Robustness Gym: Unifying the NLP Evaluation Landscape

Karan Goel[*1], Nazneen Rajani[*2], Jesse Vig[2],
Samson Tan[2], Jason Wu[2], Stephan Zheng[2], Caiming Xiong[2], Mohit Bansal[3], and Christopher Ré[1]

[1]Stanford University
[2]Salesforce Research
[3]UNC-Chapel Hill
[1]{kgoel,chrismre}@cs.stanford.edu
[2]{nazneen.rajani,jvig}@salesforce.com
[3]{mbansal}@cs.unc.edu

Abstract

Despite impressive performance on standard benchmarks, deep neural networks are often brittle when deployed in real-world systems. Consequently, recent research has focused on testing the robustness of such models, resulting in a diverse set of evaluation methodologies ranging from adversarial attacks to rule-based data transformations. In this work, we identify challenges with evaluating NLP systems and propose a solution in the form of Robustness Gym (ℝ𝔾),[1] a simple and extensible evaluation toolkit that unifies 4 standard evaluation paradigms: subpopulations, transformations, evaluation sets, and adversarial attacks. By providing a common platform for evaluation, Robustness Gym enables practitioners to compare results from all 4 evaluation paradigms with just a few clicks, and to easily develop and share novel evaluation methods using a built-in set of abstractions. To validate Robustness Gym's utility to practitioners, we conducted a real-world case study with a sentiment-modeling team, revealing performance degradations of 18%+. To verify that Robustness Gym can aid novel research analyses, we perform the first study of state-of-the-art commercial and academic named entity linking (NEL) systems, as well as a fine-grained analysis of state-of-the-art summarization models. For NEL, commercial systems struggle to link rare entities and lag their academic counterparts by 10%+, while state-of-the-art summarization models struggle on examples that require abstraction and distillation, degrading by 9%+.

1 Introduction

Advances in natural language processing (NLP) have led to models that achieve high accuracy when train and test data are independent and identically distributed (i.i.d.). However, analyses suggest that these models are not robust to data corruptions [Belinkov and Bisk, 2018], distribution shifts [Hendrycks et al., 2020, Miller et al., 2020], or harmful data manipulations [Jia and Liang, 2017], and they may rely on spurious patterns [McCoy et al., 2019] for prediction. In practice, these vulnerabilities limit successful generalization to unseen data and hinder deployment of trustworthy systems. A consequence of this is the proliferation of public-use systems that were later revealed to be systematically biased [Hamilton, 2018, 2020, Hao, 2019, Kayser-Bril, 2020, Knight, 2019, Stuart-Ulin, 2018], such as recruiting tools biased against women.

---

[*]Equal contribution. KG, NR, and JV made significant contributions.
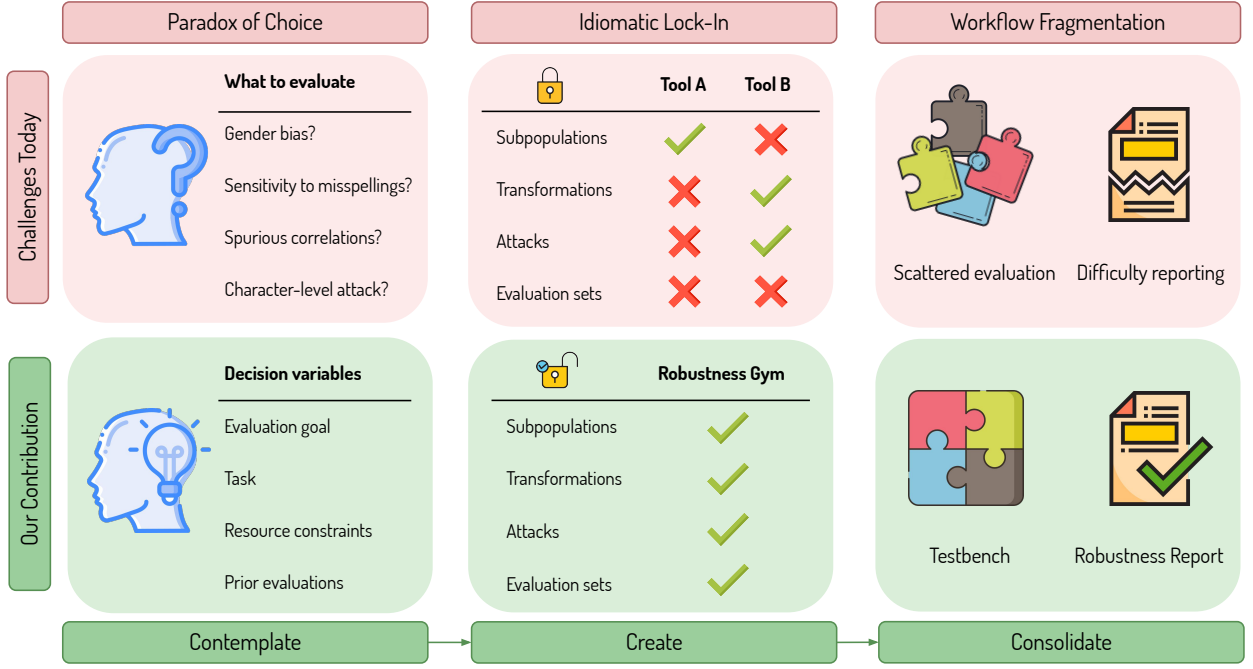[1] https://robustnessgym.com/

Figure 1: (top) challenges faced by practitioners evaluating their models, and (bottom) the Contemplate → Create → Consolidate evaluation loop uses Robustness Gym to address these challenges.

While researchers and practitioners are aware of these problems, it remains common practice to report performance solely on i.i.d. test data. Ideally, evaluation would continually test a model's capabilities on examples that it is likely to see when deployed, rather than produce a static artifact of the model's performance. This process can be complex for practitioners, since there is no systematic method to prioritize what to evaluate, which evaluation methods to apply, and how to leverage or share the findings of previous evaluations. In summary, current evaluation practices face three challenges (Figure 1, top):

1. **Paradox of choice (Section 2.3).** Even without writing a single line of code, practitioners are often confused about what evaluations to run. This stems from a lack of prior research on how to evaluate, especially guidance that is sensitive to the practitioner's task, evaluation needs, and resource constraints. This confusion becomes a key challenge when treating evaluation as a continual process, since prior evaluation attempts and findings should influence a practitioner's future evaluation decisions.

2. **Idiomatic lock-in (Section 2.4).** When determining the evaluation they want to run, practitioners must also choose an appropriate tool. We identify 4 distinct evaluation idioms supported by existing tools and research—subpopulations, transformations, adversarial attacks and evaluation sets. Each tool uses bespoke abstractions to serve a subset of these idioms (e.g. adversarial attacks on words), requiring users to glue together multiple tools to perform a broad evaluation that mixes idioms.

3. **Workflow fragmentation (Section 2.5).** As practitioners evaluate, they need to save progress, report findings and collaborate to understand model behavior. Existing solutions to save progress are tool- and idiom-specific, lack versioning and provide limited support for sharing. Existing reporting templates [Mitchell et al., 2019] are free-form, and have not successfully incentivized users to report findings e.g. we find only 6% of Huggingface [Wolf et al., 2020] models have any evaluation information reported.

In response to these challenges, we introduce Robustness Gym ($\mathbb{RG}$), a simple, extensible, and unified toolkit for evaluating robustness and sharing findings. We embed $\mathbb{RG}$ in a new paradigm for continually evaluating

models: the Contemplate → Create → Consolidate evaluation loop (Figure 1, bottom). In this loop, we envision that researchers and practitioners will:

1. **Contemplate (Section 3.1)** what evaluation to run next. We provide guidance to practitioners on how key variables—their task, evaluation needs and resource constraints—can help prioritize which evaluation to run next. We describe the influence of the task via the task schema and known prior evaluations, needs such as testing generalization, bias, or security, and constraints such as expertise, access to compute, and human resources. We describe how these decisions could evolve as more evaluations are conducted.

2. **Create (Section 3.2)** *slices* of data in $\mathbb{RG}$, where each slice defines a collection of examples for evaluation, built using one or a combination of evaluation idioms. $\mathbb{RG}$ supports users in a simple two-stage workflow, separating the storage of side information about examples (CachedOperation), away from the nuts and bolts of programmatically building slices across all 4 idioms using this information (SliceBuilder). This workflow allows users to quickly implement new ideas, minimize boilerplate code, and seamlessly integrate existing tools.

3. **Consolidate (Section 3.3)** slices and findings for faster iteration and community sharing. $\mathbb{RG}$ users can organize slices into a TestBench that can be versioned and shared, allowing a community of users to collaboratively build benchmarks and track progress. For reporting, $\mathbb{RG}$ provides standard and custom robustness reports that can be auto-generated from testbenches and included in paper appendices.

To demonstrate how this process benefits practitioners, we outline how 3 users with varying expertise can evaluate a natural language inference (NLI) model using $\mathbb{RG}$. Novice users (Section 4.1) can rely on predefined testbenches for direct evaluation. Intermediate users (Section 4.2) can create new slices using SliceBuilders available in $\mathbb{RG}$, and then construct their own testbenches. Finally, advanced users (Section 4.3) can use their expertise to add custom slices. All of these users can generate a shareable Robustness Report (Figure 4).

We validate the Contemplate → Create → Consolidate process using a 3-hour case study (Section 4.4) with Salesforce's commercial sentiment modeling team. The team's main goal was to measure the bias of their model (*contemplate*). We tested their system on 172 slices spanning 3 evaluation idioms, finding performance degradation on 12 slices of up to 18% (*create*). Finally, we generated a single testbench and robustness report for the team, summarizing these findings (*consolidate*). A post-study questionnaire found that the team considered $\mathbb{RG}$ to be easy to use, and indicated that they are very likely to integrate $\mathbb{RG}$ into their workflow.

Robustness Gym can be used to conduct new research analyses with ease. To validate this, we conduct the first study of academic and commercially available named entity linking (NEL) systems, as well as a study of the fine-grained performance of summarization models.

1. **Named Entity Linking (Section 5.1)** We compare commercial APIs from MICROSOFT, GOOGLE and AMAZON to open-source systems BOOTLEG, WAT and REL across 2 benchmark datasets (WIKIPEDIA, AIDA). We find that commercial systems struggle to link rare or less popular entities, are sensitive to entity capitalization and often ignore contextual cues when making predictions. MICROSOFT outperforms other commercial systems, while BOOTLEG displays the most consistent performance across a variety of slices. On AIDA, we find that a simple heuristic NEL method outperforms all commercial systems.

2. **Summarization (Section 5.2).** We propose and implement 5 subpopulations that capture summary abstractiveness, content distillation, information dispersion [Grusky et al., 2018], positional bias, and information reordering [Kedzie et al., 2018]. We compare 7 models on the CNN-DailyMail dataset across these subpopulations. All models struggle on summaries that discard content, require higher amounts of abstraction or contain more entities. Surprisingly, models with very different prediction mechanisms make similar errors, suggesting that existing metrics are unable to capture meaningful performance differences.

Robustness Gym continues to be under active development, and we welcome feedback and suggestions from the community.

2 Current Evaluation Challenges

We describe the problem of evaluating machine learning models, motivate a shift towards continual evaluation, lay out 3 challenges today in making this shift, and situate this in the context of existing tools and work.

2.1 Model Evaluation

Generally, validation of a trained model for a task consists of evaluating the model on a set of examples that are drawn from the training distribution [Bishop, 2006]. Assuming identical train and test distributions (i.i.d. data), validation performance estimates the model's performance at test time.

In practice, the train and test distributions can be different [Taori et al., 2020]. This distributional shift is a natural consequence of changing real-world conditions and evolving expectations of the model's capabilities. For instance, a model that detects entities in news articles will become outdated as new entities emerge over time. Standard validation overestimates true performance in this case, since it does not preempt performance degradation due to changing conditions. Researchers and practitioners in these circumstances often rely on intuition and an understanding of their domain to create evaluations and perform model selection.

Recent work suggests that models often exploit spurious correlations when making predictions [McCoy et al., 2019] and are not robust when evaluation moves beyond i.i.d. data [Hendrycks et al., 2019]. This lack of robustness makes models susceptible to failure under even the slightest distributional shifts [Miller et al., 2020], or when deployed [Stuart-Ulin, 2018]. Systematic and continual evaluation is necessary to understand the model's limitations, and as we discuss next, standard evaluation practices often fall short.

