Metamorphic Testing and Certified Mitigation of Fairness Violations in NLP Models

Pingchuan Ma[1,2] , **Shuai Wang**[1]* and **Jin Liu**[2]
[1]The Hong Kong University of Science and Technology
[2]Beijing Electronic Science and Technology Institute
pingchuan@ieee.org, shuaiw@cse.ust.hk, liujin@besti.edu.cn

Abstract

Natural language processing (NLP) models have been increasingly used in sensitive application domains including credit scoring, insurance, and loan assessment. Hence, it is critical to know that the decisions made by NLP models are free of unfair bias toward certain subpopulation groups. In this paper, we propose a novel framework employing metamorphic testing, a well-established software testing scheme, to test NLP models and find discriminatory inputs that provoke fairness violations. Furthermore, inspired by recent breakthroughs in the certified robustness of machine learning, we formulate NLP model fairness in a practical setting as (ϵ, k)-fairness and accordingly smooth the model predictions to mitigate fairness violations. We demonstrate our technique using popular (commercial) NLP models, and successfully flag thousands of discriminatory inputs that can cause fairness violations. We further enhance the evaluated models by adding certified fairness guarantee at a modest cost.

1 Introduction

Recent natural language processing (NLP) achievements have been depending on the progressive development of AI techniques. To date, machine learning (ML) models have become the cornerstone for most real-world NLP applications, such as text classification, information extraction, and question answering (QA). In particular, NLP systems have been commercialized for various real-life purposes (crime prediction, credit scoring, etc.), imposing a strong demand for the precision and reliability of the underlying models.

Despite this spectacular progress, however, recent studies have demonstrated that AI models are likely to encounter severe legal or ethical issues, when the decisions are unfairly made by being biased against certain subpopulations within sensitive attributes (e.g., those of a particular race, gender, nationality, or sexual orientation) [Kusner *et al.*, 2017; Bolukbasi *et al.*, 2016; Sheng *et al.*, 2019]. Indeed, recent studies have pointed out that black people are more likely to be classified as having negative emotions when completing emotion classification tasks [Rhue, 2018]. For sentiment analysis (SA) tasks, predictions could be changed by tokens related to identity groups as well [Garg *et al.*, 2019]. To date, the *fairness* of ML models has been violated in many unwanted and potentially harsh ways, raising deep concern regarding their rapid adoption in areas such as crime prediction, credit scoring, and hiring [Barocas *et al.*, 2019].

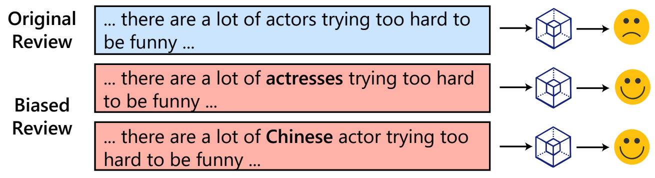
Original Review
... there are a lot of actors trying too hard to be funny ...
Biased Review
... there are a lot of **actresses** trying too hard to be funny ...
... there are a lot of **Chinese** actor trying too hard to be funny ...

Figure 1: Fairness violations in sentiment prediction.

We present a motivating example by training a CNN model for SA with the Large Movie Review Dataset.[1] Here, we present three samples with the corresponding labels and scores. It is easy to see that when sensitive attributes, including gender and nationality, are changed or explicitly specified in the samples, the results of the sentiment prediction would change and lead to a potentially unintended bias.

Existing research has proposed various techniques to expose fairness violations in NLP models. However, these works suffer from various drawbacks, such as relying on heavy-weight (unscalable) statistical or symbolic analysis tools, using pre-defined templates for input generation, or processing only structured data tables [Galhotra *et al.*, 2017; Udeshi *et al.*, 2018; Aggarwal *et al.*, 2019]. Inspired by relevant research in software engineering, we advocate for a focus on formulating the analysis of model fairness as a specific software testing task where typical NLP models are treated as a "black-box." In particular, we propose employing a well-established software testing scheme named *metamorphic testing* (MT) to expose discriminatory inputs that lead to unfair predictions of NLP models [Chen *et al.*, 1998]. To test software, MT works by 1) perturbing arbitrary inputs to generate a set of mutations, and 2) checking whether a given metamorphic relation (MR) still holds w.r.t. the mutated inputs. Each MR, usually an invariant, depicts necessary properties of the

*Corresponding Author

[1]https://ai.stanford.edu/~amaas/data/sentiment/

target software.

While defining MR to assert model fairness is straightforward (see Section 2), generating perturbed natural language inputs to test model discrimination is challenging. For instance, existing works identify and replace certain tokens with hand-coded mapping rules (e.g., "he" $\rightarrow$ "she") [Kiritchenko and Mohammad, 2018]. However, hand-coded rules are hard to fully explore subpopulations within arbitrary sensitive attributes and thus prone to missing certain discriminations. Defining such mappings also requires considerable manual effort. In this research, we leverage advanced NLP methods to systematically and automatically gather subpopulations within arbitrarily sensitive attributes and construct mappings to perturb *unstructured* natural language sentences. We further rule out "unrealistic" perturbations w.r.t. standard language fluency metrics [Ge *et al.*, 2018]. Hence, defects exposed in this work are primarily due to fluent and common sentences, presumably implying real-life issues.

