With Such Friends, Who Needs Enemies? Using A "Friendly" Question Type Discriminator For Robust QA Training

Stanford CS224N Custom Project
Track: RobustQA

Filippos Nakas
Symbolic Systems MS Program
Stanford University
fnakas@stanford.edu

Abstract

Despite the proven benefits of domain adversarial learning in promoting the learning of domain-invariant representations by QA transformer models, we hypothesize that this addition might inadvertently also lead the model to become insensitive to the question types of the input data. Our original proposal is to counteract this problem with the incorporation of an additional "friendly" question type discriminator (QT discriminator for short) that is trained to predict the training data's question types from the QA model's hidden states. In contrast to domain adversarial training, our QA model was initially meant to try and "help" instead of fool the QT discriminator so that the latter would be forced to structure its representations more heavily around the question type of its input. Our experimentation, however, showed that the model shows the most significant OOD improvement over the baseline and vanilla adversarial training when its objective is to merely force the QT discriminator to make determinate predictions (vs random) without adding the restriction of them being correct. Given this freedom during training, we discover to our great surprise that the QA model often chooses to fool instead of help the QT discriminator by using representations that the latter indeed confidently interprets as being associated with a determinate question type - though not necessarily the correct one. In the analysis, we provide a plausible explanation for why this improves robustness and use it as justification to propose this method as a potentially generalizable tactic of easily incorporating domain knowledge about what properties of the data are robust into the in-domain training process. Finally, we acknowledge the limitations of this work in fully evaluating the effectiveness and theoretical justification of this new approach and point towards future work that could surpass these limitations.

1 Key Information to include

- TA mentor: Angelica Sun
- External collaborators, external mentors, sharing project: No

2 Introduction

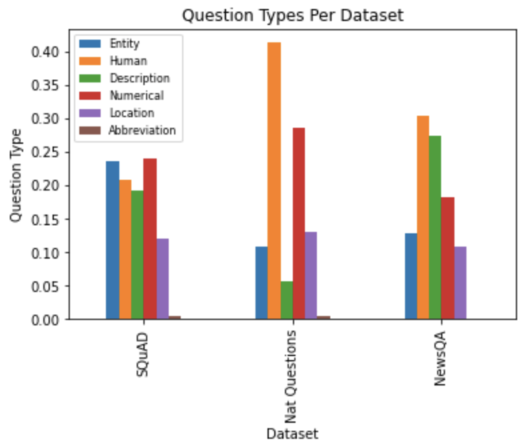
A well-known problem even with state of the art QA models that can outperform humans in datasets like SQUAD is that they fail to generalize well to performing QA on different domains that those they were trained on. For large pre-trained models like BERT, this limitation is normally addressed with some success with extensive fine-tuning on the test inputs of the out-of-domain test sets (in addition to the in-domain fine-tuning) or even additional unsupervised pre-training on relevant domain data

[1]. However, this approach faces both practical and "philosophical" limitations. On the practical side, additional pre-training can be computationally intensive on the good scenario in which there is adequate data. Even worse, if the task in question is few-shot QA this approach is unavailable. On the philosophical side, the need for additional pre-training casts serious doubt on the idea that the parameters learned by the original model are accurate and generalizable representations of the deep holistic structure of language.

Successful alternatives to these methods which depend on enriching the scale of either the data or the model include using data augmentation and more powerful baseline models. The organizers of the MRQA robustQA contest in 2019 [2], observe that, despite reaching state-of-the-art results, most such approaches improve robustness to a degree roughly proportional to the increase in in-domain performance. This suggests that these OOD performance improvements might be a natural consequence of general performance improvement and are thus again not necessary indicative of a independent increase in pure robustness, all things being equal. This point is possibly related to the fact that a unifying feature of all aforementioned approaches is that the learning process in itself remains fundamentally unchanged. Hence, a natural idea springing from these considerations is to somehow explicitly incorporate the end of attaining robust representation of the input into objective of the training process itself. Domain adversarial training is a characteristic realization of this idea.

The high level strategy is to add a feed-forward adversarial discriminator network that is simultaneously trained to read a part BERT's final hidden layer in order to guess the domain of the encoded input data. We then add a term in the QA model's loss function that motivates it to "fool" the adversarial network and make it unable to guess better than random. In being trained to additionally minimize this quantity, the QA model is thus forced to learn hidden encoding representations that are not recognizably linked to the properties of a particular domain, for that would indirectly also provide information to the adversary that helps it guess correctly.

