# Mixture of Experts and Back-Translation to improve QA robustness

Stanford CS224N Default Project - Robust QA Track

**Vy Thai**
Department of Computer Science
Stanford University
vythai@stanford.edu

**Gokul Dharan**
Department of Computer Science
Stanford University
gdharan@stanford.edu

## Abstract

This work improves the generalization of a DistilBERT-based Question Answering (QA) model with the addition of a Mixture of Experts (MoE) layer as well as through data augmentation via back-translation. QA models generally struggle to perform in contexts that differ from those present in the model's training data. As a step towards addressing this limitation, our MoE implementation effectively learns domain-invariant features without explicitly training each expert on individual subdomains. We also apply top-k sampling back-translation and introduce a new technique to more effectively retrieve the answer span from the back-translated context. We find that the addition of the MoE layer yields an improvement of 3.19 in F1 score on an out-of-domain validation set, with back-translation granting a further 1.75 in F1 score. This represents a net improvement of 10.1% over the DistilBERT baseline.

## 1 Introduction

Question-Answering (QA) is a natural language comprehension task in which a model must find the answer to an input question within a given context. Although QA models have shown strong performance when evaluated on domains that the training data is sourced from, they struggle to generalize to other domains. This significantly hinders the applicability of these models in real-world contexts where QA domains are unlikely to be identically distributed when compared to the model's training data. Our approach is strongly motivated by the results of the MRQA 2019 shared task [1], which was a competition that evaluated submitted models' QA robustness. Among the entries, the MoE approach in [2] presented a significant generalization boost to a BERT-based model when combined with multi-task learning. We evaluate a similar approach on DistilBERT without multi-task learning, as it requires additional data that may not necessarily be available when training QA models. We also re-initialize the last DistilBERT layer to improve fine-tuning stability as suggested in [3]. Finally, we add back-translation (BT) as a means of boosting the variance of the training data to improve model generalization. We find that MoE and BT present a significant improvement over a naive DistilBERT baseline.

## 2 Related Work

**DistilBERT** DisilBERT [4] is a smaller version of BERT that seeks to closely replicate the performance of the full-sized model. Using DistilBERT allows us to evaluate various strategies quickly with a limited amount of compute.

**Mixture of Experts** The results of the MRQA 2019 shared task [1] revealed that BERT-based structures may be fundamentally limited in their generalization performance, with the top-performing

submissions using XLNet. The top-performing BERT-based entry was CLER [2], which uses a full-sized BERT model along with a Mixture of Experts layer as well as multi-task learning and an ensemble. Motivated by this work, we seek to evaluate the generalization capability gained by Mixture of Experts on a DistilBERT transformer architecture in a more limited setting where we do not assume the availability of the requisite data for multi-task learning. We also evaluate the effectiveness of an additional loss term to penalize convergence of expert networks. CLER also does not include data augmentation via back-translation.

**Back-translation** [5] shows that back-translation provides significant improvements in reading comprehension on the competitive SQuAD1.1 benchmark. However, [6] shows that back-translation did not help in-domain or out-of-domain performance in question-answering on various datasets in the MRQA 2019 Shared Task. The authors suggest this may be due to the pre-trained model (XLNet) already capturing the language variation exposed by back-translation. They also questioned if the improvement shown on SQuAD1.1 by [5] would still apply for other datasets. Also, [7] investigated different methods of generating the synthetic sentences and found that back-translation using sampling and noisy beam search is more effective than greedy search in most settings. Our work evaluates the effectiveness of back-translation on a BERT-based model on datasets other than SQuAD1.1.

## 3 Approach

Our baseline is a DistilBERT-based QA model (DistilBERTQA) from Huggingface [4] [8]. In addition to MoE and back-translation (BT), we also aim to leverage learnings from [3] which suggest that re-initializing some BERT layers can boost fine-tuning performance.

