Are Multilingual BERT models robust? A Case Study on Adversarial Attacks for Multilingual Question Answering

Sara Rosenthal, Mihaela Bornea, Avirup Sil
IBM Research AI
Thomas J. Watson Research Center, Yorktown Heights, NY 10598
{sjrosenthal, mabornea, avi}@us.ibm.com

Abstract

Recent approaches have exploited weaknesses in monolingual question answering (QA) models by adding adversarial statements to the passage. These attacks caused a reduction in state-of-the-art performance by almost 50%. In this paper, we are the first to explore and successfully attack a multilingual QA (MLQA) system pre-trained on multilingual BERT using several attack strategies for the adversarial statement reducing performance by as much as 85%. We show that the model gives priority to English and the language of the question regardless of the other languages in the QA pair. Further, we also show that adding our attack strategies during training helps alleviate the attacks.

1 Introduction

Most recent advances in question answering (QA) have focused on achieving state-of-the-art results on English datasets (Rajpurkar et al., 2016, 2018; Yang et al., 2018; Kwiatkowski et al., 2019). In this work, we focus on multilingual QA using the MLQA dataset (Lewis et al., 2020) which contains parallel examples in seven languages. The MLQA dataset only contains a dev and test set, thus they apply a zero-shot (ZS) approach by training using SQuAD v1.1 (Rajpurkar et al., 2016), which is in English, and achieve impressive results.

However, (Jia and Liang, 2017; Wang and Bansal, 2018) showed that SQuAD models (Hu et al., 2018) are fragile when presented with adversarially generated data. They proposed AddSentDiverse, which produces a semantically meaningless sentence containing a fake answer that looks like the question grammatically, and adds it to the context. Most recently (Maharana and Bansal, 2020) has extended the attacks shown in (Wang and Bansal, 2018) and has shown a comprehensive analysis focused on English QA, but attacks on multilingual QA has still not been explored.

We ask the following research questions:

(1) Are multilingual QA models trained with mBERT robust? Our main focus is exploring adversarial attacks in a multilingual setting. Therefore, we specifically target MBERT. We show that all of our approaches successfully attack the MBERT model in all the MLQA languages. Further, our adversarial attacks have a stronger impact on MBERT compared to BERT.

(2) Does MBERT give preference to certain languages? Although the expectation is that a multilingual language model should treat all languages the same, we hypothesize that the model may prefer to find an answer in certain languages. This could be based on the size of the data used to build the embeddings in MBERT, the language of the training data used to fine-tune MBERT, or the language of the question in the example. We empirically show the effect of our attacks for all language combinations in MLQA to address this. Our experiments show that when the adversarial statement is in the language of the question the attack has the largest impact. In addition, MBERT gives priority to English regardless of the language the question is asked in. We find similar trends when augmenting our MBERT model with translated data during training though the MT data does alleviate the priority given to English in some cases.

(3) Can training with adversarial data alleviate some of the susceptibility to attacks? After we have shown that we can successfully attack the MLQA system we turn to addressing the flaws in the model by teaching the model to overcome the attacks. We show that including adversarial statements during training helps alleviate the attacks, especially when using data augmentation strategies to translate our adversarial MLQA data into other languages.

We explore these questions by showing that adversarial attacks affect the MLQA dataset by
2 Related Work

There are several prior strategies (Jia and Liang, 2017; Wang and Bansal, 2018; Wallace et al., 2019; Maharana and Bansal, 2020) that introduce adversarial sentences to distract RC systems (Hu et al., 2018; Shao et al., 2018) reducing SOTA performance by almost 50%. (Sen and Saffari, 2020) don’t add adversarial statements but analyze how BERT QA models handle missing or incorrect data and question variations. However, all this work focuses on English attacks. Further, although many multilingual QA datasets exist (He et al., 2017; Asai et al., 2018; Mozannar et al., 2019; Artetxe et al., 2020; Lewis et al., 2020), no prior work has explored adversarial evaluation and exposed vulnerabilities over large pre-trained multi-lingual language models. (Devlin et al., 2019).

Prior work on defending against rogue attacks as been explored in QA in recent years (Jia and Liang, 2017; Wang and Bansal, 2018; Maharana and Bansal, 2020). The earlier work (Jia and Liang, 2017; Wang and Bansal, 2018) trained a BiDAF (Seo et al., 2016) QA system using adversarial mutated data and most recently (Maharana and Bansal, 2020) use reinforcement learning to select the adversarial training policies to help create a robust model. Our approach is similar to (Jia and Liang, 2017; Wang and Bansal, 2018), but we also include data-augmentation of translated QA pairs in our model which helps build multi-lingual models that are less susceptible to attacks. We leave exploring more sophisticated defense strategies as future work.

