Applying Question Part-of-Speech Tag Extraction to Generate Adversarial Training Examples for Reading Comprehension Tasks

Collin Matz
Department of Computer Science
University of Texas
cam22838@utexas.edu

Abstract

Dataset artifacts in reading comprehension tasks introduce a challenge for creating robust models: How do we train models that are more robust to these artifacts? To address this concern, we introduce a new methodology for generating adversarial examples for question-answering datasets called STRUCTADV. STRUCTADV uses part-of-speech tag parsing to generate sentences that appear to mimic and have high n-gram overlaps with questions. We evaluate this approach on the ELECTRA-small model (Clark et al., 2020) and find that compared to the base SQuAD dataset (Rajpurkar et al., 2016), ELECTRA-small trained on data augmented by STRUCTADV performed better on exact match and F1 metrics, showing the potential for dataset augmentation with this system.

1 Introduction

With the recent surge in popularity of large-language models (LLMs) over the last few years, developing trust in these systems has become an essential part of their normalization in society. As such, determining causes and developing strategies to mitigate errors associated with LLMs is at the forefront of research in the field of natural language processing. One such type of error is that arising from spurious relations learned by the model via dataset artifacts. These errors can be exploited by showing data to the model that do not contain these artifacts, causing the model to perform below baseline metrics.

In this work, we explore an approach based on the previous work of adversarial text generation for reading comprehension tasks (Jia and Liang, 2017). This previous work explores three novel adversarial generation methods, ADDSENT, ADDANY, and ADDCOMMON. The new approach explored here is based on the ADDANY method as illustrated in Figure 1.

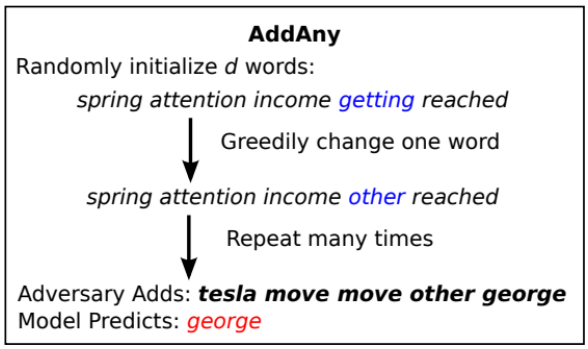

Figure 1: An illustration of the ADDANY adversary (Jia and Liang, 2017).

ADDANY is a useful method for generating adversarial data, but it lacks the independence ADDSENT and ADDCOMMON have from the model. ADDANY requires running a search over the model's answer distributions assuming the model returns a distribution at all. However, it's strength comes from the simplicity of not needing to generate grammatically correct sentences. Our approach aims to marry the strengths of ADDSENT and ADDANY into one simple algorithm.

2 Approach

2.1 ELECTRA-small Evaluation

We begin by evaluating the ELECTRA-small model's performance on two datasets: SQuAD and SQuAD Adversarial[1]. The SQuAD dataset contains 86,000 training instances of question and answer (Q&A) training examples, and 10,570 validation examples. The ELECTRA-small model trained on the SQuAD dataset for three epochs achieves decent metrics for both exact match and F1 score, as shown in Figure 2. However, when evaluated on SQuAD Adversarial, the model performed far worse, decreasing in both metrics by

[1]https://huggingface.co/datasets/stanfordnlp/squad_adversarial

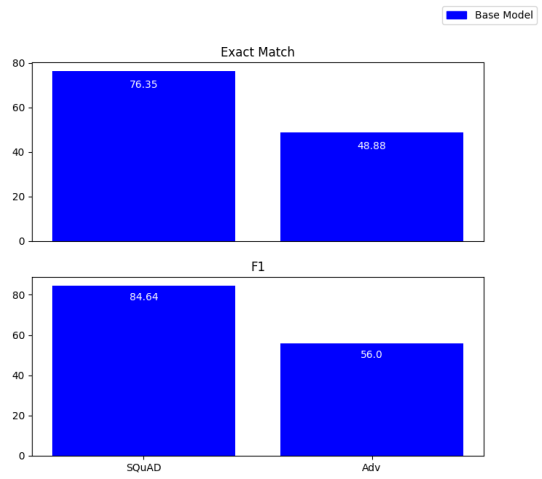

Figure 2: ELECTRA-small exact match and F1 scores on the SQuAD and SQuAD adversarial datasets.

almost 30%. This decrease in performance when switching to the adversarial dataset is highly indicative of the dataset artifacts defined in the previous section being present.

Further inspection of the model's predictions on SQuAD adversarial are indicative of the model looking for n-gram matches between questions and answers. Figure 3 shows a break down of the n-gram overlap percentage between the adversarial sentence and the question found in the SQuAD Adversarial dataset. These findings are also consistent with the findings from Jia and Liang, 2017, where they discussed the impact of an exact n-gram match between question and sentence with model performance.

