**Generating Adversarial Examples for Question Answering**

**Analysis of the Failure of Microsoft's Twitter Chatbot Tay**

A Thesis Prospectus
In STS 4500
Presented to
The Faculty of the
School of Engineering and Applied Science
University of Virginia
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By
Kevin Ivey

October 27, 2022

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISORS

Benjamin Laugelli, Department of Engineering and Society
Yanjun Qi, Department of Computer Science and Affiliated Faculty of Center for Public Health Genomics

## Introduction

Artificial intelligence and machine learning are playing an increasing role in daily life, from virtual assistants like Alexa to self-driving cars like Tesla. However, such machine learning models are susceptible to so-called adversarial attacks that modify the input to the model in a way imperceptible to humans, yet drastically change the output of the model. I will improve on existing "Question-answering" (QA) adversarial attacks by developing an algorithm to automatically attack a QA model without human intervention, thus allowing the solution to scale across domains and models. This will allow researchers to more easily test their models against adversarial attacks so as to be more robust.

However, there are important social factors that must be considered during the development and release of such an algorithm. To better understand these factors, I will investigate the non-technical factors, such as the lack of a release schedule, that contributed to the failure of Tay, a Twitter chat bot developed by Microsoft. By only considering the technical factors of developing an adversarial attack for QA models, and ignoring the non-technical factors, the issue is only partially resolved.

Therefore, given that the development of a QA adversarial attack is sociotechnical, consideration must be given to both the social and technical aspects. In what follows, I propose a technical project that seeks to develop an adversarial attack for QA models without human intervention. Additionally, I will examine the case study of Tay to better understand the factors that led to the failure of a real-world machine learning model.

## Technical Project Proposal

Natural language processing is a field of deep learning which seeks to learn information directly from human speech. A common problem in natural language processing is QA – a

format of problems with a non-trivial question that must be understood to provide the corresponding answer (Gardner et al., 2019). QA is commonly used in systems that answer human-posed questions with the expectation of an answer, such as virtual assistants (Gardner et al., 2019). However, like many deep learning models, QA models have been shown to be brittle to adversarial attacks, which perturb correctly classified samples causing the model to misclassify the sample (Alzantot et al., 2018). Such attacks may have no meaningful semantic or syntactic changes from the unperturbed sample, so that a human would not be fooled by the changes. Previous work in generating adversarial examples for QA models has been conducted, but there is no accepted standard on how a successful attack is defined.

Researchers at Stanford have developed adversarial attacks for QA Models by adding "distracting sentences" to the input paragraph that do not contradict the correct answer nor confuse humans. These adversarial sentences are generated in two different ways. The first alters the question to be semantically similar to the original question, creates an answer for the generated question, combines the two into the declarative form, and then syntactically fixes the sentences and filters them via human annotators (Jia & Liang, 2019). The second simply randomly initializes a sentence of common words and then randomly seeks changes to lower the model's confidence in its answer. In both cases, an attack is deemed successful if the model generates a new answer when the distracting sentence is added to the context. However, the use of human annotators in the first approach severely limits the usefulness of such an attack, as it can not be automatically done by the computer, and the second approach has no guarantees that the generated sentence does not contradict the answer or that it is semantically or syntactically correct. Similarly, researchers at Oregon State University developed a method for attacking QA models that relies directly on human input (Rahurkar et al, 2020). Users with a background in

deep learning were given a context paragraph, the corresponding question, and the answer, and were asked to write a paragraph with the goal of fooling the model. A successful attack was deemed as any attack where the original answer did not match the newer answer and that the adversarial paragraph preserved the semantics of the original context, as judged by humans. While these attacks were successful, the reliance upon humans severely limits the ability for such an approach to scale.

This technical project aims to design an adversarial attack for QA models without the use of humans and to integrate the work into the existing TextAttack library. TextAttack is a Python library for "adversarial attacks, data augmentation, and adversarial training in NLP" (Morris et al, 2020). The design for generating such adversarial attacks is still to come, but will likely involve a combination of word-level perturbations, sentence-level additions, and similarity metrics to ensure that the resulting example is semantically and syntactically similar to the original example. Note that special consideration must be taken to not alter the question, as then the original answer is no longer valid, or the answer, as then the original question is no longer valid.

Training and testing data will be obtained from the SQuAD dataset. The SQuAD dataset contains over 100,000 questions created from Wikipedia articles "where the answer to each question is a segment of text from the corresponding reading passage" (Rajpurkar et al., 2016). SQuAD was chosen as the dataset as it is well-curated, covers a wide range of topics, and is large in size. This dataset will be incorporated into TextAttack to allow for users to easily generate adversarial examples. To evaluate the performance of the attack, the percentage of successful attacks will be calculated over a randomly chosen subset of the dataset.

**STS Project Proposal**

On March 23, 2016, Microsoft released a chatbot, Tay, on Twitter that was supposed to emulate a teenage girl (Schwartz, 2019). According to Microsoft, Tay was "created for 18- to 24-year-olds in the U.S. for entertainment purposes" (Lee, 2016). Initially, Tay was given a baseline behavior by being trained on a dataset containing material written by professional comedians (Schwartz, 2019). However, the goal was for Tay to learn from its online interactions and emulate those behaviors in subsequent conversations. Through the duration of its lifespan, Tay's interactions with Twitter users caused it to rapidly devolve. A coordinated attack by users on the websites Reddit and 4Chan caused Tay to tweet "wildly inappropriate and reprehensible words and images" (Lee, 2016). By the time Microsoft brough Tay offline, a mere 16 hours after its launch, Tay managed to have over 93,000 tweets containing racist, sexist, and abusive language (Bridge et al. 2021). While Microsoft apologized for the incident, stating that while they had prepared for abuse from users, they "had made a critical oversight for this specific attack" (Lee, 2016).

