Adversarially Improving Adversarial Performance of QA models

Yoni Lerner
Department of Computer Science
Stanford University
Stanford, CA 94305
yonlern@stanford.edu

Abstract

Question answering models have made leaps and bounds in recent years with the introduction of standardized tasks like SQuAD[3]. However, recent work has shown that even state-of-the-art models are highly susceptible to even simple manually-generated adversarial examples, casting doubt on whether they truly represent machine comprehension. A robust system to generate diverse adversarial examples for SQuAD could help better diagnose this problem and even remedy it by augmenting datasets with generated examples. We propose the Tag Team Adversary architecture (TTA) to solve this problem. TTA utilizes two competing teams, each consisting of a QA model and adversarial model, along with a judge discriminator network to generate new valid contexts for existing SQuAD context-query-answer triplets. TTA can be used to augment data as a pair of QA models are trained or evaluated, or it can be used with fixed pretrained QA models to generate an augmented dataset.

1 Introduction

Machine comprehension is an important area of active research in natural language processing, with applications in search engines, automated medicine, and more. One popular operationalisation of machine comprehension is the task of question answering, where a model is some context information and then asked a question about it, and must produce an answer. SQuAD is a dataset of labeled examples for this task.[3] A SQuAD model is given a paragraph of context and a question, and must select a substring of the context as its answer. Recent years have seen SQuAD models greatly increase in performance, with SQuAD 1.1 models beating human performance and SQuAD 2.0 models on the cusp as of this writing. However, the question remains whether these models are achieving human-level comprehension; while question-answering is a useful task by itself, machine comprehension is the larger goal.

Recent work has cast doubt on whether these models truly comprehend what they are reading. Jia et al. have shown that SQuAD models are easily fooled in ways that humans are not.s.[2] For example, when simple unrelated sentences are constructed to structurally resemble the query and are tacked on to the end of the context, model performance plummets. Humans are unaffected by these sentences since they rightly ignore them as irrelevant. This implies that current techniques produce models that work well but rely on things like sentence structure and word similarity more than real comprehension of the context.
These adversarial example generation techniques are valuable diagnostic tools, but can't be used as data augmentation tools because of their manual construction. For example, the distractor sentence is always at the end of the paragraph to ensure that the SQuAD answer for the given query and context remains valid. So a model could show increased performance by simply always ignoring the last

Article: Oxygen
Context: Oxygen is a chemical element with symbol O and atomic number 8. It is a member of the chalcogen group on the periodic table and is a highly reactive nonmetal and oxidizing agent that readily forms compounds (notably oxides) with most elements. By mass, oxygen is the third-most abundant element in the universe, after hydrogen and helium. At standard temperature and pressure, two atoms of the element bind to form dioxygen, a colorless and odorless diatomic gas with the formula O2. Diatomic oxygen gas constitutes 20.8% of the Earth's atmosphere. However, monitoring of atmospheric oxygen levels show a global downward trend, because of fossil-fuel burning. Oxygen is the most abundant element by mass in the Earth's crust as part of oxide compounds such as silicon dioxide, making up almost half of the crust's mass.
Question: What is the atomic number of the element oxygen?
Answer: 8

Figure 1: Example of SQuAD data. When reporting the answer, the model would not produce "8" directly but instead indicate its position in the context.

sentence. TTA instead generates mutations to the context through the use of a network, and preserves validity using an adversarial competitive structure and a judge discriminator.

2 Related work

Jia et al. were the first to demonstrate that SQuAD models are easily fooled by adversarial examples. [2]. They proposed two simple methods for adversarially augmenting SQuAD examples.
First, they proposed the AddSent algorithm. AddSent creates adversarial sentences that target QA algorithms that overly focus on finding a sentence with the correct *structure* to answer the question, as opposed to ones that focus on *content*. AddSent creates adversarial sentences by taking the query, mutating it to be a structurally identical but semantically unrelated query, and then transforming the query from a question into a statement. The mutation is done by replacing proper nouns with different but similar ones (as judged by GloVe) and flipping nouns and adjectives with their antonyms (acquired by consulting WordNet). For example, the query "What ABC division handles domestic television distribution?" might be mutated into "What NBC division handles foreign television distribution?". AddSent then generates a fake answer to the question by choosing a fake answer corresponding to the original answer's 'type' (e.g. an abbreviation). Then, the mutated query is combined with the answer to make it a declarative sentence using manual rules. Jia et al. also checked over these generated sentences and corrected mistakes in them using crowdsourcing.

