# Robust Question Answering: Adversarial Learning

Stanford CS224N Default Project: Robust QA

**Prathyusha Burugupalli**
ICME
Stanford University
pratb@stanford.edu

**Mohak Goyal**
Management Science & Engineering
Stanford University
mohakg@stanford.edu

**Veer Shah**
ICME
Stanford University
veer@stanford.edu

We do not share any part of this project with any other class.

## Abstract

In the NLP task of question-answering, state-of-the-art models perform extraordinarily well, at human performance levels. However, these models tend to learn domain specific features from the training data, and consequently perform poorly on other domain test data [1]. In order to mend this issue, we adopt the adversarial training approach to learn domain invariant features in existing QA models, based on Lee et. al. [2]. In this approach, the QA model tries to learn hidden features that the discriminator, which tries to classify the domain of the question-answer embedding from the hidden features, unsure of its prediction, thereby learning domain-invariant features. We study modifications this model, in particular the impact of weights on the adversarial loss on the model's performance. We also study other techniques such as data augmentation and answer re-ranking in order to make our model more robust. Our work is limited in that we only train models on a subset of the training data available to us due to the cost of training time. However, we can conclude that changing the weight of the adversarial model results in marginal changes in performance. Furthermore, although the adversarial model exhibits improvements over our baseline, data augmentation proves to be a more effective technique in making the model robust on our of domain data given the subsampled training data. Re-ranking answer spans does not seem to produce favorable results.

## 1 Introduction

Even though many deep learning models surpass human-level performance on various task, they perform poorly on out-of-domain datasets. Thus, much research has been devoted to proposing methods that help generalize models to unseen sources of data, and often the current literature fall into two buckets – domain adaptation, where models are fine-tuned for a specific target domain, and domain generalization, where models are not given data from the target domain.

For the task of Question Answering, even the most successful models on SQuAD [1] cannot perform as well on other datasets, making the issue of robustness especially relevant in the QA literature. In our work, we study a domain generalization technique to examine how models that are not fine-tuned to particular domains perform on those domains. Much research has been devoted to the goal of domain generalization; for example Balaji et. al. propose using meta-regularization in a meta-learning framework [3]. Other research proposes breaking down the parameters of a model into domain-specific and domain-agnostic while training, and then using the domain-agnostic parameters for predictions on unseen domains [4].

We pursue an adversarial training framework to achieve robustness, inspired from the Generative Adversarial Network [5], originally implemented for Computer Vision. Since adversarial networks have been used in domain generalization before as in Ganin et. al. 2016 [6], Lee et. al. [2] hypothesized that adversarial techniques can be extended to the QA domain as well. In particular, they propose complementing a standard QA model with a domain classifier and training the QA model to confuse the domain classifier, thereby learning domain invariant features of the training data. We explore this framework and a baseline QA model, making modifications to both pipelines and training both models on the same input data in order to effectively compare the performances of the models.

We show that the adversarially trained model can indeed perform slightly better than a vanilla baseline QA model. However, due to the computational complexity of training both a QA model and a discriminator simulatenously, the adversarially trained model comes with drawbacks; we only train both models on a subsample of the data due to time restraints. We also show that the baseline model is directly helped through data augmentation techniques, and can outperform any of the adversarial models we have trained. We also examine the performances of these models on subsets of the data, which invite future testing of hypotheses. We acknowledge that future work must be done, given the limitations of our research.

## 2 Related Work

A variety of methods have been tested to evaluate robustness of QA models on out-of-sample data. Recently, GPT-3 has demonstrated great capacity in few-shot learning in a broad range of NLP tasks, but has done so partly due to its 175 billion parameters [7]. Thus, in many real-world settings, it may be unrealistic or at the very least challenging to deploy or test GPT-3.

