Improving QA System Out of Domain Performance Using Data Augmentation

Stanford CS224N Default Robust QA Project

Lauren Pendo
Institute for Computational and Mathematical Engineering
Stanford University
lpendo@stanford.edu

Amelia Gilson
Department of Statistics
Stanford University
agilson@stanford.edu

Abstract

We investigated the effectiveness of different data augmentation and sampling techniques to improve the robustness of the pre-trained DistilBERT question answering system on out of domain data. We trained the DistilBERT model on the in domain data and then experimented with fine-tuning using augmented versions of the out of domain data. To generate the additional data-points we performed random word deletion, synonym replacement, and random swapping. We found that all the fine-tuned models performed better than the baseline model. Additionally, we found that our optimal synonym replacement model performed the best on the out of domain test set, and that the combination model of synonym replacement and deletion also led to increased performance over the baseline. Overall, we conclude that data augmentation does increase the ability of our question answering system to generalize to out of domain data and suggest that future work could look further into applying combinations of these data augmentation techniques.

1 Introduction

In recent years question and answering (QA) systems have become widely used in many modern technology applications, such as search engine querying and virtual assistants. However, despite recent advances in QA modeling, these systems still struggle to generalize to a specific domain without specialized training data and information about that domain's distribution. For this reason, the bias introduced from the training dataset can limit the QA model's ability to generalize to other tasks and this means that it is hard to apply these models to problems where a large amount of training data is not available. This motivated us to experiment with a variety of data augmentation and sampling techniques to try to increase the robustness of the pre-trained DistilBERT QA system. Data augmentation tools have been shown to improve the performance and generalizability of other machine learning models. Therefore, our approach for a more robust question answering system is to implement and compare different text-based data augmentation techniques[1]. We applied the data augmentation to expand our out of domain training dataset to better the model's performance on the out of domain validation and test datasets [2].

2 Related work

In "An Exploration of Data Augmentation and Sampling Techniques for Domain-Agnostic Question Answering", the authors experimented with data augmentation on QA systems using back translation [1]. To obtain their back translations, they trained an 8-layer, seq2seq model with attention to translate from English to German and back to English. They then applied two sampling techniques to select which of the augmented data points were included in their final dataset and tuned over the probability that each additional point was included in the final dataset. They concluded that data augmentation did not increase in domain or out domain performance. However, this paper only developed one

method for generating the augmented dataset (through back translation) which inspired us to explore if using other data augmentation techniques could result in the performance increases that back translation alone could not.

To further motivate our approach, we also reviewed the paper "EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks" which analyses the performance of four different data augmentation operations: synonym replacement, random insertion, random swap, and random deletion [3]. The authors found that by implementing these techniques they were able to achieve particularly strong performance increases for smaller datasets. They also found that these methods can substantially reduce over-fitting when training on smaller datasets. Given our task and the size of our out of domain data we therefore decided to implement similar text-based data augmentation techniques.

3 Approach

We began by training the baseline, which involved fine-tuning the DistilBERT model (a smaller version the original BERT model) using the in-domain training and validation sets. We saved this model and then updated our code to allow the different data augmentation techniques to perform fine-tuning on this baseline. In this way, all of the additional fine-tuning was implemented by first reading in this pre-trained baseline's parameters and then training on the augmented out of domain training set. Our first data augmentation approach was to increase the size of the out of domain dataset by creating new questions and contexts with random deletion. For each out of domain context-query pair, we deleted a word with probability p_a, where the 'a' represents 'augment' [1]. We then created new IDs for these examples and appended them to our original data dictionary. The code was inspired by [3], but we implemented our own delete functions.

The second approach we used to generate additional query-context pairs was synonym replacement, where we replaced a fixed percent of the words (again calling the proportion of words replace p_a) in each context and query with a synonym. To generate the synonyms we used the "glove-wiki-gigaword-200" model that contains 400,000 vectors and is trained on data from Wikipedia 2014 + Gigaword 5 (6B tokens, uncased). The code for this was heavily borrowed from assignment 1. We then wrote code to replace words from the query-context pairs at random with their "most similar" synonym according to this model. With this, we also created a list of words that were never replaced with synonyms such as "the", "of" and "to". Because these words appeared with such high frequently in the dataset, we wanted to to avoid creating new context-query pairs that only replaced these words. In addition, we did not consider replacing any words that contained numbers or capital letters (using capital letters as an indicator for proper nouns). This was done because these types of words lacked valid synonyms.

