RobustQA: Adversarial Training with Hyperparameter Tuning

Stanford CS224N Default Project

Virat Singh
Department of Computer Science
Stanford University
virats@stanford.edu

Abstract

In this project, I used adversarial training and hyperparameter tuning to build a question answering system that can adapt to unseen domains with only a few training examples from the domain. From a high-level perspective, there are two model architectures: the baseline model provided by the starter code and my own adversarial model. To compare the performance of the two model architectures, I experiment with ADAM debiasing, various batch sizes, and weight decay tuning.

1 Introduction

One of the more challenging tasks in natural language processing is to build a question answering system that can generalize well to unseen domains and data. Recent advancements in the natural language processing have delivered neural question answering models that can outperform humans on datasets with discrete domains, like SQuAD[1], which is a dataset based on Wikipedia. However, for real-world applications, we need our question answering systems to generalize to numerous domains, and ideally, to hundreds if not thousands of domains. Studies over the past few years have demonstrated that state-of-the-art models oftentimes will overfit on the training data and struggle to generalize to broader datasets without additional training and finetuning [2].

To prevent question answering models from overfitting on their training data and domains, it is important that the models are able to learn domain invariant features. By learning domain invariant features, a question answering model should be able to perform better on new, unseen datasets because its features should not be overfit on domain-specific information. One way to learn domain invariant features is to use adversarial training [3]. The goal of domain adversarial training is to encourage the model to learn domain invariant features that avoid encoding domain-specific knowledge and information.

In this project, I built a question answering system that aims to generalize well on unseen out-of-domain data by using adversarial training [3] and hyperparameter finetuning. For my question answering system, I drew inspiration from Lee et al.[4] and re-implemented the adversarial QA system that they developed for their MRQA 2019 submission.

2 Related Work

Pre-trained Language Models

Over the past few years, the dominant, state-of-the-art model architectures in natural language processing have been built upon the transformer model that was introduced in 2017 by Vaswani et al[5]. The original transformer architecture that was introduced consisted of both an encoder and decoder model and was based solely on attention mechanisms. Building on top of the transformer

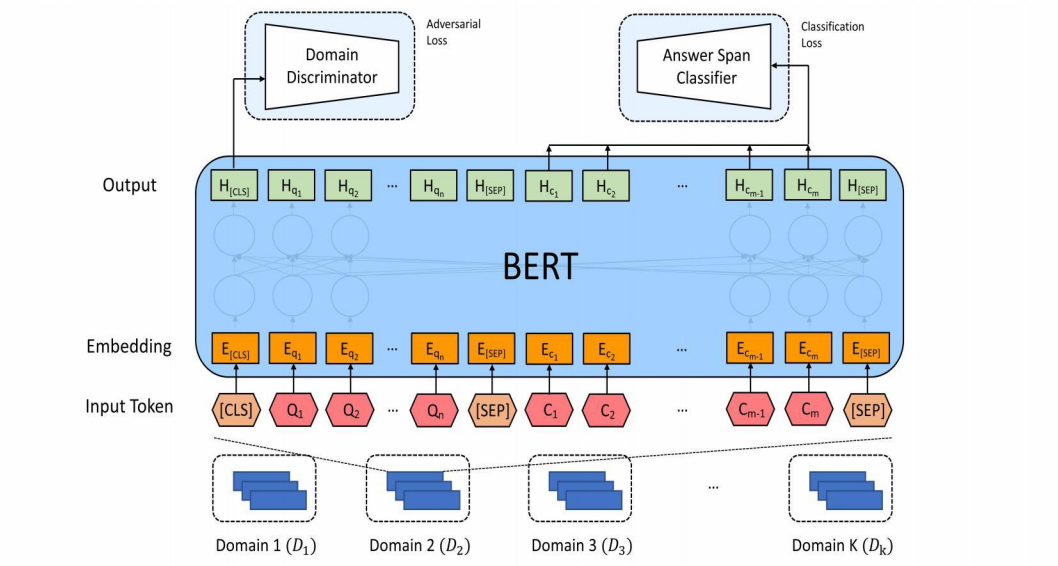

Figure 1: The overall training procedure for the question-answering system that uses adversarial training and is proposed by Lee et al.[4] for their MRQA 2019 submission. The diagram is humbly borrowed from their paper.

architecture, Devlin et al. introduced BERT[6], the Bidirectional Encoder Representations from Transformers in 2018. BERT proposed a novel way to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both the left and right context of a given input sample throughout all of the layers in the encoder stack [6]. BERT was designed to be be easily fine-tuned on downstream tasks - such as question answering - by adding one additional output layer.

