Multiple Domain Question-Answer Generation

Kimberly Lu

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-44

http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-44.html

May 15, 2019

Acknowledgement

I would like to thank Richard Shin for his advice and feedback throughout the course of this project, as well as my advisor Dawn Song and reader John DeNero for their feedback.

Multiple Domain Question-Answer Generation

Kimberly Lu
Department of Electrical Engineering and Computer Science
University of California, Berkeley
kimberlylu@berkeley.edu

Abstract

In this work we explore the usefulness and practicality of domain adaptation and multi-domain learning methods in question-answer generation. Unlike recent work in question-answer generation which focuses on processing single-domain data to create synthetic reading comprehension datasets (Du and Cardie, 2018), we propose a question-answer generation system that can adapt to datasets containing multiple domains while still achieving similar or better performance in single domains compared to a baseline. We apply our system, consisting of an answer extraction system and a question generation system, to the SQuAD and SciQ reading comprehension datasets and evaluate its efficacy in mixed- and single-domain settings. Our domain adaptation method achieves higher performance than baselines on the mixed-domain and SciQ datasets in both answer extraction and question generation.

Problem Definition and Motivation

In recent years, phishing attacks have grown more common and more sophisticated. The effectiveness of such attacks means the best defense against phishing is to automatically detect and deflect attacks without allowing the human victim to be exposed. Due to the rise of spear phishing, which are highly personalized and targeted attacks on just one or a few victims in an organization, it is sometimes extremely difficult to verify if an email is legitimate or not. For instance, the attacker may impersonate someone in the victim's contact book. In such a case, it would be advantageous to have a bot capable of engaging in dialogue with the attacker and verifying their identity in lieu of the human to detect these subtle spear phishing attacks.

One way in which this can be achieved is through an automated question-answer generation system that can verify a sender's identity through generating security questions. Such a system would use data about a user's previous communications with the sender in order to create relevant questions that verify whether the email sender is truly who they claim to be. This system would be able to effectively draw from prior email and chat text to generate a question that only the real email sender would be able to answer. Since security question generation has many uses outside of phishing defense, this system can be applied in many different security applications, not just phishing.

With this motivation, we investigate methods to apply domain adaptation to the question-answer generation task. In our particular use case, the task is to achieve high performance over multiple-domain data in a setting where the domains seen in training encompass all domains seen during inference, that is, for the set of domains in the training data D_p and the set of domains in the test data D_q, $D_q \subseteq D_p$. Such domains would include emails, chat messages, and other types of communication. Email and chat texts differ greatly in content and structure. We wish to generate high-quality question-answer pairs for each domain in the dataset, necessitating that the question-answer generation system captures domain-specific information. This differs from other work in domain adaptation that focus on training a model in a source domain such that it can achieve high performance in an unknown target domain, and shares similarities with the problem of multi-domain learning (Dredze and Crammer, 2008). We aim to improve the question-answer generation system's overall performance on a dataset with multiple domains while still maintaining comparable or better performance on each domain individually.

Recently, research has been directed towards using question-answer generation systems to create reading comprehension datasets for training models on various NLP tasks (Du and Cardie, 2018). Similar to previous work, we approach this problem by first extracting potential answers from a text input, and then using these answers and the input to generate questions relevant to the answers. However, the problem of domain adaptation on this task is still little-explored. To this end, we contribute the following:

- The implementation and evaluation of an answer extraction system which leverages BERT pre-trained models to identify answer spans in a document. We also apply a domain adaptation method to this model and assess its impact on the model's performance on both single- and multiple-domain data.
- The implementation and evaluation of a domain adaptation method on the QG-Net question generation model architecture, assessed on both single- and multiple-domain data.
- Proposed methods in which domain adaptation in this task can be improved to achieve even higher performance than the results in the aforementioned implementations.

Context: Newton's laws and Newtonian mechanics in general were first developed to describe how forces affect idealized point particles rather than three-dimensional objects. However, in real life, matter has extended structure and forces that act on one part of an object might affect other parts of an object. For situations where ...

Question: What may a force on one part of an object affect?
Answer: other parts of an object

Question: What did Newton's mechanics affect?
Answer: idealized point particles rather than three-dimensional objects

Figure 1: Example context sequence from the SQuAD dataset (Rajpurkar et al., 2016) with ground truth question and answer pairs. Text spans highlighted in pink are answer spans. The context has been truncated for brevity.

Related Work

Question-answer generation is a multi-faceted problem that has been tackled in various ways in the past. Until recently, most work has focused on either automated question answering or question generation and not both together.

Answer Extraction

We frame the task of extracting answer spans from an input sequence as a token classification problem, a research area that has already received much attention in tasks such as named entity recognition, part-of-speech tagging (Huang, Xu, and Yu, 2015), and other token classification tasks (Devlin et al., 2018).

In the context of question-answer generation, the majority of previous work on answer extraction has focused on automated question answering. Question answering systems take in an input question and context, and output an answer to the question that is a span of text in the context. Effective systems for automated question answering have been implemented with machine learning approaches. Cui et al. (2016) used recurrent networks with nested attention to tackle cloze-style reading comprehension tasks. Yu et al. (2018) proposes a convolutional neural network with attention that trains significantly faster than recurrent network approaches while also achieving better performance. Devlin et al. (2018) pre-trained transformers (Vaswani et al., 2017) on a wide range of tasks and fine-tuned them for question answering and achieved very high performance, demonstrating the advantages of transfer learning. The SQuAD dataset (Rajpurkar et al., 2016) is commonly used for training and evaluation question answering tasks. It is a collection of Wikipedia article texts with labeled answer spans and associated questions created through crowdsourcing.