Finally, we implemented two additional data augmentation methods by randomly swapping words or sentences. For the method which randomly swapped words, we applied this method to a fixed percent of the words based on the length of the context or query (again calling the proportion of words swapped p_a) in both the context and query independently. However, we only applied the method which randomly swapped sentences to the context strings given that each of the queries is itself only one sentence. For this method, we randomly swapped a fixed percent of the sentences relative to the total number of sentences in the context. Therefore using this method p_a represents the proportion of total sentences swapped. We implemented the swapping technique at both the word and sentence level to try to get a better understanding about the granularity of the data that the model is learning. We did not do this for either of the other methods given that we could not generate valid "synonyms" for sentences and we did not want to delete entire sentences from either the context or query.

Calling the original query-context pair (q, c) and each element in the generated pair (q', c'), we then independently chose each of the generated elements to be included in the augmented dataset with with probability p_s, where the 's' represents 'sample'. Note that in this selection method, the original pair (q, c) is always included and additional data-points are randomly added. Using this method, it is also possible that a new data point contains both of the generated elements (q', c') or that it contains a combination of the generated and original pair (q', c) or (q, c'). The methodology behind this selection technique was from paper [1] but we wrote our own code to implement this.

Longpre et al. [1] also proposed a sampling method called "Active Learning" where the authors attempted to select the augmented data-points (in this case generated through back translation) that

were the most different from their original counterparts. Using this method they wanted to prioritize including new data-points that add variety to the data distribution and were attempting to avoid introducing repetitive signals. They did this by quantifying the "difficulty" of a back-translated example using the F1 score and creating different probability functions that used the F1 scores as weights. For our model we only used the first sampling method, where each context-query pair is sampled with uniform probability. However, we are still training over adding more similar or dissimilar query-context pairs by additionally tuning over the percent of the words replaced, deleted, or swapped. In this way, we try to achieve the same results as their "Active Learning" sampling method but from our data augmentation techniques and not from the sampling method.

With all four data augmentations implemented, we then constructed a method to combine and repeat different augmentation techniques. When creating combinations, we passed the original query-context pairs through each of the four data augmentation's described before, appending the new (q', c') pairs to the training data. Each data augmentation technique was applied to the original data meaning for example that we did not append swapped sentences and then randomly delete words from the same samples. As described below in the results, we then removed swap and swap sentences as they performed worse than delete and synonym replacement and implemented a repeating augmentation, where we called delete and synonym replacement on the original data multiple times.

4 Experiments

We created our baseline model by fine-tuning the pre-trained DistilBERT model using the in domain SQuAD, NewsQA, and Natural Questions datasets provided as part of the default project [4]. We then further fine-tuned the baseline model on the out of domain DuoRC, RACE, and RelationExtraction datasets (without doing any data augmentation) as a second baseline to compare our augmentation techniques. The results of these two models when evaluated on the out of domain validation sets are shown below in table 1. For fine-tuning on the out of domain datasets without data augmentation we kept all tuning parameters set to their default values, with the exception of eval-every, which we set to 50. This was changed because this model trains for far fewer steps compared to the model trained on the in domain dataset. Our evaluation metrics are EM and F1 scores.

Model	Split Strategy	p_a	p_s	EM	F1
Baseline	NA	NA	NA	33.246	48.432
Finetune OO-Domain	NA	NA	NA	35.0798	50.092

Table 1: Baseline Dev Results

4.1 Validation Datasets

We then tried fine-tuning the baseline model using each data augmentation technique individually. For each of these techniques we experimented with different values of p_a and p_s. More specifically, we tested over a grid of values satisfying $p_a \in \{0.5, 0.1\}$ and $p_s \in \{0.6, 0.8, 1\}$

We first tested random deletion as an augmentation strategy with different values of p_a and p_s as well as different splitting strategies. The results of this are shown below in table 2. We found, using, $p_a = 0.1$, $p_s = 1$, that the regex splitting method performed better than whitespace splitting for this data augmentation technique. Thus for the other random deletion tests, we only used regex splitting to remove all punctuation. We found that the highest EM and F1 were achieved with $p_a = .05$ and $p_s = 0.6$. This is in contrast to future experiments, where we saw $p_s = 1$ performing the best. However, in this case $p_s = 1$ does not have a significantly worse F1 score, with a difference of 0.27. However, the difference in EM is 1.31.

