Domain Adversarial Training for QA Systems

Stanford CS224N Default Project
Mentor: Gita Krishna

Danny Schwartz
Stanford University
deschwa2@stanford.edu

Brynne Hurst
Stanford University
brynnemh@stanford.edu

Grace Wang
Stanford University
gracenol@stanford.edu

Abstract

In this project, we examine a QA model trained on SQuAD, NewsQA, and Natural Questions and augment it to improve its ability to generalize to data from different domains. We apply a method known as *domain adversarial training* (as seen in [1]) which involves an adversarial neural network attempting to detect domain-specific model behavior and discouraging this to produce a more general model. We explore the efficacy of this technique as well as the scope of what can be considered a "domain" and how the choice of domains affects the performance of the trained model. We find that, in our setting, using a clustering algorithm to sort training data into categories yields a performance benefit for out-of-domain data. We compare the partitioning method used by Lee et al. and our own unsupervised clustering method of partitioning and demonstrate a substantial improvement.

1 Introduction

One of the most challenging problems in deep learning is adapting models to out-of-domain data. (Out-of-domain here meaning data outside the training data distribution, and in-domain meaning data well-reflected by the training data distribution.) Question Answering (QA) models specifically do not generalize well to datasets that are significantly different than the data they are trained on. These models tend to overfit to in-domain data and require additional fine-tuning to achieve similar performance on other, out-of-domain datasets. We present a potential solution to this overfitting problem via *domain adversarial training*, as described in [1] by Lee et al.

Domain adversarial training is a method to modify a model's training objective to encourage the model to avoid domain-specific overfitting. As can be seen in Figure 2 (in Appendix A), the model is broken into two components: a domain discriminator and a QA model. The QA model is trained to predict answer spans given a training example context and a question. The role of the discriminator is to predict the domain of a training example from internal features learned by the QA model. The discriminator acts as a regularizer, pushing the QA model to learn domain-invariant features. After training, the discriminator can be discarded as it is not used in forward inference.

This method requires practitioners to select a group of domains that the training data belong to and partition each example into one of these domains. We show that the selection of this group of domains significantly impacts the effectiveness of this technique. Specifically, we propose an improvement to the strategy used in [1]. Rather than partition the data based on its original source (e.g., Wikipedia or CNN), we partition the data by extracting semantic and stylistic features from the text and using K-means clustering on those features. We show that using this partitioning technique improves performance on out-of-domain validation sets by a substantial margin when compared to a baseline model trained without a domain adversarial objective.

2 Related Work

A variety of techniques have been explored in recent NLP research to improve the out-of-domain performance of question-answering systems.

For instance, in [2], Gururangan et al. investigate the use of multiphase adaptive pretraining by further pretraining a transformer model with unlabeled data from the domain of a specific task. The authors present the performance gains from domain-adaptive pretraining alone, an improvement on top of that by adapting to task-specific unlabeled data, and another approach with task adaptation on an augmented corpus using simple data selection strategies.

In [3], Ribeiro et al. introduce the actionable *semantically equivalent adversarial rules* (SEARS) that are useful in detecting undesirable behavior (i.e., bugs) in black-box models for domains including machine comprehension and sentiment analysis. These bugs are often instances where replacing a single word with another that is almost semantically equivalent causes a model's behavior to change. Ribeiro et al. posit that certain semantically-similar word pairs that cause these bugs ('rules') can be used to create additional training examples based on existing training data. The authors demonstrate how to extract a set of rules from a model that can be used to generate semantically similar training examples that drive the model's behavior to avoid these kinds of bugs while maintaining accuracy.

The most important paper we encountered in designing this project was [1]. In this paper, Lee et al. attempt to build a QA model capable of performing well on out-of-domain data by constraining their model such that it learns domain-agnostic features. The authors begin by assuming the existence of what they call a performant domain-invariant classifier. This domain-invariant classifier does not have hidden features that identify question/context pairs as belonging to a specific domain so, theoretically, it should perform similarly across domains. The authors propose an adversarial network architecture and a corresponding loss function to optimize a BERT-based question answering model with this domain-invariance constraint. A QA model attempts to predict an answer, and a discriminator trains the QA model to learn domain-invariant features.