Figure 1: Distribution of Question Types for each dataset in the in-domain data)



But this might make us wonder: is **all** information that can help distinguish between domains harmful to robustness? Clearly not. Although one can surely find other powerful examples, we concerned ourselves with the data's question types. We began with the observation that question types are not similarly distributed for each dataset (see Figure 1). We then hypothesized that learning to leverage the data's question types is beneficial to the robustness of the model, because the relation of question type and answer type is deeply connected to the task in a domain-invariant manner (this hypothesis was confirmed by the experimental data; see Experiments section). The observation suggests that if the model chooses representations based on the input's question type, the adversarial model could use their correlation to better predict the input's domain. But since this opposes the adversarial objective, domain adversarial training may be accidentally teaching the QA model to become question-type "blind" (this was also confirmed by the experiments) and thus indirectly damage its robustness.

This led us to the idea of incorporating an additional "friendly" network whose purpose would be to counteract only the deleterious effects of domain adversarial training. It's architecture and input are identical to that of the adversarial model with the only difference being that it would instead predict the data's question types and its objective should be somehow made to be aligned with that of the QA model. We thus preliminarily used the negative of the analogous adversarial component loss (KL-divergence with uniform distribution of empirical frequencies of the classes) which does not strictly guarantee that the objectives of the QA and question discriminator are aligned, naively assuming that this would naturally happen. But to our great surprise, given the freedom of "choosing" whether to fool or help the QT, the gradient back-propagation leads the QA model to go unexpectedly often for the latter. Even more unexpectedly, forcing the model to strictly help the question discriminator by substituting the latter's negative log likelihood loss for the negative KL divergence term, eliminates all the observed jumps in performance. This bizarre result will be given a plausible interpretation in the analysis section.

3 Related Work

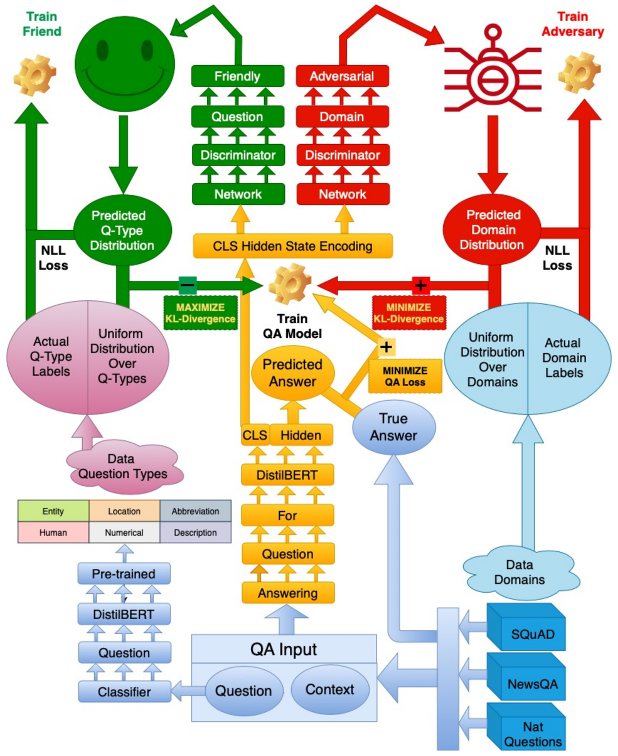
This work builds on a long series of progressive developments in the use of adversarial training from its birth as a tool for training generative models to its most recent application in various NLP tasks. In the context of this paper, this progression culminates in the 2019 paper "Domain-agnostic question-answering with adversarial training"[3] on which we rely heavily for both theoretical and technical background. The original adversarial network architecture was devised in 2014 by Ian Goodfellow and others [4] and was originally used as a method for creating generative models. Its two main components are a generator network G that is somehow meant to learn the input distribution and an adversarial network D that receives G's outputs as input and tries to determine whether the generated data come from the original distribution or not. G's objective is now re-defined as minimizing the success of D (i.e. maximising D's loss function). By simultaneously training both networks on their respective goals through back-propagation, G effectively learns how to generate data that is indistinguishable from the input distribution. Following the success of this method, others soon realized that it can also be used to improve the generalisability of neural network representations used for non-generative tasks which involve data coming from different domains.