We further propose techniques to enhance erroneous NLP models by mitigating fairness violations. To date, there are existing works on mitigating fairness failures in NLP models, e.g., with data augmentation and unbiased word embedding [Zhao *et al.*, 2018; Park *et al.*, 2018; Sun *et al.*, 2019]. Nonetheless, these works require to access and re-train erroneous models. In this research, we advocate for a new focus on mitigating fairness failures in the prediction phase, and therefore, the proposed technique can be applied to even black-box models. Inspired by recent breakthroughs in certified ML robustness [Lecuyer *et al.*, 2019; Mu *et al.*, 2019; Cohen *et al.*, 2019], we formalize fairness notation in terms of (ϵ, k)-fairness (cf. Section 3.1), and we propose a neutral phase that can piggyback on a (black-box) NLP model to smooth its outputs: the smoothed outputs are certified to preserve our (ϵ, k)-fairness.

Contribution. We propose MT-NLP, the first automated framework to identify and mitigate unfair predictions of NLP models. MT-NLP is designed as a pipeline to 1) employ advanced NLP techniques to systematically explore subpopulations within sensitive attributes identified in arbitrary natural language sentences, 2) perturb sentences regarding the identified sensitive attributes and perform MT on NLP models to detect fairness violations, and 3) formalize model fairness in a certified setting as (ϵ, k)-fairness and accordingly mitigate unfair predictions. As the proposed framework does *not* leverage any implementation detail of NLP models, it is robust to any kinds of ML models (e.g., DNN or SVM) in a black-box setting and therefore can directly analyze any (remote) models. Our extensive evaluation delineates the capabilities of de facto commercial NLP services provided by industry giants and also commonly-used local NLP models by exposing in total 2,874 discriminatory inputs (among which 441 are from commercial NLP services). We further enhance the evaluated (commercial) NLP models w.r.t. the certified guarantees at a modest cost.

2 Metamorphic Testing (MT)

Metamorphic testing (MT) has been widely used to automatically detect software defects [Chen *et al.*, 1998]. The strength of MT is due to its capability to assert software correctness via metamorphic relations (MRs). Each MR denotes necessary, usually invariant, properties of the software. For instance, to test the implementation of $\sin(x)$, instead of knowing the expected output of every input x, we can assert if a MR $sin(x) = sin(\pi - x)$ always holds when vastly mutating x. A properly defined MR alleviates the need for testing oracle (ground truth), thus making software testing significantly more flexible. We now define NLP model fairness addressed in this research and formulate the MT procedure accordingly.

Let $\mathcal{M} : \mathcal{X} \rightarrow \mathbb{R}^n$ be a model which takes a natural language sentence x as its input, model output is denoted as $\text{argmax}\, \mathcal{M}(x)$. Assume a perturbation function $\mathcal{P}$ perturbs x into a set S such that for every $x' \in S$, there exists a sensitive attribute $\mathcal{A}$ and $x_p \neq x'_{p'}$ where $p, p' \in \mathcal{A}$. In other words, $\mathcal{P}$ perturbs x by inserting or changing one token in x from p to p', assuming p and p' are both subpopulations within $\mathcal{A}$. x and x' deem a pair of *discriminatory inputs*, in case their model predictions are not identical. Hence, we check the following MR to identify fairness violations:

$$\text{argmax}\, \mathcal{M}(x) = \text{argmax}\, \mathcal{M}(x')$$

and the corresponding MT workflow can be formulated as:

```
Algorithm 1: MT for Model Fairness Violation
  Input: model $\mathcal{M} : \mathcal{X} \rightarrow \mathbb{R}^n$, input $x$, sensitive
         attribute $\mathcal{A}$, knowledge graph $G := (V, E)$
  Output: discriminatory input set $S_{dis}$
1 $S_{dis} \leftarrow \emptyset$;
2 $S \leftarrow \mathcal{P}(x, \mathcal{A}, G)$;
3 foreach $x' \in S$ do
4     if $\text{argmax}\, \mathcal{M}(x) \neq \text{argmax}\, \mathcal{M}(x')$ then
5         $S_{dis} = S_{dis} \cup \{x'\}$
6     end
7 end
8 return $S_{dis}$;
```

Consistent with existing research [Galhotra *et al.*, 2017; Udeshi *et al.*, 2018], sensitive attributes $\mathcal{A}$ (e.g., gender or race) are chosen by users to enable a more adaptive design. MT-NLP leverages a knowledge graph G during the perturbation stage (introduced shortly in Section 2.1). We now elaborate on the design of perturbation function $\mathcal{P}$.

2.1 Sentence Perturbator $\mathcal{P}$

As mentioned above, enabling an automated workflow to identify and mutate sensitive attributes in unstructured natural language sentences is challenging. Existing works leverage hand-coded mapping rules, and therefore, the capability of finding defects is limited by the currently known mappings. In contrast, we strive to systematically explore and mutate the sensitive attributes without using any predefined mappings.