We chose to model our experiments based on this paper since the adversarial training mechanism is well-explained, the authors have made the code available on Github, and we are interested in GANs, which operate using a similar principle. Specifically, the authors do not explore different ways to select "domains", opting instead for the simplest possible approach: mapping each example to a domain encompassing all training examples from a particular source (e.g., SQuAD). This results in a very small set of domains used as the training data for their discriminator model, only 6. This raises the substantial risk of overfitting to common patterns in the 6 training domains used. We decided to replicate the approach Lee et al. used and experiment with various ways of partitioning training examples into domains.

3 Approach

As in [1], we use a 3-layer feed-forward neural network as a discriminator to classify the domain of training examples using a piece of the QA model's hidden state, $\mathbf{h}_{\text{CLS}}$, as input. Their original model architecture can be seen in Figure 2. We modify the model slightly by using DistilBERT as the pre-trained language model. We also define "domain" differently, as explained in Section 4.1.

To train the discriminator, we use the following loss function ,

$$\mathcal{L}_{\text{discrim}} = -\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{d}_k^{(i)} \cdot \log\left(\hat{\boldsymbol{d}}_k^{\,(i)}\right), \tag{1}$$

where $\hat{\boldsymbol{d}}_k^{\,(i)}$ is the discriminator's predicted probability that training example i belongs to domain k, N is the total number of training examples, and $\boldsymbol{d}_k^{(i)}$ is a one-hot vector that specifies the actual domain k that example i belongs to.

The job of the QA model is to trick the discriminator by learning domain-invariant features. The loss term for the QA model without the domain-invariance penalty is,

$$\mathcal{L}_{\text{QA}} = -\frac{1}{N}\sum_{i=1}^{N} \left[\boldsymbol{y}_s^{(i)} \cdot \log\left(\hat{\boldsymbol{y}}_s^{\,(i)}\right) + \boldsymbol{y}_e^{(i)} \cdot \log\left(\hat{\boldsymbol{y}}_e^{\,(i)}\right)\right], \tag{2}$$

where $\boldsymbol{y}_s^{(i)}$ is a one-hot vector that specifies the actual starting position s of the answer for example i, $\hat{\boldsymbol{y}_s}^{(i)}$ is the QA model's vector of predicted probabilities of starting positions for example i. Similarly $\boldsymbol{y}_e^{(i)}$ and $\hat{\boldsymbol{y}_e}^{(i)}$ encode the actual ending and predicted ending positions.

The domain-invariance term,

$$\mathcal{L}_{\text{invariance}} = \frac{1}{N}\sum_{i=1}^{N} KL(U||\hat{\boldsymbol{d}}^{(i)}), \tag{3}$$

for the QA model is the Kullback-Leibler divergence between the uniform distribution U over all domains and the discriminator's actual domain predictions. The goal here is to encode the information in the hidden states in such a way where it's impossible for the discriminator to distinguish between domains. This term effectively regularizes the network, making it more difficult to overfit to domain-specific patterns.

The full loss function for the domain-invariant QA model,

$$\mathcal{L}_{\text{composite}} = \mathcal{L}_{\text{QA}} + \lambda\mathcal{L}_{\text{invariance}}, \tag{4}$$

is composed with a new hyperparameter, λ, that emphasizes the relative importance of the invariance loss term. The authors of [1] recommend using 0.01 as the value of λ.

We used stochastic gradient descent with momentum to optimize the discriminator and we used the AdamW algorithm to optimize the QA model. For each batch of training data, we first compute $\mathcal{L}_{\text{composite}}$ to perform a parameter update on the QA model and we then compute $\mathcal{L}_{\text{discrim}}$ on the same batch to perform a parameter update on the discriminator. We configured our training procedure so that multiple discriminator updates could be performed for every QA update.

4 Experiments

4.1 Data

We trained our model with three in-domain datasets, and evaluated it with three out-of-domain datasets.