The main idea, introduced in 2015 [5] and tested primarily for image classification tasks, is to organically incorporate the adversarial learning task to the original supervised learning task. Instead of just focusing on optimizing the parameters to predict the correct labels, we penalize parameter configurations that produce representations which allow an adversarial network to determine the domain from which a particular input is coming from. As before, the adversarial network tries to become a good discriminator and the classifier tunes its parameters so that - in addition to predicting the correct labels - it fools the discriminator. This method is shown to increase out-of-domain performance, the reason being that classifier is forced by the discriminator to learn representations that fool the discriminator and are thus not speciously linked to properties of a particular domain. By taking another step towards generalization researchers in 2017[6] observed increases in performance by applying this method to the NLP task of dependence parsing by connecting the output of their bidirectional-LTSM parser to a discriminator module. Finally, the 2019 paper that inspired this work applies this architecture on top of BERT to enhance its training's robustness to performance with out-of-domain data, confirming once again that it can improve OOD performance.

A significant source of inspiration for coming up with the friendly component is the rationale behind the seminal 1991 paper by Jacobs, Hinton et al.[7] called "Adaptive mixtures of local experts". In this paper, the authors propose splitting up the learning task amongst a collection of local "expert" models each of which ends up learning a different prediction strategy that suits a particular type of input. The collective outputs of each expert are modulated by something like a stochastic gating mechanism that determines which expert to call upon depending on its suitability. Prima facie, this approach seems to represent the exact opposite training philosophy than that of adversarial learning: it actively promotes building specialized representation for different domains of data. However, this seeming opposition dissipates once one considers that different partitions of data may be better suited to one of the two treatments depending on their relation to the objective of training. For the purposes of performing robust QA on a dataset comprised of smaller datasets from different sources, using a mixture of expert sub-models could be very hazardous in the likely scenario that the experts end up specializing on the individual component datasets. If there was, however, some way of "forcing" these experts to specialize in subsets of data which are defined by features that posses a deep domain-invariant connection to the nature of task - case in point, question types for QA -, specialization could end up being helpful. In light of this, the incorporation of the QT discriminator into the training process can be viewed as an alternative way of producing am implicit mixture of experts realized in this case as distinct "families" of representations exhibit by a single model's encoding hidden node value instantiations for different types of input. Although we chose to apply this method for the particular purpose of correcting the potential deficiencies of domain adversarial learning, an interesting future work would be to explore its use as a self-sufficient method that purely competes with mixture-of-expert approaches. If it is proven to adequately induce positive specialization, it has the added benefits of being relatively simple to implement on top of existing architectures and fitting naturally with large neural models such as transformers.

4 Approach

Figure 2: Visualization we created to help understand the intricacies of the training process. The forward pass follows the direction of the arrows for each of the three networks. Note that in the forward pass, the paths through the discriminators are also part of the QA model's path, as they contribute to the calculation of its loss. Each of the backward pass follows the reverse direction, with the added caveat that the discriminator parameters are only updated by the back-propagation of their own loss function and not additionally from the QA's loss even though they contribute to its calculation.



As mentioned in the introduction, we rely heavily on the domain adversarial training implementation described in the 2019 paper "Domain-agnostic question-answering with adversarial training"[3]. The coding implementation is built on the starter code provided for robustQA track and, unless otherwise stated, is our own with a few exceptions that we will describe now. Because the original paper gave no description of the discriminators architecture, in just this particular case, we copied it almost directly from the codebase https://github.com/seanie12/mrqa/blob/master/model.py. We also looked at the code-base to make sure we had interpreted correctly the uniform distribution calculation in the KL divergence loss as well as to determine whether the discriminator parameters are updated by the QA model's backward pass. Finally, to classify the questions we made small adaptations on the following script written by Danilo Croce (https://github.com/crux82/AILC-lectures2021-lab) that trains a DistillBERT classifier to distinguish 6 question types. We will now introduce the formalism.