Fine-tuning pre-trained models has often showed to generalize to out-of-domain tasks. Recent work by Gao et. al. has shown that smaller models like BERT, when fine-tuned using prompts (asking the language model to auto-complete a masked prompt) and demonstrations (training examples based on a template prompt) can outperform "vanilla" fine-tuning methods [8].

Other work has shown that augmenting training data can help improve performance on out-of-sample test data. For example, Gan et. al. focused on generating a richer training data set by first creating a model to produce paraphrased questions, and then feedings these questions as input for current QA models, a data augmentation technique that improves robustness on adversarial test sets [9].

Language has unique structure and syntax, which is presumably invariant across domains; some research has been dedicated to use this language-specific property in order to improve model robustness. In order to incorporate syntax information between a candidate answer and the question, Wu et. al. explore modelling syntactic similarity between the proposed answer and the given question, with moderate improvements over their baselines [10]. Another method to improve robustness is incorporating information from knowledge bases that encode relationships between different parts of a sentence in triplets – subject, relation, object – into a QA model architecture [11], which improves over several baselines on AddSent and AddOneSent data, a database of adversarial examples [12].

Most promising work attempts to fine tune an existing language model and/or augment training data, so we pursue similar methods in our work.

## 3 Approach

We first implemented the baseline model in accordance to the instructions given to us by the CS224N course staff here. We use this baseline model as a comparison for our adversarially trained DistilBERT model. The baseline system finetunes DistilBERT [13] on all of the training data. DistilBERT is a smaller and much faster version of BERT with about $40\%$ fewer parameters than BERT and is about $60\%$ faster to train. It retains $95\%$ of the performance of BERT on the SQuAD QA task.

In order to implement an adversarial training framework, we adapt code from Lee et. al. [2] available online at `https://github.com/seanie12/mrqa`. We pursue a training framework as depicted in Figure 1. Our code is available here: `https://github.com/veer-shah/DeepRobustQA`.
The QA model uses a pre-trained DistilBERT to embed the input tokens into the final hidden layer. This layer is then fed into both the domain discriminator and the answer span classifier. The answer span classifier is a simple linear layer that transforms the hidden layer logits for the predicted answer start and end into a final prediction. We also see the effect of re-ranking the candidate answers produced by DistilBERT, which we further explain in the **Experiments** section.

The discriminator is trained to minimize the cross-entropy loss of domain predictions. Let the true domain category be $l$ and the hidden representation of the question-passage pair be $h \in \mathbb{R}^d$. Let $K$ be the number of domain classes in the training data and $N$ be the total number of training examples. The loss function for the discriminator $D$ is:

$$\mathcal{L}_D = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} \log P_\phi(l_i^{(k)} | h_i^{(k)}).$$

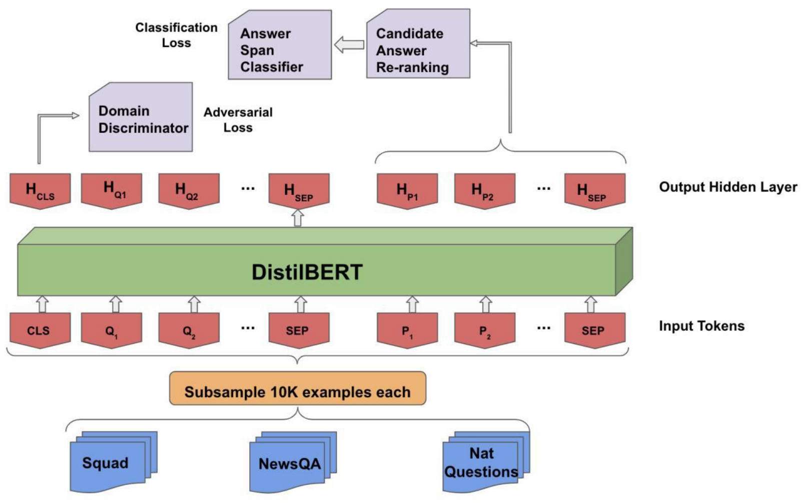

Figure 1: Model Training Procedure as seen in [2]