Given a sentence written in natural language, our observation reveals two primary mutation opportunities: 1) nouns denoting subpopulations within sensitive attributes, and 2) adjectives assigning sensitive attributes toward a (neutral) noun. Inspired by this observation, $\mathcal{P}$ conceptually captures both noun and adjective tokens for perturbation. The high-level workflow is presented as follows:

Algorithm 2: Sentence Perturbator $\mathcal{P}$

```
Input: input $x$, sensitive attribute $\mathcal{A}$, knowledge graph
       $G := (V, E)$
Output: perturbation set $S_c$
1  $\mathcal{S}_C \leftarrow \emptyset$;
2  $T \leftarrow$PartOfSpeech($x$);
3  foreach $t \in T$ and $t$ is noun do
4      if IsPerson($t, G$) then
5          $S_C \leftarrow S_C \cup$AnalogyMutate($x, t, \mathcal{A}, G$);
6          $S_C \leftarrow S_C \cup$ActiveMutate($x, t, \mathcal{A}, G$);
7      end
8  end
9  $\mathcal{S}_C \leftarrow$FluencyFilter($S_C$);
10 return $S_c$;
```

The proposed perturbation procedure takes a natural language sentence x, a user chosen sensitive attribute $\mathcal{A}$ (e.g., race), and also a knowledge graph G as the inputs. Knowledge graph connects words with labeled edges, enabling a systematic understanding of natural language. Here, we utilize a popular knowledge graph, ConceptNet [Speer *et al.*, 2017], that is particularly designed for NLP tasks. Each node on ConceptNet represents a word or phrase of natural language, and labeled edges on ConceptNet, such as `IsA`, `UsedFor`, and `CapableOf`, denote relations between connected nodes. To implement `IsPerson`, `AnalogyMutate`, and `ActiveMutate` (introduced soon), we extensively use edge `IsA` on ConceptNet, which throughly explores "attributes" of a given node. For instance, by querying the `IsA` edges of "gay", we know that "gay" is a type of "lgbt", a sensitive attribute which can likely provoke fairness violations. Note that `IsA` represents asymmetric relation. For the rest of the paper, we use `IsA` to denote the forward query while $\texttt{IsA}_{rev}$ is vice versa.

Perturbator $\mathcal{P}$ starts by performing standard part-of-speech (POS) analysis to assign a POS tag (e.g., noun, verb, and adjective) to each word [Manning *et al.*, 2014]. Noun tokens will be kept for use (line 3). We then check whether a noun token refers to a person (line 4; `IsPerson` will be explained soon). Given the sentence and the pinpointed tokens, we then propose two strategies to generate testing inputs by changing either the noun tokens with `AnalogyMutate` (line 5) or their associated adjective tokens with `ActiveMutate` (line 6). In addition, while the "naturalness" of the synthesized sentences does not have a direct influence on testing, we augment the realism of the test inputs by preserving synthesized sentences that are "fluent" (line 9). Hence, synthetic sentences triggering fairness violations could presumably imply real-life harshness. We implement a standard sentence fluency metrics proposed in [Ge *et al.*, 2018].

We now elaborate on the motivation and design of `IsPerson`. Given the highly flexible meanings of natural language tokens, our observation shows that perturbing *arbitrary nouns* does not always generate useful sentences for testing fairness violations. For instance, it could likely become confusing to change "pen" into "pencil." Hence, the application scope of our analysis is fine-tuned to focus on the most common setting — human related factors. Intuitively, "model fairness" is highly correlated to attributes of people. In fact, this design choice is consistently taken by the majority, if not all, of previous research in this field [Galhotra *et al.*, 2017; Udeshi *et al.*, 2018; Aggarwal *et al.*, 2019].

Let node denoting "human" on ConceptNet be h, to decide whether t represents a human being, `IsPerson` is implemented to check the reachability between h and t via `IsA` edges. For instance, "terrorist" and "policewoman" both deem human-related nouns, since "terrorist" node directly connects to h, and "policewoman" can also reach to h via two hops on the graph. Also, for each `IsA`, ConceptNet assigns a "weight" to denote the confidence score of such knowledge relationship. When querying the reachability, `IsPerson` is empirically designed to use edges with weight larger than 1.0. Besides h, some other nouns can assuredly represent human as well. Hence, we summarize a set, $\mathcal{I}_h$ (see Appendix A), to subsume human-related nouns. t deems a human-related noun, only if at least one $h' \in \mathcal{I}_h$ is k-hop reachable by t (k is configured as two for the implementation). To speedup the search, we also empirically summarized some non-human tokens as the early stop criteria of `IsPerson`.

Analogy Mutation

Limiting the perturbation target to only human-related tokens progressively improves the quality of the perturbed sentences. Nevertheless, it is still challenging to present a flexible design for mutating noun tokens. In general, while there are character-level mappings that can be used to mutate certain words, for instance, changing "er" to "ress" in nouns such as "waiter", these rules suffer from limited generality and may fail to find similar tokens in many corner cases (e.g., "actor" $\rightarrow$ "actress" where "actor" does not end with "er").

At this step, we propose a flexible and effective analogy mutation method, `AnalogyMutate`, that is inspired by the well-known "word analogy" techniques in manipulating word embedding vectors [Ethayarajh *et al.*, 2019]. Intuitively, if the relations between two pairs of words are similar, the distances between the two pairs of word-embedding vectors are similar. Hence, given d which is a noun token and p, p' which are subpopulations within a sensitive attribute, we obtain d', an analogy of d, with the following equation:

$$W(d') = W(d) + W(p') - W(p)$$

where $W(\cdot)$ denotes the embedding vector of a given word.

