TuringBox: An Experimental Platform for the Evaluation of AI Systems

Ziv Epstein*, **Blakeley H. Payne***, **Judy Hanwen Shen**, **Casey Jisoo Hong**, **Bjarke Felbo**, **Abhimanyu Dubey**, **Matthew Groh**, **Nick Obradovich**, **Manuel Cebrian**, **Iyad Rahwan**,
MIT Media Lab
{cebrian, irahwan}@mit.edu

Abstract

We introduce TuringBox, a platform to democratize the study of AI. On one side of the platform, AI contributors upload existing and novel algorithms to be studied scientifically by others. On the other side, AI examiners develop and post machine intelligence tasks to evaluate and characterize the outputs of algorithms. We outline the architecture of such a platform, and describe two interactive case studies of algorithmic auditing on the platform.

1 Introduction and Motivation

As the proliferation of artificial intelligence continues, algorithmic bias within AI systems has become a popular topic of scientific study [O'Neil, 2017; Friedman and Nissenbaum, 1996; Sweeney, 2013; Hannak *et al.*, 2014]. Despite growing interest in both the academic and public spheres, researchers who wish to study AI in general – and algorithmic bias in particular – face several challenges, which are outlined below:

1. **Reproducibility:** The first challenge researchers face is the difficulty in replicating AI systems of interest. A recent study showed only 6% of 400 authors at two top AI conferences shared their new algorithm's code, implying most state of the art algorithms are unable to be replicated for further examination [Hutson, 2018].
2. **Accessibility:** The second challenge researchers face is the increasing opacity of AI systems. Due to the use of proprietary training data or even the proprietary nature of many commercial algorithms, it is difficult or even impossible for computer scientists to access the underlying models of a system.
3. **Efficiency:** The third challenge researchers face is a problem of inefficiency. Due to the difficulty to reproduce or access important AI systems, researchers who wish to audit AI systems are often relegated to studying a small number of systems (for example, one computer vision API) as opposed to a class of systems (such as all commercial computer vision APIs).

We introduce TuringBox as a mechanism to face these challenges by allowing researchers to evaluate the output of AI algorithms in a controlled, procedural way. TuringBox also provides a standardized benchmarking tool in a cloud environment.

*Equal contribution.

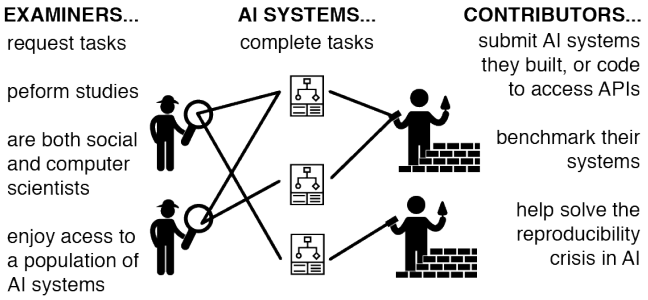

Figure 1: A schematic of the platform for AI.

2 System Overview

The schematic of the TuringBox framework is shown in Figure 1. On one side of the platform, AI *contributors* upload algorithms in various forms. First, they will upload implementations of existing AI systems. Second, they will upload their own novel AI systems. Third, contributors will upload scripts they developed which access APIs. To incentivize uploads, TuringBox eases the benchmarking process of conducting AI research by automatically comparing a contributor's uploaded algorithm to other algorithms on the platform with respect to accuracy, fairness, or other qualities determined by the examiners. For each upload, contributors will gain reputation points on the platform as a function of its performance in these categories.

On the other side of the platform, AI *examiners* investigate the output of AI systems. As with previous investigations, these AI system are studied agnostic to their underlying system architecture, and are thus represented only by their inputs and outputs [Larson *et al.*, 2016; 2017; Buolamwini and Gebru, 2018].

These investigations may take one of two forms. First, an examiner can browse the platform for existing algorithms to be studied. An examiner will provide input data to the selected algorithms, which are specified by structured input and output, and then analyze the output of the algorithms. Second, an examiner can post a *machine intelligence task*, which

calls for the creation of new algorithms by the contributor side of the platform. Examiners also receive reputation points as a function of the quality of their studies, as determined by their peers as well as other contributors.

2.1 Technical Architecture

Contributors upload Python or Javascript files to the platform, which integrates them into the codebase. The platform ensures the file represents a well-structured algorithm (defined as a canonical input output scheme) and is verified upon upload by feeding a dummy input data source (in the input data scheme) into the algorithm and making sure the code runs correctly.

Examiners interact with the platform via a intuitive GUI. Using the interface, they can specify the algorithms they want to access, and the dataset they wish to input. The request is then sent to the server, which returns results after server-side computation has terminated.

The system uses a virtualization technology to enforce a fine-grained security policy during computation. This ensures that the uploaded algorithms are performing the advertised computation and are not abusing the cloud computation environment.

3 Demonstration

In this demonstration, users will have the opportunity to perform two experiments on the TuringBox platform from both the contributor and examiner perspectives. These case studies have been carefully selected to represent two important domains in artificial intelligence: computer vision and natural language processing.