2.2 Towards Continual Evaluation

We view evaluation as a continual process from the practitioner's perspective. In practice, constant re-evaluation is necessary in order to assess if a model should continue to be used in light of new information about its limitations. By contrast, traditional evaluation addresses challenges that relate to generating a static artifact of the model's performance (e.g., computing an aggregate measure of performance on a test set [Bishop, 2006] or more fine-grained measures using a suite of tests [Ribeiro et al., 2020].

Prior work on the construction of evolving benchmarks [Kiela et al., 2020] introduced dynamic evaluation, allowing a community of practitioners to collaboratively build challenging benchmarks. We focus here on the individual's perspective, and how to equip them with tools that support the continual evaluation paradigm. This raises a fresh set of challenges that are not traditionally addressed by standard evaluation.

Next, we identify three of these challenges—the paradox of choice, idiomatic lock-in and workflow fragmentation—and highlight how existing tools and research fall short of addressing them.

2.3 Challenge 1: The Paradox of Choice

Ideal. Given a practitioner's task, needs, constraints and prior knowledge, give them guidance on what evaluation to run next.

Challenges. Evaluation is a complex, unstructured process for practitioners, since it can be confusing to choose what evaluation to run next. These decisions are frequent in continual evaluation. Here, practitioners accumulate an understanding of their model's limitations and manage changing needs, which should (ideally)

Evaluation Idiom	Tools Available	Research Literature (focusing on NLI)
Subpopulations	Snorkel [Ratner et al., 2017], Errudite [Wu et al., 2019]	Hard/easy sets [Gururangan et al., 2018] Compositional-sensitivity [Nie et al., 2019]
Transformations	NLPAug [Ma, 2019]	Counterfactuals [Kaushik et al., 2019], Stress test [Naik et al., 2018], Bias factors [Sanchez et al., 2018], Verb veridicality [Ross and Pavlick, 2019]
Attacks	TextAttack [Morris et al., 2020], OpenAttack [Zeng et al., 2020] Dynabench [Kiela et al., 2020]	Universal Adversarial Triggers [Wallace et al., 2019], Adversarial perturbations [Glockner et al., 2018], ANLI [Nie et al., 2020]
Evaluation Sets	SuperGLUE diagnostic sets [Wang et al., 2019] Checklist [Ribeiro et al., 2020]	FraCaS [Cooper et al., 1994], RTE [Dagan et al., 2005], SICK [Marelli et al., 2014], SNLI [Bowman et al., 2015], MNLI [Williams et al., 2018], HANS [McCoy et al., 2019], Quantified NLI [Geiger et al., 2018], MPE [Lai et al., 2017], EQUATE [Ravichander et al., 2019], DNC [Poliak et al., 2018], ImpPres [Jeretic et al., 2020], Systematicity [Yanaka et al., 2020] ConjNLI [Saha et al., 2020], SherLIiC [Schmitt and Schütze, 2019]

Table 1: Tools and literature on robustness for NLP, with a focus on NLI as a case study. Some tools support multiple types of evaluations, for example, TextAttack supports both augmentations and attacks. For additional related work, see Section 6.

guide future evaluation decisions. The goal is to help practitioners answer questions like: "Should I test for gender bias next?" or "Should I analyze generalization to longer inputs?".

This aspect of evaluation remains understudied, because the focus has remained on prescribing and using a particular form of evaluation (e.g. inspect performance on perturbed examples). Existing tools such as CheckList [Ribeiro et al., 2020] and TextAttack [Morris et al., 2020] provide significant support on how to write code for particular evaluations, but give little guidance on what a user should run next. Existing research has studied questions related to the theory of generalization [Hendrycks et al., 2020] but very little is known about how to systematically evaluate models that will encounter distributional shift.

While there are no easy answers to these questions, we initiate a study of how practitioners can systematically make these decisions (Section 3.1), by identifying key decision variables such as their task, evaluation needs, resource constraints and history of evaluations.

2.4 Challenge 2: Idiomatic Lock-In

Ideal. Equip the practitioner with flexible tools to create and utilize evaluation examples that are best suited to the evaluation they want to run.

Challenges. Once developers decide what they want to evaluate, they can suffer from lock-in to a particular *idiom* of evaluation after they adopt a tool. Our analysis suggests that most tools and research today serve a subset of 4 evaluation idioms:

1. **Subpopulations.** Identifying subpopulations of a dataset where the model may perform poorly.

   Example: short reviews (< 50 words) in the IMDB review dataset.

2. **Transformations.** Perturbing data to check that the model responds correctly to changes.

   Example: substituting words with their synonyms in the IMDB review dataset.

3. **Attacks.** Perturbing data adversarially to exploit weaknesses in a model.

   Example: adding the word "aaaabbbb" to the end of reviews makes the model misclassify.

# Model Cards		% of Models
Total	2133	64.6%
Non-empty	922	27.9%
Any evaluation info	197	6.0%
# Models	3301	100.0%

Table 2: Prevalence of evaluation information in model cards on the HuggingFace Model Hub (`huggingface.co/models`).

4. **Evaluation Sets.** Using existing datasets or authoring examples to test generalization and perform targeted evaluation.

   Example: authoring new movie reviews in the style of a newspaper columnist.

These idioms are not exhaustive, but shed light on how evaluation is typically conducted. In Table 1, we use this categorization to summarize the tools and research available today for the natural language inference (NLI) task. As an example, TextAttack [Morris et al., 2020] users can perform attacks, while CheckList [Ribeiro et al., 2020] users author examples using templates, but cannot perform attacks.

Tools vary in whether they provide scaffolding to let users build on new evaluation ideas easily. Tools often provide excellent abstractions for particular idioms, (e.g., TextAttack [Morris et al., 2020] scaffolds users to easily write new adversarial attacks). However, no tool that we are aware of addresses this more broadly for evaluation that cuts across idioms.

All of these limitations can make it difficult for practitioners, who are forced to glue together a combination of tools. Each tool meets different developer needs, and has its own abstractions and organizing principles, which takes away time from users to inject their own creativity and expertise into the evaluation process.

We address these challenges with Robustness Gym (Section 3.2), which uses an open-interface design to support all 4 evaluation idioms, and provides a simple workflow to scaffold users.

2.5 Challenge 3: Workflow Fragmentation

Ideal. Enable practitioners to store, version and share evaluation data, communicate findings and collaborate to take advantage of others' work.

Challenges. As practitioners evaluate, they need to keep track of progress and communicate results. Evaluation tools today let users save their progress, but provide no support for semantic versioning [Preston-Werner, 2013] and sharing findings. This is made more difficult when trying to consolidate evaluations and results across multiple tools. General-purpose data storage solutions solve this problem, but require significant user effort to customize.

Reporting findings can be difficult since there is no consensus on how to report when performing evaluation across multiple idioms. Attempts at standardized reporting suggest guidelines for what to report [Mitchell et al., 2019], but are free-form, leaving the responsibility of deciding what to report to the user.

To study whether existing tools incentivize reporting, we scraped model cards [Mitchell et al., 2019] for all available Huggingface models [Wolf et al., 2020] (as of 09/22/2020). Model cards are free-form templates for reporting that contain an entry for "Evaluation" or "Results", but leave the decision of what to report to the user. Huggingface provides tools for users to create model cards when submitting models to their model hub.
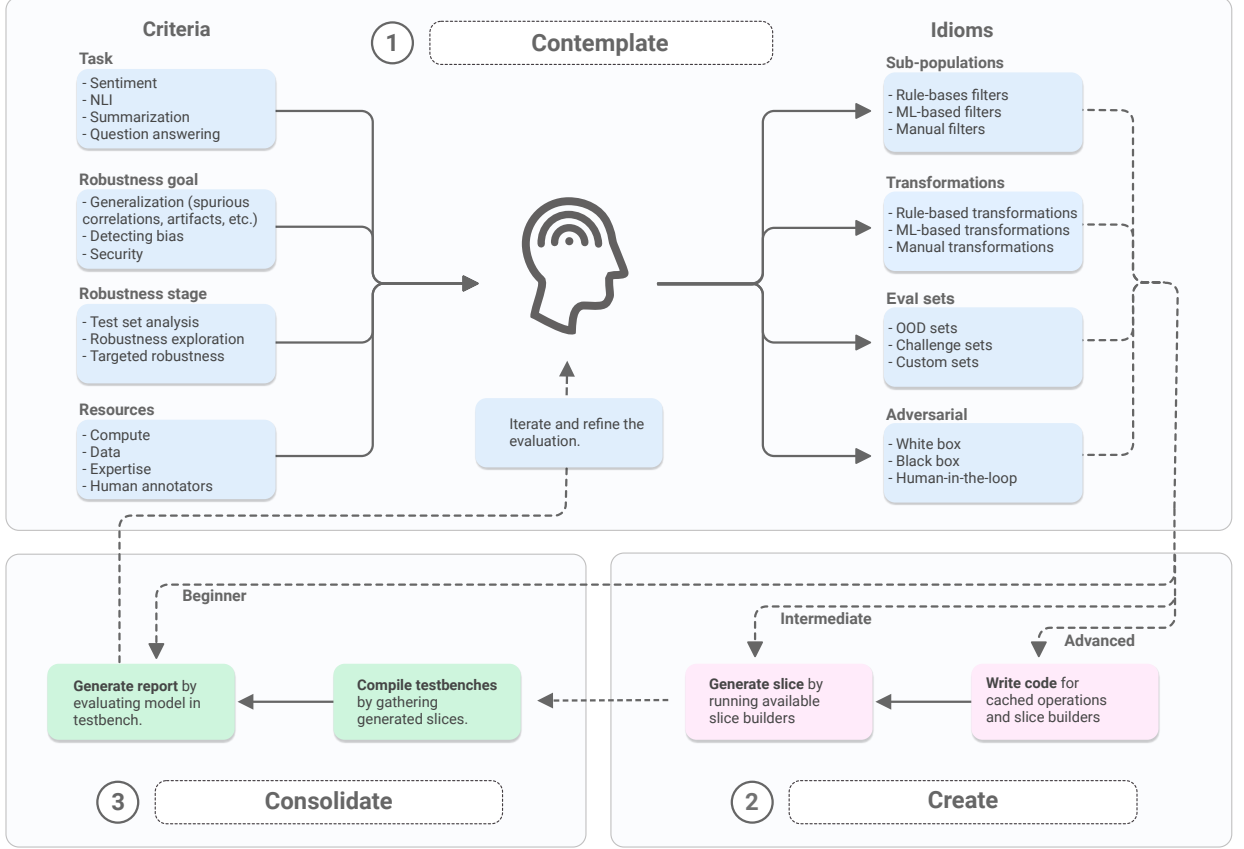
Figure 2: Illustration of the Contemplate → Create → Consolidate loop (Section 3).

Our findings are summarized in Table 2. Only a small fraction (6.0%) of models carry model cards with any evaluation information. Qualitatively, we found low consistency in how users report findings, even for models trained on the same task. This suggests that it remains difficult for users to report evaluation information consistently and easily.

In Section 3.3, we describe the support that Robustness Gym provides for versioning evaluations in test-benches, and easily exporting and reporting findings with Robustness Reports.

3   Continual Evaluation Workflow

To address the challenges we highlighted in the previous section, we propose the Contemplate → Create → Consolidate loop for performing continual evaluation. In this framework, practitioners will

1. **Contemplate** (Section 3.1) what evaluation to run next, using guidance on key decision variables,

2. **Create** (Section 3.2) slices of data for evaluation using Robustness Gym,

3. **Consolidate** (Section 3.3) findings using the testbench and reports in Robustness Gym.

Figure 2 illustrates this continual evaluation loop, which we describe in more detail next.

3.1 Contemplate: Navigating the Evaluation Landscape

As we highlighted in Section 2.3 (Paradox of Choice), practitioners may find it difficult to choose the appropriate evaluation among the large number of possibilities. We provide guidance to practitioners by focusing on key decision variables: the task, the evaluation goal, the resource constraints, and the history of prior evaluations. We connect these variables to decisions about which evaluation idiom and particular evaluations may be most appropriate. We emphasize that there is no "silver bullet" here, and our goal is to initiate research in how to make these decisions systematically.