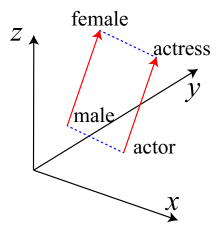

Figure 2: The parallelogram structure of analogous words.

Algorithm 3: Analogy Mutation

```
Input: input $x$, token $t$, sensitive attribute $\mathcal{A}$;
       knowledge graph $G := (V, E)$, word
       embedding table $W : t \mapsto v$
Output: Analogy Mutation Set $S$
1  $S \leftarrow \emptyset$;
2  $S_p \leftarrow G$.IsA$_{rev}$($\mathcal{A}$);
3  $p_t \leftarrow \text{argmin}_{p_i \in S_p} \|W(p_i) - W(t)\|_2$;
4  $S_p \leftarrow S_p \setminus \{p_t\}$ // set minus;
5  foreach $p_i \in S_p$ do
6      $t_M \leftarrow W^{-1}(W(p_i) + W(t) - W(p_t))$;
7      $x' \leftarrow x$.replace($t, t_M$);
8      $S \leftarrow S \cup \{x'\}$;
9  end
10 return $S$;
```

Inspired by word analogy techniques, the above equation provides an effective way of mutating d according to certain sensitive attributes. For instance, assume p and p' are subpopulations within the "gender" attribute (e.g., p as "male" and p' as "female"), Fig. 2 provides a sample mapping by mutating "actor" to its analogy noun "actress" which can potentially trigger discrimination on "gender." Similarly, given p as "Christianity" and p' as "Islamism", the given translation can map "Church" into "Mosque."

The `AnalogyMutate` is defined in Algorithm 3. Given a sensitive attribute chosen by users, we first query the knowledge graph and backwardly fetch its type nodes (line 2). For instance, suppose "gender" is the chosen sensitive attribute $\mathcal{A}$, knowledge graph G can help to collect nodes who are "types" of gender, including both "male" and "female." We then iterate each element in S_p and identify p_t whose word embedding has the shortest distance with t; we exclude this word (line 4), since that presumably indicates a subpopulation (e.g., "male") within $\mathcal{A}$ that is highly related to t (e.g., "actor"). line 5–9 concretize the analogy mutation introduced above with the inverse function of $W(\cdot)$ and iterates remaining tokens p_i (e.g., "female") within S_p.

Active Mutation

Algorithm 4 presents `ActiveMutate` used in Algorithm 2. To avoid potential conflict, we rule out t if it already has adjectives ahead. This can be easily done with part-of-speech analysis and is omitted in Algorithm 4. In addition, to prevent awkward perturbations such as "female actor", we check if t is "neutral" w.r.t. attribute $\mathcal{A}$. To this end, we extract subpopulations within a given sensitive attribute $\mathcal{A}$ by querying the knowledge graph (line 2). Then, we iterate each element $p_i \in S_p$ and leverage function `HasIsAEdge` to check if p_i is reachable by t on the graph (line 4); t is "neutral" w.r.t. p_i if no edge is found in between. If t is neutral, we re-iterate S_p (line 8) and insert a sensitive adjective ahead (line 9).

Algorithm 4: Active Mutation

```
Input: input $x$, token $t$, sensitive attribute $\mathcal{A}$;
       knowledge graph $G := (V, E)$
Output: Active Mutation Set $S$
1  $S \leftarrow \emptyset$;
2  $S_p \leftarrow G$.IsA$_{rev}$($\mathcal{A}$);
3  foreach $p_i \in S_p$ do
4      if $G$.hasIsAEdge($t, p_i$) then
5          return $\emptyset$;
6      end
7  end
8  foreach $p_i \in S_p$ do
9      $t_M \leftarrow p_i + t$ // modify $t$ with adj. $p_i$;
10     $x' \leftarrow x$.replace($t, t_M$);
11     $\mathcal{S} \leftarrow \mathcal{S} \cup \{x'\}$;
12 end
13 return $\mathcal{S}$;
```

3 Certified Mitigation

Recent advances in certified robustness provide a rigorous framework for provable defense against adversarial examples. However, it is generally non-trivial to extend existing methods to NLP tasks with discrete domain since the magnitude of adversarial sentences are not measurable in practice. Nevertheless, given an arbitrary sentence x, since we have constituted its perturbation set S, set S indeed enables the certified mitigation (introduced soon). Also, while related works propose to employ statistical methods like Interval Bound Propagation to achieve verified NLP model robustness against adversarial word substitutions [Huang *et al.*, 2019; Jia *et al.*, 2019], we note that these works cannot directly enhance black-box NLP models.

3.1 (ϵ, k)-Fairness

In this section, we formulate fairness for individual samples derived from the expected output stability bound [Lecuyer *et al.*, 2019].

Definition 3.1 A function $f : \mathcal{X} \rightarrow [0, b)^N$ and a sample x_0 are (ϵ, k)-fairness w.r.t. sensitive attribute $\mathcal{A}$, if $\exists S \subseteq \mathcal{X}_{\mathcal{A}}, |S| = k, \forall x, x' \in \{x_0\} \cup S$ subject to

$$f(x) \preceq e^{\epsilon} f(x') \quad (1)$$

where $\mathcal{X}_{\mathcal{A}}$ denotes the set of all possible mutations of x_0 on $\mathcal{A}$, $|S|$ denotes the cardinality of S and $\epsilon \geq 0$.