Question Generation

Early work in question generation used rule-based approaches that often required well-written templates with blanks to be filled in by relevant text spans in the input context (Ali, Chali, and Hasan, 2011). However, with the rise of deep learning and large QA datasets like SQuAD, recent work has successfully used recurrent neural network architectures to achieve better performance on this task. Recent approaches treat the problem as a sequence-to-sequence learning task similar to machine translation. Input sequences are first encoded into a feature-rich representation using word embeddings and an architecture such as a convolutional neural network or a bi-directional LSTM (biLSTM). A decoder then uses the encoded input to generate an output sequence. These systems require the answer span to be known *a priori*, so as to generate relevant questions.

Du, Shao, and Cardie (2017) use a biLSTM with attention to encode a given context and answer pair and then feeds this information into an LSTM decoder to generate a question sequence. Yuan et al. (2017) uses both supervised and reinforcement learning with a biLSTM and augments word vectors with part of speech tags. Wang et al. (2018) extends on the biLSTM approach by augmenting word vectors with additional information such as named entity tags, part-of-speech, and whether a token is part of an answer span. During decoding, generated question tokens are picked probabilistically from an output word distribution using an LSTM or from the input context using a pointer network.

Creating Question-Answer Datasets

More recently, work has been done on generating question-answer pairs from unlabeled text. There are a wide range of applications of question-answer generation, aside from the aforementioned security application. Question-answer generation can be used to create test questions for educational purposes or even generate datasets for question-answering networks to train on. Du and Cardie (2018) first does answer span identification using methods similar to named entity recognition, then uses the identified answer spans to perform question generation. Input contexts are encoded using a LSTM with additional features such as a co-reference position feature embedding and answer labels. The encoded input is then fed into a decoder with an attention-copy mechanism. By using this method, they are able to generate question-answer pairs from unlabeled text, creating a synthetic question answering dataset. Kumar et al. (2018) uses a pointer network to select an answer span from the input context and appends answer information and rich linguistic features to the input representation. They encode the input with a biLSTM with attention and output a generated question using an LSTM decoder.

Our work aims to apply domain adaptation methods to question-answer generation to achieve high performance across multiple domains of text. Previous work such as (Du and Cardie, 2018) focus on a single text domain, for instance a set of Wikipedia articles. For an application such as generating security questions for phishing protection, the input text could be parts of emails or chat conversations,

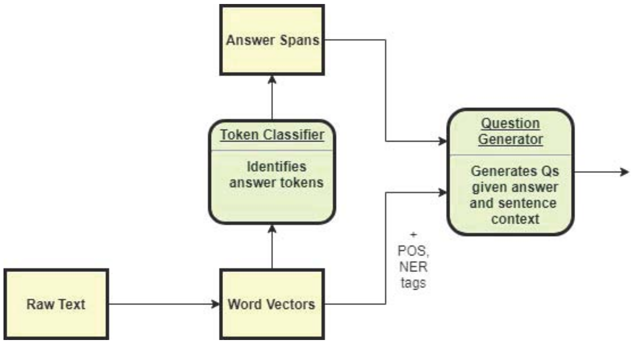


Figure 2: Overview of the question-answer generation system.

which differ greatly in structure and content compared to Wikipedia articles. In such a case it is important to be able to adapt to different types of text. To my knowledge, domain adaptation in question-answer generation has not yet been explored, although methods for domain adaptation in machine translation (Britz, Le, and Pryzant, 2017; Chen and Cardie, 2018) share some similarities.

Approach

In order to generate question-answer pairs from a source text, we first identify continuous spans in the text that constitute answer candidates. These are considered question-worthy text spans. For each candidate answer span, a question Q is generated based on the given span and the sentence from the context that contains the answer span, such that Q asks a question relevant to the span that may rely on other information in the sentence. We additionally apply domain adaptation methods to both the answer extraction system and the question generation system to improve performance over multiple domains.

Answer Span Extraction

We treat answer span extraction as a token classification task and use a pre-trained BERT model (Devlin et al., 2018) as a base for our token classifier. The BERT model, or Bidirectional Encoder Representations from Transformers, has representations pre-trained on cloze tasks and next-sequence prediction which allows it to be fine-tuned for both token- and sequence-classification tasks. Fine-tuned BERT models have achieved state-of-the-art performance on a wide variety of tasks. For this task, we fine-tune the BERT-base cased model, the smaller of the two model architectures introduced in (Devlin et al., 2018) with approximately 110M parameters. Better performance may be achieved by the BERT-large model at the cost of taking significantly longer to fine-tune.

The answer span extraction system takes in paragraph contexts and a list of answer spans found in each context. The contexts are tokenized into WordPieces (Wu et al., 2016). Each token's input representation T is created by summing its WordPiece embedding E_w and BERT positional embedding E_p. A segment embedding E_s used for next-sentence tasks is also added, but for single-sequence tasks such as this one it is a constant value.