Figure 2 visualizes and enumerates these decision criteria (top-left), embeds them in our evaluation loop, and highlights the actions available to the user in the form of which evaluation idiom and specific evaluation to choose (top-right). We describe the decision criteria below.

Task. We consider scenarios when the practitioner's task can suggest evaluations that are already known or easily available:

- **Existing research.** Prior work serves as a good starting point for evaluations that models commonly succeed against, as well as those where models typically fail (e.g., for NLI, it is well-known that examples with negation [Naik et al., 2018] are difficult for models to classify).

- **Datasets.** Tasks that are well-studied have evaluation sets that are publicly available (e.g., MNLI [Williams et al., 2018] and HANS [McCoy et al., 2019] for NLI). These serve as a useful starting point for evaluation, although users should be aware of the danger of overfitting to these evaluation sets [Dwork et al., 2015].

- **Input/output structure.** The structure of the task may constrain the types of evaluations that may be performed. For example, subpopulations based on lexical overlap may only be applied when the task is a function of two or more inputs (e.g., natural language inference accepts as input a premise and hypothesis). Prior research on similarly structured tasks can provide inspiration on a new or understudied task.

Evaluation goals. We consider 3 broad evaluation goals: testing generalization (spurious correlations, sensitivity to noise, distributional artifacts), detecting bias (gender bias, dependence on sensitive attributes), and ensuring security (vulnerability to malicious users). The practitioner's interest in these goals should influence the evaluations they choose.

- **Generalization.** Predefined out-of-distribution data splits may be used to evaluate a model's ability to generalize outside of the specific dataset set on which it was trained [Gardner et al., 2020, Koh et al., 2020]. Challenge datasets (e.g., HANS [McCoy et al., 2019]), can identify a model's ability to overcome spurious correlations in the training set (e.g., lexical overlap). Similarly, subpopulations can be constructed to leverage existing examples in a model to test particular generalization capabilities. Transformations such as paraphrasing may be used to augment the dataset with examples with differing surface-level features to test the model's reliance on artifacts in the training set.

- **Detecting bias.** Depending on the task, evaluation sets may be available to test for a model's bias with respect to particular protected attributes (e.g., gender bias in coreference in the case of Winogender [Rudinger et al., 2018] and Winobias [Zhao et al., 2018]). If no existing datasets exist, they may be synthesized by performing hand-crafted transformations with respect to particular protected attributes [Sharma et al., 2020] or subpopulations that contain particular groups considered.

- **Security.** A user might be interested in security and understanding their system's vulnerabilities, for example, a spammer may try to use adversarial attacks to bypass a spam email filter [Biggio et al., 2013]. Towards this end, the user should focus their evaluations on adversarial attacks.

Resource constraints. Constraints are central to determining the evaluations feasible for a practitioner.

- **Compute.** If compute is limited (e.g., no GPUs are available), subpopulations may be most appropriate since they can reuse predictions while attacks should be avoided since they can be extremely compute-intensive.

- **Data.** Access to data can be a bottleneck that dictates what evaluations are possible. Some tasks may require the use of proprietary or protected data (e.g., clinical notes in hospitals, or customer data in a company, making procurement and use more difficult). Transformations applied to existing data, such as with generative modeling, can be valuable in narrowing the data gap in this case.

- **Human resources.** Some evaluation strategies require a large amount of manual effort (e.g., creating custom evaluation sets). Evaluation strategies that require constructing hand-crafted rules (e.g., subpopulations), may also be time consuming. Standard transformations (e.g., paraphrasing), that augment existing datasets may help alleviate these efforts, or automated approaches to creating synthetic datasets (e.g., few-shot generation using GPT-3 [Brown et al., 2020]), may be preferred.

- **Expertise.** A user's expertise will determine whether they are able to create custom evaluations versus relying on existing ones. Domain expertise may be required to author custom evaluation sets or write custom rules for generating subpopulations. Technical expertise may be needed to write customized code for certain types of robustness tests (e.g. adversarial attacks), and should be sought if required.

Prior evaluations. The history of prior evaluations and the stage of robustness testing will also influence the choice of the next evaluation to perform. We describe 4 evaluation strategies that practitioners can use to guide continual evaluation efforts.

- **Easy → Hard.** Initial evaluations might focus on simple tests such as robustness to standard transformations (e.g., synonym substitution). Models shown to be robust against these simpler tests might then be tested on harder challenge sets or adversarial attacks.

- **Coarse → Fine.** Early evaluation should typically focus on coarse evaluations with large slices of data (e.g., performance on long vs. short inputs). Later stages of evaluation should drill-down into fine-grained slices of relevance to model deployment (e.g., queries about the Beatles in a question-answering system).

- **Explore → Exploit.** Early evaluation stages are more exploratory, as users sift through a large number of slices to search for weaknesses. Over time, it becomes more clear where a model is more or less performant, and later evaluation stages can exploit this knowledge to develop a more fine-grained understanding of performance.

- **Generic → Targeted.** Initial evaluations can draw on prior knowledge and community know-how of common evaluations. As evaluation proceeds, focus shifts to developing new evaluations that are most appropriate to the user's goal of deploying their model.

As evaluation proceeds, users should consider keeping prior evaluations as a form of regression testing [Wahl, 1999]. Much like in software, changes to the model should not degrade performance on slices where the model previously performed well.
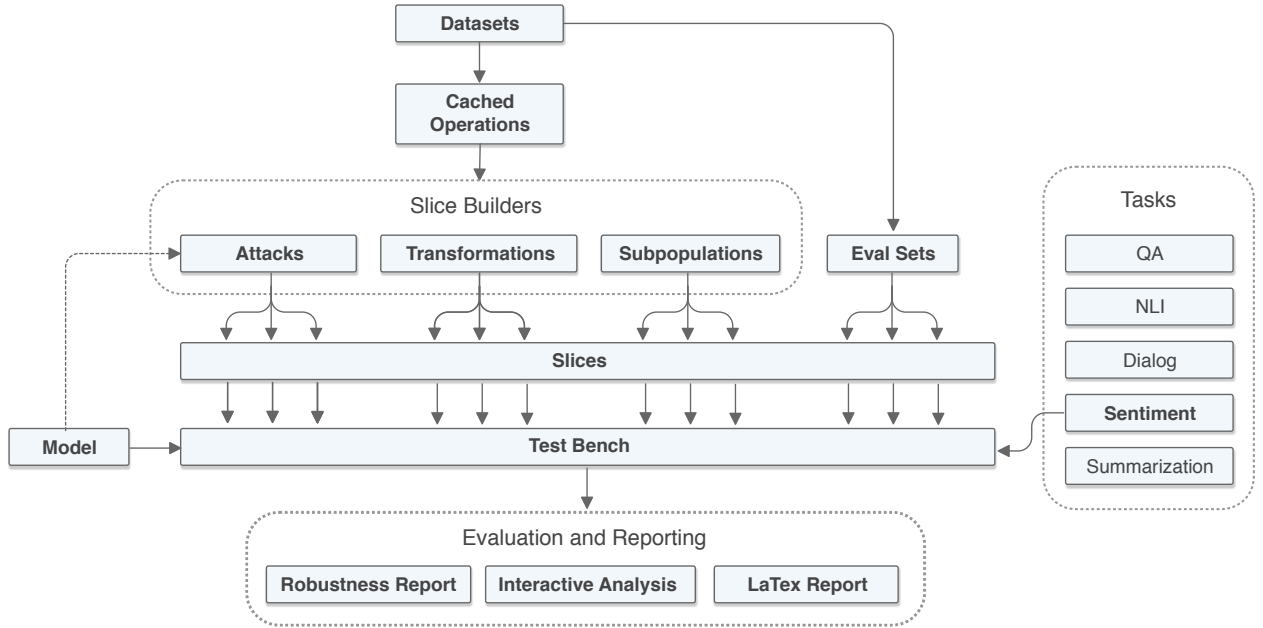
Figure 3: Robustness Gym system design and workflow.

3.2 Create: Robustness Gym

As highlighted in Section 2.4 (Idiomatic Lock-In), practitioners can get locked into a single tool that supports only a few evaluation idioms. We introduce Robustness Gym (ℝ𝔾), a toolkit that enables broad evaluation across multiple idioms. Figure 3 provides a visual depiction of the abstractions in ℝ𝔾 while Python examples for ℝ𝔾 are in Tables 4, 5 and 6 of the appendix. At a high level, ℝ𝔾 breaks robustness testing into a two-stage workflow:

1. **Caching information.** First, practitioners typically perform a set of common pre-processing operations (e.g., tokenization, lemmatization) and compute useful side information for each example (e.g., entity disambiguation, coreference resolution, semantic parsing) using external knowledge sources and models, which they cache for future analysis.

   A large part of practitioner effort goes into generating this side information—which can be expensive to compute—and into standardizing it to a format that is convenient for downstream analysis. This layer of complexity can make it difficult for them to share their evaluation with others.

   RG Support. CachedOperation is an abstraction in ℝ𝔾 to derive useful information or generate side information for each example in a dataset by (i) letting users run common operations easily and caching the outputs of these operations (e.g., running the spaCy pipeline [Honnibal et al., 2020]); (ii) storing this information alongside the associated example so that it can be accessed conveniently; (iii) providing a simple abstraction for users to write their own operations.

2. **Building slices.** Second, practitioners use the examples' inputs and any available cached information to build *slices*, which are collections of examples for evaluation based on any of the 4 evaluation idioms.

   RG Support. SliceBuilder is an abstraction to retrieve available information for an example and create slices of data from them by (i) providing retrieval methods to access inputs and cached information conveniently when writing custom code to build slices; (ii) providing specialized abstractions for specific evaluation idioms: transformations, attacks and subpopulations.

10

This breakdown naturally separates the process of gathering useful information from the nuts and bolts of using that information to build slices. Table 3 contains examples of CachedOperations and SliceBuilders that will be available in Robustness Gym.

Robustness Gym relies on a common data interface provided by the datasets library from HuggingFace [Wolf et al., 2020], which is backed by Apache Arrow [Foundation, 2019]. This ensures that all operations in Robustness Gym interoperate with HuggingFace models, and can be exported easily.

3.3 Consolidate: Share Testbenches and Findings

As highlighted in Section 2.5 (Workflow Fragmentation), users can find themselves consolidating evaluation results across several tools and evaluation idioms. Robustness Gym addresses this fragmentation by providing users a TestBench abstraction. Using this, users can assemble and version a collection of slices, which represents a suite of evaluations. Robustness Gym tracks the provenance of these slices, making it possible to identify (i) the data source that the slice originated from; (ii) the sequence of SliceBuilders by which a slice was constructed. This makes it possible for another user to reproduce or redo analysis in a collaboration, through sharing of a TestBench.

Robustness Gym also provides a general-purpose tool for creating *Robustness Reports* for any model on a TestBench. Users can also use Robustness Reports on their own, allowing them to generate reports for evaluations that are not performed in ℝ𝔾.

To incentivize standardization in reporting, ℝ𝔾 includes *Standard Reports* for several tasks. The Standard Report is comprehensive, static and is backed by a TestBench that contains slices from all evaluation idioms. It can either be generated in a PDF or LaTeX format to be added to the appendix of a paper[2]. Reports reduce user burden in communicating findings, and make it easier to standardize reporting in the community. In the future, Robustness Gym will also include an interactive tool for generating reports that allows users to pick and choose slices of interest based on their robustness goals and constraints.

4 User Personas in Robustness Gym

In this section, we discuss how users with varying expertise can use Robustness Gym to perform continual evaluation and robustness testing. We describe user personas at 3 skill levels—beginner, intermediate, and advanced—and explain a possible path through the Contemplate → Create → Consolidate process for each of them. In every case, we assume that the user's goal is to analyze the performance of an NLI model. Figure 2 illustrates how these user personas can be situated into this workflow.

4.1 Scenario I: Beginning User

Contemplate. The user's goal is to perform exploratory robustness testing for the NLI task. Because the user is new to NLP and robustness testing, they lack the knowledge to choose specific slices or write custom slices. Therefore they decide to run the Standard Report for NLI.