Dataset	Question Source	Passage Source	Train	Dev	Test
in-domain					
SQuAD [4]	Crowdsourced	Wikipedia	86,558	10,507	-
NewsQA [5]	Crowdsourced	News articles	74,160	4,212	-
Natural Questions [6]	Search logs	Wikipedia	104,071	12,836	-
out-of-domain					
DuoRC [7]	Crowdsourced	Movie reviews	128	128	1,503
RACE [8]	Teachers	Examinations	128	128	1,502
RelationExtraction [9]	Synthetic	Wikipedia	128	128	1,500

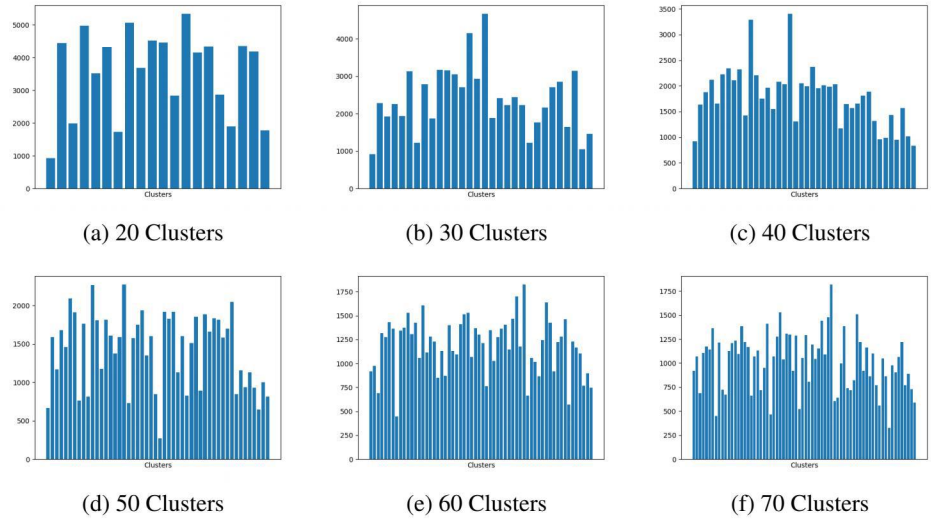
Table 1: Dataset statistics. These numbers indicate the number of passages in each dataset, not the number of questions.

We used the SQuAD 1.1 dataset, the NewsQA dataset, and the Natural Questions dataset for training, supplementing them with 128 examples from each of our out-of-domain datasets. Some examples from each of these datasets were separated for use as validation data. To support the domain adversarial training, we used the `scikit` [10] K-means algorithm to cluster the training examples into domains. To produce input features for clustering, we started by computing TF-IDF features for each context. TF-IDF is a method to compute how relevant an individual word is to a document in a collection of documents. Each example in our corpus could have an associated TF-IDF score for a particular word. Before applying TF-IDF, we cleaned each context by removing stop-words and lemmatizing each word in the context. After cleaning, we computed the TF-IDF vectors using

`scikit's TfidfVectorizer` [10], ignoring terms that occurred in more than 70% or less than 0.01% of the context paragraphs. We then kept the TF-IDF scores of the 300 remaining candidate terms with the highest document frequency. We found that increasing this number often led to extremely imbalanced clusters, so we empirically determined 300 to be a reasonably informative value without causing extreme cluster imbalance that would make training our discriminator difficult. After extracting the vector of TF-IDF features for each training example, we normalized the TF-IDF vectors by their L2 norm to have magnitude 1.

In addition, we extracted the following custom features from the raw, uncleaned context for each training example: average sentence length, maximum sentence length, minimum sentence length, percentage of adjectives, percentage of coordinating conjunctions, percentage of nouns, percentage of prepositions, maximum word repetition (maximum number of times one word is repeated in sequence), number of alphanumeric words, number of commas, average sentence sentiment (as computed by the `NLTK` library [11]), and number of unique words used. These custom features were normalized to have zero mean and unit variance across training examples, then they were scaled to the average magnitude of the TF-IDF features and multiplied by a tunable constant to modulate their relative influence in the K-means algorithm. After observing some cluster outputs, we determined that the best value to use for this constant was 6. We concatenated the scaled custom features and the TF-IDF features to produce a vector of features for each example. Before clustering, we normalized each of those vectors by their L2 norm so they would have magnitude 1.

Finally, we ran K-means with K = 20, 30, 40, 50, 60, and 70 to determine the best number of clusters. As can be seen in Figure 1, the results were well balanced for each run. We chose to test our QA model with 40 clusters because the 40 cluster set had the smallest difference between the largest cluster and the smallest cluster. We also chose to train the model with 20 clusters because we had hypothesized that a high number of clusters would inhibit the discriminator.