For synonym replacement we also compared different methods to split each context or query from a string to a list of strings (where each string in the list is a word). We initially used the findall method in the "regular expression" python package which removed all special characters and white spaces by splitting on these characters. We also tried implementing our own splitting function, where we first split on white space without removing any of the special characters. Then, while randomly iterating through the words in the query or context, we fed each selected word into a method which removed a predefined list of special characters. This way we did not remove all special characters from the input

Model	Split Strategy	p_a	p_s	EM	F1
Random Deletion	Strip Punctuation	0.1	1	34.293	49.308
Random Deletion	Strip Punctuation	0.05	1	34.55	50.20
Random Deletion	Strip Punctuation	0.05	0.8	34.03	49.71
Random Deletion	**Strip Punctuation**	**0.05**	**0.6**	**35.86**	**50.47**
Random Deletion	Strip Punctuation	0.05	0.4	34.82	50.05
Random Deletion	Whitespace	0.1	1	33.77	48.58

Table 2: Random Deletion Dev Results

Model	Split Strategy	p_a	p_s	EM	F1
Synonym Replacement	Strip Punctuation	0.1	0.5	34.031	49.759
Synonym Replacement	Strip Punctuation	0.05	0.5	33.770	49.410
Synonym Replacement	Strip Punctuation	0.1	1	33.51	49.67
Synonym Replacement	White space	0.1	1	33.77	51.57
Synonym Replacement	**White space**	**0.1**	**.8**	**34.55**	**51.94**
Synonym Replacement	White space	0.1	.6	32.98	50.23
Synonym Replacement	**White space**	**0.05**	**1**	**35.34**	**51.84**
Synonym Replacement	White space	0.05	.8	31.94	50.94

Table 3: Synonym Replacement Dev Results

string, just the characters from the words for which we were generating synonyms. We tested both of these splitting methods by fine-tuning the baseline model, the results of which are in table 3. We can see by looking at the EM and F1 scores that the models which first split on white space outperform the models that strip all punctuation. The best EM score was achieved using the model that split on white space and had p_a and p_s equal to .05 and 1 respectively. The model with the highest F1 score was achieved using p_a and p_s equal to 0.1 and .8 respectively.

Then we fine-tuned the baseline model using the random word swap and random sentence swap methods, the results of which are shown in table 4. We can see that these two swap methods perform similarly to each other across the different p_a and p_s values. The random word swap method with p_a = 0.1 and p_s = 1 had the highest F1 score of 49.67 and the random sentence swap method with p_a = 0.1 and p_s = .8 had the highest EM score of 34.29. However, neither of these methods achieved an F1 score as high as the random deletion and synonym replacement models. Additionally, both the EM and F1 scores for all of these models was lower than the evaluation metrics for the baseline model fine-tuned on the out of domain data set without any data augmentation. This suggests that training using data points with swapped words and sentences impedes the model's ability to learn which is likely a result of the models over-fitting to the training data.

Model	p_a	p_s	Split Strategy	EM	F1	Split Strategy	EM	F1
Random Swap	0.1	1	Sentences	33.51	49.25	**Words**	**33.51**	**49.67**
Random Swap	0.05	1	Sentences	32.98	49.09	Words	33.25	48.3
Random Swap	0.1	.8	**Sentences**	**34.29**	**49.61**	Words	34.03	49.63
Random Swap	0.1	.6	Sentences	33.77	48.92	Words	34.03	48.99
Random Swap	0.05	.8	Sentences	33.25	48.46	NA	NA	NA
Random Swap	0.05	.6	Sentences	33.25	48.29	NA	NA	NA