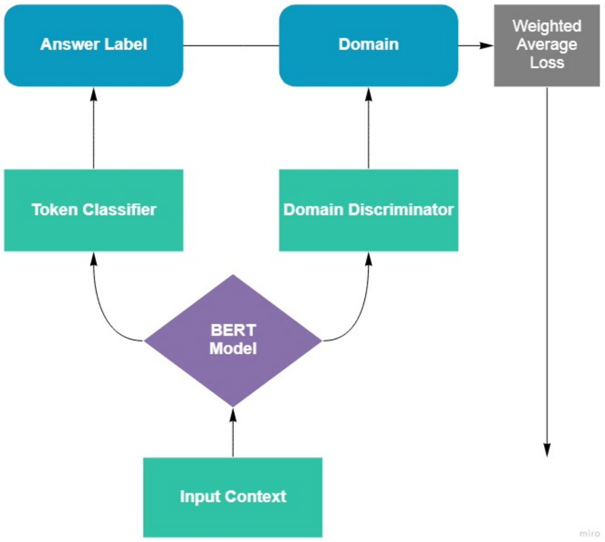


Figure 3: Visualization of the answer extraction system showing the domain discriminator. The weighted average loss is backpropagated such that the model is encouraged to learn features that are unique to each domain but still capture useful information for answer token classification.

$$T = E_w + E_p + E_s$$

The list of answer spans is then processed to assign a label to each token indicating whether it is part of an answer span or not. The pre-trained BERT transformer takes in the paragraph context and outputs the final hidden state representation of the context, a fixed-size 768-element vector. This final hidden state is then fed into a linear classifier that learns to classify each token from the BERT output. During backpropagation both the token classifier and the BERT model are updated using the token classifier's loss.

Since this system predicts an answer label for each token rather than text spans, the token labels must be processed to reconstruct answer spans from the predictions. This is done both to facilitate evaluation of the results and to enable the output to be used for the question generation system. All adjacent tokens labeled as an answer are considered part of the same answer span.

Domain Adaptation To apply domain adaptation to the answer span extraction task, we modify the BERT model by adding a discriminator that jointly predicts the domain of the given context from the encoders final hidden state. The discriminator is a linear classifier that takes in a fixed-dimension, pooled representation of the input sequence. During training, we minimize the weighted average of the token classifier's loss and the discriminator's loss. This combined loss is then used to update the token classifier, domain classifier, and BERT model.

Intuitively, the addition of this discriminator forces the BERT model to learn a feature representation that captures

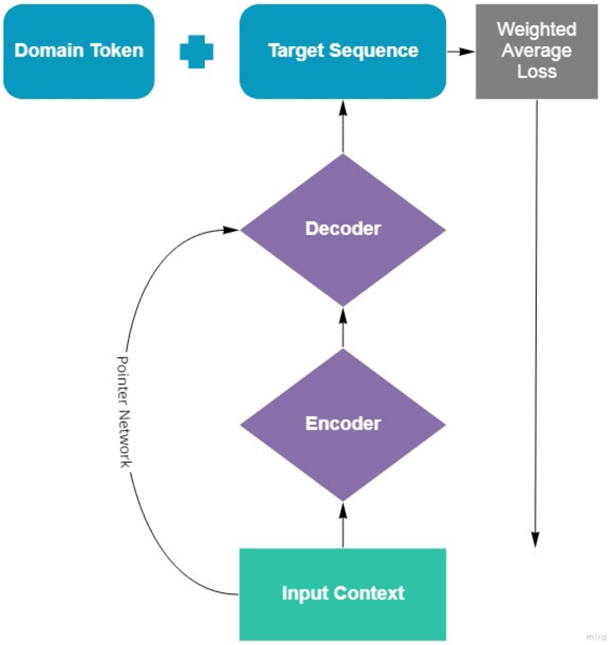


Figure 4: Visualization of the question generation system showing the domain discriminator. The domain token is prepended to the target sequence during training only.

domain-related information so that the domain discriminator can more easily differentiate between domains. These domain-specific features help the classifier maintain good performance in each domain in the dataset while also being able to flexibly shift gears when performing inference on a dataset with multiple domains mixed together. At the same time, the BERT model must continue to capture features that are useful for token classification. In tandem, these two effects cause the BERT model to learn to output a feature representation with domain-specific features that are useful for answer identification.

Question Generation

We frame the problem of question generation as a sequence-to-sequence task (Sutskever, Vinyals, and Le, 2014), mapping an input sequence (a context sentence) to an output sequence (a question). We use the QG-Net model architecture (Wang et al., 2018) which consists of a biLSTM encoder that encodes word vectors into hidden states, and a LSTM decoder that outputs the predicted question.