The QA model is trained to minimize the Kullback-Leibler (KL) divergence between the uniform distribution over $K$ classes and the discriminator's predictions. In other words, the QA model is trained to confuse the discriminator. The final loss of the QA model is $\mathcal{L}_{QA} + \lambda\mathcal{L}_{adv}$, where $\lambda$ is a hyper-parameter controlling the importance of the adversarial loss. The overall model alternates between optimizing the QA model and the discriminator. The adversarial loss is given by:

$$\mathcal{L}_{adv} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k} KL(\mathcal{U}(l)||P_\phi(l_i^{(k)}|h_i^{(k)})).$$

## 4 Experiments

Here we describe the experiments we have conducted on both the baseline model as well as our adversarially trained model.

### 4.1 Data

We use the datasets provided by the staff. The training datasets are obtained from the SQuAD [1], NewsQA [14], and Natural Questions [15] databases – these data make up the "in-domain" data. The validation and test datasets are from the DuoRC [16], RACE [17], and RelationExtraction [18] databases – these data constitute the "out-of-domain" data. Our goal is to train a model on data obtained from a certain subset of domains that performs well on previously unseen data from entirely different domains.

The training data (SQUAD, NewsQA, and Natural Questions) have 50,000 question-answer examples each, which allows complex models like DistilBERT to take advantage of a large training pool. However, because our adversarially trained model effectively trains 2 models concurrently (both the answer span prediction model as well as the domain discriminator), its training time is prohibitively expensive. When attempting to train our adversarial model with all 150,000 training examples, the estimated training time for one epoch was approximately 6 days. We hypothesize that the forward computation and backpropagation process for two models contribute to this training time.

Given time constraints of the class and credit constraints on our Azure virtual machines, we decided to train all of our models on a subsample of the training data. Specifically, we subsampled 10,000 examples from each of SQUAD, NewsQA, and Natural Questions, for a total of 30,000 question answer pairs. The same subsampled data is used in the training pipeline of all our model experiments. [1]

[1]We were unaware that we were able to request VMs with more computational power to speed up the training process until the final week of the quarter, so we made adjustments to our training process to work with the VM suggested to use in the Azure Guides.

### 4.2 Evaluation method

We use the F1 and EM metrics as described in section 1.2 of this handout. Briefly, F1 is a measure of the predicted answer span's accuracy derived from precision and recall; F1 the harmonic mean of precision and recall. EM, on the other hand, is a measure of whether or not the predicted answer span matches the true answer span exactly. Since each question-answer example comes with 3 human annotated answers, we consider the maximum F1 and EM calculated across the 3 true answers.

### 4.3 Experimental details

We conducted 5 experiments in total, and observed the effect of re-ranking the candidate predicted answer spans for each of the experimental models. We detail each approach, and the re-ranking algorithm below.

### 4.4 Data Augmentation

For data augmentation procedures we surveyed the literature from [19] and [20]. We studied the procedures of back-translation, word substitution, random word deletions, random word insertion, and random word scrambling. Out of these procedures we decided to implement back-translation and word substitution with synonyms. The implementation details of each of the two data augmentation methods follow.

#### 4.4.1 Back Translation

We considered back-translation from three different languages i.e., German, French, and Hindi. However, we finally used French because of the robustness of the translations available. We consider applying back-translation to both the questions and the passages. The back translation of the passages proved to be very complicated because the answer span cannot be extracted robustly after the passage has been altered. Therefore we applied back-translation to the questions only.

Since Google translate API calls are reliable and efficient, we chose to use it for out project. However, we encountered two issues with using Google translate: first, there was an issue with Google not allowing us to make several requests in a short interval of time, and second, there were instances when the API call would not go through, therefore stalling the script. To solve the first issue, we used pauses of 50 seconds after every 100 API calls. This method slowed down our data augmentation procedure, however it worked satisfactorily and gave us a throughput of 1 example every second. We could generate our back-translated datasets in about 9 hours. For the second issue of unresponsive API, we implemented a procedure of waiting for a maximum of 2 seconds on an call before making a retry.