### 3.1 Mixture of Experts (MoE)

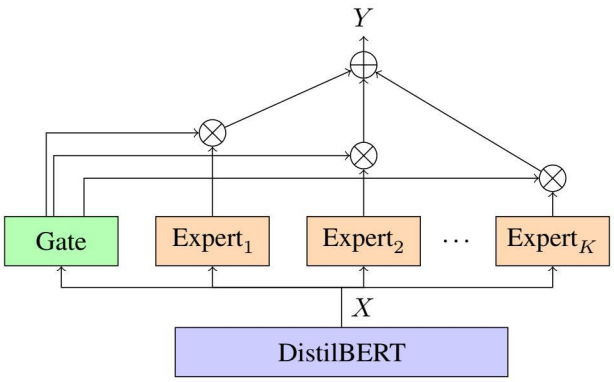


Figure 1: Model Architecture

We implement our own version of the DistilBERTforQuestionAnswering baseline based on its source code [8]. We then add an MoE layer before the output linear layer which yields the predicted logits, as shown in Figure 1. This is done with a series of $K$ one-layer MLPs that act as "experts" $E$ and a single linear layer with softmax output that acts as the gating network $G$. Both the gate and the experts take the last hidden layer of the DistilBERT model $X$ as input, and the output of the MoE layer $Y$ is the same size as $X$ and is defined as in [2]:

$$Y = \sum_{i=1}^{K} G(X)_i E_i(X).$$

In addition to the task-specific loss, [2] adds

$$\mathcal{L}_{importance} = \lambda_{\text{importance}} CV(\sum_{b \in B} G(b))$$

where $B$ is the minibatch and $CV$ is the coefficient of variation. When correctly weighted by the $\lambda_{importance}$ hyperparameter, this encourages a uniform distribution of weights across experts for a given batch. This ensures that no single expert is consistently assigned high weight, in which case the MoE layer reduces to a single MLP.

One possible approach for training an MoE model is to train each expert network on a particular domain, and then train the gating network to take a weighted sum of expert outputs that hopefully generalizes across domains. However, this technique has the natural limitation that each expert specializes on one of the in-domain datasets by construction, which may not necessarily be the best way to learn features in the DistilBERT output that are conducive to domain-invariance. Therefore, we adopt a more unsupervised approach to specializing experts in the same style as in [2], by training all experts and gates simultaneously on all of the training data. The intended goal is that the random initialization of expert and gate weights naturally lead to the specialization of experts on features that are more useful for robust, domain-invariant classification than a weighted sum across domain-specific experts.

Although this approach has shown promising results in [2], we seek to further improve performance by explicitly encouraging the experts to diverge from one another. One potential weakness of the method outlined above is that experts may converge to the same weights, meaning that multiple experts in the model specialize on the same features of the DistilBERT output. This hinders the expressiveness of the model, and in the worst case, will ensure the model is no more expressive than a model that simply has one expert network. To encourage each expert to specialize on different features of the DistilBERT output space, we add what we call an "expert convergence" loss based on the inverse-variance of the expert weights. To do so, we enumerate the $N$ weights in each of the expert networks. Then, let $w_j \in \mathcal{R}^K$ denote a vector of the $j^{\text{th}}$ weight in each of the $K$ experts. We then penalize the inverse coefficient of variation of each of these vectors to maximize variance in weight values for each element

$$\mathcal{L}_{\text{expert convergence}} = \frac{\lambda_{\text{expert convergence}}}{\frac{1}{N}\sum_{j=1}^{N} CV(w_j)}.$$

We then define our loss as follows:

$$\mathcal{L} = \mathcal{L}_{QA} + \mathcal{L}_{importance} + \mathcal{L}_{\text{expert convergence}},$$

where $\mathcal{L}_{QA}$ is the cross-entropy loss of the answer logits as implemented in [8]. Note that in Section 5.2, we find that the latter loss is ineffective, so our final model only uses the QA and importance losses.

### 3.2 Back-translation for Data Augmentation (BT)

For our model, we build on the implementation presented in [6] with some major modifications. Specifically, we experiment with query, answer, and context paraphrases generated by multiple translation models such as HelsinkiNLP [9], Fairseq [10] as well as the paid-service Google API [11]. For the Fairseq model, besides greedy search, we also try two other alternatives motivated by findings in [7]: beam search with added noise and top-k sampling. For the first alternative, we set $k = 5$ for beam search and use noising techniques on the input sentences as mentioned in [12] with some small modifications. Each sentence to be translated has a $50\%$ chance of being word-swapped and a $50\%$ chance of being word-deleted, with these two events being independent. Note that we only perform deletion after swapping. We use the `nlpaug` package to swap adjacent words randomly and delete random words on $10\%$ of words in a sentence. For the second alternative, top-k sampling, we set $k = 50$ and experiment with different temperature values to change the variance of back-translated paraphrases. For generating context paraphrases, as suggested in [6], we use the simple `Sentencizer` pipeline in SpaCy [1] to perform sentence segmentation and then translate each sentence independently.