The most recent adversarial work, (Maharana and Bansal, 2020) extends (Wang and Bansal, 2018) and expands the pool of distractor statements by synonyms replacement and paraphrasing. They use the distractors to attack an English QA system and they show that training with such distractors helps the system overcome these attacks. However, training with adversarial statements causes a loss in performance of the English system when evaluated on the original data, without the attack. A more sophisticated approach is needed to preserve the performance on the original dataset. We find that training our model with translated multilingual data preserves the performance of the original systems when the attack is not present. Finally, some cross-lingual experiments with de, ru and tr are shown, but they do not use multilingual models or any multilingual attacks. They apply the translate-test
method to translate to English, predict the results on the translation and align the result back to the original language.

3 Attack Methodology

Prior QA attack strategies include adding adversarial sentences to distract RC systems (Jia and Liang, 2017; Wang and Bansal, 2018; Wallace et al., 2019) and using question variations (Sen and Saffari, 2020). Our attack approach extends the AddSentDiverse algorithm (Jia and Liang, 2017; Wang and Bansal, 2018)) which builds adversarial distractor examples by converting the question Q into a statement S using several manually defined rules. Similarly, our goal is to generate an adversarial S that is semantically similar to Q but identifiable by a human reader as incorrect. We aim for grammatically correct sentences, using our generated rules but do not enforce this. We generate four different attack approaches. The attacks create various types of adversarial statements to confuse the QA system. Further, we explore generating adversarial statements in a multi-lingual setting. The steps for converting an English Q to an adversarial S are shown in Figure 2 and described in detail below.

3.1 Step 1: Markup Question

We take as input Q, and run two linguistic preprocessing steps: 1) Universal Dependency Parsing (UDP) (McDonald et al., 2013) using a model similar to (Qi et al., 2018) and 2) Named Entity Recognition (NER) (Sang and De Meulder, 2003) using the publicly available Spacy toolkit. We find the focus words (e.g. which, what etc.) using their corresponding POS tags (e.g. WRB, WB) generated by the parser. We perform a depth-first search on the parse and mark all POS tokens that are on the same level or a child of the focus word as part of the question rule. This approach creates over 9000 patterns in the SQuAD training set, some occurring only once. Frequent patterns include “what nn”, “what vb”, “who vb”, “how many”, and “what vb vb” accounting for over 40% of the training set. In addition, we also mark up all the entities in the question. We give priority to entities tagged by the NER that are not part of the question pattern. However, when such entities are not found, we look at nouns and then verbs to ensure better coverage. “what vb” is the pattern found in “What is the oldest cafe in Paris?” as shown in Figure 2.

3.2 Step 2: Convert Question to Statement

The pattern found in step 1 is used to choose from 8 rules based on the common question words: {“who”, “what”, “when”, “why”, “which”, “where”, “how”} and a catchall for any pattern that does not have question words (these are usually due to ill-formed questions or misspellings such as “Beyonce’s grandma’s name was?”). The rule converts the question Qx into a statement Sx containing the tagged question entities and adds an <ANSWER> placeholder. If the first question word found in the pattern is “what”, the rule “what vb” will replace “what” with <ANSWER>. For example, “<ANSWER> is the oldest cafe in Paris”, as shown in Figure 2. Sometimes, the answer is added to the end of the statement. The “when vb vb” pattern will trigger the rule for “when” which converts “When did Destiny’s Child release their second album?” to “Destiny’s Child released their second album in <ANSWER>”.

3.3 Step 3: Generate Adversarial Statements

Given question Q and statement S, four attack statements are generated which replace <ANSWER> and/or question entities based on the attack. The candidate entities are randomly chosen from the entities found in the SQuAD training data based on their type. The type of the answer entity is chosen based on the entity that the system predicts for the dev/test question. The candidate entities are applied to create the adversarial statement using the following transformations from most complex to most simple. An example for each of the attacks is shown in Figure 2.