QG-Net takes in pairs of answers and the sentence that contains each answer. The sentences are tokenized and converted into vectors using the GloVe (Pennington, Socher, and Manning, 2014) word embeddings, then are additionally augmented with features that indicate whether the word is part of an answer span, its part of speech, whether it's a named entity, and its word case. That is, for an input sentence with word vectors $\{c_1, c_2, ...c_n\}$ then each word vector c_i is modified as follows:

$$\tilde{c_i} = [c_i, ANS, POS, NER, CAS]$$

During encoding, a biLSTM processes the input sentence $\{\tilde{c_1}, \tilde{c_2}, ...\tilde{c_n}\}$ in the forward and backward directions to produce a hidden state for each word vector that captures dependencies on words that come before and after it. During decoding, a LSTM generates a predicted sentence one token at a time. The token generated at timestep t is dependent on the encoding of the input sequence and all previously generated tokens. That is, if s_t is the hidden state of the tth generated token,

$$s_t = LSTM(C, s_{t-1})$$

where C is the encoded input context. During decoding, each token output is probabilistically chosen from either an output vocabulary distribution (determined by the decoder) or a distribution over the words in the input context (determined by a pointer network). The addition of the pointer network helps increase question relevance by using words from the input. Therefore the final generated token is chosen from a union of the question vocabulary V^Q and the set of unique words in the input context V^S. Beam search is used to select the best candidate output sequence.

Domain Adaptation To apply domain adaptation to the question generation task, we prepend a domain token to the target sequences in the training data. This method is referred to as ”target token mixing” in Britz, Le, and Pryzant (2017). Similar to a discriminator used in the answer extraction task, this forces the encoder to output a feature representation of the input that captures domain-related data, so that the decoder can output the correct domain token during training. Additionally, since each word generated during decoding is affected by the words that come before it, the predicted domain token also has the effect of altering the probability distribution of words generated in the question. This allows the question generator to identify the domain of the input during inference and use that information to adjust the output accordingly.

The domain token at the start of the predicted sequence is removed during evaluation, and target sequences in the dev set do not contain a domain token.

Training and Experimental Setup

Datasets

We use the SQuAD (Rajpurkar et al., 2016) and SciQ (Johannes Welbl and Gardner, 2017) datasets as our two domains during training and evaluation. SQuAD consists of paragraph contexts from Wikipedia articles and more than 100K question-answer pairs associated with these contexts. In the dataset, answers are a contiguous span of text from their associated context. The questions in SQuAD are generated by human crowdworkers through Amazon Mechanical Turk. The SQuAD dataset consists of 18,896 contexts in the training set with 86,832 associated question-answer pairs, and 2,067 contexts in the dev set with 10,526 associated question-answer pairs.

The SciQ dataset consists of paragraph contexts from science textbooks and 13,679 multiple-choice questions associated with these contexts. These multiple choice questions are also crowdsourced, but unlike SQuAD, the answers are

Method	Eval data	Precision			Recall			F-measure		
		Prop	Bin	Exact	Prop	Bin	Exact	Prop	Bin	Exact
Train on SQuAD	SQuAD	50.16	52.43	20.61	35.08	46.10	**20.11**	**41.29**	**49.06**	**20.35**
	SciQ	13.82	16.75	8.70	**59.55**	**64.00**	**34.06**	22.44	26.56	13.86
Train on SciQ	SquAD	**53.67**	**53.86**	13.41	2.27	3.93	1.05	4.36	7.33	1.94
	SciQ	51.56	53.31	35.36	41.24	46.06	31.76	**45.83**	**49.42**	33.46
Trained on mixed data	SQuAD	48.51	50.79	19.62	**35.33**	**46.28**	19.91	40.88	48.43	19.76
	SciQ	**52.38**	**54.08**	**38.40**	36.93	41.21	29.70	43.32	46.78	33.49
Mixed w/domain adaptation	SQuAD	50.07	52.54	**20.63**	32.81	43.08	18.72	39.64	47.34	19.63
	SciQ	51.28	52.81	38.10	39.83	43.39	32.00	44.83	47.64	**34.78**

Table 1: Precision, recall, and F1 metrics for each answer extraction model evaluated on single-domain data.

Method	Precision			Recall			F-measure		
	Prop	Bin	Exact	Prop	Bin	Exact	Prop	Bin	Exact
Train on SQuAD	34.59	37.11	15.32	**39.26**	**49.41**	**22.41**	36.77	42.38	18.20
Train on SciQ	**53.26**	**54.56**	**28.66**	8.85	11.20	6.20	15.18	18.58	10.19
Trained on mixed data	48.75	50.93	21.40	35.10	45.18	20.95	**40.81**	47.89	21.17
Mixed w/domain adaptation	50.73	53.06	22.61	34.00	43.82	20.45	40.71	**48.00**	**21.48**

Table 2: Precision, recall, and F1 metrics for each answer extraction model evaluated on mixed-domain data. The evaluation set consists of an equal number of SQuAD and SciQ examples.

not necessarily text spans from the context. We convert SciQ into a SQuAD-like format to facilitate training and testing on both datasets. The multiple-choice questions are converted into question-answer pairs by pairing each question with its correct answer. In addition, if an answer is not a contiguous text span from the associated context, we remove this example from the dataset. After processing, this leaves us with 9,574 contexts in the training set with 9,219 associatd question-answer pairs and 825 contexts in the dev set with 793 associated question-answer pairs.

For training and testing answer extraction, we assign a label to each token in the context indicating whether it is part of a ground truth answer span. For question generation, we take each answer and the sentence containing the answer and pair them with the associated ground truth question.

To evaluate the effectiveness of our domain adaptation methods, we also created a mixed dataset that consists of an equal number of SQuAD and SciQ examples with domain labels. As SciQ is smaller than SQuAD, we use the full SciQ train and dev sets when creating the mixed dataset. For training and testing answer extraction, each example in the training and dev set has a domain label that must be predicted jointly along with the token labels. For question generation, the domain label is prepended to the target sequence for the training set and no domain labels are provided during evaluation.