(a) 20 Clusters (b) 30 Clusters (c) 40 Clusters

(d) 50 Clusters (e) 60 Clusters (f) 70 Clusters

Figure 1: K-means Clustering Statistics. Each bar represents a cluster, and the y-axis of each plot is the number of training examples in the cluster. The 20 cluster set and the 40 cluster set were used during training.

4.2 Evaluation method

To evaluate performance of our model during training, we were specifically interested in monitoring the $\mathcal{L}_{\text{invariance}}$ and $\mathcal{L}_{\text{composite}}$. We expected to see $\mathcal{L}_{\text{composite}}$ trending downward for both the in-domain and out-of-domain data. We also expected $\mathcal{L}_{\text{invariance}}$ to reach a steady-state equilibrium, indicating that the discriminator was not able to learn to predict domains and the QA model was learning domain-invariant features.

To evaluate our output, we looked at the Exact Match (EM) and F1 metrics averaged across the entire dataset (in-domain was evaluated separately from out-of-domain). Exact Match is a strict metric,

requiring the model output to exactly match the ground truth answer. F1 is more forgiving, and is the harmonic mean of precision and recall. For questions with more than one ground-truth answer, we take the max of the EM and F1 scores.

To observe the trend in the described metrics throughout training, see Figure 4 in Appendix A.

4.3 Experimental details

4.3.1 Model Configurations

As a baseline, we used the QA model found in the starter code for the project without the additional domain adversarial objective. The rest of our experiments concern models that use the domain adversarial objective with different domain partitioning schemes. We used a domain partitioning scheme similar to the scheme used in [1] to compare their approach to our K-means-based approach. This partitioning scheme is denoted in our results table as “Source-Based" and simply maps each example to the dataset it originally came from (e.g., an example from SQuAD is in the “SQuAD" domain, etc.). We also evaluated our K-means-based partitioning scheme with a 40-cluster partition and a 20-cluster partition. Each of these three domain adversarial models used hyperparameters selected via individual searches as described in Section 4.3.2.

We fine-tuned each model for 3 epochs as that is what we had selected for our baseline. All of our models seemed to converge by this point. Our experiments would often take about two full days to train depending on the step multiplier we chose for the discriminator as part of our hyperparameter search. We used a batch size of 32 because we empirically determined that was the maximum that the hardware was capable of.

4.3.2 Hyperparameter Search

We used the `RayTune` [12] library to write a hyperparameter search routine to determine the best hyperparameters to use during training. We performed separate searches using a subset of our training data for our source-based clustering model, our 20-cluster model, and our 40-cluster model. Each search was run for 2 epochs over the data used. Table 2 contains the hyperparameters we selected to use on the full dataset. We ultimately selected our choice of hyperparameters because the KL Divergence and the QA model loss were trending down (see Figure 3 for training curves). This indicates that with these hyperparameters, the QA model was better able to trick the discriminator.

The "adversarial loss weight" hyperparameter is the λ introduced in [1]. The “step multiplier" is how many parameter updates were performed on the discriminator for every parameter update performed on the QA model. Increasing the step multiplier dramatically increased the training duration, so we did not explore values larger than 3.

QA Parameters			
	Learning Rate	**Weight Decay**	**Adversarial Loss Weight**
Source-Based	9.1803E-05	1.6613E-02	1.1364E-02
20-Cluster	5.22044177664971E-05	1.0524918464003E-03	0.119641324416047
40-Cluster	8.72772969749864E-05	8.55232316497858E-02	8.47815672455572E-03
Discriminator Parameters			
	Learning Rate	**Momentum**	**Step Multiplier**
Source-Based	6.2666E-06	0.89467	1
20-Cluster	8.75765078084943E-05	0.912875330349223	3
40-Cluster	1.93834168124229E-05	0.857915590911954	3