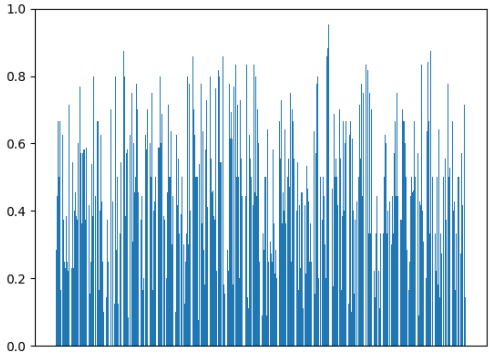

Figure 3: Distribution of n-gram overlap percentages between adversarial sentence and question in SQuAD Adversarial.

2.2 StructAdv

Here, we introduce StructAdv, a method for generating adversarial Q&A examples based on AddAny. AddAny runs a local search over a model's output distribution to determine which sequence of words to insert into a question's context. StructAdv also generates a random string of words to use as adversarial content but does not use the model's output to do so. StructAdv works in three steps for each example of a dataset.

Step 1 is to create a set of predefined sequences of part-of-speech (POS) tags. The POS tags for each sequence are taken from the Penn Treebank collection of POS tags (Marcus et al., 1993). The selected sequences are intended to describe very simple sentence structure that can represent facts. As such, not every POS tag was included as a possible candidate. Table 1 shows the set of POS tag sequences used for StructAdv. At this stage, we also mitigate the discrepancy between POS tag names from Penn Treebank and the Python library spaCy[2] by defining a mapping between equivalent tags.

POS Tag Sequences								
DT	NN	VBZ	JJ					
NNP	VBZ	DT	JJ	NN	IN	NN		
PRP	VBD	DT	NN	TO	VB	PRP		
IN	DT	NN	VBZ	JJ	CC	JJ		
DT	NN	VBD	IN	DT	JJ	NN		
DT	NN	VBD	CD	JJ	NNS			
PRN	NN	VBZ	CD	JJ	NN	IN	DT	NN
DT	NN	IN	CD	NNS	VBP	JJ		
CD	NNS	VBD	IN	DT	JJ	NN		
DT	NN	VBD	PRP	CD	NNS	TO	VB	
PRP	MD	VB	CD	JJ	NN	IN	NN	
CD	JJ	NNS	VBZ	JJ	CC	JJ		

Table 1: List of POS tag sequences used in StructAdv.

Step 2 proceeds in two parts. First, we define a key-value mapping for POS tags and words. The set of keys in this mapping is the unique set of POS tags from the tag sequence set defined in the previous step. The values for these keys are lists that store the parsed words from our question. We pre-populate select lists with words that are useful for maintaining sentence structure in an effort to produce grammatically correct sentences in the case that a question does not contain some POS tag that we expected, as illustrated by Figure 4. Once we have this mapping defined, we sort each word

[2]https://spacy.io/

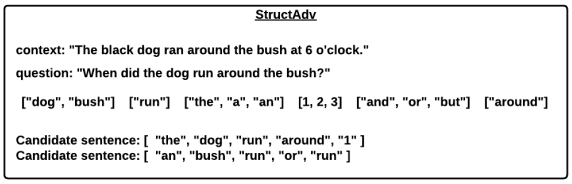

Figure 5: Examples of STRUCTADV generating candidate sentences to inject into a single question context.

of the question into its respective list, based on the POS tag associated with that word.

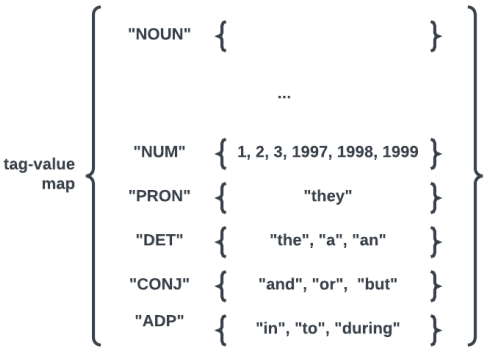

Figure 4: Example of the key-value mappings from POS tags to words. The POS tags here are as defined in spaCy, not Penn Treebank. All pre-defined words are shown.

After both the tag sequence set and tag-word mapping are populated, we move on to Step 3. We randomly select K POS tag sequences from our set. In this work, we set K = number of tag sequences. For each tag sequence in our selection, we attempt to randomly assign words from our tag-word mapping that are associated with the POS tags in the sequence. If we attempt to replace a tag with a word from a list that is empty, we simply skip the tag. After we have generated a candidate sentence for every selected tag sequence, we select the longest candidate sentence under the assumption that, since the most words matched, the quality of this sentence should be higher than the others. The overall process of STRUCTADV is shown in Figure 5. Step 3 is repeated randomly one, two, or three times for each dataset example in order to introduce a wide range of content for the model to see.