For each case study, we will provide the user with AI systems and datasets to complete the experiment. As a contributor, the user will be able to upload an AI system and use TuringBox's automatic benchmarking tool. As an examiner, users will be able to upload a customized dataset to the platform and select which algorithms to test. Users will then be able to see the output for each algorithm selected for the specified input data and have an opportunity to examine the results. We describe each case study below.

3.1 Case Study 1: Disparate Treatment by Body Type in Commercial Computer Vision APIs

Recently, computer vision systems have been shown to exhibit racial and gender biases [Buolamwini and Gebru, 2018]. In this demonstration, users will examine and quantify bias in commercial computer vision systems with respect to a previously unexamined source of bias: body type.

In order to quantify bias, users will be able to select which commercial computer vision algorithms they would like to test and what outputs (for example: text labels, not-safe-forwork scores, demographic information, etc.) each algorithm should return per image. The user will also specify which data set they would like to input to their selected algorithms. We will provide a dataset of images labeled for body type as a default input data set, but also provide functionality for the user to upload their own, or use the webcam in real time.

The user can then analyze each algorithm as well as the entire class of commercial computer vision algorithms for evidence of disparate treatment between body types.

3.2 Case Study 2: Sentiment Classification Bias by Gender, Ethnicity, and Age in NLP systems

Recent research has highlighted corpora and dataset biases learned by NLP models and the troubling potential impact of biased NLP systems [Bolukbasi *et al.*, 2016; Zhao *et al.*, 2017; Blodgett *et al.*, 2016]. While most existing studies focus on characterizing and removing bias from subcomponents of NLP models, there has only been limited research of the degree to which end-to-end real-world systems exhibit problematic biases.

We provide a demonstration that allows users to easily investigate the biases of real-world systems for sentiment classification. Users enter keywords related to demographic attributes (e.g. gender, ethnicity, age) and examine the biases that arise across different APIs. Our demonstration quantifies the amount of bias related to the keywords by measuring differences in sentiment when doing keyword replacement (e.g. replacing 'white' with 'black'). The keyword replacements are done across a large text corpus to obtain robust measures of bias. Through this demonstration, users can easily test and characterize the output of black box NLP algorithms.

3.3 Demonstration of Algorithm Sandbox for Contributors

For both of the two case studies above, we will also demonstrate our system's ability to integrate new algorithms into the underlying codebase. This integration protocol will enforce the following constraints: first, that uploaded algorithms are well-structured for the tasks they claim to complete, and second, that the algorithms compute under the enforcement of a fine-grained security policy. For this demonstration, participants will upload local files corresponding to pre-trained neural networks. In real time, they will be able to visualize the system verifying and integrating the algorithm, as well as the system benchmarking it against other algorithms on the platform after it has been integrated.

4 Conclusion

Methods for comparing similar algorithms – both in the contexts of algorithmic auditing and benchmarking – has been stymied by a lack of tools and protocols. In this demonstration, we showcase a novel cloud-based framework that offers both methodological and technological contributions. TuringBox offers a standardized methodology of considering algorithms only by their inputs and outputs, thus allowing comparison across a broad class of models. TuringBox employs a virtualization environment that provides a robust solution to security concerns as well as cloud computation at scale. This methodology as well as the underlying technology address the challenges of reproducibility, accessibility, and efficiency. In addition to these technological and methodological contributions, our demonstration offers two novel and highly interactive case studies in the field of algorithmic bias.

References

[Blodgett *et al.*, 2016] Su Lin Blodgett, Lisa Green, and Brendan T. O'Connor. Demographic dialectal variation in social media: A case study of african-american english. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1119–1130, 2016.

[Bolukbasi *et al.*, 2016] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in Neural Information Processing Systems*, pages 4349–4357, 2016.

[Buolamwini and Gebru, 2018] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency*, pages 77–91, 2018.

[Friedman and Nissenbaum, 1996] Batya Friedman and Helen Nissenbaum. Bias in computer systems. *ACM Trans. Inf. Syst.*, 14(3):330–347, July 1996.

[Hannak *et al.*, 2014] Aniko Hannak, Gary Soeller, David Lazer, Alan Mislove, and Christo Wilson. Measuring Price Discrimination and Steering on E-commerce Web Sites. In *Proceedings of the ACM Internet Measurement Conference (IMC'14)*, Vancouver, Canada, Nov 2014.

[Hutson, 2018] Matthew Hutson. Artificial intelligence faces reproducibility crisis, 2018.

[Larson *et al.*, 2016] J Larson, S Mattu, L Kirchner, and J Angwin. How we analyzed the compas recidivism algorithm. propublica, 2016.

[Larson *et al.*, 2017] J Larson, J Angwin, L Kirchner, and S Mattu. How we examined racial discrimination in auto insurance prices. propublica, 2017.

[O'Neil, 2017] Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books, 2017.

[Sweeney, 2013] Latanya Sweeney. Discrimination in online ad delivery. *Queue*, 11(3):10, 2013.

[Zhao *et al.*, 2017] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2979–2989, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.