Table 2: Hyperparameters Used During Training

4.4 Results

Generally, the best validation performance we obtained was with the 40-cluster partition. Our best model (using K-means with 40 clusters to define the domains in the training set) obtained an EM

score of 40.528 and an F1 score of 58.408 on the out-of-domain test set. Considering the modest improvements seen in [1] (about 1.5-2 points higher on both EM and F1), we are fairly surprised that our clustering scheme was able to get 5 or more points of improvement in both metrics on our out-of-domain validation sets. Part of the improvement may be because of our inclusion of a small number of out-of-domain training examples, but this did not make our source-based model better than our baseline. There is an intuitive argument to be made about the efficacy of our clustering approach; the source-based approach does not attempt to prevent the QA model from overfitting to categories of examples within a single data source or across data sources. Our clusters were based on features that should not be particularly informative to the QA model in determining the answers to questions, so it makes sense that more broad regularization over these clusters leads to better out-of-domain performance.

The baseline model we used performed better on the in-domain validation datasets. This is to be expected, as removing the possibility of overfitting to domain-specific patterns will have an adverse impact on a model's performance on examples in that domain. Interestingly, our best-performing model on the out-of-domain validation sets is the second-best performing model on the in-domain validation sets. We believe this is at least partially because the strength of the regularization (the "adversarial loss weight" parameter in Table 2) was greater for our 20-cluster model and source-based model.

We believe that one reason our 40-cluster results were better than our 20-cluster results on the out-of-domain data is because the baseline model is overfitting to more than 20 distinct groups of examples in the training data. The 40-cluster domain adversarial objective is able to more thoroughly regularize the model over these large groups because 40 clusters can approximate them better than 20 can.

The full set of validation performance metrics that we obtained can be seen in Table 3.

in-domain (results on the validation set)								
Model	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
	SQuAD		**News QA**		**Natural Questions**		**Average**	
Baseline	**63.33**	**77.01**	**39.27**	**57.51**	**52.8**	**69.43**	**54.77**	**70.51**
Source-Based	59.24	74.36	37.94	55.04	50.19	66.9	51.79	67.95
20 Cluster	60.19	74.32	37.73	54.72	49.67	66.09	51.88	67.51
40 Cluster	62.82	76.45	38.82	56.55	51.77	67.7	54.02	69.35
out-of-domain								
Model	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
	Race		**Relation Extraction**		**DUORC**		**Average**	
Baseline	21.09	34.34	38.28	63.89	31.75	38.82	31.68	47.12
Source-Based	18.75	32.03	40.62	67.55	27.78	41.07	29.06	46.92
20 Cluster	20.31	33.46	48.44	**71.76**	31.75	40.55	33.51	48.63
40 Cluster	**23.44**	**35.67**	**49.22**	71.10	**35.71**	**45.27**	**36.126**	**50.706**

Table 3: Experimental Results

Interestingly enough, the out-of-domain data we used for validation sets are from the same sources as some of the data used as out-of-domain validation sets in [1]. Lee et al. use the same validation metrics as we do, so we can directly compare their performance change to ours (see Table 4).

These three datasets prove to be among the least improved among the out-of-domain validation sets used in [1]. They aren't directly comparable to our results because Lee et al. used different training data and a different BERT architecture, but it is interesting to note that our 40-cluster model is quite a bit more effective at improving our baseline's performance on examples from these three datasets than Lee's model was at improving their baseline's performance.

Model	EM	F1	EM	F1	EM	F1
	Race Dataset		Relation Extraction Dataset		DUORC Dataset	
BERT-base	**28.23**	39.51	**73.33**	**83.89**	42.78	53.32
Domain-adv BERT	26.50	**39.73**	72.67	83.53	**45.97**	**57.89**
Relative Improvement	-1.73	0.22	-0.66	-0.36	3.19	4.57

Table 4: Lee et al. results

5 Analysis

We saw the greatest improvement on the RelationExtraction dataset [9], which is not surprising given that its passages are selected from the same source as SQuAD and Natural Questions, two of our in-domain datasets. However, on the Race dataset, our Source-Based and 20-cluster models actually performed worse than our Baseline model (see Table 3). In this case, the Race dataset is the least similar to our in-domain datasets (it is sourced from English exams rather than an online source like Wikipedia [8]). Examples from the Race dataset can be seen in Table 5.