#### 4.4.2 Word Substitution

For the word substitution procedure on the datasets, we chose an intuitive method of replacing individual words with their synonyms. We considered synonyms which were one word or two words long. For example, for the word *connect,* we considered both *join* and *bring together* as valid substitutions. We also considered hyphenated words in our procedure.

We tried four methods to generate the candidate synonyms. Those are: (1) *synsets* function of NLTK [2], (2) *lemmas* function of NLTK, (3) PyDictionary [3], and (4) using GloVe. Out of these, we found the *lemmas* function of NLTK to be the best method in terms of human evaluation of the quality of synonyms. The PyDictionary made API calls to synonyms.com and was therefore very slow (we needed to make 20 API calls per example, on average). The other two methods did not produce good quality synonyms.

For choosing between valid candidates for substitutions, we tried two methods. The first is a simple random procedure which picks one of the options uniformly at random. The second method was using the NLTK toolkit [21]. The NLTK toolkit provides a function to compare the similarity score of two words. However, we realised that using the random procedure was as good as the similarity score based method in terms of human evaluation of the quality of word substitution. The similarity score based method is much more restricted since it can take only individual words as inputs. for example, it cannot compare 'bring together' with 'connect'. Therefore we used the random method.

Taking inspiration from [22], we decided to use perform word-substitution only on the nouns and verbs in the sentence. The reason behind this decision is that changing adjectives and articles would change the meaning of the sentecne drastically. Similar to [22], we used *spacy* and *en-core-web-sm* libraries to

[2] `https://www.geeksforgeeks.org/get-synonymsantonyms-nltk-wordnet-python/`

[3] `https://pypi.org/project/PyDictionary/`

classify words as nouns or verbs in a sentence. As we performed word substitution on the passages, the location as well as the text of the answer were changed. To take care of this, we modified the answer start point in the modified training example and also updated the answer text whenever our method modified some parts of the answer in the passage.

A major concern with this procedure was that our method did not check if the substituted word made sense in the context of the sentence. For example 'connecting the dots' should not be replaced by 'bringing together the dots', but our procedure cannot make this distinction. We brainstormed on trying different procedures for dealing with this issue. We considered using off-the-shelf models for classifying sentences as grammatically and logically correct or not. We decided against using such models because of the restriction on using large pre-trained models outside of DistilBert for our project.

#### 4.4.3 Vanilla Baseline Models

We trained two baseline models. One of the baselines used only the 30,000 training examples we subsampled, and the other used an augmented dataset.

The baseline models both use a DistilBertForQuestionAnswering pre-trained model (6 layers, 768 hidden size, 12 attention heads, and 66M parameters) to predict answer spans. We initiated the learning rate to 3e-5, used a batch size of 16, and ran each of the models for 1 epoch. We used the AdamW optimizer, as was used given to us by the course staff. The model took approximately 4 hours to train.

#### 4.4.4 Adversarial Models

We adversarially trained three models. The models only differed in their $\lambda$ parameter, as we wanted to see the effect of a higher weighted adversarial loss on the performance of the model.

The models share most details with each other. All adversarial models use a DistilBert pre-trained model (6 layers, 768 hidden size, 12 attention heads, and 66M parameters) to embed the input tokens, and an additional linear layer for answer span classification; these two models compose the QA model. The QA model is trained using a dropout rate of 0.1. The discriminator has 4 layers (with the final layer outputting logits for domain prediction), a hidden layer size of 786, and a dropout rate of 0.1. We initiated the learning rate to 3e-5, used a batch size of 32, and ran each of the models for 1 epoch. We used the BertAdam optimizer, as was used in Lee et. al. [2]. For all models, we use a KL monotonic annealing procedure, which gradually increases the importance of the adversarial loss over training. Each model took approximately 30 hours to train under these parameters.