**RARQ: Random Answer Random Question**

The adversarial statement has a Random Answer entity and one Random Question entity is changed. This approach is similar to the technique found in prior work (Wang et al., 2020). Our attack approach extends the technique found in prior work (Wang et al., 2020) and uses question variations (Sen and Saffari, 2020). The attacks create various types of adversarial statements to confuse the QA system. Further, we explore generating adversarial statements in a multi-lingual setting. The steps for converting an English Q to an adversarial S are shown in Figure 2.
et al., 2018), however they only explore monolingual.

RAOQ: Random Answer Original Question
The adversarial statement has a Random Answer entity but the Original Question entities remain. This may appear to be too harsh of an adversary because the statement can look like a correct answer. However, we found that in most cases (19/20 times) people can successfully determine that this statement is adversarial and it should be clear that the adversarial statement does not belong when it is in a different language than the rest of the context. We provide further discussion on this attack in Section 8.

NARQ: No Answer Random Question
The adversarial statement has No Answer and one Random Question entity is changed.

NAOQ: No Answer Original Question
The generated adversary has No Answer and the Original Question entities remain.

3.4 Step 4: Translate
The adversarial statements created in step 3 are always generated for English questions. Since we are evaluating a multilingual dataset and model, all statements are translated into the six other languages available in the dataset.

3.5 Step 5: Insert Statements in Context
The generated statements are inserted in random positions in $C_x$ as in (Wang et al., 2018). This produces a new instance $(Q_x, C_y, A_y, S_z)$ where $x, y, z \in L$ are the languages for the question, context, and statement respectively and they need not be the same. We avoid always adding distractors at the beginning or end of the context (Jia and Liang, 2017) as QA systems can easily learn to predict such distractors.

4 Data
SQuAD v1.1 Our primary training dataset is SQuAD v1.1 (Rajpurkar et al., 2016) which is an English only dataset. Training on English data and evaluating on a multilingual dataset is considered a zero shot (ZS) setting. The SQuAD training set is large consisting of over 87,000 QA pairs. Each instance has a question $Q$, context $C$, and an answer $A$ within the context in English: $(Q_{en}, C_{en}, A_{en})$.

MT-SQuAD: Augmented Translation Data
Since we only have the SQuAD English examples to train our system, we use a commercial transla-
tion API⁴ to expand our training data. For every example in SQuAD, we translate the question and the context to six other languages: German (de), Spanish (es), Arabic (ar), Hindi (hi), Vietnamese (vi) and Chinese (zh). We obtain the gold answer for the translated examples by aligning the gold answer in the English context to the translated context. We perform the answer alignment by placing pseudo HTML tags around $A_{en}$ and then translate $C_{en}$. In the majority of cases, the translated answer is marked by the same tags inside the translated context. We discard the translation when the answer alignment does not succeed⁵.

### MLQA Dataset

We evaluate our model on MLQA (Lewis et al., 2020), a large parallel multilingual QA dataset consisting of 7 languages. Each MLQA instance has $(Q_x, C_y, A_y)$ where the question and context language need not be the same. We focus on the more comprehensive task, generalized cross-lingual transfer (G-XLT), by extracting answer $A_y$ from context $C_y$ in language $y$ given question $Q_x$ in language $x$, for all language pairs. There is no training data in MLQA, encouraging ZS approaches using datasets such as SQuAD. We use the official evaluation metric of the MLQA dataset, the mean F1 scores for all $(x, y)$ language pairs.

Note that although we focus on the SQuAD-like MLQA corpus in our experiments, our approach can be applied to any reading comprehension task and any corpus.

### 5 Experimental Setup

Our experiments focus on the MLQA task using the mBERT pre-trained language model. In all cases our model is the same but the training and test data used differs for each attack and defense. We train our QA models using mBERT with our two datasets described in Section 4: SQuAD (en only) and MT-SQuAD. This results in two models: mBERT$_{QA}$ and MT-mBERT$_{QA}$.

The mBERT$_{QA}$ and MT-mBERT$_{QA}$ model architecture is a standard MRC/QA model built on top of mBERT (Devlin et al., 2019) which processes two separate input sequences, one for the question and one for the given context and train two classification heads on top of mBERT, pointing to the start and end positions of the answer span.