The data used for the QA model are tuples (c_i^l, q_i^l, y_i^l) denoting the i^{th} context, question, span data-point triplet in the l^{th} training domain out of total of K training domains ($K = 6$ for the original paper and $K = 3$ for ours. y_i^l is itself a tuple of $y_{i,s}^l$ and $y_{i,e}^l$ denoting respectively the start and end position of the gold standard answer span. The model's purpose is to predict these two position for each data-point, given the context and the question. The loss function uses standard negative log loss across all domains:

$$\mathbf{L_{QA}} = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k}[logP_\theta(y_{i,s}^k|c_i^k, q_i^k) + logP_\theta(y_{i,e}^k|c_i^k, q_i^k)]$$

In the original domain-adversarial training paper, in addition to minimizing this loss function, the training must also fool a feed-forward Discriminator D (), which is fed some hidden state h from within the BERT architecture containing some high-level representation of the context and query (in this case, the hidden state of the [CPS] position). In our own approach, we additionally use an identically structured feed-forward network that receives the same hidden state and attempts to predict the question type of each data point. During training the domain "adversarial" and the QT friendly discriminators train their parameter sets ϕ, and ω to respectively predict the domain-label l and the question-label q of each data-point using h. In the original paper, the QA model must learn to minimize its adversary's success by minimizing the KL-divergence of its softmax predictions with the uniform distribution over the domain relative data sizes (in essence, trying to make its best guess to approach being random):

$$\mathbf{L_{adv}} = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} KL(U(l)||logP_\phi(l_i^k|h_i^k))$$

,where KL is the KL-divergence and U(l) is the uniform distribution over the K classes based on their empirical frequencies in the data.

In our new version, the QA model must additionally learn to **maximize** the friendly QT network's prediction certainty - though not necessarily, accuracy - by **maximizing** the KL-divergence of between its prediction softmax over the data's question types and the uniform distribution over the question types, where M is the number of question types and q^m is the m^{th} question label:

$$\mathbf{L_{friend}} = -\frac{1}{N}\sum_{m=1}^{M}\sum_{i=1}^{N} KL(U(q)||logP_\omega(q_i^m|h_i^m))$$

The reason for which we use this objective instead of the discriminator's NLL loss will be dealt with extensively in the analysis section. These three loss functions will now be combined into one using the hyper-parameters λ_{adv} and λ_{friend}:

$$\mathbf{L} = \mathbf{L_{QA}} + \lambda_{adv}\mathbf{L_{adv}} - \lambda_{friend}\mathbf{L_{friend}}$$

5 Experiments

5.1 Data

We are training the data on the three in-domain and three OOD datasets provided in the RobustQA project and depending on whether we are doing in-domain training or secondary OOD fine-tuning, we use their respective validation data-sets. The question classifier we borrow from the external script is trained on the coarse grained-version (six classes) of the Question Classification dataset in English (https://cogcomp.seas.upenn.edu/Data/QA/QC/)

5.2 Evaluation method

We use the standard EM and F-1 scores and compare our results against our own trained vanilla and adversarial baselines, as well as the scores of the project's development and test leaderboards. Moreover, we evaluate the model both before and after a limited amount of fine-tuning to trace its effect on performance.

5.3 Experimental details

In-Domain Training Due to time and resource contraints we could not afford to serisously experiment with all hyperparameters and we thus chose to focus primarily to on those that bear a direct relevance to the model architecture and were observed to lead in the greatest differences in performance in preliminary tests. We decided not to experiment with the number of epochs, learning rate, batch size, dropout rate and optimizer; they were set respectively to the standard values of 3e-05, 16, 0.1 and AdamW (we also used same AdamW for the two discriminator networks). After including the two networks, training on both Azure and Google Colab roughly doubled from 4 hours to 8-9 hours.

We experimented extensively with the two hyperparameters controlling the loss function an well as the type of loss function used for the friendly component. A few experiments with each loss function were sufficient to rule in favour of the negative KL divergence as opposed to the NLL. For pure adversarial training we found the optimal value with lee(2019)[3] of 0.5 having tested two higher and two lower values. For adversarial friendly training we first experiment with parameter pairs that were either different or equal and quickly realized that the latter is better. Having then randomly chosen 0.3 on one of our first trials, we attained the best possible results and were not able to beat it by any of the future trials using 0.2, 0.25, 0.35, 0.4 and 0.5.
Best in class for vanilla adversarial: $\lambda_{adv} = 0.5$, $\lambda_{friend} = 0$
Best in class for adversarial-friendly: $\lambda_{adv} = 0.3$, $\lambda_{friend} = 0.3$
Finally, we experiment with merging the "abbreviation" with the "description" question type and rejected as it clearly worsened performance.