Evaluation Metrics

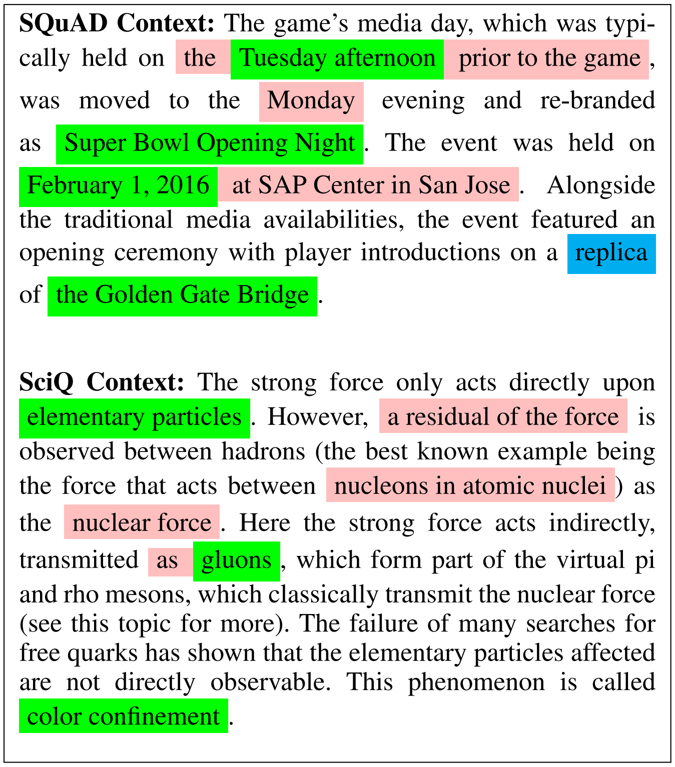
For answer span extraction, we evaluate our models performance using precision, recall, and F1 scores calculated against the ground truth answer spans. The boundaries of answer expressions are difficult to define even by human annotators (Wiebe, Wilson, and Cardie, 2005) so besides exact match (EM) metrics, we also use two types of soft precision and recall metrics: proportional overlap and binary overlap. Proportional overlap gives partial credit to predicted answer spans that overlap with a ground truth answer span, proportional to the amount of overlap (Johansson and Moschitti, 2010). Binary overlap counts any predicted answer span that overlaps with a ground truth answer span as correct (Breck, Choi, and Cardie, 2007).

For question generation, we evaluate our model's performance using BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014), and ROUGE-L (Lin, 2004). BLEU measures n-gram precision against ground truth questions while penalizing overly short sequences. The METEOR metric measures n-gram recall while also taking into account synonyms, paraphrasing, and stemming. ROUGE-L is a text summarization metric that measures the longest matching n-gram between the predicted and ground truth question.

Finally, we link our answer extraction and question generation systems with domain adaptation methods to produce a generated set of question-answer pairs from the mixed dataset. We then apply a top-performing question-answering system Document Reader (Chen et al., 2017) which has been pre-trained on SQuAD to our generated dataset and evaluate its performance. During evaluation we use the official SQuAD evaluation script which calculates exact match and F1 scores for the answers generated by the question-answering system.

Experiments

Training Data For both the answer extraction system and question generation system, we train four separate models. One model is trained solely on the full SQuAD set without any domain adaptation (”baseline SQuAD model”). One model is trained solely on the full SciQ set without domain adaptation (”baseline SciQ model”). One is trained on a the mixed dataset, which has an equal number of SQuAD and



SQuAD Context: The game's media day, which was typically held on the Tuesday afternoon prior to the game, was moved to the Monday evening and re-branded as Super Bowl Opening Night. The event was held on February 1, 2016 at SAP Center in San Jose. Alongside the traditional media availabilities, the event featured an opening ceremony with player introductions on a replica of the Golden Gate Bridge.

SciQ Context: The strong force only acts directly upon elementary particles. However, a residual of the force is observed between hadrons (the best known example being the force that acts between nucleons in atomic nuclei) as the nuclear force. Here the strong force acts indirectly, transmitted as gluons, which form part of the virtual pi and rho mesons, which classically transmit the nuclear force (see this topic for more). The failure of many searches for free quarks has shown that the elementary particles affected are not directly observable. This phenomenon is called color confinement.

Figure 5: Example answer spans in one SQuAD and one SciQ context sequence. Ground truth answer spans are highlighted in pink, predicted answer spans in blue, and spans where the predicted answer overlaps with a ground truth answer are highlighted in green.

SciQ examples ("baseline mixed model"). Finally a model that uses domain adaptation techniques is trained on the mixed set with domain labels ("domain adaptation model").

Model Parameters To fine-tune BERT for the answer extraction task, we used a batch size of 18 and 5 epochs for each of the models we trained. For each model, we used a learning rate of 3e-5 and a maximum sequence length of 512 tokens for context sequences. Contexts longer than 512 tokens were truncated, while contexts shorter than 512 tokens were padded. This length was chosen because it is the maximum length supported by BERT's positional embeddings. Only 0.03% of the SQuAD dataset and 0.1% of the SciQ dataset are longer than 512 tokens and were truncated.