Create. The user is able to create the report with a few clicks in the ℝ𝔾 interface. They select "Standard Report", "Ternary Natural Language Inference" (task), "SNLI" (dataset), "BERT-Base" (model), and click "Generate Report".

Consolidate. The Standard Report, shown in Figure 4 provides a detailed snapshot of various robustness tests for NLI models. The tests may include Subpopulations (e.g., HASNEGATION, LEXICALOVERLAP),

---

[2]See Figure 8 in the appendix.

	Type	Instantiation	Examples
Rule-based	Filters	HasPhrase	■ Subpopulation that contains negation.
		HasLength	■ Subpopulation that is in the `{X}` percentile for length.
		Position	■ Subpopulation that contains `{TOKEN}` in position `{N}`.
	Logic	IFTTT recipes	■ If example ends in `{ING}` then transform with backtranslation.
		Symmetry	■ Switch the first and last sentences of a source document to create a new eval set.
		Consistency	■ Adding "aaaabbbb" at the end of every example as a form of attack.
	Template	Checklist	■ Generate new eval set using examples of the form "`I {NEGATION} {POS_VERB}.`"
Machine	Classifier	HasScore	■ Subpopulation with perplexity `{>X}` based on a LM.
		HasTopic	■ Subpopulation belonging to a certain topic.
	Tagger*	POS	■ Subpopulation that contains `{POS_NOUN}` in position `{N}`.
		NER	■ Subpopulation that contains entity names with non-English origin.
		SRL	■ Subpopulation where there is no `{AGENT}`.
		Coref	■ Subpopulation that contains the pronouns for a particular gender.
	Parser*	Constituency	■ Transform with all complete subtrees of `{POS_VP}` in the input.
		Dependency	■ Subpopulation that has at least 2 `{POS_NP}` dependent on `{POS_VP}`.
	Generative	Backtranslation	■ Using a seq2seq model for transformation using backtranslation.
		Few-shot	■ Using GPT3 like models for creating synthetic eval sets.
	Perturbation	Paraphrasing	■ Synonym substitution using EDA.
		TextAttack	■ Perturbing input using TextAttack recipes.
Human or Human-in-the-loop	Filtering	Figurative text	■ Using humans to identify subpopulation that contain sarcasm.
	Curation	Evaluation sets	■ Building datasets like ANLI, Contrast sets, HANS, etc.
		Data validation	■ Using human-in-the-loop for label verification.
	Adversarial	Invariant	■ Perturbing text in a way that the expected output does not change.
		Directional	■ Perturbing text in a way that the expected output changes.
	Transformation	Counterfactual	■ Transforming to counterfactuals for a desired target concept.

Table 3: Sample of slice builders and corresponding data slices along with example use cases that can either be used out-of-the-box or extended from Robustness Gym. ■ → subpopulations, ■ → evaluation sets, ■ → transformations and ■ → adversarial attacks. ∗ marked are CachedOperations and the rest are SliceBuilders.

Transformations (e.g., SYNONYMAUG, KEYBOARDAUG) [Ma, 2019], Attacks (TEXTATTACK) [Garg and Ramakrishnan, 2020, Morris et al., 2020], and Evaluation Sets [Bowman et al., 2015]. The user gleans several initial insights from this report. For example, they see that the model is vulnerable to common typing mistakes due to low accuracy on the KEYBOARDAUG slice; the predicted class distribution column further reveals that this noise causes the model to predict `contradiction` significantly more frequently than `entailment` or `neutral`. The user is able to easily share the generated PDF of this report with their colleagues, with whom they can iterate on additional robustness tests for misspellings.

4.2 Scenario II: Intermediate User

Contemplate. This user is interested in exploring gender bias in NLI models. Specifically they would like to test cases where specific gendered pronouns are present in the premise or hypothesis. They are willing to write minimal code to instantiate existing SliceBuilder classes with custom parameters but do not want to write the code from scratch. Therefore they decide to create slices using built-in subpopulation SliceBuilders.

Create. The user applies the existing HASPHRASE class in order to create subpopulations with female pronouns in the hypothesis:

```
    subpopulations = HasPhrase(['her', 'she']) # instantiate
    slices = subpopulations(snli, ['hypothesis']) # apply to data
```

Consolidate. The user generates a report for immediate analysis and makes the TestBench available on GitHub in order to collaborate with the broader community.

4.3 Scenario III: Advanced User

Contemplate. This user is interested in performing robustness tests for spurious correlations in NLI related to surface-level similarities between premise and hypothesis. They are particularly interested in evaluating whether models rely on the premise and hypothesis being of similar length in order to detect entailment. As they are performing a novel analysis, they plan on writing custom logic to create the appropriate slices. They consider two types of slices: subpopulations and transformations, as described below.

Create. The user utilizes the existing SCORESUBPOPULATION class, which constructs subpopulations using arbitrary scoring functions. They create a custom scoring function `len_diff`, which returns the absolute difference in length between the hypothesis and premise, and then create a SliceBuilder for the subpopulation of examples that score in the top 10% as follows:

```
    s = ScoreSubpopulation(intervals=[('90%','100%')], score_fn=len_diff)
```

The user also utilizes existing SliceBuilders such as the LEXICALOVERLAP class, which creates subpopulations based on the lexical overlap between premise and hypothesis. Additionally, they transform the dataset using classes such as EASYDATAAUGMENTATION [Wei and Zou, 2019]. They can then compose this transformation with the custom SliceBuilder described earlier to create a larger evaluation set.

Consolidate. The user generates a report for immediate analysis, and also generates an appendix for a paper to share results with the research community. They make their code and testbench available on GitHub so that others may reuse and refine their approach.

4.4 Commercial Sentiment Analysis Case Study

We validate the Contemplate → Create → Consolidate workflow with Robustness Gym through a real-world case study. We conducted a 3-hour long virtual case study with a member of the team that built the Einstein sentiment system which is part of Salesforce's cloud offerings.[3]

Pre-study questionnaire. Our pre-study questionnaire elicited information on the team's task (e.g., sentiment modeling, question answering, etc.), what metrics they use for evaluation (e.g., accuracy, F1, etc.), how they evaluate robustness (e.g., standard validation/testing, out-of-distribution data, bias testing, attacks) and what their evaluation goals are (e.g., security, generalization). Their responses suggest that their evaluations were mainly on a proprietary validation set that included some out-of-distribution data, and their main interest was in understanding the potential bias for identity groups.

We also asked them to rate on a Likert scale $(1-5)$, whether they would "like to evaluate the robustness of [their] model more thoroughly than [they] do today." (agreement $4/5$) and "would benefit from having a library that gives [them] the tools to evaluate the robustness of [their] models." (agreement $5/5$). The format and other details about the questionnaire are in Appendix A.1.

Study. The study took place during the COVID-19 pandemic and was conducted virtually with the user from the sentiment team. Due to difficulties in guiding them virtually, one of the authors shared their screen and
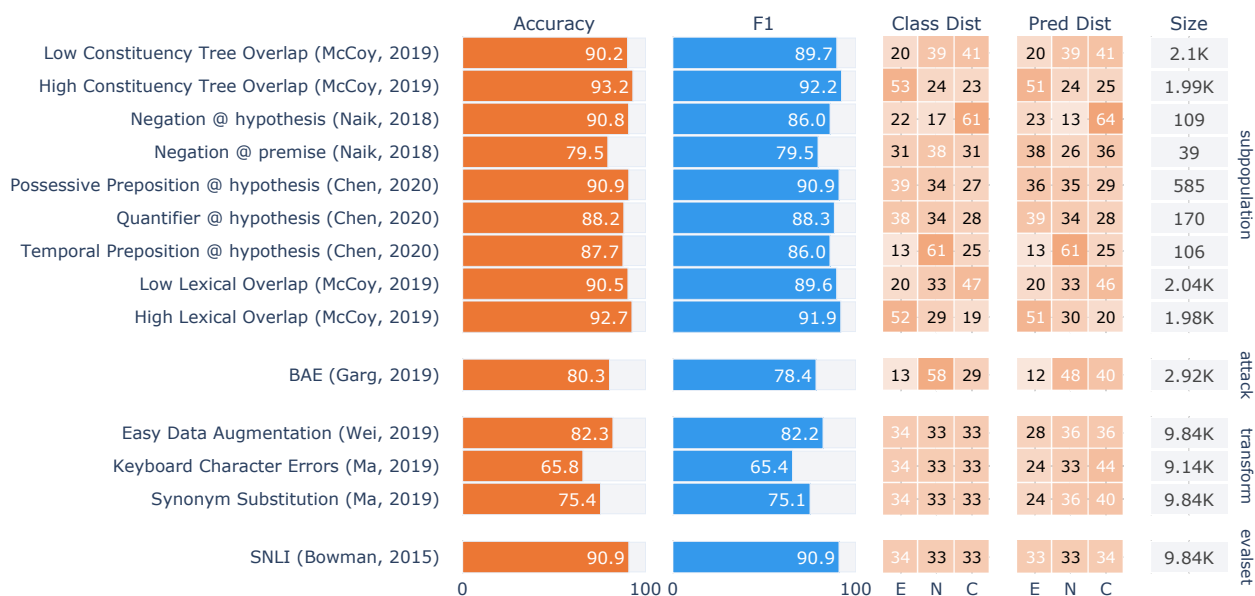
---

[3]https://einstein.ai/products/community-sentiment

	Accuracy	F1	Class Dist (E N C)			Pred Dist (E N C)			Size	
Low Constituency Tree Overlap (McCoy, 2019)	90.2	89.7	20	39	41	20	39	41	2.1K	subpopulation
High Constituency Tree Overlap (McCoy, 2019)	93.2	92.2	53	24	23	51	24	25	1.99K	
Negation @ hypothesis (Naik, 2018)	90.8	86.0	22	17	61	23	13	64	109	
Negation @ premise (Naik, 2018)	79.5	79.5	31	38	31	38	26	36	39	
Possessive Preposition @ hypothesis (Chen, 2020)	90.9	90.9	39	34	27	36	35	29	585	
Quantifier @ hypothesis (Chen, 2020)	88.2	88.3	38	34	28	39	34	28	170	
Temporal Preposition @ hypothesis (Chen, 2020)	87.7	86.0	13	61	25	13	61	25	106	
Low Lexical Overlap (McCoy, 2019)	90.5	89.6	20	33	47	20	33	46	2.04K	
High Lexical Overlap (McCoy, 2019)	92.7	91.9	52	29	19	51	30	20	1.98K	
BAE (Garg, 2019)	80.3	78.4	13	58	29	12	48	40	2.92K	attack
Easy Data Augmentation (Wei, 2019)	82.3	82.2	34	33	33	28	36	36	9.84K	transform
Keyboard Character Errors (Ma, 2019)	65.8	65.4	34	33	33	24	33	44	9.14K	
Synonym Substitution (Ma, 2019)	75.4	75.1	34	33	33	24	36	40	9.84K	
SNLI (Bowman, 2015)	90.9	90.9	34	33	33	33	33	34	9.84K	evalset

Figure 4: Robustness Report for Natural Language Inference using `bert-base` on SNLI.

conducted all the experimentation throughout the 3-hour period.

We followed the Contemplate → Create → Consolidate loop, and we highlight the key steps that we took through the study period.

- **Contemplate (1).** We first identified resource constraints—the user provided us with their evaluation data, and gave us black-box access to their model[4]. We used a CPU-only MacBook Pro for all computation. Since the team had previously analyzed the sensitivity of their model to mentions of race/gender/religion/ethnicity, we decided to first verify performance on subpopulations of their dataset with identity-sensitive words.

- **Create (1).** We constructed slices for evaluating performance using a SliceBuilder that searched for identity words. We found no degradations compared to the average performance of the model on nine identity-sensitive words.

- **Contemplate (2).** Next, after discussion with the user, we considered whether the model could have performance disparities along different topics.

- **Create (2).** We next evaluated the model on subpopulations that contained topic-specific words. We found that the model did poorly on some topics, with performance degradations of up to 18%.

- **Contemplate (3).** Next, we set out to understand whether the model was robust to input perturbations. The user highlighted instances of input noise that they wanted to gather more information on.

- **Create (3).** We used 4 different transformations for simulating typing errors and paraphrasing text, and found that performance degraded by 6%.

