcal{D}_j\right\}_{j=1}^{L}$ of out-of-domain datasets. (In our situation, $L = 3$ since we are evaluated on three out-of-domain datasets.)

Then, our question-answering model is a standard question-answering model from DistilBERT, as given to us by the starter code for the default final project. We are trying to minimize the negative log-likelihood of an answer $\mathbf{y}$ for the in-domain datasets. Since on our reading comprehension questions, we expect the answer $\mathbf{y}$ to be a portion of the context for our question, we represent $\mathbf{y}$ as $(\mathbf{y}_s, \mathbf{y}_e)$, which are the start and end positions of the answer within our context paragraph. Then, we are trying to minimize the sum of

$$-\log P_\theta(\mathbf{y}_{s,k}|\mathbf{q}_k, \mathbf{c}_k) - \log P_\theta(\mathbf{y}_{e,k}|\mathbf{q}_k, \mathbf{c}_k)$$

over all questions and contexts in all the training datasets, which we will call $\mathcal{L}_{\text{QA}}$.

Then, we will create an adversarial loss for our model, so that our final model loss is $\mathcal{L}_{\text{QA}} + \lambda\mathcal{L}_{\text{adv}}$, where λ is a hyperparameter that decides how much the adversarial loss is valued in the final calculation.

To calculate $\mathcal{L}_{\text{adv}}$ we first consider the discriminator, which is a machine learning model with loss $\mathcal{L}_D$. Here, we are trying to minimize the negative log-likelihood of the probability of a domain category given the final hidden state of the question-answering model, or the sum over all questions and contexts over all domains of

$$-\log P_\phi(l_i|\mathbf{h}_i),$$

where $\mathbf{h}_i$ is the hidden layer and l_i is the domain.

Then, the adversarial loss tries to minimize the difference between the output of the discriminator's prediction and a random prediction, which it takes to be the uniform prediction $U(l)$. It does so by calculating the Kullback-Leibler(KL) difference between these two, so that the adversarial loss is the sum over all contexts and questions of

$$\text{KL}(U(l), P_\phi(l_i, \mathbf{h}_i)).$$

4.2 Baseline

The baseline for the model is the default baseline described in the Robust Question-Answering default final project description. It trains a DistilBERT question-answering model on the three in-domain datasets and evaluates them on the three out-of-domain datasets.

4.3 Code

The code for our project was built upon the starter code given for the default final project and was an adapation of the code for the Domain-Agnostic Question Answering with Adversarial Learning paper to work for our given task.

There are three main modifications we made to the starter code. Instead of using the default DistilBERT question-answering model, we used the model from the Github for the paper, which combined a question-answering model and a discriminator (which was simply a combination of linear layers and ReLUs) in the way described in the math section above. We then modified the model from the paper so that the question-answering model used was a DistilBERT model instead of a Bert model and the input taken was in the format given by the starter code, which is more easily compatible with DistilBERT, rather than the format assumed in the original code. Then, we modified the evaluation function from the starter code so that it was compatible with the evaluation output for this model rather than assuming the evaluation output was that of a standard DistilBERT model. We also changed the way that the model was saved from training and loaded for evaluation so that it was loading the full model and not just changing the parameters for the DistilBERT model, because this is what allowed all the layers of our model to be saved for the evaluation process.

Then, the next modification we made was to the Trainer class of the train.py file, where we saved the original Trainer class as a BaseTrainer and made a subclass called AdvTrainer, which modified the train method of the BaseTrainer to call the forward method and calculate the loss of both the question-answering model and the discriminator. In order to do so, we used a method to calculate the running average loss which was taken from the github for the paper.

The final modification we made was for the discriminator to work properly. For this to be the case, we needed to store not only the input data as a question, context, answer triple, but also store a fourth key, which we called "label," which had numerical ID value depending on the domain the question, context, answer triple was taken from. This label is what the discriminator would predict. Thus, we changed the processing of the input data to store this label alongside the question, context, and answer it was already storing.

5 Experiments

We ran three experiments on this data, besides the baseline evaluation.

5.1 Data

The datasets we are using are provided by the default Robust Question Answering project. The three in-domain datasets we are given are the Stanford Question-Answering Dataset (SQuAD), the News Question Answering Dataset (NewsQA), and the Natural Questions dataset. The three out-of-domain datasets are Duo Reading Comprehension (DuoRC), Reading Comprehension Dataset from Examinations (RACE), and RelationExtraction. These datasets are all different forms of reading comprehension data, from different sources, and questions from the out-of-domain dataset appear in small quantities in the training data. For more information on the datasets, see the Robust Question-Answering default final project handout. The data from the dataset appears in a question-context-answer triple, which we also parse to include a label that tags each triple with the ID number (from 0 to 5) that we've given each dataset. The question-answering model is given the question and context and is asked to predict the answer (in terms of start and end positions within the context) while the discriminator is given the last hidden layer of the question-answering model and asked to predict the label.