Intuitively, the definition requires that the outputs of at least k biased mutations are bounded by the fairness parameter ϵ. Naturally, higher k indicates a higher confidence toward the stated fairness parameter. In a special case where $\epsilon = 0$, the outputs of all the k mutations are equal.

3.2 Neutral Layer Design

Enabled by the (ϵ, k)-fairness formulation, this section proposes the design of a neutral layer that can piggyback the (black-box) NLP models and mitigate its fairness violation. To this end, we reuse the set of perturbed sentence S generated in the testing phase.

Inspired by the binary version of random response [Warner, 1965] and how it is generalized into a multivalue version by the staircase mechanism [Kairouz *et al.*, 2016], we design a certified mitigation mechanism to achieve (ϵ, k)-fairness as follows:

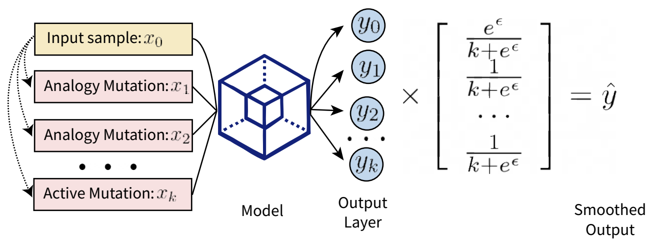
Input sample: x_0
Analogy Mutation: x_1
Analogy Mutation: x_2
Active Mutation: x_k
Model
y_0
y_1
y_2
y_k
Output Layer
$\times \begin{bmatrix} \frac{e^\epsilon}{k+e^\epsilon} \\ \frac{1}{k+e^\epsilon} \\ \cdots \\ \frac{1}{k+e^\epsilon} \end{bmatrix} = \hat{y}$
Smoothed Output

Figure 3: Workflow of the certified mitigation mechanism.

Theorem 1. *Given a model $\mathcal{M}$, an input x and its corresponding set of perturbations S, the certified model f is under (ϵ, k)-fairness guarantee if*

$$f(x) = \frac{e^\epsilon}{k + e^\epsilon}\mathcal{M}(x) + \sum_{x' \in S} \frac{1}{k + e^\epsilon}\mathcal{M}(x') \tag{2}$$

The certified mechanism allocates weights to each element in $\{x\} \cup S$ and yields a smoothed output $\hat{y}$. Overall, this method imposes no need to change the model training process, but enabling provable mitigations at inference time. Hence, it is generic and applicable toward arbitrary (black-box) models. Fig. 3 illustrates the mitigation process over input sample x_0 by using its corresponding perturbed sentences. We further give proof of Theorem. 1 on its fairness guarantee in Appendix B.

As depicted in Fig. 3, the proposed mitigation leads to additional computing overhead. Overall, for each input x, we compute its perturbation set S, incurring k additional predictions (recall $|S| = k$). Nevertheless, since the prediction phase of each sample is independent, it is feasible to parallelize the $k + 1$ prediction phases and reduce the processing time.

4 Evaluation

While the proposed technique is capable of testing any NLP task, in the evaluation, we decide to focus on one widely-used NLP task, sentiment analysis (SA). SA has been well commercialized for daily usage and constantly reported to be affected by discriminatory inputs [Dixon *et al.*, 2018; Kiritchenko and Mohammad, 2018].

MT-NLP allows users to chose any reasonable sensitive attributes $\mathcal{A}$ as inputs. In the evaluation, we take "gender", one frequently concerned factor in fairness, as the sensitive attributes of $\mathcal{A}$. We leave it as one future work to equip MT-NLP with other sensitive attributes, for instance "religion."

4.1 Perturbation

From 20,000 sample sentences in the Large Movie Review test dataset, our perturbator $\mathcal{P}$ finds 17,165 "mutable" sentences; mutable sentences contain at least one human-related noun token (see `IsPerson`). $\mathcal{P}$ then generates in total 174,183 perturbed sentences as the test inputs, among which 30,859 are produced by analogy mutations and 143,324 are from active mutations. Hence, for one mutable sentence x, the average size of its perturbation set S is 10.2, indicating that the proposed technique reveals considerable amount of test inputs for arbitrary natural language sentence.

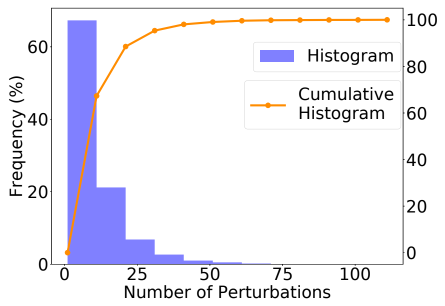

Figure 4: Perturbation distribution.

NLP Models and Services	# of Tested Sentences	# of Fairness Violations
Amazon Comprehend API	28,425	197
Google NLP API	30,504	140
Microsoft NLP API	28,721	104
CNN	174,183	1,111
Logic Regression (LR)	174,183	1,322

Table 1: Result overview.