Domain-agnostic Adversarial Training

In their MRQA 2019 submission, Lee et al.[4], designed a domain-agnostic adversarial training architecture for the question answering task, which is outlined in Figure 1. Their model consisted of a standard QA model, which was built on top of BERT[6], and a second discriminator model, which was a simple three-layer neural network classifier. During training, the standard QA model and discriminator model constantly compete, with the QA model trying to identify the start and end positions of an answer span and the discriminator model trying to identify the domain label of the QA model's question-answer pair. Their architecture showed better performance than the baseline model, which was a standard QA model only.

Hyperparameter Training and Finetuning

Despite the success of the original BERT[6] model in NLP tasks, the original authors omitted ADAM debiasing from their implementation, a decision which has shown to cause instability in BERT's downstream performance [7]. In their paper, Zhang et al.[7] observed that adding back the ADAM debiasing to BERT's optimizer improved the BERT model's stability and performance. Lee et al.'s[4] QA model reused the original BERT ADAM optimizer, which omits ADAM debiasing. In my experiments, I am including ADAM debiasing for all of my models as it has shown to increase model performance and stability.

3 Approach

3.1 Main approach

In this project, I experimented with two types of question answering systems: the baseline model and an adversarial model. For both models, I applied ADAM debiasing to increase stability and

	Dataset	Question (Q)	Context (C)	Train	Dev	Test
in-domain	SQuAD[1]	Crowdsourced	Wikipedia	50,000	10,507	-
	NewsQA [9]	Crowdsourced	News articles	50,000	4,212	-
	Natural Questions [10]	Search logs	Wikipedia	50,000	12,836	-
oo-domain	DuoRC [11]	Crowdsourced	Movie plots	127	126	1,248
	RACE [12]	Domain experts	Examinations	127	128	419
	RelationExtraction [13]	Synthetic	Wikipedia	127	128	2,693

Table 1: Statistics for datasets used for building the QA system for this project. Question and Context refer to data sources from which the questions and passages were obtained. Table borrowed from [14]

performance. Furthermore, I experimented with a batch sizes of 8, 16, and 32. Lastly, I experimented with removing weight decay on the bias and layer norm weight parameters. The code for my baseline and adversarial models is available at `https://github.com/virattt/robustqa`.

3.2 The baseline model

As is mentioned in the default final project handout, the baseline model fine-tuned DistilBERT[8], which is a smaller, distilled version of BERT[8]. Furthermore, the loss function used was the negative log-likelihood (cross-entropy) loss for the start and end positions of a given question's answer span from the dataset.

3.3 The adversarial model

For the adversarial model, I re-implemented the question answering system that Lee et al.[4] proposed in their paper for MRQA 2019 and modified it to fit our project's experimentation flow. The model is divided into two components, a standard question answering model and a discriminator model, which is a three-layer neural network classifier.

The QA model is trained the same way as the baseline model above: to predict the start and end positions of a given question's answer span within a context passage. The discriminator model was trained to predict the domain of the QA model's hidden representation state. The QA model's hidden representation was provided by the [CLS] token that the BERT[6] / DistilBERT[8] model produces at the end of every forward pass during training.

The goal of the QA model was to "fool" the discriminator model such that the latter was unable to predict the former's domain label. Once the discriminator model was unable to predict the QA model's domain label, then it was assumed that the QA model had learned domain-agnostic features [4].