⁴https://www.ibm.com/watson/services/language-translator/
⁵Hindi is the least successful, only translating 24% of the data. The rest translate at least 90% of the data.

| Dataset | SQuAD Dev | MLQA Dev | MLQA Test |
|---------|-----------|----------|-----------|
| BERTb   | mBERT     | BERTb    | mBERT     |
| ORIG    | 86.0      | 69.2     | 69.2      | 80.4      |
| RARQ    | 47.8      | 36.7     | 38.0      | 37.3      | 37.9      |
| NAQ    | 76.9      | 59.1     | 55.4      | 58.8      | 56.2      |
| RAQ    | 39.3      | 13.0     | 12.3      | 29.2      | 11.8      |
| NAOQ   | 87.2      | 70.2     | 59.9      | 69.6      | 60.0      |

Table 1: F1 scores for the original dataset and attack (C$_{en}$, Q$_{en}$) with adversarial S$_{en}$ for SQuAD and MLQA using BERT$_{QA}$ and mBERT$_{QA}$ models. The attacks that have the biggest impact are highlighted in bold.

We compute G-XLT at test time by extracting answer $A_z$ from context $C_z$ in language $z$ given question $Q_y$ in language $y$ for all language pairs. The test results are computed on the MLQA dataset. We use the official evaluation metric used by the MLQA dataset and report the performance of our models on the G-XLT task with the mean F1 scores for all $(Q_l, C_l)$ language pairs.

### 6 Attack Experiments

#### 6.1 English Only

We first explore the robustness of mBERT$_{QA}$, when trained on SQuAD, with the attacks using English data from SQuAD and MLQA to create the adversarial statement $S_{en}$ as $(Q_{en}, C_{en}, A_{en}, S_{en})$. We contrast the multilingual mBERT$_{QA}$ to the English BERT-Base (BERT$_b$$_{QA}$) on the same attacks. Table 1 shows the results for these experiments. We find that the attack has a significantly bigger impact when using the mBERT$_{QA}$ model than using BERT$_b$$_{QA}$ for the SQuAD dataset, even though mBERT$_{QA}$ performs 3 points better than BERT$_b$$_{QA}$ on the original (ORIG) dataset that does not contain adversarial statements. Similarly, the attack with the MLQA dataset is far more successful on the mBERT$_{QA}$ model compared to BERT$_b$$_{QA}$. Interestingly, the NAOQ attack is not successful on both datasets when using BERT$_b$$_{QA}$. We suspect that the model understands...
We attack both

The attacks that have the biggest impact are highlighted

Table 2: G-XLT results for the original test set and

that no answer entity is available and the syntax of

that no answer entity is available and the syntax of the

statement helps steer the model to the cor-

Next, we look at adding the adversarial statement in the language of the question, $S_{Q_l}$ as $(Q_x, C_y, A_y, S_y)$ where $x, y \in L$ and may or may not be the same language. These attacks have the strongest impact, indicating that the model gives a preference to the language in the question regardless of whether it matches the language in the context.

We also explore adding an English adversarial statement, $S_{en}$, to the context $C_y$ for all of the test examples as $(Q_x, C_y, A_y, S_{en})$. English is the language of training data in mBERT$_{QA}$. Individual per language results (not visible in the table) showed that it causes a significant decrease in performance for all attacks ranging from 13.3 to 37 points compared to the original test set. Furthermore, adding $S_{en}$ to $C_y$ where $y \neq en$ also causes a significant decrease in performance similar to or worse than $C_{en}$, even though the statement is in a different language than the rest of the context and the question may or may not be in English.

We explore this further using machine translation to add the adversarial statement in German, $S_{de}$ and Chinese, $S_{zh}$. The $S_{de}$ attack is slightly weaker than the $S_{en}$ attack and the $S_{zh}$ attack is considerably weaker than the $S_{en}$ attack; the model has a strong preference for predicting English statements regardless of the language of $Q_x$ and $C_y$ and prefers languages more similar to English than those that are more distant. Furthermore, the preference for English compared to German is also observed when using mt-mBERT$_{QA}$ which is trained with data in 6 languages. However, the attack on Chinese is much stronger when using mt-mBERT$_{QA}$ compared to mBERT$_{QA}$ indicating that the model does learn to distinguish between more distant languages in a non-ZS approach. We also notice that with mt-mBERT$_{QA}$ the non-adversarial model performs significantly better, but the attack is stronger. This may be due to the added noise from translation.