We trained 3 adversarial models, each with a different $\lambda$ parameter: $\{0.01, 0.05, 0.1\}$. We endeavored to study the effect of the importance on adversarial loss on model performance. Intuitively, a higher $\lambda$ parameter would incentivize the QA model to learn more domain-invariant features, but penalizing the model too much too early in training may result in a loss of performance generally. Thus, we train three models to observe the effect of this parameter on performance.

#### 4.4.5 Re-ranking Algorithm

We came across an interesting technique for adversarial QA in [23]. The technique consists of extracting top 20 candidate answers from the QA model and re-ranking them on the basis of maximum overlap of named entities between the question and the sentence containing the candidate answer. Based on the intuitive similarity between adversarial QA and domain-agnostic QA, we decided to try this method. The results we obtain using this technique are modest and are listed in Tables 1 and 2.

The authors of [23] uses a pre-trained transformer-based named entity recognition (NER) system to extract the named entities from the question and from the sentences containing the top 20 candidate answers. Then these answers are re-ranked on the basis of the number of common named entities with the question. This procedure was shown to give very good results on the *Addsent* and *AddOneSent* datasets for adversarial QA.

We deviated from the procedure in [23] on one key aspect. We did not have access to a large pre-trained NER model. Therefore, we used a proxy for named-entities. We extracted words starting with capital alphabets and considered them as named-entities in our algorithm. One important observation was that our baseline model picked many of the top 20 candidate answers from the same sentence in the passage. In this case, the re-ranking algorithm did not change the relative ranks of these candidate answers. After analysing some of the examples of the procedure, we realized that this algorithm is very

| Model | F1 | EM |
|---|---|---|
| Baseline | 40.45 | 25.92 |
| Baseline with Augmented Data | 43.52 | 26.18 |
| Adversarial Model with $\lambda = 0.01$ | 40.74 | 24.87 |
| Adversarial Model with $\lambda = 0.05$ | 40.65 | 25.13 |
| Adversarial Model with $\lambda = 0.1$ | 39.89 | 23.82 |
| Baseline with Re-Ranking | 41.02 | 25.92 |
| Baseline with Augmented Data and Re-Ranking | 42.57 | 25.39 |
| Adversarial Model with $\lambda = 0.01$ and Re-Ranking | 39.01 | 23.30 |
| Adversarial Model with $\lambda = 0.05$ and Re-Ranking | 39.36 | 23.82 |
| Adversarial Model with $\lambda = 0.1$ and Re-Ranking | 38.40 | 22.77 |

Table 1: Comparison of Model's on Dev Set

specific to adversarial examples appearing in *Addsent* and *AddOneSent*, and does not extend very well to the domain-agnostic QA problem.

### 4.5 Results

In Table 1, we can see a comparison of all models on the dev set. These data comprise 126 examples from DuoRC [16], 128 examples from RACE [17], and 128 examples from RelationExtraction [18]; the evaluation metrics are produced on all three datasets combined.

Since all of our models use a pre-trained DistilBERT model as the main engine for embedding input tokens, we can isolate the impact of changes in the input data (using augmented data or not), the training procedure (adversarially trained or not), and the ranking of the predicted answer span (using our re-ranking algorithm or not).

While many of the models have similar F1 and EM scores, the baseline model with augmented data clearly outperforms all models on both evaluation metrics. **The baseline model with augmented data, on the RobustQA test leaderboard, achieves an F1 score of 54.39 and EM score of 35.76**. This model achieves the best test scores of all of our submissions. Differences in the F1 and EM scores on the test set compared to the dev set can be attributed to differences in the distribution of the data (i.e. the data sample in dev set may constitute a different distribution of data than the one in the test set).