A unique feature of our implementation as compared to [5] and [6] is our algorithm for retrieving the translated context's answer span. When we segment the context sentences, we identify and tag

[1] `https://spacy.io/api/sentencizer`

| | Original and translated sentence | Original and retrieved translated answer |
|---|---|---|
| 1 | In late 2011, she took the stage at **New York's Roseland Ballroom**... | New York's Roseland Ballroom |
| | At the end of 2011, she took the stage at the **Roseland Ballroom in New York**... | Roseland Ballroom in New York |
| 2 | ...described Beyoncé as the **greatest entertainer alive** | greatest entertainer alive |
| | ...described Beyoncé as **the greatest living artist** | the greatest living artist |
| 3 | ...hospitalized, but her injuries are **not life-threatening**... | not life-threatening |
| | ...hospitalized, but her injuries do **not endanger her life**... | not endanger her life |

Table 1: Example of back-translation using new method of retrieving answer span

the source sentence that contains the answer span. We can then assume that our back-translated answer span should be found in the back-translation of the tagged sentence. We first try to retrieve new answer using exact string matching, as described in [6]. When that fails, [6] used heuristic techniques to find the new start and end tokens in the translated sentence based on the start and end tokens of the original answer, using character-level 2-grams. However, we find many cases where this algorithm fails to find the correct answer span, such as when the word order is changed in the paraphrase. For example, in Table 1, the original answer span is, "New York's Roseland Ballroom," while in the translated sentence, this answer span has changed to, "Roseland Ballroom in New York." To successfully identify these answer spans, our approach finds the span of words in the translated sentence that yields the highest BLEU-score on word and character levels (1,2,3,4 n-gram) compared to the original answer, given a certain minimum threshold. To take into account where the new answer span might have a different length, we find all the answer span candidates ranging from $(length_{original} - 3)$ to $(length_{original} + 3)$, where $length_{original}$ is the number of words in original answer. Another common failure mode for answer matching occurs when the back-translation process may have found synonyms that do not necessarily match the original answer span in terms of BLEU score. To reduce the frequency of this failure mode, we compute this BLEU score-based matching on both the back-translated and original answer, as the back-translated answer will often have the same choice of words as in the back-translated context. In summary, given an example with original answer $o$ and translated answer $t$, the accumulated score for each candidate span of words $s$ is given below:

$$\begin{aligned} score(s) = w \cdot (3 \cdot BLEU_{word}(o, s) + BLEU_{word}(t, s)) \\ + (1 - w) \cdot (3 \cdot BLEU_{char}(o, s) + BLEU_{char}(t, s)) \end{aligned} \tag{1}$$

where:

$$w = \begin{cases} 0.3, & \text{if } length_{original} == 1, \\ 0.7, & \text{if } length_{original} > 1 \end{cases} \tag{2}$$

This implementation prioritizes the original answer in defining the overall score, as it is often more reliable in identifying the answer span in the back-translated context. We also use the weight $w$ to prioritize character-level BLEU scores when original answer has only one word and word-level BLEU scores when the original answer has multiple words. For the RACE dataset, since answers can be longer than one sentence, we use a similar technique to find the position for the last consecutive 5-word span of the original answer across the new translated sentences after identifying the start position using the above method. We also further increase the diversity of the dataset by choosing the translated query with probability $0.9$ for any augmented example.

### 3.3 Ensemble

We know that BERT models are prone to fine-tuning instability, so we seek to boost the model's test set performance by summing the start and end logit distributions from multiple models. Specifically, given $n$ models trained on distinct random seeds, where the $i^{\text{th}}$ model outputs start and end logits $s^i$ and $e^i$ respectively, our test-time logit outputs are given by

$$s^{out} = \sum_{i=1}^{n} s^i, e^{out} = \sum_{i=1}^{n} e^i$$

# 4 Experiments

## 4.1 Datasets and Setup

**Data** We use three in-domain reading comprehension datasets - 50,000 examples each from Natural Questions [13], NewsQA [14], and SQuAD [15] - for fine-tuning our pre-trained distilbert-base-uncased model. The model will then be further fine-tuned on the out-of-domain training dataset consisting of 127 examples from each of the RelationExtraction [16], DuoRC [17], and RACE[18] datasets. The validation set also consists of these three out-of-domain (ood) datasets. The data is processed in chunks of size 384 with a stride of 128.