- **Contemplate (3).** Lastly, they wanted to investigate whether the model was robust to larger distributional shifts in the inputs.

---

[4] We could not access the model directly, but could give them examples to fetch predictions.

- **Create (3).** We downloaded and used an open-source sentiment dataset, and found that performance degraded by 5%.

- **Consolidate (1).** We collated all the slices into a testbench, and generated a report to share with other members of their team.

We performed 3 iterations of (Contemplate → Create), resetting the evaluation objective after each iteration, and using Robustness Gym to investigate them. Overall, we evaluated the system on 172 different subpopulations, 1 open-source evaluation set from the internet, and 4 different transformations, all in the 3-hour period. We observed a total of 12 subpopulations where performance degraded significantly. This performance deterioration occurred under all 4 types of transformations as well. Lastly, since we did not have access to the model for training, we made prescriptions for augmentation-based training to improve performance on examples where the model underperformed.

Post-study questionnaire. We conducted a post-study questionnaire with the user, where we asked them to provide feedback on Robustness Gym and the overall study. We elicited feedback on "how likely [they] were to incorporate Robustness Gym in [their] workflow" (very likely $5/5$), and the perceived "ease of use of Robustness Gym" (high $5/5$). In feedback related to the utility of the 4 evaluation idioms in Robustness Gym, they found subpopulations to be "very insightful", and were enthusiastic about the ability to perform various evaluations in a single tool. Lastly, the robustness report gives information on how the team could make improvements and work towards adopting continual evaluation for their system.

5 Experimental Results using Robustness Gym

Robustness Gym makes it easy for researchers and practitioners to perform novel analyses of existing tasks and models. To demonstrate this, we use Robustness Gym to investigate fine-grained performance on 2 tasks—named entity linking (NEL) and text summarization. For NEL, we present the first fine-grained analysis of NEL across 3 widely used commercial APIs, and 3 state-of-the-art academic systems. For summarization, we analyze 7 state-of-the-art models for text summarization trained on the CNN/DailyMail dataset.

5.1 NEL on Commercial APIs

We analyze the fine-grained performance of commercial and state-of-the-art-systems for named entity linking (NEL). NEL is a fundamental component of both search and question-answering systems such as conversational assistants, and has a widespread impact on the performance of commercial technology. Given some text, the NEL task involves the identification of all entity mentions, and contextualized linking of these mentions to their corresponding Wikipedia entries, e.g., "She drove a Lincoln to Lincoln" would link the first mention of Lincoln to `Lincoln_Motor_Company` and the second mention of Lincoln to `Lincoln,_Nebraska`. Each identified mention (e.g., "Lincoln") is typically mapped to a candidate list of Wikipedia entries (e.g., all "Lincoln"-related Wikipedia entries) before disambiguation. Our goal is to use Robustness Gym to understand where existing NEL systems fall short.

Systems. We consider 3 commercially available NEL APIs: (i) GOOGLE Cloud Natural Language API[5], (ii) MICROSOFT Text Analytics API[6], and (iii) AMAZON Comprehend API[7,8]. We compare them to 3 state-

---

[5] https://cloud.google.com/natural-language
[6] https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/
[7] https://aws.amazon.com/comprehend/
[8] AMAZON only performs named entity recognition (NER) to identify mentions of named-entities in text, so we use it in conjunction with a simple string matching heuristic to resolve entity links.

	Amazon	Google	Microsoft	Bootleg	Rel	Wat	Size	
everything	49.9	49.2	66.8	78.7	51.2	69.2	49.5K	all
popular	68.7	82.7	71.7	83.9	80.4	88.1	8.32K	
everything	64.9	73.3	73.2	85.1	71.6	83.2	15.5K	head
kg-relation	61.6	80.3	69.9	83.8	77.3	85.5	5.07K	
one-of-the-two	48.7	65.3	65.3	77.2	65.3	78.6	885	
share-1-type	79.5	81.5	89.1	94.7	85.4	92.8	1.25K	
strong affordance	65.1	73.3	73.1	85.5	71.2	83.2	14.4K	
unpopular	30.3	37.8	49.5	79.2	33.7	50.6	650	
everything	44.4	40.4	65.1	77.2	44.3	65.1	30K	torso
kg-relation	55.5	53.4	77.5	86.9	50.3	72.2	6.23K	
one-of-the-two	43.0	39.2	66.5	80.6	46.5	68.2	2.13K	
share-1-type	45.4	44.8	77.4	88.2	57.9	76.2	2.78K	
strong affordance	45.6	41.1	66.2	78.5	44.3	65.5	24.4K	
unpopular	23.6	21.2	45.4	61.7	24.8	43.3	3.51K	
everything	33.8	21.6	55.1	65.2	23.6	46.2	4.04K	tail
kg-relation	44.0	30.5	70.4	80.4	19.6	51.8	901	
one-of-the-two	45.5	31.2	65.6	74.9	34.4	57.7	279	
share-1-type	31.5	24.5	62.9	80.7	38.0	64.0	461	
strong affordance	34.9	22.5	57.8	67.9	23.6	46.5	3.14K	
unpopular	16.7		37.4	44.1		28.1	449	
everything	25.0	15.6	50.8	66.4	18.0	36.7	128	toes
kg-relation	27.8	16.7	66.7	75.0	22.2	47.2	36	
one-of-the-two	50.0	41.7	58.3	75.0	41.7	75.0	12	
share-1-type	14.3	28.6	64.3	78.6	35.7	64.3	14	
strong affordance	26.9	18.3	55.8	69.2	19.2	37.5	104	
unpopular	30.0		40.0	40.0		20.0	10	

Figure 5: Robustness Report for NEL on Wikipedia. Performance reported using the Recall metric.

of-the-art systems and a heuristic baseline: (i) BOOTLEG [Orr et al., 2020], a self-supervised system for NEL, (ii) REL [van Hulst et al., 2020], a system that combines existing state-of-the-art approaches, (iii) WAT [Piccinno and Ferragina, 2014] an extension of the TAGME [Ferragina and Scaiella, 2010] linker, and (iv) POP, our simple heuristic baseline that picks the most popular entity among a set of candidate entities.

Datasets. We compare these methods on examples drawn from two datasets: (i) WIKIPEDIA, which contains $100,000$ entity mentions across $37,492$ sentences from a 2019 Wikipedia dataset, and (ii) AIDA, the AIDA test-b dataset.

Metrics. For WIKIPEDIA, we compare performance on recall[9]. For AIDA, we compare performance on Macro-F1.

---

[9] WIKIPEDIA is sparsely labeled and we do not report precision or F1 scores, which can be misleading.

5.1.1 Analysis on WIKIPEDIA

Slices. In line with Orr et al. [2020], we consider 4 groups of slices—head, torso, tail and toe—that are based on the popularity of the entities being linked. Intuitively, head examples involve resolving popular entities that occur frequently in WIKIPEDIA, torso examples have medium popularity while tail examples correspond to entities that are seen rarely. Toe entities are a subset of the tail that are almost never seen. We consider 5 subpopulations from [Orr et al., 2020] within each group,

- *kg-relation* contains examples where the entity being linked is related to another entity in the sentence. This serves as useful contextual information that can aid disambiguation.

- *one-of-the-two* contains examples where the gold entity is one of the two most popular candidates in the list of candidates, and both have similar popularity. These examples require careful use of context to disambiguate.

- *share-1-type* contains examples where the sentence contains 3 consecutive entities that share the same type affordance. These type affordances can be used as contextual cues for disambiguation.

- *strong-affordance* contains examples where the sentence has words that are highly associated (as measured by tf-idf) with the gold entity's type(s). Again, these words can be used as contextual cues.

- *unpopular* contains examples where the gold entity is the second or less popular entity in the list of candidates, and the most popular entity is at least $5\times$ more popular than the second. These examples require the model to overlook popularity in favor of preferring a more uncommon entity.

Lastly, we also consider performance on *popular* entities which correspond to examples where the entity mention corresponds to one of the top 800 most popular entity mentions.

Bootleg is best overall. Overall, we find that BOOTLEG is the best-performing system, while MICROSOFT is the best-performing commercial system. BOOTLEG outperforms other systems by a wide margin, with a 12 point gap to the next best system (MICROSOFT), while MICROSOFT in turn outperforms other commercial systems by more than 16 points.

Performance degrades on rare entities. For all systems, we find that performance on head slices is substantially better than performance on tail/toe slices. BOOTLEG is the most robust across the set of slices that we consider[10]. In particular, we note that GOOGLE and AMAZON struggle on tail and torso entities, while MICROSOFT's performance degrades more gracefully. GOOGLE's model is particularly adept at popular entities where it outperforms MICROSOFT by more than 11 points.

5.1.2 Analysis on AIDA

For AIDA, we compare performance on Macro-F1, since AIDA provides a dense labeling of entities (and therefore computing precision is meaningful). Similar to WIKIPEDIA, we find that BOOTLEG is the best-performing system overall on AIDA, while MICROSOFT is the best-performing commercial system.

Sensitivity to capitalization. Both GOOGLE and AMAZON are sensitive to whether the entity mention is capitalized. GOOGLE's performance goes from 54.1% on sentences where all gold-labeled entities are capitalized to 38.2% on sentences where no gold-labeled entities are capitalized. Similarly, MICROSOFT degrades from 66.0% to 35.7% on these slices. This suggests that mention extraction in these models is quite sensitive to capitalization. In contrast, AMAZON, BOOTLEG and WAT have stable performance, regardless of capitalization.

---

[10]We note that this may partly be a consequence of the set of slices we use, which are taken from Orr et al. [2020].

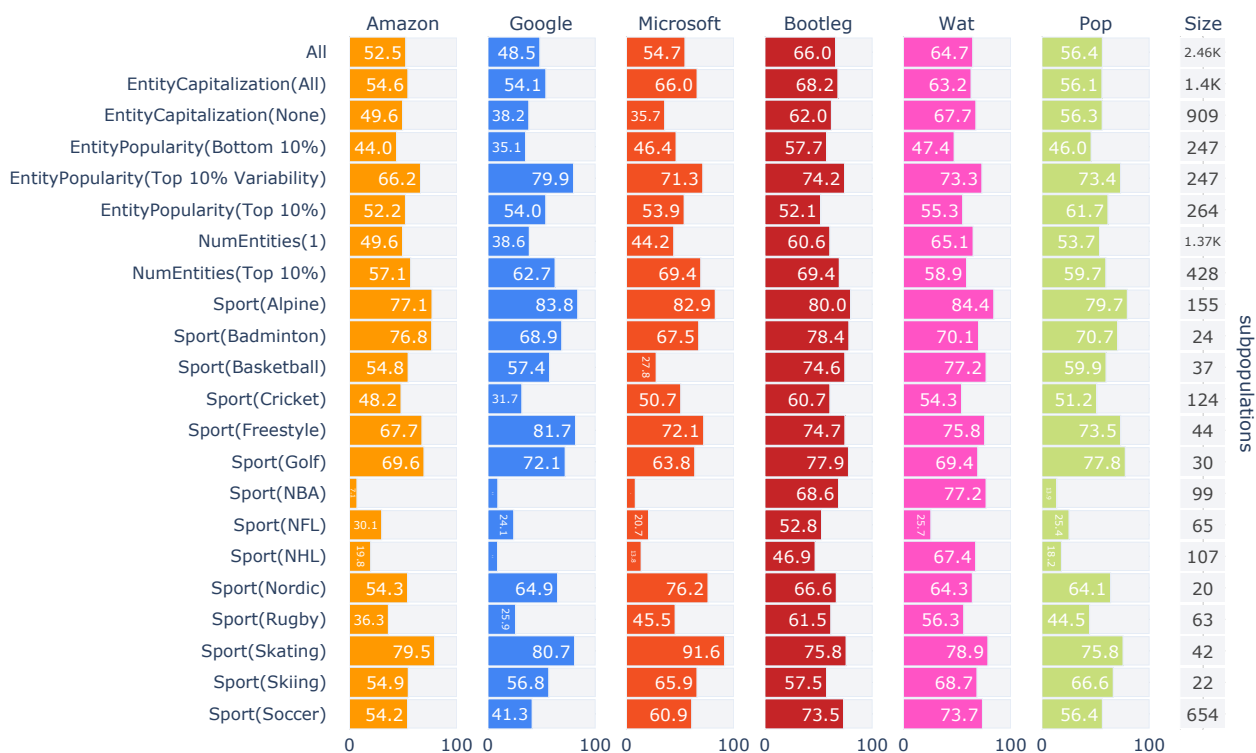| subpopulations	Amazon	Google	Microsoft	Bootleg	Wat	Pop	Size
All	52.5	48.5	54.7	66.0	64.7	56.4	2.46K
EntityCapitalization(All)	54.6	54.1	66.0	68.2	63.2	56.1	1.4K
EntityCapitalization(None)	49.6	38.2	35.7	62.0	67.7	56.3	909
EntityPopularity(Bottom 10%)	44.0	35.1	46.4	57.7	47.4	46.0	247
EntityPopularity(Top 10% Variability)	66.2	79.9	71.3	74.2	73.3	73.4	247
EntityPopularity(Top 10%)	52.2	54.0	53.9	52.1	55.3	61.7	264
NumEntities(1)	49.6	38.6	44.2	60.6	65.1	53.7	1.37K
NumEntities(Top 10%)	57.1	62.7	69.4	69.4	58.9	59.7	428
Sport(Alpine)	77.1	83.8	82.9	80.0	84.4	79.7	155
Sport(Badminton)	76.8	68.9	67.5	78.4	70.1	70.7	24
Sport(Basketball)	54.8	57.4	27.8	74.6	77.2	59.9	37
Sport(Cricket)	48.2	31.7	50.7	60.7	54.3	51.2	124
Sport(Freestyle)	67.7	81.7	72.1	74.7	75.8	73.5	44
Sport(Golf)	69.6	72.1	63.8	77.9	69.4	77.8	30
Sport(NBA)				68.6	77.2		99
Sport(NFL)	30.1	24.1	20.7	52.8	25.7	25.4	65
Sport(NHL)	19.8		13.8	46.9	67.4	10.2	107
Sport(Nordic)	54.3	64.9	76.2	66.6	64.3	64.1	20
Sport(Rugby)	36.3	25.9	45.5	61.5	56.3	44.5	63
Sport(Skating)	79.5	80.7	91.6	75.8	78.9	75.8	42
Sport(Skiing)	54.9	56.8	65.9	57.5	68.7	66.6	22
Sport(Soccer)	54.2	41.3	60.9	73.5	73.7	56.4	654