The adversarial models tend to marginally outperform the vanilla baseline model on F1, but fall short of the vanilla baseline on EM. This suggests that adversarially training models given our training configuration (number of training examples, epochs, learning rate, etc,) does not produce noticeable improvements in performance. There may be a few reasons for this. First, we have subsampled the data so the adversarial training could not take advantage of the full variety of question-answer examples. Arguably, with more examples, the baseline with augmented data could also perform correspondingly better; however, when comparing the baseline model trained on the entire training set with the baseline model trained on an augmented data set using the entire training set, we see that the F1 score on the entire dev set actually drops from approximately 48 to 47. Thus, improvements using data augmentation may not necessarily scale with the size of the training data and must be studied further. Secondly, the baseline model gets to see twice the number of examples in its training than the adversarial model does, so it inherently has a data advantage over the adversarial model.

We observe that changes in the lambda parameter produce marginal differences in the adversarial models' F1 and EM metrics. We believe this behavior is due to the restricted number of batch iterations the models are trained since we only train for one epoch. Thus, due to the monotonic annealing schedule for the $\lambda$ parameter, the model loss does not weight the adversarial loss highly enough for a significant portion of training in order to result in significant changes across models.

Notably, the re-ranking algorithm does not produce better answer predictions. In fact, as we see in Table 1, the re-ranking algorithm only improves the baseline model; all other models see both their F1 and EM scores suffer when using re-ranking for their answer span prediction. Note that the re-ranking method improves the results only when the QA model fails to identify the correct sentence to look at for the answer. After inspecting several examples, we noted that our QA models typically looked at the correct sentence but did not extract the correct set of words from the sentence. Answer re-ranking doesn't help in this scenario. Further, we notice that the *Race* dataset is closer to adversarial evaluation than the

| Model | DuoRC | | Race | | Relation Extraction | |
|---|---|---|---|---|---|---|
| | F1 | EM | F1 | EM | F1 | EM |
| Baseline | 34.57 | 28.57 | 25.46 | 14.06 | 61.23 | 35.16 |
| Baseline with Augmented Data | 38.82 | 29.37 | 27.69 | 11.72 | 63.98 | 37.5 |
| Adversarial Model with $\lambda$ = 0.01 | 35.95 | 29.37 | 25.05 | 10.94 | 61.16 | 34.375 |
| Adversarial Model with $\lambda$ = 0.05 | 37.17 | 30.16 | 24.80 | 12.5 | 59.93 | 32.81 |
| Adversarial Model with $\lambda$ = 0.1 | 34.41 | 26.19 | 25.10 | 11.72 | 60.09 | 33.59 |
| Baseline with Re-ranking | 34.81 | 26.98 | 27.61 | 16.41 | 60.53 | 34.38 |
| Baseline with Augmented Data and Re-ranking | 37.30 | 27.78 | 27.36 | 11.72 | 62.98 | 36.72 |
| Adversarial Model with $\lambda$ = 0.01 and Re-ranking | 32.67 | 26.19 | 24.95 | 10.16 | 59.30 | 33.59 |
| Adversarial Model with $\lambda$ = 0.05 and Re-ranking | 35.88 | 28.57 | 24.76 | 11.72 | 57.37 | 31.25 |
| Adversarial Model with $\lambda$ = 0.1 and Re-ranking | 32.87 | 25.39 | 24.72 | 10.93 | 57.53 | 32.03 |

Table 2: Comparison of Model's on Subsets of Dev Set

other datasets. This is because it is collected from examinations for high school students. In this setting, answer re-ranking helps improve the performance of the baseline QA model by 2 points on the F1 score.

## 5 Analysis

Quantitative analysis as in our **Results** section help us to understand the general accuracy and performance of our models. To better understand why our models may behave the way they do, we explore two methods for qualitative analysis. First, we evaluate each of our models on each of the individual datasets in the dev set, and analyze what the models' performances on these subsets tell us about the inner workings of the model. We also analyze specific answer examples produced by the best performing models, as they may indicate in which cases the model excels and in which cases the model fails.