**Method and Details** We evaluate the model on the out-of-domain validation set via EM and F1 scores based on the overlap between output answers and gold answers. We do not evaluate in-domain performance since the focus of this project is on building a model that can succeed in domains with little training data. All of the models discussed in this section include a pre-trained DistilBERT [4] model (distilbert-base-uncased) that is then fine-tuned on the in-domain training set for 3 epochs at a learning rate of $3 \times 10^{-5}$. Then, a second-stage fine-tuning is performed on the smaller out-of-domain training set for 30 epochs at the same learning rate, with the checkpoint that performs best on the validation set being saved to avoid overfitting. Our best MoE model has 12 experts, each with a hidden size of 1024, $\lambda_{\text{importance}} = 0.01$, and $\lambda_{\text{expert convergence}} = 0$. The justification for this configuration is given in Section 5.2. Back-translation experiments are then conducted on this MoE model.

# 5 Results & Analysis

## 5.1 Summary

| Model | F1 | EM |
|---|---|---|
| DistilBERTQA | 48.89 | 34.03 |
| MoE | 52.08 | **37.17** |
| MoE+BT | **53.83** | **37.17** |

Table 2: Out-of-domain validation scores

We find that our MoE+BT model significantly boosts out-of-domain performance, with an F1 improvement of +4.94 over baseline. Of this, the MoE architecture is responsible for a gain of +3.19, with data augmentation via back-translation yielding a further gain in F1 of +1.75. Figure 2 shows the validation scores for the second-stage fine-tuning process on out-of-domain data, which reveals the high noise that occurs as a natural consequence of such a small validation set of approximately 128 examples per dataset. Regardless, the MoE and MoE+BT models clearly outperform the baseline. We also note that the baseline does not substantially improve in validation performance throughout the second stage of fine-tuning.

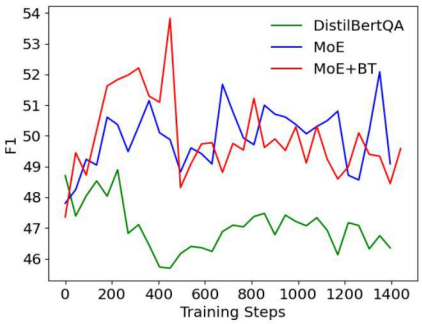


Figure 2: Validation F1 for fine-tuning on out-of-domain data

### 5.2 Model Architecture

| MoE Variant | F1/EM | F1/EM | F1/EM |
|---|---|---|---|
| Number of experts $\{8, \mathbf{12}, 16\}$ | 49.18/34.03 | **52.08/37.17** | 49.32/34.82 |
| Expert hidden size $\{512, \mathbf{1024}, 2048\}$ | 51.02/36.13 | **52.08/37.17** | 48.35/34.82 |
| $\lambda_{\text{importance}}\{0.001, \mathbf{0.01}, 0.1\}$ | 49.02/34.82 | **52.08/37.17** | 48.04/33.51 |
| $\lambda_{\text{expert convergence}}\{\mathbf{0}, 1, 10\}$ | **52.08/37.17** | 51.13/35.34 | 50.88/34.29 |

Table 3: Ablation across MoE components on fixed seed (ood validation performance)

Without back-translation, we conduct a fairly extensive study of the performance of various MoE configurations in Table 3. We note the large degree of variance as each of the parameters are changed, sometimes even causing the MoE model to perform worse than the DistilBERTQA baseline. This reflects both the limitation of being able to evaluate models with only around 128 examples from each dataset and the inherent and well-known fine-tuning instability of BERT-based models [3]. Ideally, we would replicate these experiments across many random seeds to build confidence in the results, as done in [3], but that would require an infeasible amount of compute for the scope of this project. Therefore, we proceeded with the best MoE variant found on a fixed seed, as highlighted in Table 3.

One key takeaway from these results is that the naive expert convergence penalty is ineffective. This is due to the fundamental drawback of our implementation: it relies on enumerating weights for the expert network and only penalizes convergence for weights that correspond to the same index in this enumeration. Therefore, any permutation of expert hidden nodes and corresponding weights will yield a network that may not be penalized by our implementation even though the output of the expert will be identical to any other permutation of this network. Essentially, this loss penalty can only prevent exact duplicates of a given expert down to the order of the weights and hidden nodes, so any performance gain in encouraging expert variance is offset by the added noise in the training loss, as shown in Table 3. As a result, our chosen variant has $\lambda_{\text{expert convergence}} = 0$, nullifying the expert convergence loss.