6.3 Attack Strategies

Our four different strategies for generating adversaries perform similarly across all experiments. RAQO is consistently the strongest attack because it looks the most like the actual answer even though humans can easily identify that it is not the correct answer. At minimum, it causes a 30 point reduction (see RAQO $S_{de}$ attack in mt-mBERT$_{QA}$ model in Table 2) but in its worst attack it causes a whopping
We have shown that we can successfully attack the system using several scenarios and exploit vulnerabilities in multilingual QA models. Now we follow prior approaches (Jia and Liang, 2017; Wang and Bansal, 2018; Maharana and Bansal, 2020), to explore how we can make the model more robust to multilingual adversarial attacks. We achieve this by augmenting the model with examples of the attack strategies during training in addition to the original (ORIG) data so that it can learn to spot these adversarities on its own, as described in (Maharana and Bansal, 2020). To avoid the over-fitting of the statement structure we add randomness to the template and we also randomize the location of the points of confusion inside the context.

Our approach for generating statements is the same as during the attack strategy, except that in contrast to the attack strategy, in this case we choose random entities to replace the answer using the gold answer instead of the prediction. We train a separate model for each attack ORIG + {RARQ, NARQ, RAOQ, NAOQ}, and then evaluate how the new models trained with the adversarial data perform on the various adversarial test sets. We explore the defense with both the mBERT\textsubscript{QA} and MT-mBERT\textsubscript{QA} models. In the mBERT\textsubscript{QA} model all attack statements during training time are in English. In the MT-mBERT\textsubscript{QA} model they are in all languages. The results are shown in Table 3. The ORIG column shows the original training model with the attacks as in Table 2 for comparison.

The ORIG rows of Table 3 show how the models perform when there are no adversarial statements. The defense models that contain the adversarial statements suffer a loss (2-4 points) in performance when testing with mBERT\textsubscript{QA} on the original data. On the other hand, all MT-mBERT\textsubscript{QA} defense models obtain consistent scores and maintain the performance of the original model when testing with the original dataset where no adversarial statements exist. We also find that all the defense training models can effectively protect the QA system against the attacks, in most cases reaching near original performance if the attack is the same as defense strategy (the diagonal results; e.g. the RARQ training model recovers best for the RARQ test set), with consistently better performance for the MT-mBERT\textsubscript{QA} model. The models can even recover well for different attack strategies. This indicates that the defense models may be able to withstand other types of attacks than the ones tried here. For the mBERT\textsubscript{QA} model RAOQ recovers better using all other training models. We expect the similarity to the question causes difficulty during training.

Initial experiments applying all attacks at once was slightly better than the individual models indicating that such an approach could be useful. Our insights show that basic defense strategies assist with over-fitting of multilingual QA models that are under attack. We show particularly positive success when augmenting the MT model with attack statements during training. We leave exploring more complex models such as adversarial networks (Goodfellow et al., 2014, 2015) on top of BERT as future work.

8 Discussion and Analysis

We have shown that our attacks that are generated automatically by converting the question to a statement are successful in fooling the system. It is important that the attacks are easily detectable by people to ensure that our attacks are fair in exploiting flaws in the system. Specifically, with the RAOQ attack, these can appear to be natural sentences that could answer the question. If outside knowledge is unknown, the actual answer and the adversarial RAOQ statement can both appear correct without context. However, these sentences will usually not flow with the rest of the passage and it will be the only mention of the entity. The correct entity will likely be mentioned more than once in the passage. Finally, to the average person with external knowledge it should be easy to recognize that the entities do not belong with the rest of the passage. We verified this through an annotation task on 100 English instances using two native English speakers (50 each), 20 for each of the four attacks and 20 for the original instances with no
Table 3: Average G-XLT results for the Defense models for the SQuAD and MT training data per test set and the statement language. The first column (ORIG) shows the attacks as found in Table 2 for comparison. The header row shows the different training strategies which include the original data and one attack strategy. The attack strategy during test time is shown by row as strategy + S Lang. The defense models per column with the biggest improvement are highlighted in bold.