Question	Which name may have something to do with “gladness"?
Context (shortened)	“Every year in English-speaking countries, people list the most popular names...In Britain *a parent today might call their little girl Grace, Jessica or Ruby*...In China names have very clear meanings. If a girl is called *Mei, her name means “beautiful". If a boy is called Wu*, his name means “like a soldier". Names in English-speaking countries are like this too. The girl’s name **Joy** is probably partly chosen because the parents wish their daughter to be joyful and bring joy to others...Another reason why kids get the names they do is that parents want to *name their boy or girl* after someone who is famous, such as an actor, a pop music star or a sports star."
Model	**Answer**
Baseline	Mei, her name means "beautiful". If a boy is called Wu
20-Cluster	a parent today might call their little girl Grace, Jessica or Ruby.
40-Cluster	name their boy or girl
Question	Why did the author decide to help the man?
Context (shortened)	“There is always a man who stands on different comers of the street in our city, holding a sign that reads ‘Will work for food for my family’... As I was sharing that feeling with my daughter and her friend, I decided that I needed to help this man. I wanted **to show the girls the importance of helping others**, not about worrying whether he was legitimately struggling or not...I told the man that the girl wanted to help him *because she was worried about him being cold.*"
Model	**Answer**
Baseline	because she was worried about him being cold.
20-Cluster	she was worried about him being cold.
40-Cluster	because she was worried about him being cold.

Table 5: Some validation examples from the Race [8] dataset (on which our 20-Cluster and 40-Cluster models performed worse than the baseline). Each model received an EM and F1 score of 0.0 for these answers. Note that the ground-truth answer is in bold.

Portions of each context were cut out for brevity, but these examples illustrate some of the issues our model encountered with the Race dataset. To answer these questions correctly, our model would have had to develop effective features for text that is written in a much different style than the majority of our training data. We believe that it would have been difficult for our model to learn features like

this given the small amount of data it saw from this domain and the dramatic difference between this distribution and that of our training data, even with the aid of the discriminator (though the 40-cluster model's modest improvement on this dataset was likely due to the discriminator's inclusion). Data augmentation techniques or additional data gathering to include a wider variety of out-of-domain data could potentially help improve our models performance with these types of questions and contexts.

6 Conclusion

We demonstrated that a domain adversarial training objective can be enhanced by choosing a finer-grained domain partitioning scheme than what was used in [1]. Specifically, we describe a method of partitioning domains using TF-IDF and K-means clustering and demonstrate that it yields a significant improvement over our baseline and a domain partitioning scheme based on the one used in [1]. We learned that the choice of domain partitioning scheme makes a significant difference in the effectiveness of this type of regularization.

There are several limitations of this project. Because fine-tuning took multiple days, we were limited in the number of experiments we could run within the deadline. We did not have sufficient time to perform ablation studies on the effects of different features or TF-IDF configurations in our domain partitioning scheme on the fine-tuned model. Additionally, our hyperparameter searches were done over a subset about 100 times smaller than our training dataset. We could have theoretically improved our hyperparameter search with efforts to make this subset a more balanced representation of examples in the partitioned domains. We also found it difficult to do a decent analysis, partially because our out-of-domain validation set wasn't particularly large (less than 400 context paragraphs in total), so there is less statistical certainty associated with the out-of-domain validation set performance improvements we found. We also would have liked to observe the average EM and F1 scores for each of our domain clusters to determine if certain clusters performed better than others. However, we were unable to categorize the validation set into our K-means clusters due to an error in saving the K-means parameters.

The method of domain adversarial training, although somewhat complex, seems quite underdeveloped. If we had more time, we could have explored the implications of using different inputs to the domain discriminator model (perhaps we could perform some kind of attention over all of the transformer's final hidden layer states and use the result as an input to the discriminator), as we still don't feel that $\mathbf{h}_{\text{CLS}}$ is an obviously superior choice. We also recognize that there is potential for multiple different discriminators (that would be trained for different domain partitions) to be applied in concert, adding one loss term for each to the QA model's loss. This would be more expensive at training time, but it presents an interesting solution to the problem of having to choose a domain partitioning scheme from multiple candidates—multiple can be chosen at once! If we had more time on this project, this would definitely be the next thing to try.

We based the feature vectors for our K-means clustering purely on functions of the context paragraphs, but the questions contain potentially useful information as well. Incorporating the questions for each example into these feature vectors is another potential improvement that could be explored.