Second, they proposed the AddAny algorithm. AddAny creates a distractor 'sentence' which is an ungrammatical string of words. AddAny uses local search to choose the set of words that leads to poorest performance by the model. However, AddAny does not enforce anything about the added words; they may not be compatible with the original (context, query, answer) triplet. In practice, though, they are usually a string of gibberish words containing many query words but not forming any sort of sensible content.
For both algorithms, the generated sentence is appended to the end of the existing context. The sentence could not be randomly inserted into the context because that would mean disturbing the flow of the paragraph, which may change the meaning of the context (for example by changing what pronouns refer to). This is one major limitation of these methods, because it means that they cannot be used for data augmentation; the model could simply learn to ignore the last sentence.

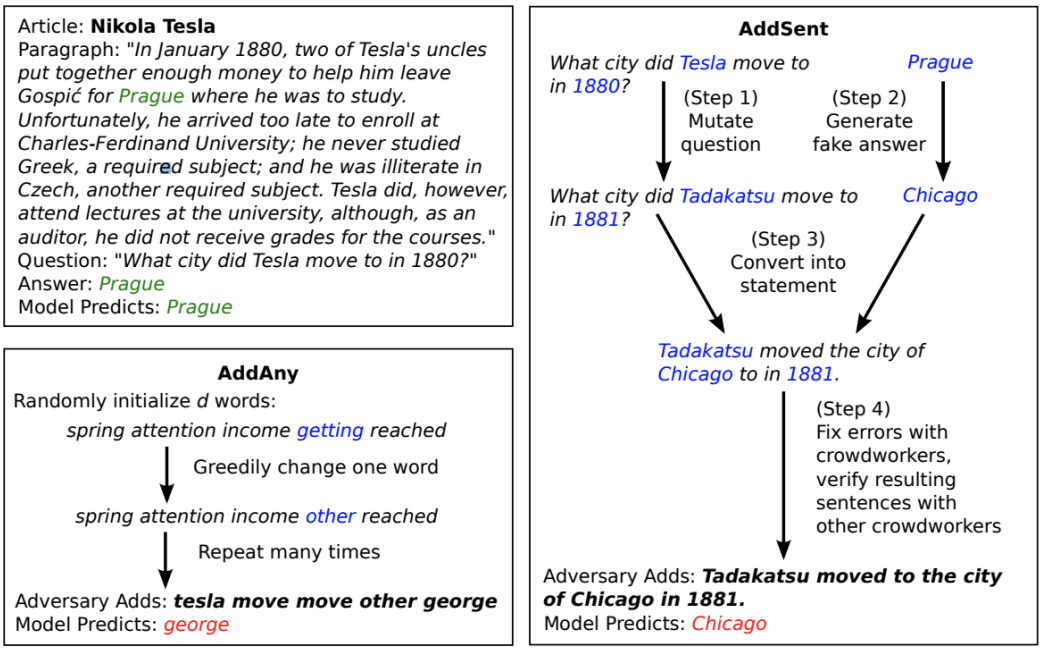

Figure 2: A SQuAD example augmented with AddSent and AddAny. Figure by Jia et al. [2]

The idea of using adversary networks with distinct loss functions within a larger network to enforce a difficult-to-state objective indirectly was popularized by Goodfellow et al. and the Generative Adversarial Network. [1] By training a discriminator to tell whether a generated network is or is not real, and training a generator to fool the discriminator, Goodfellow et al. encode the objective "make this image look real". TTA uses a similar idea for its adversary nets, and uses a near-identical discriminator for its 'judge'.

These methods both succeeded at fooling QA models. Jia et al. tested sixteen published SQuAD models, including BiDAF and Match-LSTM, in both single and ensemble forms. No model was robust to adversarial examples. Models showed an average drop in F1 scores from 75% to 36% for AddSent examples and to 7% for AddAny. While AddAny uses local search against the models, AddSent is a simple model-independent augmentation and still manages to tank performance. This shows these models are very unlikely to be learning much machine comprehension despite their stellar performance; human evaluation by Jia et al. showed that humans are mostly robust to AddSent.