Fig. 4 studies the perturbation results by reporting the distribution of number of perturbations derived from an input sentence x. We report that about 67.0% inputs have more than 5 perturbations, and 37.2% inputs have over 10 perturbations. The largest perburbation set for an input is 116. Overall, the promising results demonstrate that the proposed technique can automatically synthesize large amount of perturbations from arbitrary sentences, thereby enabling the systematic testing of NLP models. Furthermore, recall that given the definition of (ϵ, k)-fairness, k, denoting $|S|$, quantifies the confidence toward the stated fairness over input x. In other words, the perturbed sentences enable high confidence of certified mitigation, as will be discussed shortly in Section 4.3.

When perturbing sentences, the leveraged knowledge graph, ConceptNet, is hosted remotely. Hence, the perturbation time largely depends on the local network bandwidth. Nevertheless, we still report that our perturbation takes roughly 28 hours (on average 715 samples per hour). We interpret the overall processing time as practical and promising.

4.2 Testing Fairness Violations

Table 1 reports the evaluation results. We use three commercial SA services provided by Google, Microsoft, and Amazon for the evaluation. To our knowledge, the NLP models employed in these commercial services are not disclosed. We wrote Python scripts to query with these remote services and parse their outputs (in JSON format). In addition, we locally trained two SA models with the Large Movie Review training dataset [Maas *et al.*, 2011]. The CNN model, implemented in Keras (ver. 2.2.4), has one convolutional layer followed by two fully-connected layers. We use the standard pre-processing module of the IMDB data for embedding. As for the LR model, we use the default setting provided in Scikit-Learn (ver. 0.22.1) and BoW embedding.

While considerable amount of perturbed sentences are generated, all three commercial NLP services have limits of maximal input length that can be accepted. That is, not every sen-

NLP Models and Services	Original Sentence	Perturbed Sentence	Prediction Change
Amazon Comprehend API	. . . train to make their way to L.A. to see his **uncle** to lift the curse . . .	. . . train to make their way to L.A. to see his **aunt** to lift the curse . . .	negative $\rightarrow$ positive
Microsoft NLP API	. . . favorites are the O'Reily **brothers** . . .	. . . favorites are the O'Reily **sisters** . . .	positive $\rightarrow$ negative
CNN	. . . a gold train filled with Union soldiers . . .	. . . a gold train filled with Union **male** soldiers . . .	positive $\rightarrow$ negative

Table 2: Fairness violations detected by MT-NLP from commercial and local models.

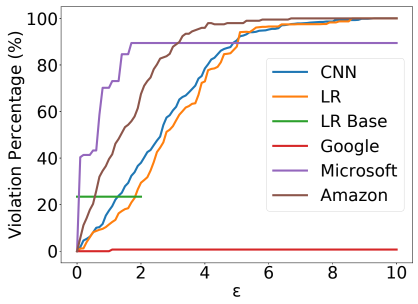

Figure 5: Fairness violation percentage w.r.t. different ϵ.

tence can be smoothly analyzed. The second column of Table 1 reports the number of successfully processed sentences, and the third column reports the number of triggered fairness violations. In general, we report that considerable number of fairness violations can be found from all the models. We interpret this as promising and intuitive; MT procedure implemented in MT-NLP is agnostic toward the specific model implementation, and it performs *consistently* across all these SA tools, despite the differences in the underlying models.

More discriminatory inputs can be found from local models compared with remote commercial services. While the implementation details of these commercial tools are not disclosed, our observation reveals an interesting finding: commercial NLP services seem to perform *pre-process* and trim off certain sensitive adjective tokens (e.g., "girly") which could likely lead to unwanted and harsh discriminations. As a result, most of the discriminatory inputs found from commercial APIs are generated by analogy mutation which directly perturbs noun tokens. In contrast, large amount of discriminatory inputs generated by both mutation strategies are found when testing the local models.

Table 2 presents three typical discriminatory inputs found by MT-NLP. The first example is found when testing the Amazon service. MT-NLP identified the gender-related noun **uncle** and successfully inferred the corresponding alternative token, **aunt**, with analogy mutation. Similarly, the second case changes **brother** into **sister** with analogy mutation. The last sample presents a sentence perturbed with active mutation on the local CNN model: a gender-related adjective, **male**, was inserted into the original sentence and provoked a fairness violation.

4.3 Certified Mitigation

By piggybacking the neutral layer toward a NLP model, we enforce the (ϵ, k)-fairness guarantee. In this section, we evaluate the effectiveness in terms of different settings. In general, ϵ quantifies the strength of the fairness guarantee, and for sufficiently small ϵ, perturbing sensitive attributes should not change the model output. Fig. 5 reports the evaluation results over different NLP models and services ("LR Base" is a baseline derived from adversarial training; will explain shortly). The percentage is calculated, by dividing the number of violations after mitigation with the total number of violations found before mitigation (i.e., the third column of Table 1).

Fig. 5 reports promising results for all the NLP models, despite that they presumably share different model structures and settings. We report that when ϵ is 0.1, on average more than 98.0% violation cases can be mitigated. Nevertheless, when ϵ is increased to 10, only certain amount of violations in Microsoft API (10.6%) and Google API (99.3%) can be mitigated. None violations are mitigated in others (therefore their corresponding percentages are "100%" in Fig. 5).