Table 4: Random Swap Dev Results

p_a	p_s	Model	EM	F1	Model	EM	F1
0.1	1	All Methods	33.51	49.09	Synonym and Deletion	34.55	50.86
0.1	.8	All Methods	33.25	49.49	Synonym and Deletion	34.29	50.31
0.1	.6	All Methods	31.41	48.8	Synonym and Deletion	34.29	50.42
0.05	1	All Methods	34.03	51.19	**Synonym and Deletion**	**36.13**	**53.03**
0.05	.8	All Methods	34.82	50.81	Synonym and Deletion	35.08	50.91
0.05	.6	All Methods	35.6	49.61	Synonym and Deletion	34.82	50.57

Table 5: Combination Dev Results

Model	p_a	p_s	Times Applied	EM	F1
Synonym and Deletion	0.05	1	1	36.13	53.03
Synonym and Deletion	0.05	1	2	35.08	50.41
Synonym and Deletion	0.05	1	3	35.6	52.65
Synonym and Deletion	0.05	1	4	34.03	50.27

Table 6: Repeat Delete and Synonym Replacement Dev Results

To further increase the size and diversity of our training data, we then decided to fine-tune models using a combination of the techniques described above. Again using a range of p_a and p_s values, we first tested models that used all four data augmentation techniques. Then, because they performed the best in our initial tests, we also tested models that just used a combination of synonym replacement and random deletion. The results of all of these experiments are shown in Table 5. We can see from this table that, when using the same p_a and p_s values, the model which only used synonym replacement and random deletion always produced a higher F1 score and, with the exception of when p_a = .05 and p_s = .6, a higher EM score. These results again helped support our hypothesis that using the random swap methods does not increase the performance accuracy.

Finally, because the optimal p_s value for most of our methods was 1, we also experimented with passing the data through the data augmentation methods multiple times to increase the p_s to multiples of 1. We did this specifically for the model which used a combination of synonym replacement and random deletion as this combination performed the best of any model when we were restricting p_s to be less than or equal to 1. As $p_a = 0.05$ performed the best on most of our combinations, we also passed this parameter through to create datasets that included 2x, 3x and 4x passes through our augmentation and sampling functions. The additional samples were created by calling our delete and synonym replacement methods multiple times on the original data and appending this new data to our training set. We can see that from this we achieved the best dev performance when we created the augmented datasets 1 and 3 times as seen in Table 6.

4.2 Test Datasets

We selected our three best performing models to run on the test dataset. The three best models were Synonym Replacement, combination of Synonym/Deletion 1x, and Synonym/Deletion 3x. All of these models had $p_a = 0.05$ and $p_s = .1$. While the combination models performed better on our validation set, the model which only trained using synonym replacement led to the best results on the out of domain test set, as seen in table 7. The F1 score of synonym replacement was 3.65% greater than the delete and synonym repeated 1 time model and 5.28% greater than the delete and synonym model repeated 3 times. The EM was 1.52% and 2.55% greater respectively.

Model	p_a	p_s	Times Applied	EM	F1
Synonym and Deletion	0.05	1	1	41.07	60.289
Synonym and Deletion	0.05	1	3	40.436	59.682
Synonym	**0.05**	**1**	**NA**	**42.569**	**61.204**

Table 7: Test Results

5 Analysis

5.1 Parameter Selection

Throughout our experiments we consistently saw that the models with $p_a = 0.05$ outperformed models with other p_a and p_s values. Because of this we hypothesize that augmenting 5% of the data introduces an appropriate additional level of entropy to the generated samples, where augmenting 10% of the data introduced too much noise for the models to train as well. This would also mean that the samples perturbed with $p_a = 0.05$ contain enough randomness so that the models do not over-fit as much as they would without data augmentation, but also still contain the appropriate answer span where the more perturbed datasets might not.

Additionally, when holding the augmentation method and p_a value constant, we saw that the models trained with $p_s = 1$ generally outperformed the models trained with lower p_s values. Since models with $p_s = 1$ include all of the new generated context-query pairs (q', c'), we hypothesise that this result means including the augmented data points with greater frequency can reduce over-fitting on the out of domain training set. Additionally, the models with $p_s = .6$ often performed similarly to or worse than the baseline model fine-tuned on the out of domain data set (without augmentation). This additionally suggests that including context-query pairs where one generated element is paired with one original element (q, c') or (q', c) can reduce the overall performance of a model.