Figure 6: Robustness Report for NEL on AIDA. Performance reported using the Macro-F1 metric.

Performance on topical entities. Interestingly, all models appear to struggle on some topical slices (e.g., on the NFL slice), all models degrade significantly, with BOOTLEG outperforming other models by 20+%. Both GOOGLE and MICROSOFT display strong performance on some topics, (e.g., GOOGLE on alpine sports and MICROSOFT on skating).

Popularity heuristic outperforms commercial systems. Somewhat surprisingly, we find that POP outperforms all commercial systems by 1.7 points. In fact, we note that the pattern of errors for POP is very similar to those of the commercial systems (e.g., performing poorly on NBA, NFL and NHL slices). This suggests that commercial systems sidestep the difficult problem of disambiguating ambiguous entities in favor of returning the more popular answer. Similar to WIKIPEDIA GOOGLE performs best among commercial systems on examples that contain the most popular entities (top 10% entity popularity).

Overall, our results suggest that state-of-the-art academic systems substantially outperform commercial APIs for NEL.

5.2 Summarization with State-of-the-Art Models

Next, we analyze the performance of state-of-the-art summarization models using Robustness Gym. We selected summarization as an instance of a text-generation task to demonstrate the versatility of Robustness Gym for prediction tasks beyond classification or sequence labeling. For example, we show how slices can be computed based not only on the input text but also the ground-truth label and other cached information. We present a unified view of robustness testing of summarization systems that is inspired by a diverse set of approaches to this problem [Grusky et al., 2018, Jung et al., 2019, Kedzie et al., 2018, Kryscinski et al.,

2019].

Models. We use model predictions for 7 models from SummEval [Fabbri et al., 2020] on the CNN/Daily-Mail [Hermann et al., 2015] test dataset: (i) LEAD-3, which uses the 3 leading sentences as a summary, (ii) NEUSUM [Zhou et al., 2018], (iii) BANDITSUM [Dong et al., 2018], (iv) JECS [Xu and Durrett, 2019], (v) T5 [Raffel et al., 2020], (vi) BART [Lewis et al., 2020], (vii) PEGASUS [Zhang et al., 2020].

Slices. Below, we define several heuristics for identifying subpopulations of summarization datasets for robustness testing. See Appendix A.3 for additional details.

- *abstractiveness* is the degree to which the reference summary requires consolidating and reframing content from the source document [Grusky et al., 2018]. Summaries range from extractive, where a subset of sentences is directly selected from the source document to abstractive.

- *distillation* is the degree to which the reference summary discards content from the source document. Highly distilled summaries require models to carefully select what to present in the summary.

- *position* is the average location—in the source document—of where information in the summary comes from. High positions require models to use information that appears later in the source document.

- *dispersion* is the degree to which the reference summary uses content that is distributed broadly across the article versus concentrated in a particular region. High dispersion requires the method to attend to multiple parts of the source document to consolidate information.

- *ordering* is the similarity with which content in the source and reference summary are ordered. Summaries that change or reverse the ordering of content require models to reason over how to best present contextual information.

We also consider slices based on the length of the source document and the number of contained entities, which serve as proxy measures for the complexity of content to be summarized.

5.2.1 Analysis on CNN/DailyMail

We include a Robustness Report in Figure 7, and describe results below.

Models struggle to abstract and distill. All models perform worst on the "most distilled" subpopulation, i.e. on examples where the model must discard a large amount of information in the article to construct a summary. Models also struggle on the examples that required the most abstraction. In contrast, both extractive and abstractive models excel on extractive examples ("least abstractive").

Abstractive models have less positional bias. Extractive models have large gaps in performance between examples where the summary can be constructed using the early ("earliest positions") vs. late ("latest positions") of the article e.g. all extractive models have a $9+$ point gap between these subpopulations. Abstractive models have smaller gaps, e.g. PEGASUS has a gap of only 5.9 points.

Errors are highly correlated. All summarization models, whether extractive or abstractive, degrade and improve on the same populations of data. This is surprising, since these models use quite different prediction mechanisms e.g. abstractive models like T5 appear to offer no relative advantage on the "most abstractive" examples compared to the Lead-3 baseline (both models are 9 points worse than their overall performance). We note that the development of reliable evaluation metrics in summarization continues to be an active area of research, and it is likely that current evaluation metrics are unable to capture some meaningful differences that may exist.
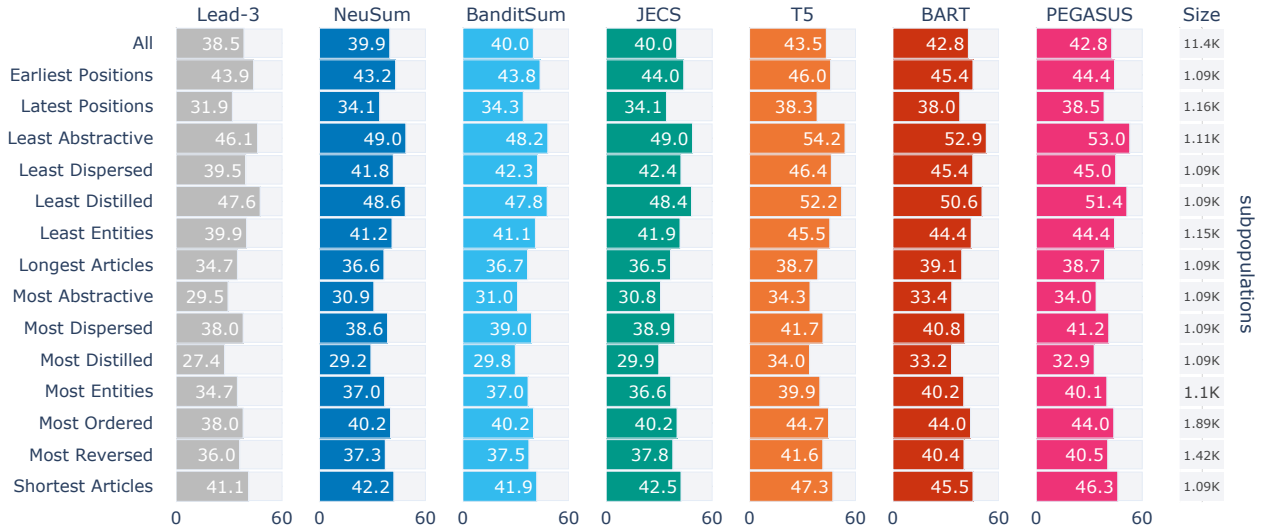
19

subpopulations	Lead-3	NeuSum	BanditSum	JECS	T5	BART	PEGASUS	Size
All	38.5	39.9	40.0	40.0	43.5	42.8	42.8	11.4K
Earliest Positions	43.9	43.2	43.8	44.0	46.0	45.4	44.4	1.09K
Latest Positions	31.9	34.1	34.3	34.1	38.3	38.0	38.5	1.16K
Least Abstractive	46.1	49.0	48.2	49.0	54.2	52.9	53.0	1.11K
Least Dispersed	39.5	41.8	42.3	42.4	46.4	45.4	45.0	1.09K
Least Distilled	47.6	48.6	47.8	48.4	52.2	50.6	51.4	1.09K
Least Entities	39.9	41.2	41.1	41.9	45.5	44.4	44.4	1.15K
Longest Articles	34.7	36.6	36.7	36.5	38.7	39.1	38.7	1.09K
Most Abstractive	29.5	30.9	31.0	30.8	34.3	33.4	34.0	1.09K
Most Dispersed	38.0	38.6	39.0	38.9	41.7	40.8	41.2	1.09K
Most Distilled	27.4	29.2	29.8	29.9	34.0	33.2	32.9	1.09K
Most Entities	34.7	37.0	37.0	36.6	39.9	40.2	40.1	1.1K
Most Ordered	38.0	40.2	40.2	40.2	44.7	44.0	44.0	1.89K
Most Reversed	36.0	37.3	37.5	37.8	41.6	40.4	40.5	1.42K
Shortest Articles	41.1	42.2	41.9	42.5	47.3	45.5	46.3	1.09K

Figure 7: Robustness Report for summarization on CNN-DailyMail. Performance reported using the ROUGE-1 metric.

6 Related Tools and Work

Our work is related to many machine learning research areas including AutoML, Ethical ML, Interpretable ML, as well as Error Analysis.

AutoML. Automated machine learning is an area of research that focuses on creating tools that help remove the manual efforts in building machine learning systems [Snoek et al., 2012]. Traditionally, these have focused on data wrangling, feature and model selection, and hyperparameter optimization. More recently with hardware acceleration, AutoML has expanded to include neural architecture search (NAS) [Pham et al., 2018]. Although AutoML aims to provide tools for efficient and robust models, it only focuses on training and not evaluations [Feurer et al., 2015]. Robustness Gym on the other hand focuses on removing the manual effort in evaluations of machine learning models across a suite of robustness tests.

Ethical ML. There exist reporting and analysis tools developed for ethical, fair and responsible use of ML. The Model Cards toolkit [Mitchell et al., 2019] is an example of a reporting toolkit. Examples of analysis tools include the What-if Tool [Wexler et al., 2019], FairVis [Cabrera et al., 2019], and FairSight [Ahn and Lin, 2019]. Although these toolkits provide support to report and identify biases in ML models, it is not obvious how and which biases should be tested for. Their scope is also limited to ethics. Robustness Gym is a general-purpose toolkit that supports both reporting and analysis. It provides tools for evaluating robustness while mapping out the various dimensions that a user should consider for their use case.