The sentiment analysis results of Google API can be decided according to the sign of its return value (a float number f). f manifests a negative sentiment, if it was less than zero, and vice versa. Also, the sentiment is "neutral", if f is zero. In other words, a small perturbation on f, e.g., $0 \rightarrow 0.001$, is sufficient to change its label from "neutral" to "positive". We inspect the fairness violations of Google API, and find most violations are misclassified to "neutral" from "positive" or "negative." Hence, even ϵ reaches 10 where the impact of other mutations is negligible, the mitigation can fix most erroneous predictions, by still slightly drifting f from zero.

We also design a comparison group (the "LR Base" line) where a LR model is adversarially trained using the original dataset and corresponding mutations generated by our perturbator $\mathcal{P}$. We report that with adversarial training, LR model becomes resilient toward 76.6% discriminatory inputs (only 23.4% violations, as shown in Fig. 5). In comparison, the "LR" data in Fig. 5, when ϵ is less than 1.8, achieves a better mitigation rate. This comparison demonstrates that our mitigation technique, with sufficiently small ϵ, can achieve even higher security guarantee compared with white-box adversarial training.

As for the cost, since our mitigation extends each prediction x into $k + 1$ predictions on $\{x\} \cup S$, the CPU time of each prediction becomes linear to k. As aforementioned, k is on average 10.2, which can be well parallelized on modern multi-core CPUs. Hence, we envision the elapsed real time should not vary largely in real-world usages. Overall, we interpret the evaluation results as promising: small ϵ lead to practical mitigation of fairness violations at a modest cost.

5 Related Work

Software testing techniques has been applied for testing AI applications driven by machine learning or deep learning models. To this end, well-established testing techniques, including metamorphic testing [Wang *et al.*, 2019; Dwarakanath *et al.*, 2018; Zhang *et al.*, 2018], differential testing [Pei *et al.*, 2017; Tian *et al.*, 2018], and fuzz testing [Xie *et al.*, 2018], have been successfully leveraged in detecting errors of CV applications [Tian *et al.*, 2018; Pei *et al.*, 2017; Zhang *et al.*, 2018] and NLP applications including machine translation [Sun and Zhou, 2018; He *et al.*, 2020; Yan *et al.*, 2019] and chatbots [Bozic and Wotawa, 2019].

AI model fairness has been particularly formulated as a software property and analyzed by software engineering techniques. To date, mutation testing techniques are proposed to detect discrimination in NLP models [Galhotra *et al.*, 2017; Udeshi *et al.*, 2018]. [Aggarwal *et al.*, 2019] introduces symbolic execution and local explainability to generate inputs in black-box scenarios. Nevertheless, existing research essentially uses pre-defined mapping rules (e.g., "male" $\rightarrow$ "female") to perturb *structured data tables* (e.g., the Kaggle Bank Marketing Dataset [Martinez, 2018]). In contrast, this work aims to perturb unstructured natural language sentences, thus extensively extending the application scope.

6 Conclusion

We have presented MT-NLP, a throughout and automated tool to pinpoint and mitigate fairness violations of NLP models. Our evaluation has found thousands of discriminatory inputs from commercial and local NLP models, and accordingly smooth model predictions with certified guarantees.

A Human-Related Noun Word Set $\mathcal{I}_h$

We empirically define $\mathcal{I}_h$ as:
{"human", "people", "person", "human_adult", "employee"}

B Proof of Theorem 1

In this section, we proof that the mechanism in Theorem 1 is indeed under (ϵ, k)-fairness guarantee. Let $\mathcal{M}$ denote an arbitrary NLP model, where $\mathcal{X}$ is the input domain. We first prove the case where $\mathcal{M} : \mathcal{X} \rightarrow [0, b)$, then generalize to multi-dimensional cases. As introduced in Section 2, by mutating an input x_0, we create a set of semantics-preserving mutations $\mathcal{S}_p = \{x_1, \cdots, x_k\}$. Then according to Theorem 1, we have f

$$f(x_0) = \frac{e^\epsilon}{k + e^\epsilon}\mathcal{M}(x_0) + \sum_{i=1}^{k} \frac{1}{k + e^\epsilon}\mathcal{M}(x_i) \qquad (3)$$

Derived from the definition, we can obtain a corollary. Given x, x', if $\mathcal{M}(x) \le \mathcal{M}(x')$, then $f(x) \le f(x')$ accordingly.

For the ease of presentation, given a set of mutations $\{x_0, x_1, \cdots, x_k\}$, let $\{y_0, y_1, \cdots, y_k\}$ denote the output of $\mathcal{M}$ and $\{\hat{y_0}, \hat{y_1}, \cdots, \hat{y_k}\}$ denote the output of f, respectively. Also, without loss of generality, let $i, j \in [k + 1]$, where $[k + 1] := \{0, 1, \cdots, k\}$.

Then,

$$\begin{aligned}\frac{\hat{y_i}}{\hat{y_j}} &= \frac{(e^\epsilon - 1)y_i + \sum y_n}{(e^\epsilon - 1)y_j + \sum y_n} \\ &= \frac{(e^\epsilon - 1)y_i + \sum y_n - e^\epsilon[(e^\epsilon - 1)y_j + \sum y_n]}{(e^\epsilon - 1)y_j + \sum y_n} + e^\epsilon \\ &= \frac{(1 - e^\epsilon)(\sum_{n \ne i} y_n + e^\epsilon y_i)}{(e^\epsilon - 1)y_j + \sum y_n} + e^\epsilon \le e^\epsilon\end{aligned} \qquad (4)$$

holds. Therefore, $\forall x, x' \in \{x_0\} \cup \mathcal{S}_p$, $\frac{f(x)}{f(x')} \le \frac{f(x_i)}{f(x_j)} \le e^\epsilon$. Then, applying the inequation to multi-dimensional models iteratively, $f(x) \preceq e^\epsilon f(x')$ holds.