References

[1] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 196–202, Hong Kong, China, November 2019. Association for Computational Linguistics.

[2] 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. *ArXiv*, abs/2004.10964, 2020.

[3] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Semantically equivalent adversarial rules for debugging nlp models. In *ACL*, 2018.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, 2016.

[5] Adam Trischler, T. Wang, Xingdi Yuan, J. Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. In *Rep4NLP@ACL*, 2017.

[6] T. Kwiatkowski, J. Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, C. Alberti, D. Epstein, Illia Polosukhin, J. Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Q. Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

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

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

[9] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *ArXiv*, abs/1706.04115, 2017.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Edward Loper and Steven Bird. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*, 2002.

[12] Tune: Scalable hyperparameter tuning, 2021.
Documentation available at https://docs.ray.io/en/master/tune/index.html.

A Appendix

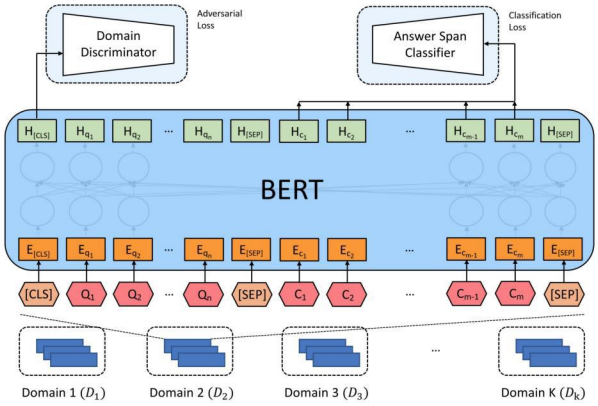
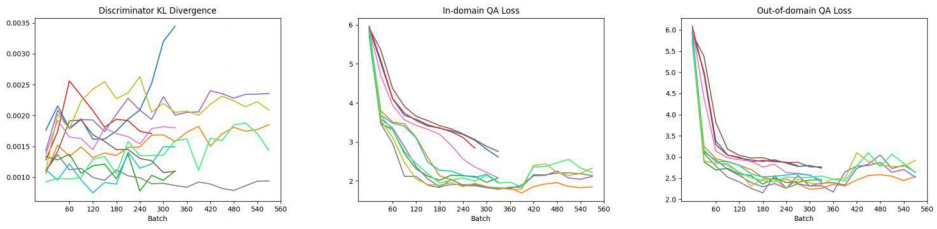
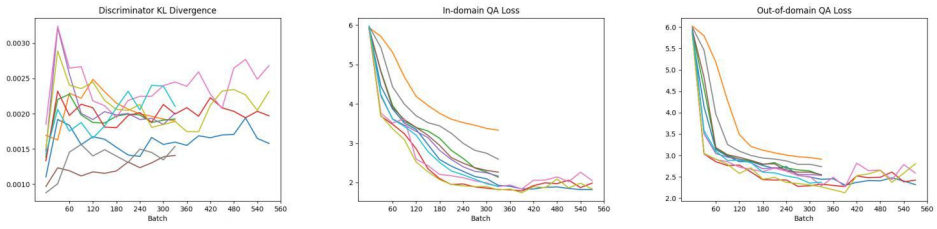


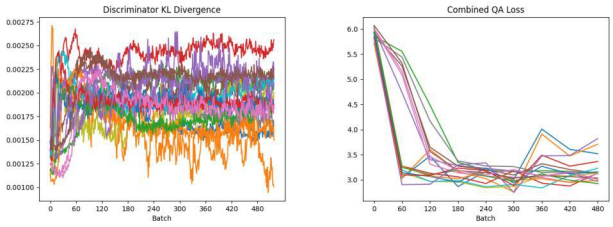
Figure 2: Overall training procedure for learning domain-invariant features from [1]. Our final model uses DistilBERT in place of BERT, and we evaluate several different domain partitioning methods.