When training the QG-Net models, we used a batch size of 64 and 20 epochs for each model and optimize using stochastic gradient descent.

Results and Discussion

We evaluate the effectiveness of our domain adaptation methods by comparing the performance of the domain adaptation model with the non-domain adaptation model trained on the mixed set (referred to as the "baseline mixed model"). We also verify that the domain adaptation model achieves comparable performance with the baseline SQuAD and baseline SciQ models in a single-domain setting.

Context 1: temporomandibular joint the temporomandibular joint (tmj) is the joint that allows for opening (mandibular depression) and closing (mandibular elevation) of the mouth , as well as side-to-side and protraction/retraction motions of the lower jaw . this joint involves the articulation between the mandibular fossa and articular tubercle of the temporal bone , with the condyle (head) of the mandible .
Human: the temporomandibular joint (tmj) is the joint that allows for opening (mandibular depression) and closing (mandibular elevation) of this ?
Baseline mixed model: what involves the articulation between the mandibular depression and closing of the lower jaw ?
Domain adaptation model: what is the name of the joint where the articulation between the mandibular and articular tubercle ?

Context 2: during the period in which the negotiations were being conducted , tesla said that efforts had been made to steal the invention . his room had been entered and his papers had been scrutinized , but the thieves , or spies , left empty-handed .
Human: what was tesla afraid someone was trying to do with his invention ?
Baseline mixed model: what were tesla 's efforts made to do ?
Domain adaptation model: what did tesla say efforts had been made to do ?

Figure 6: Comparison of the ground truth human-written questions with our domain adaptation model's questions generated from the given context. Answers are highlighted. Context 1 is from the SciQ dataset, while Context 2 is from the SQuAD dataset. Both contexts' domains were correctly identified by the domain adaptation model.

Answer Span Extraction Evaluation

Table 2 shows the precision, recall, and F1 metrics for each model evaluated on the mixed dataset. On all datasets, the domain classifier in the domain adaptation model achieved over 95% accuracy. The domain adaptation model outperforms the models trained on single domain data by a large margin. It also outperforms the mixed baseline model on precision metrics and on the binary and exact match F1 metrics. While the gains are modest, there is a clear trend of increased precision in the domain adaptation model compared to the baseline which indicates that the model is correctly learning domain-specific features that make its predictions more accurate in each domain. However, the addition of domain adaptation appears to also reduce the recall of the model on the mixed set. By comparing the performance of the domain adapation model with the baseline mixed model in Table 1 we see that this drop in recall occurs on SQuAD data. Since the proportion of answer words in

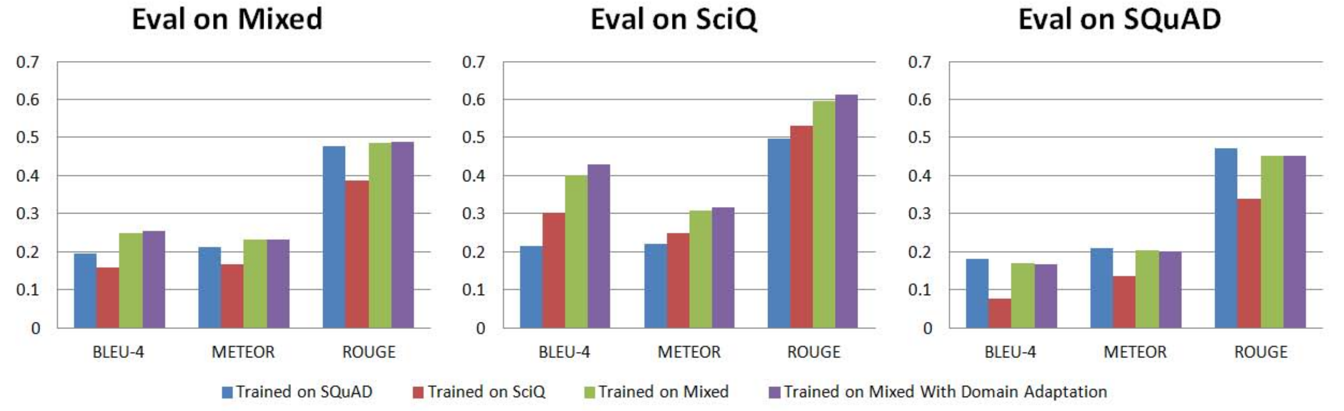


Figure 7: BLEU-4, METEOR, and ROUGE-L scores for each of the trained question generation models when evaluated on the mixed dataset, the full SciQ dataset, and the full SquAD dataset.

SciQ is much lower than in SQuAD, it may be that learning domain-specific features for the SciQ dataset inadvertently affected predictions in the SQuAD dataset.

As seen in Table 1, the domain adaptation model's performance on single-domain datasets is comparable to that of the baseline models trained on each respective domain. In addition, the domain adaptation model achieves comparable performance with the baseline mixed model on single-domain data, and even outperforms the baseline mixed model when evaluated on the SciQ dataset. This demonstrates that the addition of domain adaptation does not reduce the model's effectiveness on single domains. The fact that the domain adaptation model is able to improve its performance on the SciQ dataset by learning from additional SQuAD data indicates that the model is able to adapt what it learns from the SQuAD data to produce better predictions in the SciQ domain. From this we infer that this domain adaptation method can also be used to gain performance increases on known domains with few training examples. Thus this domain adaptation method offers modest gains in performance on mixed datasets while still performing well in a single-domain setting.