### 5.3 Back-translation Results

| Translation Model | Greedy search (de) | Beam+noise (k=5, ru) | top-k (k=50, temp=1.7, ru) |
|---|---|---|---|
| Google API | 50.86/36.13 | – | – |
| HelsinkiNLP | 50.21/34.29 | 51.62/35.86 | 50.66/34.82 |
| Fairseq | 50.31/34.03 | 51.64/35.34 | **53.83/37.37** |

Table 4: MoE+BT ood performance with various BT techniques. GoogleAPI does not have the option to use beam search or top-k sampling.

The experiments shown in Table 4 indicate that most back-translation techniques do not improve model performance over the MoE model. Indeed, every BT configuration except for top-k Fairseq shows a decrease in both F1 and EM compared to MoE. This suggests that the MoE architecture may cover many of the same benefits that are found by applying back-translation. Among the sampling methods, greedy search was the least effective and consistently decreased validation performance compared to MoE. In greedy search configurations, we also noticed that the model overfits very early in training, suggesting that the deterministic output of this method may lead to some false patterns in the training data that help our model predict the correct output. Since greedy search also overlooks future word choices, the translation quality is generally lower too, adding spurious noise to the dataset that harms performance.

Among the three translation models, Google API gives the highest performance using greedy search. Between HelsinkiNLP and Fairseq, Fairseq shows a significant increase in the top-k sampling configuration. To further analyze these results, we also calculated the average normalized Levenshtein distance using the `python-Levenshtein` package [2] in Pypi between each original and translated

[2] `https://pypi.org/project/python-Levenshtein/`

sentence (Table 5). A lower Levenshtein distance indicates higher similarity between the source and translated sentences, which in turn indicates a less diverse back-translated dataset. The results suggest that Google API does a good job in conserving the semantic information in sentences, which limits diversity of the back-translated paraphrases. On the other hand, the best BT method - Fairseq using top-k search - has the second-highest Levenshtein distance. Interestingly, although HelsinkiNLP model using top-k sampling also gets a similar Levenshtein distance to Fairseq, it was much less effective in boosting performance, which indicates that there are factors not encapsulated by Levenshtein distance that affect back-translation quality.

| Model | Avg Levenshtein distance |
|---|---|
| GoogleAPI (de) | 0.15 |
| Fairseq (greedy, de) | 0.23 |
| Fairseq (beam+noise, ru) | 0.30 |
| Fairseq (top-k, ru) | 0.34 |
| HelsinkiNLP(top-k, ru) | 0.35 |

Table 5: Average Levenshtein distance for different techniques

### 5.4 Finetuning Strategies

We evaluated several alternate strategies for training our model, all of which were suboptimal in comparison to our chosen strategy outlined in Section 4.1. Re-initializing the last few layers in BERT models can often help boost finetuning performance [3]. However, it seems likely that the much smaller DistilBERT model cannot easily recover the performance of the pretrained model when one of its layers is re-initialized, as we see a decrease in out-of-domain performance with this strategy (-0.93 F1). We also attempted to mix out-of-domain and in-domain data during training (-2.76 F1), as well as back-translating in-domain data as well (-2.04 F1).

### 5.5 Per-Domain Analysis

| Model | DuoRC (F1/EM) | RACE (F1/EM) | RelationExtraction (F1/EM) |
|---|---|---|---|
| DistilBERTQA | 35.59/25.40 | 37.00/**22.66** | 73.89/53.91 |
| MoE | **45.18/33.33** | 35.37/20.31 | 75.60/57.81 |
| MoE+BT | 43.49/31.75 | **37.78**/21.09 | **80.05/58.59** |

Table 6: Per-domain ood validation performance

The results in Table 6 allow us to make some qualitative insights into the strengths and weaknesses of this model. Interestingly, each of the two principal components of our model, MoE and BT, seem to boost performance significantly on DuoRC and RelationExtraction respectively, rather than across all domains.