4 Experiments

4.1 Data

For my experiments, I trained my QA system on three *in-domain* reading comprehension datasets (SQuAD[1], NewsQA[9], and Natural Questions[10]). There are also three *out-of-domain* datasets (DuoRC[11], RACE[12], and RelationExtraction[13]) that were used for evaluation. For the three *in-domain* datasets, there are separate `indomain_train` and `indomain_val` sets, which I used for training. Similarly, for the three *out-of-domain* datasets, there were separate `oodomain_train` and `oodomain_val` sets. For validation, I only used the `oodomain_val`. The `oodomain_train` set was unused. Additionally, the *out-of-domain* dataset had an `oodomain_test` set, which is a held-out test set that was used for evaluation. An overview of the datasets is provided above in Table 1.

4.2 Evaluation method

The evaluation metrics used were F1 score and Exact Match. F1 score is the harmonic mean between precision and recall; Exact Match is a stricter, binary measure that evaluates whether the QA system's answer output exactly matches the ground truth answer.

	Name	ADAM Debiasing	Batch Size	Weight Decay
Baseline	Model 1	✓	32	✗
	Model 2 *	✓	16	✗
	Model 3	✓	8	✗
	Model 4	✓	8	✓
Adversarial	Model 5	✓	32	✗
	Model 6	✓	16	✗
	Model 7	✓	8	✗
	Model 8	✓	8	✓

** Model 2 is the default baseline model that was provided by the starter code.*

Table 2: A breakdown of the eight models that were implemented and used for experimentation. ADAM debiasing was applied to all models and then various combinations of batch sizes and weight decay were used.

Baseline Model For the baseline model, the loss function is the sum of the negative log-likelihood for the start and end positions of the prediction output.

The loss equation $\mathcal{L}$ for negative log-likelihood was borrowed from Lee et al.'s paper [4] and shown in (3), where l is the domain category and $\mathbf{h} \in \mathbb{R}^d$ is the hidden representation of both the question and the passage. Note that the hidden representation $\mathbf{h}$ was captured in the `[CLS]` token representation that was produced by the final layer of the DistilBERT[8] model.

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\log P_\phi(l_i^{(k)}|\mathbf{h}_i^{(k)}) \tag{1}$$

Adversarial Model For the adversarial training model, I used two loss functions and summed them together to compute the total loss for the entire QA system: the sum of the negative log-likelihood for the start and end positions of the output (3) and the minimum of the Kullback-Leibler divergence as specified in Lee et al. [4]. The loss function for the adversarial model $\mathcal{L}_{QA}$ was also borrowed from Lee et al.'s paper and is defined in (2), where N is the total number of *in-domain* examples, $\mathbf{y}_{i,s}$ is the start index of the answer in the passage and $\mathbf{y}_{i,e}$ is the end index of the answer in the passage.

$$\mathcal{L}_{QA} = -\frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\big[\log P_\theta(\mathbf{y}_{i,s}^{(k)}|\mathbf{x}_i^{(k)}, \mathbf{q}_i^{(k)}) + \log P_\theta(\mathbf{y}_{i,e}^{(k)}|\mathbf{x}_i^{(k)}, \mathbf{q}_i^{(k)})\big] \tag{2}$$

Lastly, the loss of the discriminator model, which minimizes the Kullback-Leibler divergence is specified in (3) and provided by Lee et al [4]. In their adversarial QA system, Lee et al. alternated between optimizing the QA model and the the discriminator [4] and I did the same in my adversarial system.

$$\mathcal{L}_{dis} = \frac{1}{N}\sum_{k=1}^{K}\sum_{i=1}^{N_k}\mathcal{U}(l)P_\phi(l_i^{(k)}|\mathbf{h}_i^{(k)}) \tag{3}$$

Then, the total loss for the adversarial model is $\mathcal{L}_{QA} + \lambda\mathcal{L}_{dis}$ where λ is a hyperparameter that allows for controlling the importance of the discriminator loss [4].

4.3 Experimental details

I ran four experiments each on the baseline model and the adversarial model for a total of eight experiments. A full breakdown of the individual models that were used for experimentation is provided in Table 2.