Out-Of-Domain Training

Since this part is orders of magnitude less expensive in time and computation we experimented with almost all hyper-parameters and found little reason in changing them from the default. The general rule was that optimal performance on the development set would be reached in less than ten epochs (usually in the first five) and would gradually plateau to a slightly smaller value, showing gradual increases and decreases thereafter. We thus settled at using the same batch size and learning rate for the in-domain training and evaluated at every 100 batch iterations for 10 epochs. Note that this fine-tuning step is performed using the vanilla training process of a QA model and does involve the larger network.

5.4 Results

Figure 3: In-Domain Validation F-1 and EM are indistinguishable for all three types of training.

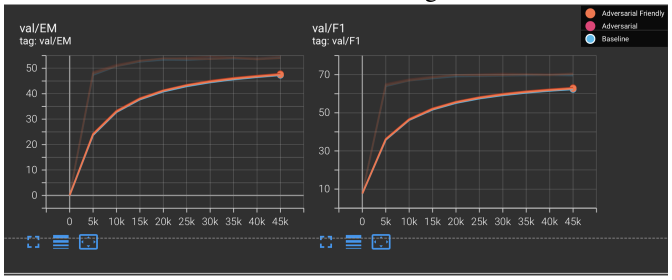


Figure 4: From left to right: Domain Discriminator Loss, Total QA Loss, Q-Type Discriminator Loss

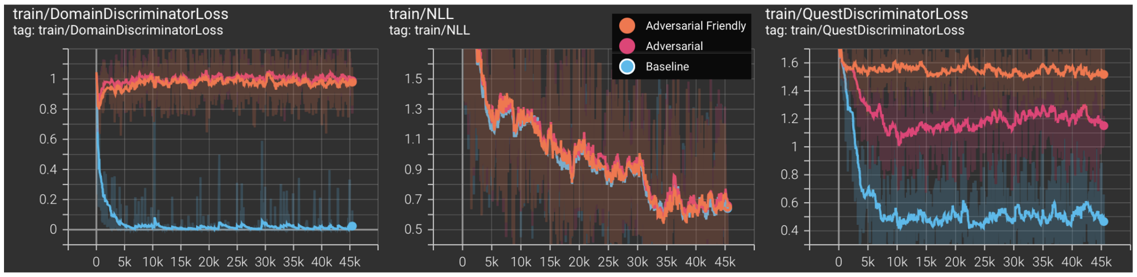


Figure 5: Performance on Dev Set

One Shot OOD					Fine-Tune OOD				
Run Type	**F1**	**vs Baseline**	**EM**	**vs Baseline**	**Run Type**	**F1**	**vs Baseline**	**EM**	**vs Baseline**
Baseline	46.71	0.00	30.37	0.00	**Baseline**	49.71	0.00	34.82	0.00
Adversarial	44.59	-2.12	28.27	-2.10	**Adversarial**	50.34	+0.63	36.13	+1.31
Adversarial Friendly	49.65	+2.94	33.25	+2.88	**Adversarial Friendly**	51.99	+2.28	36.13	+1.31

Performance on Test Set: EM: 41.032 F1: 58.364

** Please see appendix for discrepancy between fine-tune baseline score here vs the poster.

6 Analysis

Adversarial training performs significantly worse than the baseline without extra fine-tuning on the OOD data but improves significantly after OOD finetuning. This might be because the adversarial objective obstructs the QA model from gaining specialized enough representations that are adequate to the task but for the same reason allows greater flexibility for learning the new domain. The good news is that, as we expected, the friendly component we added greatly succeeds in enhancing the performance of pure adversarial learning for both one-shot and fine-tuning adjusted evaluation. This boost acquires even greater significance in light of the fact that it is not accompanied by an increase in in-domain performance, which is a safe indicator that the improved OOD performance is a pure effect of increased robustness and not a side-effect of a more general improvement in the model.