5.2 Validation Datasets

On the validation dataset, our models which implemented synonym replacement produced answers that were overall both shorter and more accurate than the baseline model (the model fine-tuned on the out of domain data without any augmentation) and the model which only used random deletion. We can see this by looking at the examples:

Question: What was the skeleton named after? **Answer:** Song
Baseline Answer: ADDIS ABABA, Ethiopia-One of the world's most famous fossils-the 3.2 million-year-old Lucy
Random Deletion Answer: ADDIS ABABA, Ethiopia-One of the world's most famous fossils-the 3.2 million-year-old Lucy
Synonym Replacement Answer: Beatles Song
Synonym/Deletion Answer: Beatles Song
Synonym/Deletion 3X Answer: Lucy, her name taken from a Beatles song

Question: What is the ending year of PO European Ferries? **Answer:** 1999
Baseline Answer: 1987 after the Herald of Free Enterprise disaster, when Townsend Thoresen was renamed PO European Ferries, until 1999
Random Deletion Answer: 1987 after the Herald of Free Enterprise disaster, when Townsend Thoresen was renamed PO European Ferries, until 1999
Synonym Replacement Answer: until 1999
Synonym/Deletion Answer: 1999 when
Synonym/Deletion 3X Answer: until 1999

There were additional instances when evaluated on the validation set where the models with synonym replacement were able to give a more specific or exact answer. For example, looking at the examples below, we can see that the synonym replacement models produced the correct answer "Tate Gallery" to a location question where the baseline and random deletion models just said "New York City". Similarly, when asked about a different art museum, the synonym models gave the correct answer "Wadsworth Atheneum Museum of Art" where the other models only said "Hartford, Connecticut".

Combined these example indicate that the noise introduced into the augmented dataset from the synonym replacement techniques helped the model reduce over-fitting, where in this case over-fitting is indicated by the baseline model needing to replicate a larger section of the context to try to answer the questions.

Question: What is the name of the place where Whaam! can be found?**Answer:** Tate
Baseline Answer: New York City
Random Deletion Answer: New York City
Synonym Replacement Answer: Tate Gallery
Synonym/Deletion Answer: Tate Gallery
Synonym/Deletion 3X Answer: Tate Gallery

Question: What is the name of the place where Whaam! can be found? **Answer:** Wadsworth Atheneum
Baseline Answer: Hartford, Connecticut
Random Deletion Answer: Hartford, Connecticut
Synonym Replacement Answer: Wadsworth Atheneum Museum of Art
Synonym/Deletion Answer: Wadsworth Atheneum Museum of Art
Synonym/Deletion 3X Answer: Wadsworth Atheneum Museum of Art

5.3 Test Datasets

Although the models that combined synonym replacement and deletion had the highest F1 and EM scores on our validation set, the synonym model with $p_a = 0.05$ and $p_s = 1.0$ performed the best on the test data. While our sampling methods were introduced to increase the size of the out of domain data, it is likely that the methods that increased the size of the data by larger factors also led to over-fitting. The synonym/deletion repeated once model increases the size of our dataset by a factor of 3 and the synonym/deletion repeated three times increases the data by a factor of 7. Consequently, while this helped improve performance on the validation dataset, it did not on the test data.

We also evaluated some of the differences between the three test submissions' generated answers. The three sets of answers differed across all models at 356 points and differed across at least two for 1,609 answers. When comparing individual datasets, the synonym model and the synonym/deletion repeated 3 times model differed the most with 1,373 mismatched answers while the synonym/deletion repeated 3 times and synonym/deletion repeated once had 1,147 different answers and synonym/deletion repeated once and synonym were the most similar with only 1,054 different answers.

We further looked at the spans of the answers where at least one model generated a different answer. The synonym/deletion repeated 3 times model had the shortest average answer span at 23.05 characters, whereas the synonym/deletion repeated once and synonym model used an average of 26.66 and 26.16 characters respectively. We plotted the distributions of answer length using a log scale in Figure 1. We can see that the synonym/deletion repeated 3 times model is highly concentrated, which indicates that using more multiples of the perturbed data made it less flexible. This model also has some distribution at the high end, which indicates the spans may be too long for a small subset. The synonym test has high values at both the tails as well as high values towards the bottom of lengths, which indicates it remained flexible to finding answers of different span lengths, which likely led to it performing the best on the test set. However, there were cases where our other models performed better and cases where all three produced seemingly acceptable (although different) answers.