	Name	$F1_{val}$	EM_{val}	$F1_{test}$	EM_{test}	ADAM Debiasing	Batch Size	Weight Decay
Baseline	Model 1	47.16	31.68	-	-	✓	32	✗
	Model 2*	**48.35**	**32.46**	-	-	✓	16	✗
	Model 3	46.28	30.89	-	-	✓	8	✗
	Model 4	46.82	30.89	-	-	✓	8	✓
Adversarial	Model 5	46.80	30.89	-	-	✓	32	✗
	Model 6	**49.24**	**33.51**	58.35	40.459	✓	16	✗
	Model 7	47.93	33.77	-	-	✓	8	✗
	Model 8	47.73	30.10	-	-	✓	8	✓

** Model 2 is the default baseline model that was provided by the starter code.*

Table 3: The evaluation results produced by the eight models after experimentation. The models were trained for 3 epochs with a learning rate of `3e-5` [4]. The **val** result were produced by evaluating the models on the *oodomain-val* dataset. The **test** result was produced by evaluating model 6 on the RobustQA test leaderboard.

From a high-level the four experiments consisted of:

1. Debiasing the ADAM optimizer
2. Varying the batch size between 32, 16 (default), and 8
3. Applying weight decay to the bias and layer norm parameters

All models were trained for three epochs each and a standard learning rate of `3e-5` was used, as proposed by Lee et al. [4] in their domain-agnostic question answering system. A full breakdown of the individual models that were used for experimentation is provided in Table 2.

ADAM Debiasing Omission The inclusion of ADAM debiasing was inspired by the work of Zhang et al. [7], in which they determined that omitting ADAM debiasing can decrease model stability and performance. By default, the baseline model's optimizer already included ADAM debiasing as it used HuggingFace's `AdamW` optimizer from the `transformers` library. That being said, the adversarial model, which was derived from Lee et al.'s [4] implementation, did not include ADAM debiasing and instead used `BertAdam` from the `pytorch-pretrained-bert` library. As a result, I applied ADAM debiasing on all of the adversarial model experiments, as well.

Batch Size Tuning It has been suggested in previous research that larger batch sizes may reduce the generalizability of a model [15][16]. The default batch size of the baseline model was 16 and the batch size that Lee et al. [4] used was 64. Even though Lee et al. used a batch size of 64 for their model, I decided to use a batch size of 16 for the "baseline" adversarial model that I built to match the baseline model that was provided by the starter code. Then, to determine the effect of batch sizes on generalizability, I experimented with reducing the batch size for both models to 8 and increasing the batch size to 32.

Weight Decay The default baseline model did not include any weight decay for the bias and layer norm parameters in the starter code. On the other hand, Lee et al.'s[4] adversarial model includes weight decay for the bias and layer norm parameters. As a result, I experimented with adding weight decay for the bias and layer norm parameters on both the default baseline model and the adversarial model.

4.4 Results

The overall results for the experiment are outlined in Table 3. As demonstrated, model X produced the best F1 score and model Y produced the best EM score. Overall, the adversarial models performed marginally better than the baseline models. Although the differences in F1 score is small across the board, there is an almost 10% delta in the EM scores of the best-performing adversarial model and the worst performing baseline model.
I expected the adversarial model to perform better than the baseline model, similar to how Lee et al.'s

[4] adversarial model performed significantly better than the baseline model that was provided for MRQA 2019. Even though my adversarial model performed better than the best baseline model, the marginal performance increase is disappointing as I expected the adversarial model to perform significantly better than the baseline model.

A few possible reasons for why my model only performed marginally better as opposed to significantly better (as Lee et al's [4] did) could be due to my model architecture being smaller (DistilBERT[8] vs. BERT[6]) and the number of training examples used being fewer. Furthermore, the MRQA training set included six out-of-domain datasets whereas our project only included three out-of-domain datasets. It could be the case that doubling the number of domains (from three to six) may have allowed Lee et al's [4] adversarial model to more effectively learn domain-invariant features.