Interpretable ML. Interpreting ML models enables a better understanding of their behavior. There exist several tools and frameworks for general purpose interpretability, including the recent Language Interpretability Tool (LIT) [Tenney et al., 2020], IBM's AI Explainability 360 [Arya et al., 2019], AllenNLP Interpret [Wallace et al., 2019], InterpretML [Nori et al., 2019], Manifold [Zhang et al., 2018], and Pytorch Captum [Kokhlikyan et al.]. DiCE is a tool focused on explanations using counterfactuals [Mothilal et al., 2020]. Interpretability and robustness are both desirable but different properties for ML models. Interpretability tools not only have different objectives but also different sets of features that are complementary to Robustness Gym (e.g., interpreting or providing causal explanations for a particular prediction after ℝ𝔾 identifies a generalization

20

problem in the model). Many of these tools focus on interactive visualization, which limits their scope to interpreting small numbers of examples and makes their results difficult to reproduce. This also makes their use susceptible to subjectivity and selection bias. By contrast, Robustness Gym can scale to large datasets (e.g., $100,000$ Wikipedia examples in Section 5.1) with ease. Testbenches provide reproducibility to the analyses conducted in $\mathbb{RG}$.

Error Analysis. Tools for error analysis help users in understanding where their models are failing. Errudite [Wu et al., 2019] supports users in exploring subpopulations of their data, while CrossCheck [Arendt et al., 2020] and Manifold [Zhang et al., 2018] focus on visualization and analysis for model comparison. Robustness Gym is complementary to these tools in that it enables users to understand likely performance degradations and preempt those before they become errors.

7   Conclusion

We introduced Robustness Gym, an evaluation toolkit that supports a broad set of evaluation idioms, and can be used for collaboratively building and sharing evaluations and results. To address challenges faced by practitioners today, we embedded Robustness Gym into the Contemplate $\rightarrow$ Create $\rightarrow$ Consolidate continual evaluation loop. Our results suggest that Robustness Gym is a promising tool for researchers and practitioners.

Acknowledgements

References

[1] Yongsu Ahn and Yu-Ru Lin. Fairsight: Visual analytics for fairness in decision making. *IEEE transactions on visualization and computer graphics*, 26(1):1086–1095, 2019.

[2] Dustin Arendt, Zhuanyi Huang, Prasha Shrestha, E. Ayton, Maria Glenski, and Svitlana Volkova. Crosscheck: Rapid, reproducible, and interpretable model evaluation. *ArXiv*, abs/2004.07993, 2020.

[3] Vijay Arya, Rachel K. E. Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Q. Vera Liao, Ronny Luss, Aleksandra Mojsilović, Sami Mourad, Pablo Pedemonte, Ramya Raghavendra, John Richards, Prasanna Sattigeri, Karthikeyan Shanmugam, Moninder Singh, Kush R. Varshney, Dennis Wei, and Yunfeng Zhang. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques, Sept 2019. URL https://arxiv.org/abs/1909.03012.

[4] Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *ArXiv*, abs/1711.02173, 2018.

[5] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[6] C. M. Bishop. Pattern recognition and machine learning (information science and statistics). 2006.

[7] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*, 2015.

[8] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[9] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. Fairvis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE, 2019.

[10] Vincent Chen, Sen Wu, Alexander J Ratner, Jen Weng, and Christopher Ré. Slice-based learning: A programming model for residual learning in critical data slices. In *Advances in neural information processing systems*, pages 9392–9402, 2019.

[11] Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, and et al. Pulman, Stephen. Using the framework. Technical report, Deliverable D6, 1994.

[12] Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer, 2005.

[13] Yue Dong, Yikang Shen, E. Crawford, H. V. Hoof, and J. Cheung. Banditsum: Extractive summarization as a contextual bandit. In *EMNLP*, 2018.

[14] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349:636 – 638, 2015.

[15] Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. Summeval: Re-evaluating summarization evaluation. *arXiv preprint arXiv:2007.12626*, 2020.

[16] P. Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). *ArXiv*, abs/1006.3498, 2010.

[17] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.

[18] Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and T. Kwiatkowski. Empirical evaluation of pretraining strategies for supervised entity linking. *ArXiv*, abs/2005.14253, 2020.

[19] Apache Software Foundation. Arrow: A cross-language development platform for in-memory data, 2019. URL https://arrow.apache.org.

[20] Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, et al. Evaluating nlp models via contrast sets. *arXiv preprint arXiv:2004.02709*, 2020.

[21] Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. *ArXiv*, abs/2004.01970, 2020.

[22] Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. Stress-testing neural models of natural language inference with multiply-quantified sentences. *arXiv preprint arXiv:1810.13033*, 2018.

[23] Max Glockner, Vered Shwartz, and Yoav Goldberg. Breaking nli systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 650–655, 2018.

[24] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1065. URL https://www.aclweb.org/anthology/N18-1065.

[25] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

[26] Isobel Asher Hamilton. Amazon built an AI tool to hire people but had to shut it down because it was discriminating against women, 2018. URL https://www.businessinsider.com/amazon-built-ai-to-hire-people-discriminated-against-women-2018-10.

[27] Isobel Asher Hamilton. Twitter is investigating after anecdotal data suggested its picture-cropping tool favors white faces, 2020. URL https://www.businessinsider.com/twitter-investigating-picture-preview-algorithm-racial-bias-2020-9.

[28] Karen Hao. Facebook's ad-serving algorithm discriminates by gender and race, 2019. URL https://www.technologyreview.com/2019/04/05/1175/facebook-algorithm-discriminates-ai-bias/.

[29] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. *arXiv preprint arXiv:1901.09960*, 2019.

[30] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. *arXiv preprint arXiv:2006.16241*, 2020.

[31] Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend, 2015.

[32] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python, 2020. URL https://doi.org/10.5281/zenodo.1212303.

[33] Paloma Jeretic, Alex Warstadt, Suvrat Bhooshan, and Adina Williams. Are natural language inference models imppressive? learning implicature and presupposition. *arXiv preprint arXiv:2004.03066*, 2020.

[34] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, 2017.

[35] Taehee Jung, Dongyeop Kang, Lucas Mentch, and Eduard Hovy. Earlier isn't always better: Sub-aspect analysis on corpus and system biases in summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3324–3335, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1327. URL https://www.aclweb.org/anthology/D19-1327.

[36] Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the difference that makes a difference with counterfactually-augmented data. *arXiv preprint arXiv:1909.12434*, 2019.

[37] Nicolas Kayser-Bril. Google apologizes after its Vision AI produced racist results, 2020. URL https://algorithmwatch.org/en/story/google-vision-racism/.

[38] Chris Kedzie, Kathleen McKeown, and Hal Daumé III. Content selection in deep learning models of summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1818–1828, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1208. URL https://www.aclweb.org/anthology/D18-1208.

[39] Douwe Kiela et al. Rethinking AI Benchmarking, 2020. URL https://dynabench.org/.

[40] Will Knight. The Apple Card Didn't 'See' Gender—and That's the Problem, 2019. URL https://www.wired.com/story/the-apple-card-didnt-see-genderand-thats-the-problem/.

[41] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. Wilds: A benchmark of in-the-wild distribution shifts, 2020.

[42] Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Jonathan Reynolds, Alexander Melnikov, Natalia Lunova, and Orion Reblitz-Richardson. Pytorch captum. URL https://github.com/pytorch/captum.

[43] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1051. URL https://www.aclweb.org/anthology/D19-1051.

[44] Alice Lai, Yonatan Bisk, and Julia Hockenmaier. Natural language inference from multiple premises. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 100–109, Taipei, Taiwan, November 2017. Asian Federation of Natural Language Processing. URL https://www.aclweb.org/anthology/I17-1011.

[45] M. Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, A. Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *ArXiv*, abs/1910.13461, 2020.

[46] Edward Ma. NLP Augmentation, 2019. URL https://github.com/makcedward/nlpaug.

[47] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. SemEval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 1–8, Dublin, Ireland, August 2014. Association for Computational Linguistics. doi: 10.3115/v1/S14-2001. URL https://www.aclweb.org/anthology/S14-2001.

[48] R. T. McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *ArXiv*, abs/1902.01007, 2019.

[49] R Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

[50] J. Miller, Karl Krauth, B. Recht, and L. Schmidt. The effect of natural distribution shift on question answering models. *ArXiv*, abs/2004.14444, 2020.

[51] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[52] John X Morris, Eli Lifland, Jin Yong Yoo, and Yanjun Qi. Textattack: A framework for adversarial attacks in natural language processing. *arXiv preprint arXiv:2005.05909*, 2020.

[53] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 607–617, 2020.

[54] Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. Stress test evaluation for natural language inference. *arXiv preprint arXiv:1806.00692*, 2018.

[55] Yixin Nie, Yicheng Wang, and Mohit Bansal. Analyzing compositionality-sensitivity of nli models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6867–6874, 2019.

[56] Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *ACL*, 2020.

[57] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[58] L. Orr, Megan Leszczynski, Simran Arora, Sen Wu, N. Guha, Xiao Ling, and C. Ré. Bootleg: Chasing the tail with self-supervised named entity disambiguation. *ArXiv*, abs/2010.10363, 2020.

[59] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

[60] Francesco Piccinno and P. Ferragina. From tagme to wat: a new entity annotator. In *ERD '14*, 2014.

[61] Adam Poliak, Aparajita Haldar, Rachel Rudinger, J. Edward Hu, Ellie Pavlick, Aaron Steven White, and Benjamin Van Durme. Collecting diverse natural language inference problems for sentence representation evaluation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 67–81, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1007. URL https://www.aclweb.org/anthology/D18-1007.

[62] Tom Preston-Werner. Semantic versioning 2.0. 0. *línea]. Available: http://semver. org*, 2013.

[63] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, M. Matena, Yanqi Zhou, W. Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

[64] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access, 2017.

[65] Abhilasha Ravichander, Aakanksha Naik, Carolyn Rose, and Eduard Hovy. EQUATE: A benchmark evaluation framework for quantitative reasoning in natural language inference. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 349–361, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1033. URL https://www.aclweb.org/anthology/K19-1033.

[66] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Association for Computational Linguistics (ACL)*, 2020.

[67] Alexis Ross and Ellie Pavlick. How well do nli models capture verb veridicality? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2230–2240, 2019.

[68] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. Gender bias in coreference resolution. *arXiv preprint arXiv:1804.09301*, 2018.

[69] Swarnadeep Saha, Yixin Nie, and Mohit Bansal. Conjnli: Natural language inference over conjunctive sentences. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8240–8252, 2020.

[70] Ivan Sanchez, Jeff Mitchell, and Sebastian Riedel. Behavior analysis of NLI models: Uncovering the influence of three factors on robustness. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1975–1985, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1179. URL https://www.aclweb.org/anthology/N18-1179.

[71] Martin Schmitt and Hinrich Schütze. SherLIiC: A typed event-focused lexical inference benchmark for evaluating natural language inference. In *Proceedings of the 57th Annual Meeting of the Association*

for Computational Linguistics, pages 902–914, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1086. URL https://www.aclweb.org/anthology/P19-1086.

[72] Shubham Sharma, Yunfeng Zhang, Jesús M. Ríos Aliaga, Djallel Bouneffouf, Vinod Muthusamy, and Kush R. Varshney. Data augmentation for discrimination prevention and bias disambiguation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, AIES '20, page 358–364, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371100. doi: 10.1145/3375627.3375865. URL https://doi.org/10.1145/3375627.3375865.

[73] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[74] Chloe Rose Stuart-Ulin. Microsoft's politically correct chatbot is even worse than its racist one, 2018. URL https://qz.com/1340990/microsofts-politically-correct-chat-bot-is-even-worse-than-its-racist-one.

[75] Rohan Taori, Achal Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt. Measuring robustness to natural distribution shifts in image classification. *ArXiv*, abs/2007.00644, 2020.

[76] Ian Tenney, James Wexler, Jasmijn Bastings, Tolga Bolukbasi, Andy Coenen, Sebastian Gehrmann, Ellen Jiang, Mahima Pushkarna, Carey Radebaugh, Emily Reif, et al. The language interpretability tool: Extensible, interactive visualizations and analysis for nlp models. *arXiv preprint arXiv:2008.05122*, 2020.

[77] Johannes M. van Hulst, F. Hasibi, K. Dercksen, K. Balog, and A. D. Vries. Rel: An entity linker standing on the shoulders of giants. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[78] N. J. Wahl. An overview of regression testing. *ACM SIGSOFT Softw. Eng. Notes*, 24:69–73, 1999.

[79] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for nlp. *arXiv preprint arXiv:1908.07125*, 2019.

[80] Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. Allennlp interpret: A framework for explaining predictions of nlp models. *arXiv preprint arXiv:1909.09251*, 2019.