3 Approach

We propose the Tag Team Adversary architecture (TTA) to create adversarial examples usable for data augmentation. TTA can generate valid mutations of existing adversarial contexts that leave them consistent with question/answer pairs already associated with them. TTA works by creating two adversarial networks to mutate given contexts from SQuAD. The adversaries are bidirectional sequence-to-sequence models, pretrained to simply reproduce their input. These adversarial networks take turns generating a mutated context. This mutated context is then passed alongside the question from SQuAD to two distinct QA models, which generate candidate answers. Adversary A is working to fool QA model 2 but not QA model 1, and Adversary B is working to fool QA model 1 but not QA model 2. This two-team competition ensures that the adversaries create difficult adversarial examples, but also that they remain valid (and hence solvable by the ally QA model), since the adversary does not have access to the question or answer and so must maintain the semantic meaning of the context for the ally to succeed. The loss of each adversary is equal to the loss of its ally QA model - λ * the loss of its enemy QA model, where λ is a hyperparameter to shift how much the adversary prioritizes fooling the enemy vs. helping the ally.

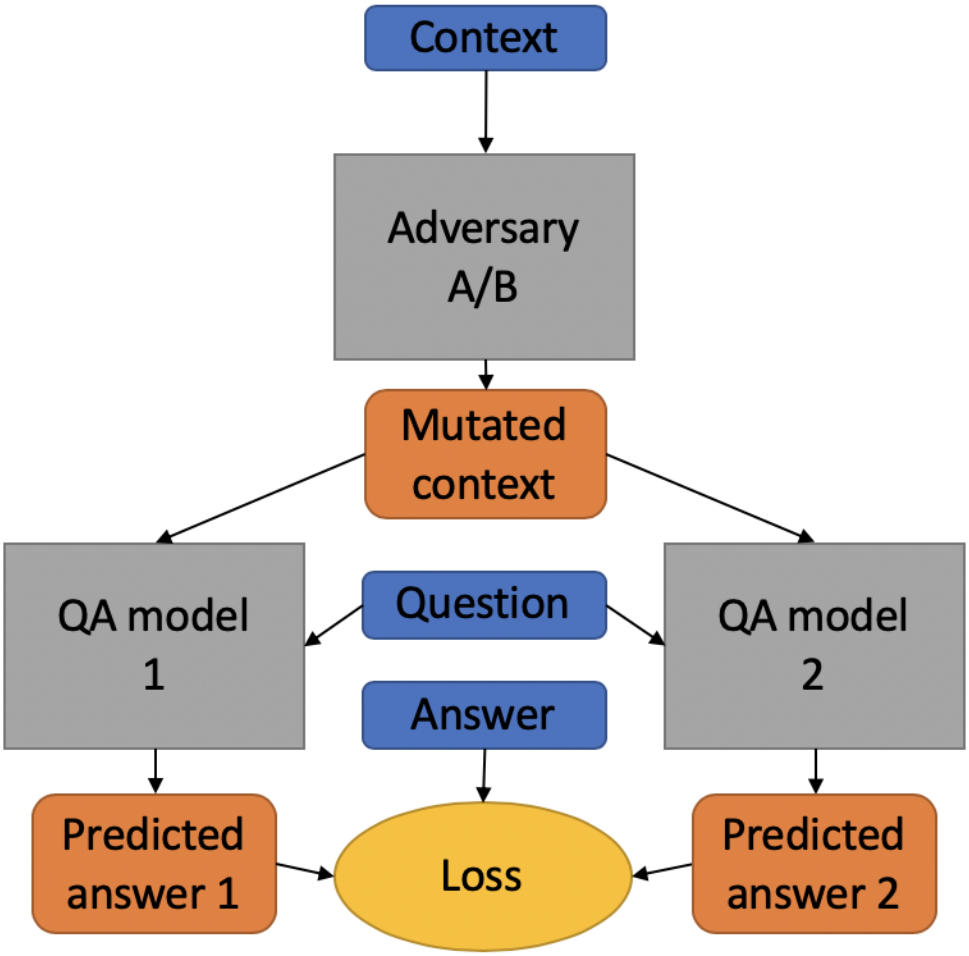

Figure 3: The TTA architecture. The adversary alternates between two models (A and B), each of which is 'teamed' with a different QA model.

Any QA model can be used with this architecture so long as it can be backpropagated through. If TTA is simply used to augment a dataset, then both QA models should be frozen, but they must be distinct from one another to break the symmetry (otherwise the loss for the adversaries will be constant). However, if the QA models are trained alongside the adversaries, additional measures are needed. A 'judge' network presides over the generated contexts in this case to make sure they stay sensible and grammatical and prevent the QA models and adversaries from overfitting to each other. This network is trained as a discriminator to predict whether its input is a true SQuAD context or a mutated context generated by an adversary. The loss function for the adversaries has another component added that penalizes them if they create a context that the judge can recognize as mutated, with a hyperparameter to scale it. This prevents contexts from straying away from valid sentences.