| Test Data | S Lang | ORIG + RARQ | NARQ + RAOQ | NAOQ + RAOQ |
|-----------|--------|-------------|-------------|-------------|
| MBERT: SQuAD Training Data |        |             |             |             |
| ORIG      | 52.1   | 50.1        | 49.6        | 48.8        | 48.3        |
| RARQ      | 27.7   | 46.9        | 42.4        | 42.3        | 32.1        |
| S_c       | 15.7   | 44.5        | 38.0        | 35.7        | 21.7        |
| S_q       | 36.3   | 44.0        | 40.7        | 40.0        | 35.5        |
| RARQ      | 36.8   | 47.1        | 48.2        | 45.1        | 48.8        |
| S_c       | 24.8   | 44.4        | 45.9        | 40.2        | 35.3        |
| S_q       | 41.0   | 44.6        | 45.0        | 42.2        | 40.0        |
| RARQ      | 17.2   | 36.1        | 34.1        | 38.0        | 25.5        |
| NARQ      | 7.5    | 33.3        | 30.9        | 33.9        | 17.9        |
| S_c       | 24.8   | 32.7        | 31.4        | 32.9        | 27.3        |
| NARQ      | 36.8   | 42.3        | 46.9        | 43.6        | 44.0        |
| S_c       | 26.8   | 38.1        | 45.3        | 41.1        | 42.4        |
| NARQ      | 39.5   | 40.0        | 42.5        | 39.9        | 40.4        |
| NAOQ      | 18.8   | 56.8        | 51.0        | 55.4        | 33.1        |
| S_c       | 20.3   | **57.6**    | 54.0        | **57.1**    | 37.2        |
| S_q       | 36.6   | 57.1        | 53.3        | 56.4        | 43.5        |
| NAOQ      | 30.3   | 56.1        | **57.6**    | 56.5        | 54.3        |
| S_c       | 29.1   | 56.7        | 57.5        | 55.9        | 54.8        |
| NAOQ      | 42.7   | 56.5        | 57.3        | 56.6        | 54.6        |
| RAQQ      | 9.2    | 47.2        | 40.9        | 54.1        | 29.5        |
| S_c       | 9.3    | 52.1        | 47.1        | 56.9        | 35.5        |
| S_q       | 22.7   | 50.3        | 45.2        | 55.1        | 38.0        |
| NAOQ      | 30.1   | 48.7        | 56.1        | 55.2        | **58.0**    |
| S_c       | 27.8   | 51.1        | 56.6        | 56.5        | 57.8        |
| NAOQ      | 41.2   | 50.8        | 56.0        | 55.9        | 57.8        |

In the above passage the adversarial statement is italicized. First, the entity National Declassification Center that is included in the sentence is clearly not the name of the artist. Further, even if one was uncertain about this, the sentence doesn’t flow with the rest of the paragraph. It reads much better if that sentence is removed. Finally, the sentence that does include the correct answer flows better on its own, and within the paragraph.

9 Conclusion

We have shown several novel adversaries that successfully attack mBERT for MLQA. Specifically, we show that the language of the adversarial statement impacts the attack with priority given to English and the language of the question regardless of the other languages in the QA pair. We also show that including such attack strategies while training our defense brings back performance without the need for complex neural network engineering. Not only do the strategies improve results for their corresponding attack, they help for all our attacks indicating model robustness. In the future, we plan to expose vulnerabilities on other multilingual LMs and datasets and explore more sophisticated defense strategies.

10 Ethics

The intended use of this work is to expose vulnerabilities in question answering systems solely as a means of making them more robust. We are aware that this work could be misused to attack question answering systems. We do not promote attacking systems for any malicious intent.
References

Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. 2020. On the cross-lingual transferability of monolingual representations. in ACL 2020.

Akari Asai, Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasu Tsuruoka. 2018. Multilingual extractive reading comprehension by runtime machine translation. arXiv preprint arXiv:1809.03275.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc.

Weí He, Kái Liu, Jíng Liu, Yajuan Lyu, Shíqi Zhao, Xínyan Xiao, Yuán Liú, Yízhong Wáng, Huá Wú, Qiaoqiao She, et al. 2017. Dureader: a chinese machine reading comprehension dataset from real-world applications. arXiv preprint arXiv:1711.05073.

Mínghao Hu, Yuxíng Peng, Zhèn Huáng, Xípeng Qiú, Fúru Wèi, and Míng Zhóu. 2018. Reinforced mnemonic reader for machine reading comprehension. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, pages 4099–4106. International Joint Conferences on Artificial Intelligence Organization.

Róbin Jíá and Pércy Liáng. 2017. Adversarial examples for evaluating reading comprehension systems. EMNLP.

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, Lílon Jónes, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quóç Lé, and Slav Peteò. 2019. Natural Questions: a benchmark for question answering research. TACL.

Patrick Lewis, Barlas Öğuz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. ACL.

Adyasha Maharana and M. Bansal. 2020. Adversarial augmentation policy search for domain and cross-lingual generalization in reading comprehension. In EMNLP.
Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *ACL*.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. *NAACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.