(a) 20-Cluster Hyperparameter Search Results

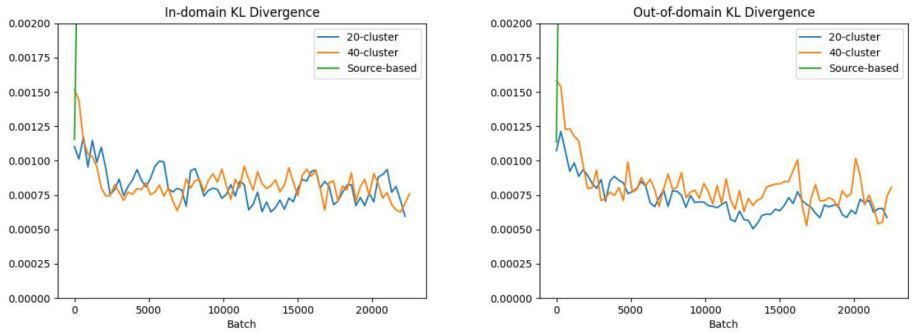


(b) 40-Cluster Hyperparameter Search Results

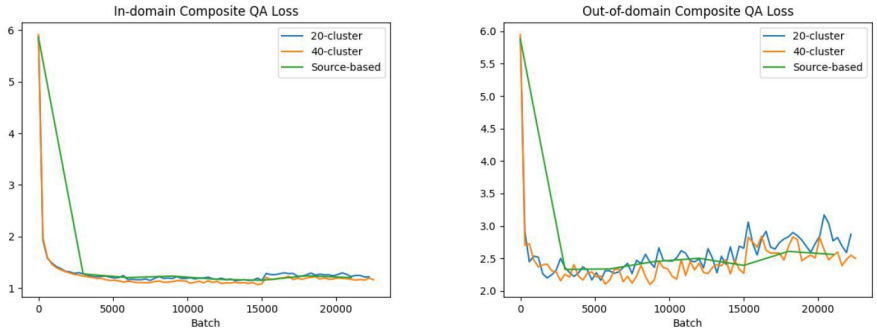


(c) Source-Based Hyperparameter Search Results

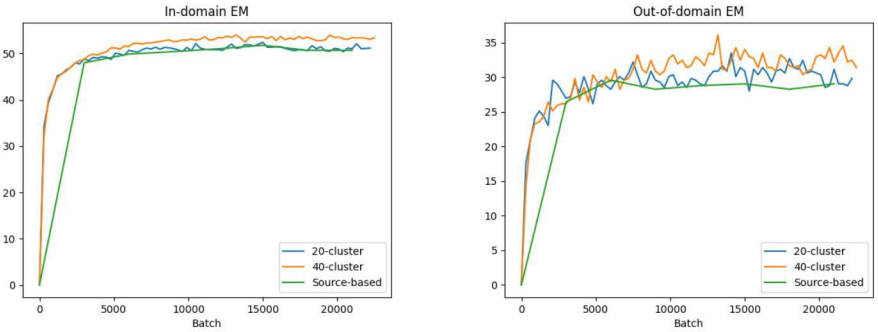
Figure 3: Hyperparameter Search Results. Each line on each plot indicates an additional run of our model with a different hyperparameter combination. (We were able to run more trials for the source-based model.)



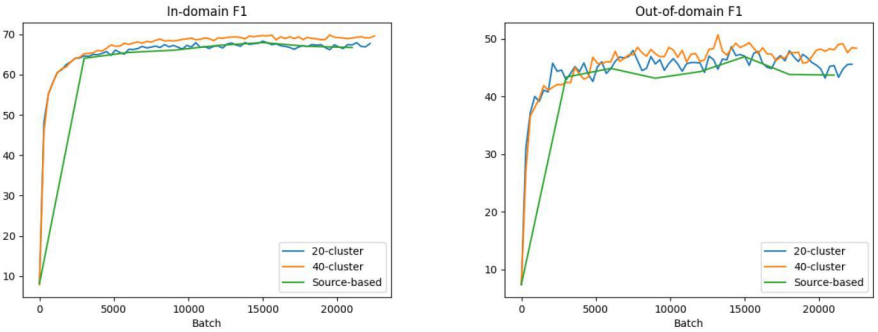
(a) KL Divergence Training Curves



(b) Composite QA Loss Training Curves



(c) Exact Match (EM) Training Curves



(d) F1 Training Curves

Figure 4: Evaluation Metric trends during training. As can be seen in these plots, our composite QA Loss improved significantly for all three of our models, but the 40-cluster model performed the best on both EM and F1.