In Table 2, we compare all models across the individual domain-level dev sets on both the F1 and EM scores. This granular detail reveals more perspective on the behavior of the models. Some broad trends are that all models tend to have better nominal F1 and EM scores on Relation Extraction data compared to DuoRC and Race – in fact, models consistently perform the worst on Race. This may be due to the way these datasets are constructed. Relation Extraction data are from triplets of the form relation-question-answer [18], where the entity identified in the question is also in the answer. So, the question answer pairs we derive from this data may result from more easily identifiable relations, such as "educated_at", "spouse_of", or "occupation." Race, on the other hand, is derived from examination data, which contain more difficult questions, such as "Why" questions, that require more reasoning.

Although the baseline model with augmented data improved over the vanilla baseline and outperformed all of the other models, it achieves lower EM scores on some models an both DuoRC and Race – the baseline model with re-ranking achieves the best EM score on Race and the adversarial model with $\lambda$ parameter 5 achieves the best EM score on DuoRC. This result is due to the fact that the baseline model with augmented data includes words in its answer span that overlap with many of the golden answers, but do not match many of them exactly. Perhaps because the model was trained on synonym data and back-translated data which changes or replaces words without necessarily matching the surrounding context, the model is less able to identify the correct context in which the true answer lies. Also, re-ranking also played a significant role in improving the vanilla baseline's performance on Race.

The adversarial models exhibit more distinct results per dataset. The value of the lambda parameter makes a very small difference to the performance of the model. Overall there is a small decrease in scores as we increase lambda, however there is no consistent trend across datasets (as seen in Table 1). This means that the adversarial training idea is not quite successful in this setting. Overall, the best value of lambda out of the ones we tried is $0.01$, which coincides with what the original paper used.

The re-ranking algorithm shows better performance on particular subsets of the dev data. In particular, we see that the Race F1 and EM scores for the adversarial models with re-ranking are similar to those metrics for the adversarial models without re-ranking. On the other datasets, though, re-ranking results in a clear drop in both F1 and EM scores. These results are consistent with the fact that in general, re-ranking seems to dampen performance.

We can also view a model's answer span predictions, and qualitatively analyze where a model tends to perform admirably and where it tends to fall short of expectation.

**EXAMPLE 1:**
**Question:** How many people still lack electricity in the world now?
**Context:** Earth Day has come and gone, but *it's a fact of daily life that 1.6 billion people around would have no electricity in their homes*. . . . *By 2030, when the Earth's population will be likely to top 8billion, 1.3 billion people will still lack electricity*. . . .
**Answer from Augmented Model:** 1.6 billion
**Answer from Baseline Model:** 1.3 billion
**Answer from Adversarial Model:** 1.3 billion
**Analysis:** Note that the data augmented model is able to understand the meaning of the question (even though the wording is different) because of the back translation. While both baseline and adversarial models were referring to the sentence where there was exact match of the word "lack".

**EXAMPLE 2:**
**Question:** Who wants to start a camp?
**Context:** Diary of a Wimpy Kid: Ivy and Bean Make the RulesBy Annie Barrows Bean's older sister Jessie goes to a summer camp called Girl Power 4-Ever, but Bean can't join her because she is too young. . . . *So Bean and her best friend, Ivy, decide to create their own camp*. At Camp Flaming Arrow, Ivy and Bean come up with all the activities and, of course, they make the rules. . . .
**Answer from Augmented Model:** Ivy and Bean
**Answer from Baseline Model:** Bean and her best friend, Ivy, decide to create their own camp. At Camp Flaming Arrow, Ivy and Bean
**Answer from Adversarial Model:** Flaming Arrow, Ivy and Bean
**Analysis:** Note that the baseline model is able to to identify the correct sentence. But, it could not extract the right words for the answer from it. While, the augmented model could extract the correct answer. We observe this pattern in many more examples, that the baseline model could identify the correct sentence but picked an incorrect set of words, which logically did not sound like an answer to the question. The data-augmented model was better in this respect.