MoE shows a major gain in performance on the DuoRC dataset. DuoRC is the only dataset to have bimodal data in the sense that its movie plot synopses contexts are sourced from both Wikipedia and IMDb [17]. Therefore, the MoE model may be able to have some experts specialize on one subdomain, Wikipedia, and others on IMDb. The authors of [17] do find that there is a significant difference between the two subdomains, with only 26% overlap in the bag-of-words for plots for the same movie sourced from the two sites. Additionally, they also note that IMDb plots tend to be longer and more descriptive. This lends credence to the idea that MoE can distinguish and meaningfully specialize on each of these domains, and is one possible reason for the large gain over the baseline performance. If the dataset indicated the source (IMDb or Wikipedia) for each context, we would be able to directly evaluate the veracity of this claim by examining MoE Gate network activations for each source.

Performance on RACE [18] did not increase significantly with any of the model variants, and interestingly, it is the dataset that requires the most complex reasoning as it includes abstract questions like,

"what's the best title for this passage?" We find that neither back-translation nor MoE significantly improve performance, and we suspect that this is due to a fundamental ceiling for BERT-based model performance on this dataset with so few training examples. In [1], a full-sized BERT-based QA model fine-tuned on 349 examples achieved an F1 of 45.3. Since DistilBERT is smaller and generally not quite as powerful as BERT, and since we only fine-tune on 127 examples, perhaps we are limited by the transformer architecture, which would explain why MoE did not boost performance on this dataset.

We hypothesize that the quality of back-translation in terms of boosting model performance is inversely related to the average context length. DuoRC has by far the longest context lengths, consisting of large paragraphs, and that is the only dataset where we saw a degradation in performance with back-translation compared to MoE. RACE consists of smaller contexts that are a few sentences long, and we see a moderate gain for MoE+BT over MoE. RelationExtraction is sourced from Wikipedia [16] and consists of very simple one-sentence contexts, like, "A similar fate met Jan Karski's older brother Colonel Marian Kozielewski," with corresponding question, "What is Jan Karski's brothers name?" RelationExtraction also sees the largest gain for MoE+BT performance. As such, we suspect that with longer contexts, the back-translation pipeline is more likely to output a critical error in the translation process that renders the context, question, or answer invalid. Furthermore, the movie synopses in DuoRC use significantly more complex phrasing than the simpler contexts of RACE, which are designed for English exams for Chinese students. This is another factor that increases the error rate of back-translation on DuoRC more than any other dataset.

## 6 Test Submission

Our best MoE+BT model achieved an F1 of 59.403 and EM of 41.537 on the held-out test set. Then, we used all of the available out-of-domain data, including the development set, and modified the second-stage fine-tuning to run for three epochs independent of validation score. This aggregated training data was then duplicated according to dataset so that the training data was proportioned in the same ratio as the test set, with an emphasis on RelationExtraction. Our final output was ensembled across three random seeds trained with this setup, and this achieved an F1 of 60.157 and an EM of 43.050.

## 7 Conclusion

We find that Mixture of Experts and back-translation are effective techniques for boosting QA domain-invariance on a DistilBERT-based model. Specifically, back-translation using top-k sampling from the Fairseq translation model is the only BT variant that presents a significant improvement on the performance over MoE. At test time, we find that applying an ensemble across random seeds further boosts generalization performance. We also find that our loss penalty to encourage experts to diverge was ineffective. Future work could involve designing a more robust means of encouraging expert divergence and transferring our MoE+BT method to a more powerful QA architecture like XLNet.

## References


[1] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. *CoRR*, abs/1910.09753, 2019.

[2] Tomoki Taniguchi Takumi Takahashi, Motoki Taniguchi and Tomoko Ohkuma. Cler: Cross-task learning with expert representation to generalize reading and understanding. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, 2019.

[3] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. Revisiting few-sample bert fine-tuning, 2020.

[4] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019.

[5] Minh-Thang Luong Rui Zhao Kai Chen Mohammad Norouzi Adams Wei Yu, David Dohan and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. 2018.

[6] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. An exploration of data augmentation and sampling techniques for domain-agnostic question answering. *CoRR*, abs/1912.02145, 2019.

[7] Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. Understanding back-translation at scale. 2018.

[8] Huggingface. Source code for transformers.models.distilbert.modeling_distilbert.

[9] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.

[10] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[11] Cloud translation | google cloud.

[12] Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. 2018.

[13] 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. *Transactions of the Association of Computational Linguistics*, 2019.

[14] 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*, pages 191–200, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[15] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

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

[17] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. *CoRR*, abs/1804.07927, 2018.

[18] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.