4 Experiments

As previously stated, we focused on augmenting the SQuAD 2.0 dataset [3]. Unfortunately, TTA proved too complex to stably implement for the scope of this project. To evaluate the success of TTA, we would use several approaches. First, using the fixed-QA version of TTA, an augmented dataset would be created. Its usability as training data would be tested by training a standard SQuAD QA model on the augmented data, then testing its performance on SQuAD, AddSent-augmented SQuAD,

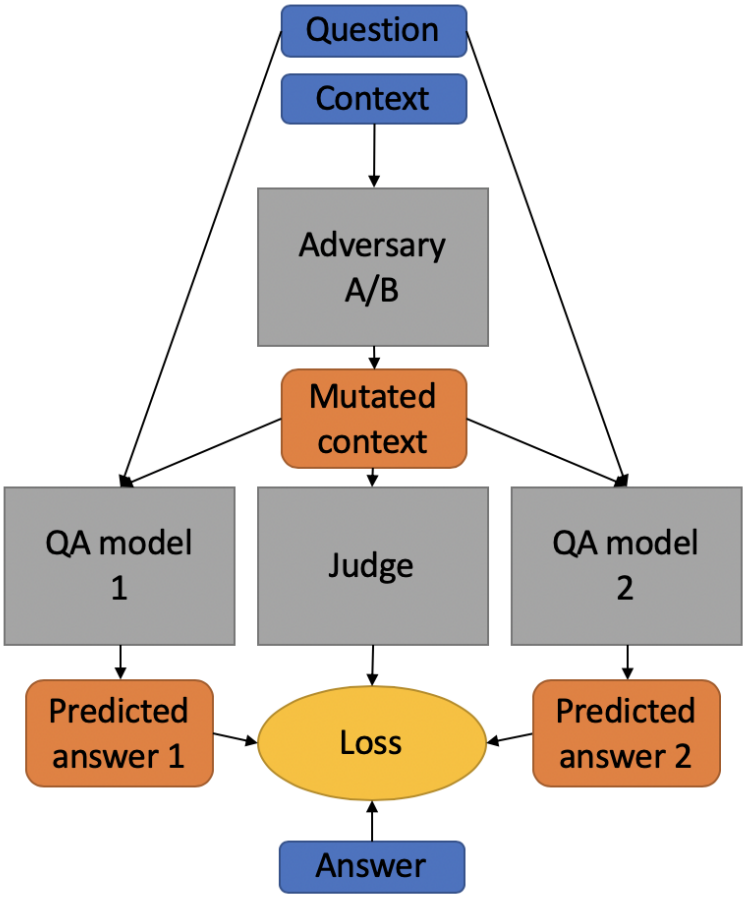

Figure 4: The TTA architecture with the judge discriminator.

and AddAny-augmented SQuAD. Then, a qualitative examination of a random sample of the dataset would check the validity of generated contexts and the diversity and quality of mutations. For the non-QA-fixed version, the QA model would be trained and its performance would be evaluated on SQuAD, AddSent-augmented SQuAD, and AddAny-augmented SQuAD.

5 Analysis

There are many potential ways to augment a context while leaving it valid, including replacements of words with synonyms, insertion of unrelated distractor sentences and clauses, some changes of order of sentences, and more; the types and numbers of different mutation types would be tallied and reported. The number of validity errors in the sample would also be tallied. For the non-QA-fixed version, the differences between the two QA models would be noted. Was different initialization enough to break their symmetry? When they give different answers, is there a consistent trend as to which one gives which answer? Is one susceptible to a different kind of mutation than the other? These questions would be investigated and reported.

6 Conclusion

TTA is an adversarial architecture for adversarial example generation in SQuAD. By relying on a tag-team competition between two pairs of networks (and potentially a judge network), it maintains validity of context while incentivizing creation of difficult mutations.

Future work would focus on simplifying this architecture, as one of its main limitations is its high complexity and parameter-count and resulting instability. For example, a better adversary architecture could help create specific desired types of mutations with far less parameters (for example one that edited the context instead of reproducing a whole new one sequence-to-sequence).

References

[1] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

[2] Robin Jia and Percy Liang. "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *CoRR* abs/1707.07328 (2017). arXiv: 1707.07328. URL: http://arxiv.org/abs/1707.07328.

[3] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.