Batch Size Tuning The results suggest that varying the batch size did not significantly help the models generalize more effectively. In fact, decreasing the batch size from 16 to 8 resulted in worse performance on both the baseline and adversarial models. Furthermore, increasing the batch size from 16 to 32 also resulted in worse performance; the drop in performance was more severe in the adversarial regime than in the baseline regime. Determining the optimal batch size for a given training regime is an inexact science and it could be that a batch size of 16 is the "sweet spot" for the pre-trained DistilBERT[8] architecture that I used in my experiments.

Weight Decay The results suggest that applying weight decay did not significantly help the model generalize more effectively. In fact, adding weight decay slightly reduced the performance of the adversarial model and slightly improved the performance of the baseline model. That being said, the deviation in performance was less than .20 F1 points for both the adversarial model and baseline model, so my conclusion is that the effect of weight decay on my models was negligible. A possible reason for this could be that weight decay may be less impactful on small model architectures like DistilBERT[8] and that we may only see the benefit of weight decay on larger model architectures like that of BERT[6].

5 Analysis

I inspected the outputs of my best-performing model, Model 6, an adversarial model. The model's predictions began to include some part or an exact match of the ground-truth answer as training steps increased, as expected. Instead of analyzing the exact match predictions, I am highlighting some of the more interesting and unique predictions.

5.1 Multiple similar answer options

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

Context: The Hudson River separates the Bronx on the west from Alpine , Tenafly and Englewood Cliffs in Bergen County , New Jersey ; the Harlem River separates it from the island of Manhattan to the southwest ; the East River separates it from Queens to the southeast ; and to the east , Long Island Sound separates it from Nassau County in western Long Island . Directly north of the Bronx are (from west to east) the adjoining Westchester County communities of Yonkers , Mount Vernon , Pelham Manor and New Rochelle . (There is also a short southern land boundary with Marble Hill in the Borough of Manhattan , over the filled - in former course of the Spuyten Duyvil Creek . Marble Hill 's postal ZIP code , telephonic area codes and fire service , however , are shared with the Bronx and not Manhattan .)

Answer: Harlem River

Prediction: Hudson River

Analysis: In the context, there are multiple geographical areas that are similar to the ground-truth answer: "East River", "Hudson River", and "river". One could argue that "Spuyten Duyvil Creek" is similar as well. Although the model predicted the incorrect answer - Hudson River - it still managed to generate a prediction relatively close to the ground-truth answer - Harlem River. I suspect that this may be due to the fact that the model was trained on word-level embeddings instead of character-level embeddings.

5.2 More detailed predictions

Question: When did beryl markham fly across the atlantic?

Context: When Markham decided to take on the Atlantic crossing , no female pilot had yet flown non-stop from Europe to New York , and no woman had made the westward flight solo , though several had died trying . Markham hoped to claim both records . On 4 September 1936 , she took off from Abingdon , England . After a 20 - hour flight , her Vega Gull , The Messenger , suffered fuel starvation due to icing of the fuel tank vents , and she crash - landed at Baleine Cove on Cape Breton Island , Nova Scotia , Canada . She became the first person to make it from England to North America non-stop from east to west . She was celebrated as an aviation pioneer .

Answer: September 1936

Prediction: 4 September 1936

Analysis: In this example, the model produced a more detailed prediction of "4 September 1936" instead of "September 1936". It may be that the model has learned some date-related features (day, month, year) and included the "4" as part of its prediction.

5.3 Cryptic context

Question: Who did the minnesota vikings lose to in the super bowl?

Context: BTableB BTrB BThB Game EEThE BThB Date EEThE BThB Winning team EEThE BThB Score EEThE BThB Losing team EEThE BThB Venue EEThE BThB City EEThE BThB Attendance EEThE BThB Ref EEThE EETrE BTrB BTdB 01 ! I EETdE BTdB 0000 January 11 , 1970 EETdE BTdB Kansas City Chiefs 02 ! Kansas City Chiefs (2 , 1 – 1) EETdE BTdB 2307 ! 23 – 7 EETdE BTdB Minnesota Vikings 01 ! Minnesota Vikings (1 , 0 – 1)

...truncated text...