**EXAMPLE 3:**
**Question:**What does the Five Friendlies mean when put together?
**Context:** Designed to express the playful qualities of five little children who form an intimate circle of friends, *the Five Friendlies also embody the natural characteristics of four of China's most popular animals*–the Fish, the Panda, the Tibetan Antelope, the Swallow–and the Olympic Flame. . . . *When you put their names together–Bei Jing Huan Ying Ni–they say "Welcome to Beijing,"* offering a warm invitation that reflects the mission of the Five Friendlies as young ambassadors for the Olympic Games.
**Answer from Augmented Model:** embody the natural characteristics of four of China's most popular animals
**Answer from Baseline Model:** "Welcome to Beijing,"
**Answer from Adversarial Model:** "Welcome to Beijing,"
**Analysis:** In this example, the model trained on data-augmented text related the words "mean" and "embody". The baseline and adversarial model referred to the right sentence and produced the correct answer. This is because the model trained on data-augmented text was trained to learn synonyms and back-translations, it confused between "mean" and "embody". We conjecture that if we had better quality and fewer word-substitution in data augmentation, then the model would not have made such mistakes.

## 6 Conclusion

The best way to tackle robustness problems in deep learning applications in NLP continues to be an open question. We have trained, examined, and analyzed several deep learning question answering models to compare the efficacy of a variety of approaches to explore this issue. From our results, we learned that adversarially training models produces slight gains in performance. Using augmented data techniques, such as word substitution and back translation, prove to be even more effective, and produced our best performing model with an F1 score of 54.39 and EM score of 35.76 on the test set. We also tested an approach in re-ranking our answer span predictions to better match the associated question, but this process only improved one of our models, and hurt the rest.

We acknowledge that our work has limitations and thus invites future research. The training data size and training time (number of epochs) were very limited in our study, and analyzing the effect increasing both of these parameters may produce interesting results since we have seen data augmentation may have decreasing marginal gains when training data size increases. Future work may also study the impact of data augmentation on the adversarial model, to analyze the impact of this techniques on different models.

## 7 Acknowledgments

We would like to thank the CS 224N instructors and teaching staff for providing helpful support throughout the project, including guidance on using Microsoft Azure and training the baseline models.

## References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[2] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. *arXiv preprint arXiv:1910.09342*, 2019.

[3] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. Metareg: Towards domain generalization using meta-regularization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. *CoRR*, abs/1710.03077, 2017.

[5] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[6] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, Franc¸ois Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural network. *Journal for Machine Learnign Research*, 2016.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[8] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. 2020.

[9] Wee Chung Gan and Hwee Tou Ng. Improving the robustness of question answering systems to question paraphrasing. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[10] Bowen Wu, Haoyang Huang, Zongsheng Wang, Qihang Feng, Jingsong Yu, and Baoxun Wang. Improving the robustness of deep reading comprehension models by leveraging syntax prior. pages 53–57, November 2019.

[11] Zhijing Wu and Hua Xu. Improving the robustness of machine reading comprehension model with hierarchical knowledge and auxiliary unanswerability prediction. *Knowledge-Based Systems*, 203:106075, 2020.

[12] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

[13] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[14] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

[15] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[16] Amrita Saha, Rahul Aralikatte, Mitesh M Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *arXiv preprint arXiv:1804.07927*, 2018.

[17] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*, 2017.

[18] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*, 2017.

[19] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *arXiv preprint arXiv:1912.02145*, 2019.

[20] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[21] Edward Loper and Steven Bird. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*, 2002.

[22] Rishi Bhalodia. Introduction to data augmentation in nlp. *Medium*, 2019.

[23] Sagnik Majumder, Chinmoy Samant, and Greg Durrett. Model agnostic answer reranking system for adversarial question answering. *arXiv preprint arXiv:2102.03016*, 2021.