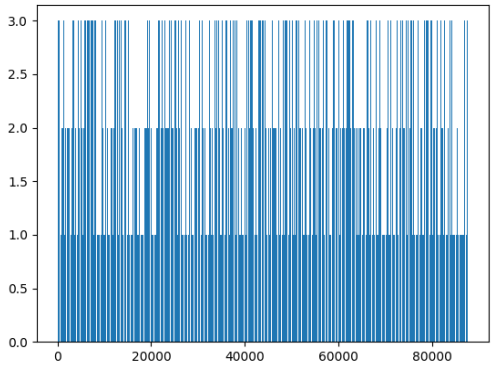

Figure 6: Number of adversarial examples added for each example from the SQuAD dataset.

2.3 Generated Adversarial Examples Analysis

STRUCTADV added 175,147 adversarial examples to the SQuAD dataset, bringing the total number of training examples in the dataset up to 262,747. The distribution of number of training examples added for each example from the original SQuAD dataset is shown in Figure 6.

3 Results

The ELECTRA-small model trained on the STRUCTADV augmented SQuAD dataset performed better on all three evaluation datasets than the model trained on the base SQuAD dataset did, as shown in Figure 7. The high performance of the base model on the STRUCTADV augmented SQuAD is not entirely surprising, as the quality of the generated sentences can vary from random generation.

The most important performance increase was for the SQuAD Adversarial datasets. This dataset contains data augmented with ADDSENT, a higher quality data augmentation than our own STRUCTADV. This performance boost implies that, although our model did not necessarily see grammatically correct sentences in training, it was able to distinguish between true and false contexts at evaluation time, which is in line with ADDANY.

4 Discussion and Future Work

One important highlight at this point is to discuss the varying lengths of datasets. While SQuAD contained 86,000 examples, the augmented SQuAD contained over 260,000 examples. The performance increase seen by the model

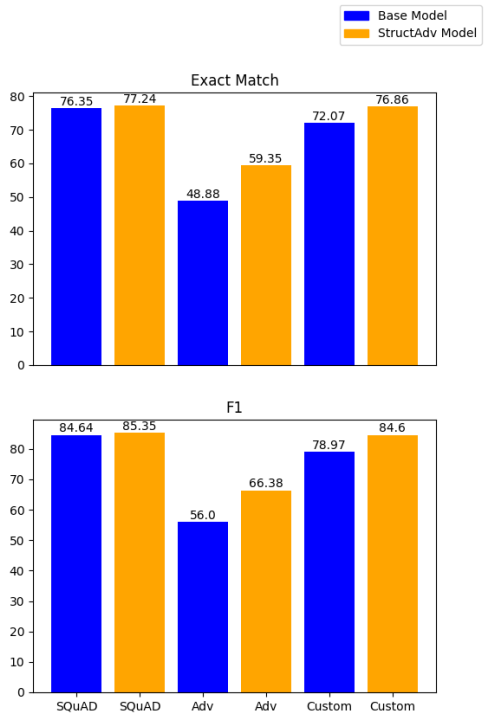

Figure 7: Comparison of ELECTRA-small performance before and after training on a StructAdv augmented dataset. Adv: Adversarial SQuAD, Custom: StructAdv Augmented SQuAD.

trained with StructAdv may attribute a portion of its performance gain to an increase in overall examples, even if duplicated. In essence, the StructAdv model was able to see the training examples more times than the base ELECTRA-small model, which may have given it an unfair advantage. In future work, it would be beneficial to restrict the number of samples in the training set for each model.

Improvements may be made to the StructAdv algorithm that enable higher quality sentences to be generated. Longer POS tag sequences with more complex structure may be added, the number of POS tags allowed by the algorithm may be increased, the generation of sequences may include grammar checking to ensure sentences actually are grammatically correct, etc.

Despite these improvements, we are confident that StructAdv provides a unique way to generate adversarial data that can be useful for researchers. Although other methods for producing adversarial data, such as AddSent, provide higher quality sentences, the ease of use and complete model independence of StructAdv are compelling reasons to continue research into the methodology.

Acknowledgments

I would like to thank Dr. Durrett and the teaching staff for AI 338 for providing the starter code and resources to perform this research task. Natural language processing can seem dauntingly complex, but the staff of this course provided an excellent overview of the field in an easily understandable and fun context!

(Jia and Liang, 2017) (Rajpurkar et al., 2016) (Clark et al., 2020) (Marcus et al., 1993)

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.