These results demonstrate that applying domain adaptation is both useful and practical in the task of multi-domain question-answer generation. Even with a simple approach, we are able to achieve performance gains on multiple domain data and adapt what it learns from one domain to improve inference in other domains. There are multiple ways in which the domain adaptation method can be modified to achieve greater improvements.

The fact that we are leveraging pre-trained BERT models could partially account for the slightness of the performance gains, as the pretraining tasks used to create the model already help it generalize across datasets. Training a BERT model from scratch would likely yield greater performance gains with this domain adaptation method.

There is also a flaw in the domain adaptation method where learned domain-specific features are not selectively used for predictions on their respective domain, but are used to make predictions on all domains. This reduces the number of domain-specific features per domain, which can cause problems in datasets with many domains. Additionally, the domain-specific features capture for other domains may reduce performance on a target domain. This is likely the reason why the domain adaptation model has lower recall scores on the SQuAD dataset compared to the baseline mixed model. To further improve performance, we could instead learn a separate set of features for each domain. This can be achieved by training a separate model for each domain. Then during prediction, the domain discriminator would first determine the domain of the example and choose the appropriate set of features to perform token classification with.

In conjunction with domain-specific feature sets, the model could also learn a set of general features that can be concatenated with domain-specific features during inference. These domain-indifferent features should capture useful information across all the datasets and therefore will improve token classifier performance in each domain. In order to extract these domain-indifferent features, we can use a domain discriminator that adversarially attempts to determine the domain from the set of features. The goal becomes to learn a set of features that are indistinguishable between domains, similar to the concept of generative adversarial networks (Goodfellow et al., 2014).

Question Generation Evaluation

Figure 7 shows the BLEU-4, METEOR, and ROUGE-L metrics for each model. During evaluation, we used the ground truth answer spans in order to get a clear idea of solely the question generation system's performance. We link the answer extraction and question generation systems together in the next section where we perform end-to-end evaluation.

The domain adaptation model is the highest performing on all metrics when evaluated on the mixed dataset. This shows that the addition of the domain token to the target sequence during training explicitly aids in question generation

Model	Dataset	Exact Match	F1 Score
Document Reader	Entire Generated Dataset	56.08	71.20
	SciQ portion	74.61	85.93
	SQuAD portion	52.86	68.65

Table 3: Performance of the pre-trained DocReader reading comprehension model on the dataset generated by our domain adaptation models. We report its performance on the entire dataset as well as each domain in the dataset.

on mixed domain data.

The domain adaptation model also has comparable or better performance on single domain data compared to the other models. In particular, it outperforms all other models by a significant margin on the SciQ dataset. This demonstrates that the domain adaptation method also helps the model adapt what it learns during training on one domain to improve the quality of question generation on different domains. SciQ has approximately half the training data size of SQuAD, so the effect is more pronounced in this dataset. In essence, the mixed dataset can be seen as augmenting the SciQ dataset with SQuAD examples. Because we are utilizing a domain adaptation method the model is able to apply what it learns from SQuAD training when doing SciQ inference. These results demonstrate that this domain adaptation method does not significantly degrade performance in a single-domain setting, and can even improve performance on domains with less training data.

Similar to the answer extraction domain adaptation method, the method we utilize from question generation focuses on teaching the model to extract domain-specific features. The performance of the domain adaptation model could be further improved on the mixed-domain dataset by also learning to extract domain-inspecific features which generalize well across all domains in the training set. These domain-inspecific features could be concatenated with domain-specific features during inference to produce higher quality output sequences.

End-to-End Evaluation

Using the domain adaptation answer extraction system and the domain adaptation question generation system we generated 3,938 question-answer pairs based off the context paragraphs contained in the mixed dev set. Unlike in the question generation evaluation, for end-to-end evaluation we generated questions based off of the answers extracted by our domain adaptation answer extraction system. The generated dataset contains an equal number of SQuAD and SciQ contexts, but only 583 of the question-answer pairs are from SciQ while 3,356 pairs are from SQuAD. We then use the single model Document Reader that has been pre-trained on the SQuAD dataset, the details of which are discussed in (Chen et al., 2017), to attempt to answer the questions in the generated dataset. Results are evaluated using the official SQuAD evaluation script which calculates exact match and F1 measures. Exact match measures the percentage of predictions that match any one of the ground truth answers exactly. F1 score measures the average amount of overlap between predicted and ground truth answers.

3 shows the performance of the Document Reader on our generated dataset. Compared to its performance on the SQuAD dev set where it achieves an exact match score of 69.5% and F1 score of 78.8%, the Document Reader performs less well on the generated dataset, indicating that it is more difficult or less coherent than SQuAD. This drop in performance makes sense in that the generated dataset contains question-answer pairs in two different domains, which inherently makes the task of question answering somewhat more difficult. Additionally, the question-answer pairs in the SQuAD dataset are human-generated, and accordingly are more coherent than generated pairs. Overall, the performance of the Document Reader on our dataset is reasonable and indicates that our generated dataset is more difficult than a single-domain dataset but not much lower in quality.