Question: What does Woody Woodpecker wish he was?
Synonym Replacement Answer: leprechaun
Synonym/Deletion Answer: were rich
Synonym/Deletion 3X Answer: were rich

In the above example our best test-set (synonym replacement) did not manage to produce the correct answer, whereas our other two submissions were able to. There were other occurrences where one of our two worse sets produced the correct answer, while the synonym replacement model did not, indicating that while the other two models did not produce the best EM and F1, they were able to learn some answers that the synonym replacement model was unable to.

This question below shows the limitations of our evaluation metrics; when we cannot see the answer, it appears that both answers should be acceptable, while only one actually matches the correct answer

span. We found that there were 956 cases where the difference in length between the synonym answer and the other answers was less than 3 characters. While this is not a perfect measure, it does indicate that there are cases where the answers may be similar, but the correct span may have been in synonym, causing the increased performance.

Question: Which replaced the YUBA League?
Synonym Replacement Answer: Basketball League of Serbia
Synonym/Deletion Answer: Basketball League of Serbia
Synonym/Deletion 3X Answer: the Basketball League of Serbia

There are also some cases where the other two models did not select the correct dependency for the named entity. In this example, the synonym model properly infers that Todd and Neil are students who heard Headmaster Nolan's speech, which requires understanding the entire context. On the other hand, the other two models selected phrases containing Norman Lloyd, the actor whose name is in parentheses directly after the reference to Nolan.

Question: Who does Nolan warn?
Synonym Replacement Answer: Todd and Neil
Synonym/Deletion Answer: stern Headmaster Nolan (Norman Lloyd
Synonym/Deletion 3X Answer: Norman Lloyd

We conclude that although synonym replacement produced the best F1 and EM, the other two models were still successful in times where the synonym replacement was not. However, these models may have exhibited over-fitting as seen in Figure 1.

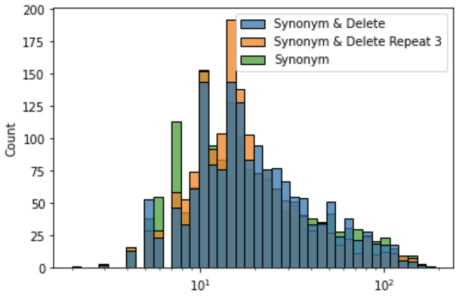

Figure 1: Distribution of Answer Lengths

6 Conclusion

In our comparison of different data augmentation techniques, we found synonym replacement led to the best EM and F1 scores on the test dataset, even though it did not have the best performance on the validation set. Our best validation performance was trained using a combination of synonym replacement and random deletion. However, the fact that this model performed worse on the test set leads us to believe that the larger training datasets led to over-fitting and did not generalize as well as the simpler data augmentation method. Moreover, we conclude that random swapping of words and sentences is not an effective data augmentation technique, as it performs worse than the model fine-tuned on the original out of domain data. Finally, in testing different probability parameters, we conclude that setting $p_a = .05$ and $p_s = 1$ leads to the highest increases in performance across different augmentation techniques.

While the larger training sets we created did not lead to the highest F1 and EM scores, they were able to successfully answer some questions from the test dataset that the synonym replacement model was not. Further, we found that in many cases, the answers were very similar between all three models, but only one answer likely exactly matched the answer span. In future work, we could further analyze these differences to find the optimal trade-off between adding additional perturbed data points and maintaining model flexibility. In addition, we could measure performance by closeness to the Exact Match, allowing for close answers to contribute to the performance of the model. Further work could also include applying more advanced sampling techniques, based on context length or answer span, as opposed to uniform random sampling.

References

[1] 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.

[2] John Miller, Karl Krauth, Benjamin Recht, and Ludwig Schmidt. The effect of natural distribution shift on question answering models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6905–6916. PMLR, 13–18 Jul 2020.

[3] Jason W. Wei and Kai Zou. EDA: easy data augmentation techniques for boosting performance on text classification tasks. *CoRR*, abs/1901.11196, 2019.

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