5.2 Evaluation method

The evaluation methods we are using are the ones given to us by the default final project, and they are both quantitative evaluation methods used to measure the accuracy of the predicted answers to the reading comprehension questions. The way the evaluation methods work are as follows:

There are three human-expert responses to each of the questions. The exact match (EM) score for a question is 100 if the model prediction exactly matches one of the three human expert answers, and 0 otherwise. Meanwhile, the F1 score for a given question and human answer is a harmonic mean of precision (p) and recall (r); that is, it is $\frac{2pr}{p+r}$. Here, precision is the percent of the model answer that appears in the human answer, and recall is the percent of the human answer that appears in the model answer. This means that precision measures the model's ability to not include excess information, and recall measures the model's ability to include all the requisite information. The maximum F1 score across all three human answers is taken.

Then, the overall F1 and EM scores are the average such scores over the whole validation or test dataset.

5.3 Experimental details

When running these experiments, we mainly used the default hyperparameters set by the starter code and the adversarial training paper. This meant we had a batch size of 16, a learning rate of 0.00003, trained for 3 epochs, and evaluated every 5000 iterations. The other defaults are taken from the adversarial training paper, and they are the hidden layer size of 768 for the model and discriminator, the dropout rate of 0.1, and the number of layers for our discriminator, which was 3. Moreover, the λ, which is as described in our math section, was set to a default of 0.01.

We ran one experiment with these defaults, but we also ran a large discriminator experiment, where the discriminator had 4 layers and λ was 0.05, and a small discriminator experiment, where the discriminator had 2 layers and λ was 0.05. We used the same standard as used in the 2019 paper, which was that we picked the checkpoint for each model that produced the highest F1 score on the validation set as the checkpoint we would use to compare the models [5].

5.4 Results

Model	**validation**		**test**	
	EM	F1	EM	F1
baseline	**33.246**	48.432	–	–
default adversarial	31.675	**48.836**	**41.353**	**59.803**
big discriminator	32.461	48.006	–	–
small discriminator	29.319	47.233	40.803	59.713

Table 1: Model performance on validation and test set.

Note that the big discriminator also received EM and F1 scores of 41.284 and 59.434 on the test set, but these were using the latest checkpoint, not the one that recieved the highest validation score, so these cannot be directly compared.

In general, the F1 and EM scores obtained here are worse than the F1 and EM scores obtained on the adversarial training paper, which obtained an average EM score of 44.10 and F1 score of 56.15 on the validation set and an EM score of 48.59 and F1 score of 64.68 on the test set. (This was when taking the model checkpoint that performed highest on the validation set.) However, when we look at improvement compared to the baseline, we can see that the adversarial model in the paper improved the F1 score by around 2 points compared to the baseline, which implies that at least part of the score increase for adversarial training in that paper compared to this experiment is due to the fact that said paper built their model on BERT, while we built our model on the smaller DistilBERT. Moreover, if we look more closely at the different training sets for the adversarial training paper, we can see that while adversarial training performed 4 points better in F1 on DuoRC, it only performed 0.2 points better in F1 on RACE and did worse on RelationExtraction. Thus, we can see that our adversarial training, which scored about 0.4 points better than the baseline in terms of F1 score, did about expected as compared to the paper we based it on.

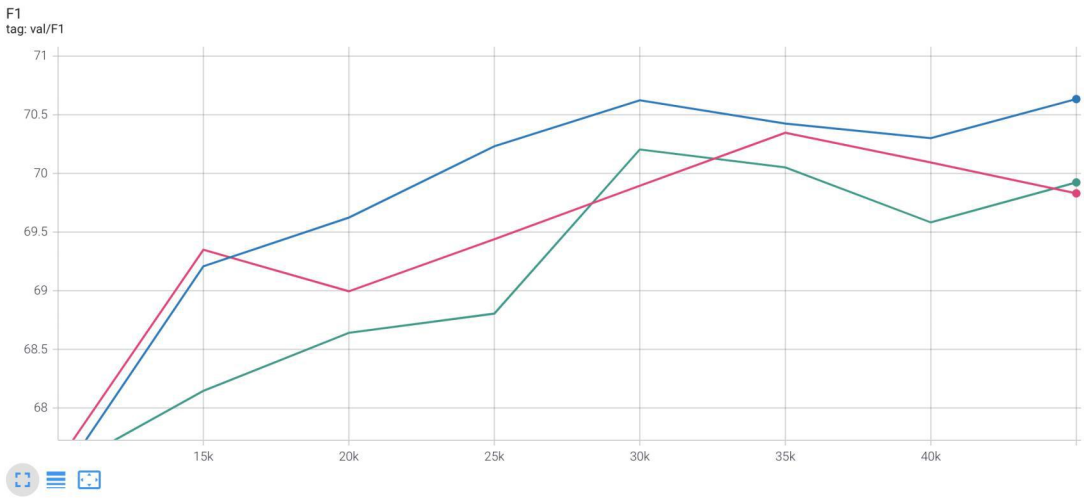

Figure 1: A graph of the F1 scores on the validation data of the three models over different training iterations, taken from TensorFlow. Default adversarial training is in blue, big discriminator is in pink, and small discriminator is in green.