However, as has been often hinted at in previous sections, the reason for this improvement proves to be very surprising. We will gradually see why by analyzing the Q-Type Discriminator Loss graph on the right of Figure 4. The blue and pink lines represent the friendly network's loss throughout baseline and adversarial training respectively, in both of which cases it plays no part in the training process. Both of these lines behave in ways that confirm the two hypotheses we stated in the introduction. First, the blue line's descent shows that the baseline model naturally gravitates towards encoding representations from which the QT discriminator can effectively decode the input's question type. Secondly, adding the adversarial objective forcefully disturbs this original tendency: the pink line does not seem able to descend beyond a certain point. This happens because if the shared input of the discriminators (final CLS hidden state of DistilBERT) allowed the friendly one to predict the question type, the adversarial one could also use the same information to leverage the correlation between domain and question type and thereby greatly improve its prediction success. So far, so good.

Things start to get interesting when we observe the behaviour of the orange line, which traces the loss of QT discriminator once the maximization of its KL-divergence from uniform is also incorporated in the objective of the QA model. Our expectation was that this would cause to the loss line to cross the lower bound imposed by the adversarial objective, and somewhat increase its proximity to the blue line. But we observe the exact opposite: the loss now seems to face an even higher lower bound. Even more surprisingly, the line does not show any tendency to move downwards even in the beginning. To understand what's happening, recall that the QA's added objective is to maximize the KL divergence between the QT discriminator's soft-max from the uniform distribution, i.e., using an encoding representation that will lead the QT in making *some* determinate choice. Having the freedom to maximize this objective by either leading the QT discriminator to predict correctly or incorrectly, the QA model seems to consistently go by the latter for a certain fraction of times. **This means that the training leads the QA model to often represent the input of a question type A in a way that the QT discriminator confidently interprets as being associated with a particular different question type B - or at least with one amongst a small subset of different question type(s).**

As we mentioned in the previous section, forcing the QA model to strictly help the QT discriminator by minimizing its NLL does not lead to increased robustness. The benefits in robustness must then be understood as the result of affording - for certain training inputs - the QA model the freedom to represent the input's encoding in its hidden states in way that leads the two discriminators to believe it has a different question type than its actual one. Observe that this is the only way in which the QA model can satisfy both discriminator components of its loss function at the same time, because whenever it instead opts for "helping" the friendly QT classifier, it inadvertently also helps the adversary due to the Q-Type - domain correlation. In such cases, the benefit of being able to satisfy both discriminator objectives in one go must be overcompensating for the cost of not using the representation that most fits the actual question type of the input for the sake of pure QA performance. This may be helping robustness in a somewhat analogous way to how adding dropout helps ordinary training. **The network becomes more flexible by allowing itself to cross-transfer prediction strategies for different question types whenever that is not too costly.** Given the generally uneven distribution of question types in the training data (see Apendix), the QA model benefits from leveraging question types with the added flexibility of also using what it has learned from different question types whenever that is a good substitute - either because one question type has fewer examples and is similar enough to one with more (say, using "human" question type representations for "entity" type questions) or because certain examples get wrongly classified by the imperfect question classifier we are using.

The relatively poorer performance of our best model in the test leader-board (lower half) compared to the dev leader-board(top 20) can be attributed to a few different reasons. To begin with, more generally, domain-adversarial training for QA does not have a history of achieving state of the art results for robust QA [2] and, as explained in the introduction, is bested by techniques that simultaneously also somewhat increase in-domain performance. The good news for our implementation is that we achieved pure increases in OOD performance while actually slightly decreasing in-domain performance, which brings us to the second point. Observing the overall higher scores in the test as compared to the dev set, we suppose that the test data had a greater resemblance to the in-domain data than the OOD used for development. Since the adversarial network is used to abstract from any particularities in the in-domain data - which naturally worsens in-domain performance -, it is natural to expect that it might be responsible for the underwhelming performance compared to methods that do not try to completely abstract from the peculiarities of the training data. Moreover, we must take into account that domain-adversarial training is probably not at its best when the in-domain has only three data sub-domains (the original paper used six). Finally, we have limited sense of the accuracy of the question classifier we are using. Although I manually checked that its performance is decent for about 100 examples, the fact that it is being trained on less than 10.000 questions that are likely to differ from our training data makes it certain that a non-negligible portion of of examples will be wrongly classified. Using an improved question classifier will almost certainly improve performance.