We also examine the performance of the Document Reader on each domain of the generated dataset separately. In the SciQ domain, the Document Reader actually performs better on our generated data than on the SciQ dev set, where it achieves an EM score of 69.09% and F1 score of 80.55%. As noted in previous sections, our domain adaptation model performs significantly better than other models when generating questions in the SciQ domain. Therefore, it is possible that the questions generated by our model are highly relevant to the associated answer and therefore easier to answer. Another possible cause for this performance increase is that our domain adaptation model is using what it has learned from the SQuAD training data to make the generated question-answer pairs more SQuAD-like. Since the Document Reader is trained on SQuAD, this would make the SciQ questions more similar to its training data.

Conclusion

We have presented domain adaptation methods to improve the performance of answer extraction and question generation in multiple-domain data and demonstrated their efficacy on the SQuAD and SciQ reading comprehension datasets. The addition of these methods produced improvements over the baseline model trained on the mixed dataset when evaluated on the mixed-domain data and on SciQ. These results demonstrate that the domain adaptation methods not only improve the model's ability to learn and do inference in multiple domains, but also can adapt what it learns in one domain to improve its performance in a separate domain. The domain adaptation model's significant performance increase in the SciQ domain indicates that these domain adaptation methods can also be used to increase performance on a known domain with few training examples by augmenting the training data with labeled examples from another domain.

Future work can improve on the domain adaptation methods presented in this paper by also learning a set of domain-inspecific features through using an adversarial domain discriminator. Further improvements can also be made by learning a separate set of domain-specific features for each domain and swapping between them based on the predicted domain of the test example.

References

Ali, H.; Chali, Y.; and Hasan, S. A. 2011. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, 58–67.

Breck, E.; Choi, Y.; and Cardie, C. 2007. Identifying expressions of opinion in context. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, 2683–2688. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Britz, D.; Le, Q.; and Pryzant, R. 2017. Effective domain mixing for neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, 118–126. Association for Computational Linguistics.

Chen, X., and Cardie, C. 2018. Multinomial adversarial networks for multi-domain text classification. *CoRR* abs/1802.05694.

Chen, D.; Fisch, A.; Weston, J.; and Bordes, A. 2017. Reading wikipedia to answer open-domain questions. *CoRR* abs/1704.00051.

Cui, Y.; Chen, Z.; Wei, S.; Wang, S.; Liu, T.; and Hu, G. 2016. Attention-over-attention neural networks for reading comprehension. *CoRR* abs/1607.04423.

Denkowski, M., and Lavie, A. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, 376–380. Baltimore, Maryland, USA: Association for Computational Linguistics.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805.

Dredze, M., and Crammer, K. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, 689–697. Stroudsburg, PA, USA: Association for Computational Linguistics.

Du, X., and Cardie, C. 2018. Harvesting paragraph-level question-answer pairs from wikipedia. *CoRR* abs/1805.05942.

Du, X.; Shao, J.; and Cardie, C. 2017. Learning to ask: Neural question generation for reading comprehension. *CoRR* abs/1705.00106.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991.

Johannes Welbl, N. F. L., and Gardner, M. 2017. Crowdsourcing multiple choice science questions. In *Workshop on Noisy User-generated Text*.

Johansson, R., and Moschitti, A. 2010. Syntactic and semantic structure for opinion expression detection. In *CoNLL*.

Kumar, V.; Boorla, K.; Meena, Y.; Ramakrishnan, G.; and Li, Y. 2018. Automating reading comprehension by generating question and answer pairs. *CoRR* abs/1803.03664.

Lin, C.-Y. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 74–81. Barcelona, Spain: Association for Computational Linguistics.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, 311–318. Stroudsburg, PA, USA: Association for Computational Linguistics.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. Doha, Qatar: Association for Computational Linguistics.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR* abs/1606.05250.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. *CoRR* abs/1409.3215.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *CoRR* abs/1706.03762.

Wang, Z.; Lan, A. S.; Nie, W.; Waters, A. E.; Grimaldi, P. J.; and Baraniuk, R. G. 2018. Qg-net: A data-driven question generation model for educational content. In *Proceedings of the Fifth Annual ACM Conference on Learning at Scale*, L@S '18, 7:1–7:10. New York, NY, USA: ACM.

Wiebe, J.; Wilson, T.; and Cardie, C. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation* 39(2):165–210.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; Klingner, J.; Shah, A.; Johnson, M.; Liu, X.; Kaiser, L.; Gouws, S.; Kato, Y.; Kudo, T.; Kazawa, H.; Stevens, K.; Kurian, G.; Patil, N.; Wang, W.; Young, C.; Smith, J.; Riesa, J.; Rudnick, A.; Vinyals, O.; Corrado, G.; Hughes, M.; and Dean, J. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144.

Yu, S.; Indurthi, S. R.; Back, S.; and Lee, H. 2018. A multi-stage memory augmented neural network for machine reading comprehension. In *Proceedings of the Workshop on Machine Reading for Question Answering*, 21–30. Association for Computational Linguistics.

Yuan, X.; Wang, T.; Gülçehre, Ç.; Sordoni, A.; Bachman, P.; Subramanian, S.; Zhang, S.; and Trischler, A. 2017. Machine comprehension by text-to-text neural question generation. *CoRR* abs/1705.02012.