Miami Dolphins 03 ! Miami Dolphins (3 , 2 – 1) EETdE BTdB 2407 ! 24 – 7 EETdE BTdB Minnesota Vikings 02 ! Minnesota Vikings (2 , 0 – 2) EETdE BTdB Rice Stadium 01 ! Rice Stadium EETdE BTdB Houston , Texas 01 ! Houston , Texas EETdE BTdB 071882 ! 71,882 EETdE BTdB EETdE EETrE BTrB BTdB 09 ! IX EETdE

Answer: Kansas City Chiefs

Prediction: Miami Dolphins

Analysis: In this example, we have a really complex, non-human-readable context that I am guessing is a table of scores for past football games. In this case, the model seems to have picked a random team - "Miami Dolphins" out of the various team names. This example was pulled from one of the final training steps, which means that the model had almost been trained for a full 3 epochs by this point. This may have been one of the few contexts that the model has seen that is not a coherent paragraph and as a result, it failed miserably.

5.4 Questionable ground-truth answer

Question: What is another general name for a religious teacher?

Context: Religious and spiritual teachers, such as gurus, mullahs, rabbis, pastors/youth pastors and lamas, may teach religious texts such as the Quran, Torah or Bible.

Answer: spiritual

Prediction: lamas

Analysis: In this example, the ground-truth answer is questionable, as it is not immediately obvious that "spiritual" is a more general name for a religious teacher than "lamas" (or its singular variant).

6 Conclusion

In this project, I successfully re-implemented the adversarial model that Lee et al. [4] designed for their MRQA 2019 submission. Furthermore, I made slight modifications to their approach by adapting their code to our project's guidelines and experimenting with hyperparameter tuning. Although the adversarial models performed better than the baseline models, the difference in performance was not as great as I would have expected. Nevertheless, I consider the marginal performance improvement that my adversarial model demonstrated to be a satisfactory achievement. Given the modest number of training domains that I used for my models (three) and that I did not use the *oodomain-train* dataset at all for training my adversarial model, the smaller than expected performance gains make sense.

Future work may include more domains for training so that the adversarial model can learn more domain-invariant features. Furthermore, one may also include data augmentation and rebalancing techniques to incorporate training examples from out-of-domain corpora. Lastly, future work should also include enhancements to the underlying adversarial model itself. The discriminator model that the adversarial model uses is a simple three-layer feed-forward neural network. One could make creative hyperparameter tuning adjustments to the discriminator model or even reuse a pre-trained network and perform fine-tuning.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[2] Dani Yogatama, Cyprien de Masson d'Autume, Jerome Connor, Tomas Kocisky, Mike Chrzanowski, Lingpeng Kong, Angeliki Lazaridou, Wang Ling, Lei Yu, Chris Dyer, and Phil Blunsom. Learning and evaluating general linguistic intelligence, 2019.

[3] Motoki Sato, Hitoshi Manabe, Hiroshi Noji, and Yuji Matsumoto. Adversarial training for cross-domain Universal Dependency parsing. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 71–79, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[4] Seanie Lee, Donggyu Kim, and Jangwon Park. Domain-agnostic question-answering with adversarial training. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, 2019.

[5] Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan N. Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. Tensor2tensor for neural machine translation. *CoRR*, abs/1803.07416, 2018.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav. Artzi. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*, 2019.

[8] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962v2*, 2019.

[9] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 2017.

[10] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *TACL*, 2019.

[11] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. DuoRC: Towards Complex Language Understanding with Paraphrased Reading Comprehension. In *ACL*, 2018.

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

[13] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *CoNLL*, 2017.

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

[15] Jorge Nocedal Mikhail Smelyanskiy Nitish Shirish Keskar, Dheevatsa Mudigere and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[16] Ibrahem Kandel and Mauro Castelli. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, 6, 05 2020.