The failure of Tay is commonly attributed to its naive "repeat-after-me" learning algorithm that did not filter out potentially sensitive subjects as well as the targeted attack by users to force Tay to learn unwanted behavior (Vorsino, 2021). While these factors did play a role in Tay's failure, this view overlooks how the complete lack of an adequate release plan led to the failure of Tay. For example, Tay immediately underwent a full public release, allowing anyone with a Twitter account to interact with the software, rather than following a small initial release that gradually grew in scope. If we continue to only see the failure of Tay as a failure in the software algorithm used and a failure of human online behavior, we will not be able to understand how the release schedule of a software can influence the success of real-world

machine learning projects. This will potentially lead to similar failures in the future that are independent of the quality of software engineering and the user base.

I argue that the poor implementation of Tay from an algorithmic perspective and the morally questionable interactions with Twitter users coupled with an underdeveloped release plan led to its failure. In order to analyze these factors, I will use Actor-Network theory, which seeks to identify a network builder as the primary actor and then follows the actors, both technical and non-technical, that contribute to the network builder's ultimate goal (Cressman, 2009). Two additional concepts are needed for the application of Actor-Network theory: translation and punctualization. Translation specifies how these actors are related in the sociotechnical network and punctualization refers to the process of putting an actor-network into a black box and considering it as a single actor in another actor-network. Applying this idea, I will analyze how Microsoft, as the network-builder, formed a network with the goal of a successful Twitter chatbot, and how it rapidly dissolved. Given that the actual algorithm of Tay is private, it must be punctalized within this network. By analyzing the network, I will gain a better understanding of the technical and non-technical actors that must be considered when designing machine learning models that learn directly from their end users. For this analysis, I will use evidence from Microsoft's press releases on Tay, discussion boards on websites such as 4Chan that discuss the attacks on Tay, and the actual Tweets from Tay itself.

## Conclusion

The result of the technical problem discussed will be a design and implementation in TextAttack of an adversarial attack for QA models that generates adversarial examples without the use of humans. The STS research paper will determine why the Microsoft artificial intelligence Twitter chatbot, Tay, failed within 16 hours of its launch. I will use Actor-Network

theory to characterize how human factors, such as the Twitter users, and non-human factors, such as the workings of the model, led to the rapid decay of Tay and forced Microsoft to bring Tay offline. Together, the technical report will address the issue of generating adversarial examples in QA models, and the STS report will analyze the specifics of how Tay, a real-world machine learning model, failed due to the presence of adversarial examples. The combined results will serve to provide the natural language processing community with a new adversarial attack and a sociotechnical analysis of why such attacks are relevant.

## References


Alzantot, M., Sharma, Y., Elgohary, A., Ho, B., Srivastava, M., & Chang, K. (2018). Generating natural language adversarial examples. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2890-2896. https://dx.doi.org/10.18653/v1/D18-1316

Bridge, O., Raper, R., Strong, N., & Nugent, S. (2021). Modelling a socialised chatbot using trust development in children: Lessons learnt from Tay. *Cognitive Computation and Systems*, *3*(2), 100-108. https://doi.org/10.1049/ccs2.12019

Cressman, D. (2009, April). *A brief overview of actor-network theory: Punctualization, heterogeneous engineering & translation*. https://summit.sfu.ca/item/13593

Gardner, M., Berant, J., Hajishirzi, H., Talmor, A., & Min, S. (2019). Q*uestion answering is a format; when is it useful?*. arXiv. https://arxiv.org/abs/1909.11291

Jia, R., & Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing,* 2021-2031. https://dx.doi.org/10.18653/v1/D17-1215

Lee, P. (2016, March 25). *Learning from Tay's introduction.* Microsoft. https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/

Morris, J., Lifland, E., Yoo, J., Grigsby, J., Jin, D., & Qi, Y. (2020). TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations,* 119-126. https://dx.doi.org/10.18653/v1/2020.emnlp-demos.16

Rahurkar, P., Olson, M., & Tadepalli, P. (2020). Human adversarial QA: Did the model understand the paragraph?. *NeurIPS 2020 Workshop on Human And Model in the Loop Evaluation and Training Strategies*, https://openreview.net/forum?id=57NC-S7o4Aw

Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,* 2383-2392. https://dx.doi.org/10.18653/v1/D16-1264

Schwartz, O. (2019, November 25). *In 2016, Microsoft's racist chatbot revealed the dangers of online conversation*. IEEE Spectrum. https://spectrum.ieee.org/in-2016-microsofts-racist-chatbot-revealed-the-dangers-of-online-conversation

Vorsino, Z. (2021). Chatbots, gender, and race on web 2.0 platforms: Tay.AI as monstrous femininity and abject whiteness. *Signs: Journal of Women in Culture and Society*, *47*(1), 105-127. https://doi.org/10.1086/715227