7 Conclusion

We have confirmed the hypothesis that our new method significantly improves domain-adversarial training for RobustQA. As hypothesized, forcing the QA model to build representations based on the input's question type increases robustness but in a different manner that what we initially suspected. The improvement crucially depends on allowing the QA model the freedom to occasionally "betray" the friendly network for certain kinds of input, by using representations corresponding to the "wrong" question type. Although we have provided a reasonable hypothesis for this outcome, we unfortunately did not have the time to inspect in detail the type of substitutions that are actually performed and the conditions under which they happen; this analysis should be undertaken in future work and will be crucial in adequately comprehending the characteristics and validity of the proposed method. In addition, our work suffers from serious limitations related to the combination of having limited computation and time resources while evaluating a training process that involves a large number of crucial modelling choices that probably significantly affect the extremely complicated interactions between the three networks. Given the notorious instability of adversarial training (made exponentially worse by the inclusion of the friendly component), checking the results across a large number of seeds is necessary to trust the results. It will also be important to check performance for different numbers of domain and question classes and a more accurate question classifier.

If the methods proves its value under these circumstances, promising avenues of future work include using a larger number of hidden states as discriminator input - possibly different for each and with different degrees of overlap -, changing the discriminator architectures, and using the friendly discriminator alone as a self-sufficient method. Finally, this method can be easily be adapted for increasing robustness in other NLP and machine learning tasks by simply choosing a dataset partition based on a feature(s) - maybe using more one friendly network- that is expected to posses intrinsically deep and domain-invariant connections with the success of the task at hand. For example, the role played by question type in QA could be played by proposition type in natural language inference.

References

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

[2] Robin Jia Minjoon Seo Eunsol Choi Danqi Chen Adam Fisch, Alon Talmor. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *EMNLP*, 2019.

[3] Donggyu Kim Seanie Lee and Jangwon Park. Domain-agnostic question-answering with adversarial training. In *MRQA@EMNLP*, 2019.

[4] Mehdi Mirza Bing Xu David Warde-Farley Sherjil Ozair Aaron Courville Yoshua Bengio Ian Goodfellow, Jean Pouget-Abadie. Generative adversarial nets. In *NIPS*, 2014.

[5] Victor Lempitsky Yaroslav Ganin. Unsupervised domain adaptation by backpropagation. In *PMLR*, 2015.

[6] Hiroshi Noji Yuji Matsumoto Motoki Sato, Hitoshi Manabe. Adversarial training for cross-domain universal dependency parsing. In *CoNLL*, 2017.

[7] S. Nowlan R. Jacobs, Michael I. Jordan and Geoffrey E. Hinton. Adaptive mixtures of local experts. In *Neural Computation*, 1991.

A Appendix

**Please note that the fine-tune baseline score presented in the experiment section is actually different and clearly higher than that presented in the poster. I realized that I accidentally used the results of an OOD fine-tune training trial that used different configuration parameters than those used for the other two results. The qualitative analysis of the results thankfully remains the same.

Figure 6: Distribution of Question Types for all In-Domain data

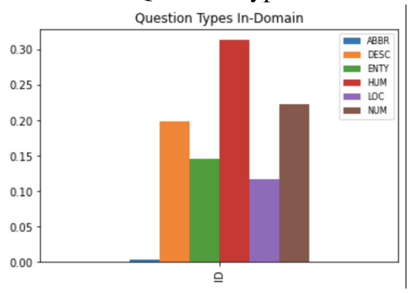
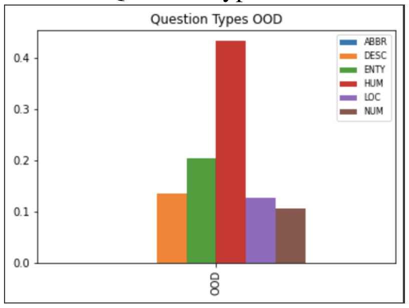


Figure 7: Distribution of Question Types for all Out-Of-Domain data


Question Types OOD
OOD
ABBR
DESC
ENTY
HUM
LOC
NUM