Moreover, the paper claimed that they found 0.01 to be the optimal λ value for their adversarial training model. We can see that both increasing and decreasing the value of adversarial loss in the model decreases the F1 score of the model, albeit slightly. This is consistent with the results found in the paper.

6 Analysis

We can consider the performance of the model as compared to the baseline on some sample validation questions. The first interesting question is the following:

Question: which river separates the bronx in new york city from manhattan island

Context: BPB The Hudson River separates the Bronx on the west from Alpine , Tenafly and Englewood Cliffs in Bergen County , New Jersey ; the Harlem River separates it from the island of Manhattan to the southwest ; the East River separates it from Queens to the southeast ; and to the east , Long Island Sound separates it from Nassau County in western Long Island . Directly north of the Bronx are (from west to east) the adjoining Westchester County communities of Yonkers , Mount Vernon , Pelham Manor and New Rochelle . (There is also a short southern land boundary with Marble Hill in the Borough of Manhattan , over the filled - in former course of the Spuyten Duyvil

Creek . Marble Hill 's postal ZIP code , telephonic area codes and fire service , however , are shared with the Bronx and not Manhattan .) EEPE

Answer: Harlem River

Though the answer is clearly the Harlem River, both the baseline and adversarial model predicted that the answer was the Hudson River. This implies that the attention for the model is off, because it would have seen that the Hudson River is the closest river to "separates the Bronx" and gone with that, instead of reading further context and realizing both that all the other rivers in the list started from the Bronx and that the Harlem River is the one that "separates it from the island of Manhattan." This implies that attention in the model needs to be modified to improve the long-term reading comprehension, and that this has not been changed through the adversarial model.

The next interesting question is based on a CNN article, and again both the adversarial model and the baseline model got the question wrong, though in different ways. The **question** here was "What was the allegation?" based on a news article about an actor who was sentenced to prison. The correct **answer** here was "attempted burglary," based on the **context** sentence "The Bronx County District Attorneys Office had sought the maximum of 15 years for Lillo Brancato Jr., who was convicted last month of attempted burglary stemming from a fatal encounter with police officer Daniel Enchautegui." However, both the baseline model and the adversarial model made predictions based on the context sentence "An actor who played a wannabe mobster in 'The Sopranos' was sentenced Friday to 10 years in prison for a botched burglary that left an off-duty New York police officer dead." That is, the baseline model predicted that the answer was "botched burglary" and the adversarial model predicted that the answer was "a botched burglary that left an off-duty New York police officer dead."

This is one question where perhaps having more domain-specific knowledge would be helpful, so that the model would know that "attempted burglary" is a more common name for a legal allegation than "botched burglary." Here, maybe a model like the shared gates adversarial network would be helpful, so that the model can also try to learn the domain-specific knowledge about the potential names for allegations. This is the only information that would seriously distinguish between the predicted answer and the correct answer, because the context sentences simply describe the former as what he was sentenced for and the latter as what he was convicted for.

Finally, we will consider one case where our adversarial model answers the question correctly and the baseline doesn't.

Question: What did he say at the university?

Answer: "I never thought any of this was going to be easy."

The context for this question is a CNN article about Obama giving a speech at a New Orleans university, and it is a complicated question because the entire context paragraph discusses a speech he gave at the university. The prediction the baseline model made was based on the context sentence "Before a cheering crowd at a town hall meeting in New Orleans, President Obama fired back at critics who accuse him of accomplishing little in his nine months in office, saying 'I'm just getting started.'" Meanwhile, the correct answer occurs a couple lines later, where the article states, "'I never thought any of this was going to be easy,' said Obama, speaking at the University of New Orleans in his first visit to the Gulf Coast city since taking office."

While the baseline model seems to have found the first line spoken by Obama in the context paragraph, the adversarial model seems to have recognized that it should be looking for both "university" and "Obama" in the context question, and therefore found the sentence that described the main point of the speech, where Obama was described to be speaking at the university. This improved searching for keywords would be one of the domain-invariant features that the adversarial model would have focused on, while the baseline model might have focused more on features that are domain-specific to the in-domain set.

Overall, we can see qualitatively that in a few specific cases, the baseline model improves upon the adversarial model in terms of finding keywords to search for, but doesn't improve upon the baseline in terms of having domain-specific knowledge or improved attention.

7 Conclusion

In this project, we were able to build a domain-adversarial model on top of DistilBERT that improved the model's robustness in performing question-answering tasks on domains that were not heavily present in the training set. Using this model, we got an improved F1 score of around 0.34 more than the baseline on the validation set. However, our model could be improved by focusing on details like modifying the model attention or adding a layer that does prioritize domain-specific information.

References

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

[2] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. Revisiting few-sample bert fine-tuning, 2021.

[3] Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. Investigating meta-learning algorithms for low-resource natural language understanding tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1192–1197, Hong Kong, China, November 2019. Association for Computational Linguistics.

[4] 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.

[5] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. *CoRR*, abs/1910.09342, 2019.