[81] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280, 2019.

[82] Jason W Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[83] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics*, 26(1):56–65, 2019.

[84] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume*

1 (Long Papers), pages 1112–1122. Association for Computational Linguistics, 2018. URL `http://aclweb.org/anthology/N18-1101`.

[85] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[86] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[87] Tongshuang Wu, Marco Tulio Ribeiro, J. Heer, and Daniel S. Weld. Errudite: Scalable, reproducible, and testable error analysis. In *ACL*, 2019.

[88] Jiacheng Xu and Greg Durrett. Neural extractive text summarization with syntactic compression. *ArXiv*, abs/1902.00863, 2019.

[89] Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, and Kentaro Inui. Do neural models learn systematicity of monotonicity inference in natural language? *arXiv preprint arXiv:2004.14839*, 2020.

[90] Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. Openattack: An open-source textual adversarial attack toolkit. *arXiv preprint arXiv:2009.09191*, 2020.

[91] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics*, 25(1):364–373, 2018.

[92] Jingqing Zhang, Y. Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. *ArXiv*, abs/1912.08777, 2020.

[93] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*, 2018.

[94] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, M. Zhou, and T. Zhao. Neural document summarization by jointly learning to score and select sentences. In *ACL*, 2018.

A Appendix

A.1 Commercial System Case Study

Pre-study questionnaire. We asked the user to fill out the first questionnaire before the user study session and the second one after the session. The pre-study form included the following questions: which NLP task is the team working on (sentiment, dialog, question answering, natural language inference, machine translation,

language modeling, summarization, and others), what metrics does the team use for evaluating their models (accuracy, P/R/F1, exact match, BLEU, ROUGE, or other generation metrics, and others), how they evaluate robustness (standard val/test datasets, out-of-distribution examples or datasets for generalization testing, axiomatic bias tests, adversarial attacks, and model cards). The form also asked the user to rate on a Likert scale of 1-5, 1 being strongly disagree and 5 being strongly agree, the following two statements: "I would like to evaluate the robustness of my model more thoroughly than I do today." and "I would benefit from having a library that gives me the tools to evaluate the robustness of my models." They rated the aforementioned agreement statements as 4/5 and 5/5 respectively.

Post-study questionnaire. The post-study questionnaire evaluated Robustness Gym in terms of ease of use and how likely they are to incorporate the gym in their workflow on a Likert scale of 1-5, 1 being "very unlikely" and 5 being "very likely". At the end of the study, the team rated "very likely" for both ease of use and eagerness for using Robustness Gym as part of their workflow.

One question was about rating the usefulness of the 4 evaluation idioms in the study. The team rated subpopulations and adversarial attacks as 5/5, transformations as 4/5, and eval sets as 3/5 on a scale of 1-5, 1 being "not useful" and 5 being "very useful". For the key takeaways of the study, the team found subpopulation slices as being "very insightful". They were very happy that they could first use adversarial attacks to probe for vulnerabilities and then use augmentations to fix them all in one tool.

A.2 Named Entity Linking

AIDA. For the AIDA test-b dataset, we follow [18] to split each passage in the dataset into examples. Each example corresponds to one sentence in the passage, pre-pended with the leading sentence that the passage starts with as context. We ignore predictions over the context sentence when calculating metrics.

A.3 Summarization

We describe the summarization slices in more detail below.

Abstractiveness. The degree to which the reference summary is abstractive versus extractive [24], based on the proportion of n-grams in the reference summary that are *not* in the article. Formally, we define the abstractiveness of a summary S given an article A as:

$$\mathtt{abstractiveness}(A, S) = 1 - \mathtt{rouge}_{\mathtt{precision}}(A, S)$$

based on several variations of Rouge (Rouge-1, Rouge-2, Rouge-L). Note that $\mathtt{rouge}_{\mathtt{precision}}(A, S)$ equals the proportion of n-grams in the reference summary that are also in the article. The abstractiveness metric is essentially the complement of the Extractive Fragment Coverage metric introduced in Grusky et al. [24].

Distillation. The degree to which the reference summary is distilled from a larger quantity of content, based on the proportion of n-grams in the article that do *not* appear in the reference summary:

$$\mathtt{distillation}(A, S) = 1 - \mathtt{rouge}_{\mathtt{recall}}(A, S)$$

Note that $\mathtt{rouge}_{\mathtt{recall}}(A, S)$ equals the proportion of n-grams in the article that appear in the reference summary.

We also consider 3 fine-grained metrics that rely on the similarities between sentences in the article and sentences in the reference summary. For these metrics, we define a sentence-similarity matrix M, where $M_{i,j}$ is a similarity score (e.g. Rouge-1) between sentence a_i in the article and sentence s_j in the summary. We provide the sentence-similarity matrix M as a built-in abstraction in Robustness Gym, from which a

variety of metrics may be decoded. Sharing this abstraction not only reduces code reuse, but also lowers the computational cost when performing multiple evaluations.

We also define a `match` function, which returns the index i of the sentence in the article with greatest similarity to the summary sentence s_j:

$$\texttt{match}(j) = \arg\max_i(M_{i,j})$$

Based on these formalisms, we define 3 metrics:

Position. The mean position of the matched sentences in the article:

$$\texttt{position}(A, S) = \sum_{j=1}^{N} \texttt{match}(j)/N$$

This metric is inspired by previous work showing that summarization models may be biased toward sentences at the beginning of an article [35, 38, 43].

Dispersion. The degree to which summary sentences match content that is distributed broadly across the article versus concentrated in a particular region. We define dispersion as the variance of the position of the matched sentences:

$$\texttt{dispersion}(A, S) = \sum_{j=1}^{N} (\texttt{match}(j) - \mu)^2/N$$

where μ is the mean match position, which equals $\texttt{position}(A, S)$, defined earlier. This metric is related to Extractive Fragment Density [24], which measures the degree to which extracted text in the summary comes from a contiguous sequence versus being broadly sourced from the article.

Order. The similarity in ordering between the summary sentences and the matched article sentences. Specifically, we compute the Spearman rank correlation between the positions of sentences in the reference summary and the positions of their matched counterparts in the article:

$$\texttt{order}(A, S) = \texttt{spearman}((\texttt{match}(j))_{j=1}^{N}, (j)_{j=1}^{N})$$

This metric is inspired by prior work in summarization evaluation that studied the effects of shuffling sentences in the source article, revealing a significant degradation in performance in news articles compared to other domains [38].

A.4 Code and Reports

Code. We provide example code snippets for Robustness Gym in Tables 4 (CachedOperation), 5 (Slice-Builder), and 6 (TestBench, Report), below.

LaTeX Report. Figure 8 is an example of a report generated in a LaTeX format. The code for the figure was auto-generated and the figure was simply included in the appendix.

Goal		Code Snippet
Caching	Create	**Create Spacy cached operation**
| | | ```python
spacy = Spacy()
``` |
| | | **Create Stanza cached operation** |
| | | ```python
stanza = Stanza()
``` |
| | | **Create a custom cached operation** |
| | | ```python
cachedop = CachedOperation(
    apply_fn=my_custom_fn,
    identifier=Identifier('MyCustomOp'),
)
``` |
| | | **Run a cached operation** |
| | | ```python
dataset = cachedop(dataset, columns)
``` |
| | Retrieve | **Retrieve all Spacy info cached** |
| | | ```python
Spacy.retrieve(dataset, columns)
``` |
| | | **Retrieve Spacy tokens** |
| | | ```python
Spacy.retrieve(batch, columns, 'tokens'
)
``` |
| | | **Retrieve Stanza entities** |
| | | ```python
Stanza.retrieve(
    batch,
    columns,
    Stanza.entities
)
``` |
| | | **Retrieve any cached operation info after processing** |
| | | ```python
CachedOperation.retrieve(
    batch,
    columns,
    my_proc_fn,
    'MyCustomOp'
)
``` |

Table 4: Code for the CachedOperation abstraction in Robustness Gym.

Goal		Code Snippet
Slice Building	**Subpopulations**	
	Create a subpopulation that generates three slices based on raw lengths in [0, 10], [10, 20] and [20, ∞)	``` length_sp = Length([(0, 10), (10, 20), (20, np.inf)]) ```
	Create a subpopulation that generates two slices based on bottom 10% and top 10% length percentiles	``` length_sp = Length([('0%', '10%'), ('90%', '100%')]) ```

	Create a custom subpopulation by binning the outputs of a scoring function	```
		custom_sp = ScoreSubpopulation(
		    [('0%', '10%'), ('90%', '100%')],
		    my_scoring_fn
		)
		``

	Transformations	
	Create EasyDataAugmentation	``` eda = EasyDataAugmentation() ```
	Create any NlpAug transformation	``` nlpaug_trans = NlpAugTransformation(pipeline=nlpaug_pipeline) ```

	Create a custom transformation	```
		custom_trans = Transformation(
		    Identifier('MyTransformation'),
		    my_transformation_fn
		)
		``

	Attacks	
	Create TextAttack recipe	``` attack = TextAttack.from_recipe(recipe, model) ```
	Evaluation Sets	
	Create a slice from a dataset	``` sl = Slice(dataset) ```
	Slice Builders	

	Run any SliceBuilder	```
		dataset, slices, membership =
		slicebuilder(
		    batch_or_dataset=dataset,
		    columns=columns,
		)
		``

Table 5: Code for the **SliceBuilder** abstraction in Robustness Gym.

Goal		Code Snippet
| | Create a testbench | ```
testbench = TestBench(
    identifier=Identifier('MyTestBench'),
    version='0.1.0'
)
``` |
| Testbench | Add slices to testbench | ```
testbench.add_slices(slices)
``` |
| | Fuzzy search testbench for slices | ```
top_k_matched_slices = testbench.search('len')
``` |
| | Bump testbench minor version | ```
testbench.bump_minor()
``` |
| | Save and load a testbench | ```
testbench.save(path)
testbench.load(path)
``` |
| | Evaluate model on slices and generate report | ```
testbench.create_report(model)
``` |
| Report | Create a custom report | ```
report = Report(
    dataframe_with_metrics,
    report_columns,
)
``` |
| | Generate figure from report | ```
figure = report.figure()
``` |
| | Generate LaTeX report | ```
latex = report.latex()
``` |

Table 6: Code for the TestBench and Report abstractions in Robustness Gym.

	Accuracy	F1	Class Dist (E N C)			Pred Dist (E N C)			Size	
Low Constituency Tree Overlap (McCoy, 2019)	90.2	89.7	20	39	41	20	39	41	2.1K	subpopulation
High Constituency Tree Overlap (McCoy, 2019)	93.2	92.2	53	24	23	51	24	25	1.99K	
Negation @ hypothesis (Naik, 2018)	90.8	86.0	22	17	61	23	13	64	109	
Negation @ premise (Naik, 2018)	79.5	79.5	31	38	31	38	26	36	39	
Possessive Preposition @ hypothesis (Chen, 2020)	90.9	90.9	39	34	27	36	35	29	585	
Quantifier @ hypothesis (Chen, 2020)	88.2	88.3	38	34	28	39	34	28	170	
Temporal Preposition @ hypothesis (Chen, 2020)	87.7	86.0	13	61	25	13	61	25	106	
Low Lexical Overlap (McCoy, 2019)	90.5	89.6	20	33	47	20	33	46	2.04K	
High Lexical Overlap (McCoy, 2019)	92.7	91.9	52	29	19	51	30	20	1.98K	
BAE (Garg, 2019)	80.3	78.4	13	58	29	12	48	40	2.92K	attack
Easy Data Augmentation (Wei, 2019)	82.3	82.2	34	33	33	28	36	36	9.84K	transform
Keyboard Character Errors (Ma, 2019)	65.8	65.4	34	33	33	24	33	44	9.14K	
Synonym Substitution (Ma, 2019)	75.4	75.1	34	33	33	24	36	40	9.84K	
SNLI (Bowman, 2015)	90.9	90.9	34	33	33	33	33	34	9.84K	evalset

Figure 8: Robustness report for textattack/bert-base-uncased-snli model on SNLI dataset. The report lays out scores for each evaluation, broken out by category. Citations: [7, 10, 46, 48, 54, 82].

Note: the LaTeX figure and caption above is auto-generated using "report.latex()".