References

[Aggarwal *et al.*, 2019] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. Black box fairness testing of machine learning models. In *ACM ESEC/FSE*, pages 625–635. ACM, 2019.

[Barocas *et al.*, 2019] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and Machine Learning*. fairmlbook.org, 2019. http://www.fairmlbook.org.

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

[Bozic and Wotawa, 2019] Josip Bozic and Franz Wotawa. Testing chatbots using metamorphic relations. IFIP-ICTSS, pages 41–55, 2019.

[Chen *et al.*, 1998] Tsong Y Chen, Shing C Cheung, and Shiu Ming Yiu. Metamorphic testing: a new approach for generating next test cases. Technical report, Technical Report HKUST-CS98-01, 1998.

[Cohen *et al.*, 2019] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, pages 1310–1320, 2019.

[Dixon *et al.*, 2018] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. 2018.

[Dwarakanath *et al.*, 2018] Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M. Rao, R. P. Jagadeesh Chandra Bose, Neville Dubash, and Sanjay Podder. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. ISSTA 2018, pages 118–128, 2018.

[Ethayarajh *et al.*, 2019] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. ACL, 2019.

[Galhotra *et al.*, 2017] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. Fairness testing: testing software for discrimination. In *ACM ESEC/FSE*, pages 498–510. ACM, 2017.

[Garg *et al.*, 2019] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. Counterfactual fairness in text classification through robustness. In *AIES*, pages 219–226. ACM, 2019.

[Ge *et al.*, 2018] Tao Ge, Furu Wei, and Ming Zhou. Fluency boost learning and inference for neural grammatical error correction. In *ACL*, pages 1055–1065, 2018.

[He *et al.*, 2020] Pinjia He, Clara Meister, and Zhendong Su. Structure-invariant testing for machine translation. ICSE '20, 2020.

[Huang *et al.*, 2019] Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving verified robustness to symbol substitutions via interval bound propagation. *arXiv preprint arXiv:1909.01492*, 2019.

[Jia *et al.*, 2019] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. *arXiv preprint arXiv:1909.00986*, 2019.

[Kairouz *et al.*, 2016] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *JMLR*, 17(1):492–542, 2016.

[Kiritchenko and Mohammad, 2018] Svetlana Kiritchenko and Saif M. Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. *CoRR*, abs/1805.04508, 2018.

[Kusner *et al.*, 2017] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NIPS*, pages 4066–4076. Curran Associates, Inc., 2017.

[Lecuyer *et al.*, 2019] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. Certified robustness to adversarial examples with differential privacy. In *IEEE S&P*, pages 656–672. IEEE, 2019.

[Maas *et al.*, 2011] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *ACL*, pages 142–150. Association for Computational Linguistics, 2011.

[Manning *et al.*, 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60, 2014.

[Martinez, 2018] Janio Martinez. Kaggle Bank Marketing Dataset, 2018.

[Mu *et al.*, 2019] J. Mu, P. Liang, and N. Goodman. Shaping visual representations with language for few-shot classification. *arXiv preprint arXiv:1911.02683*, 2019.

[Park *et al.*, 2018] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. *arXiv preprint arXiv:1808.07231*, 2018.

[Pei *et al.*, 2017] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *SOSP*, pages 1–18. ACM, 2017.

[Rhue, 2018] Lauren Rhue. Racial influence on automated perceptions of emotions. *Available at SSRN 3281765*, 2018.

[Sheng *et al.*, 2019] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. *arXiv:1909.01326*, 2019.

[Speer *et al.*, 2017] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, 2017.

[Sun and Zhou, 2018] Liqun Sun and Zhi Quan Zhou. Metamorphic testing for machine translations: MT4MT. ASWEC' 18, pages 96–100, 2018.

[Sun *et al.*, 2019] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating gender bias in natural language processing: Literature review. *arXiv:1906.08976*, 2019.

[Tian *et al.*, 2018] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. DeepTest: Automated testing of deep-neural-network-driven autonomous cars. ICSE '18, 2018.

[Udeshi *et al.*, 2018] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. Automated directed fairness testing. In *ACM/IEEE ASE*, pages 98–108. ACM, 2018.

[Wang *et al.*, 2019] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. Adversarial sample detection for deep neural network through model mutation testing. ICSE '19, pages 1245–1256, 2019.

[Warner, 1965] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[Xie *et al.*, 2018] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Hongxu Chen, Minhui Xue, Bo Li, Yang Liu, Jianjun Zhao, Jianxiong Yin, and Simon See. Coverage-guided fuzzing for deep neural networks. *arXiv preprint arXiv:1809.01266*, 2018.

[Yan *et al.*, 2019] Boyang Yan, Brian Yecies, and Zhi Quan Zhou. Metamorphic relations for data validation: a case study of translated text messages. In *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)*, pages 70–75. IEEE, 2019.

[Zhang *et al.*, 2018] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. DeepRoad: GAN-based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. ASE, 2018.

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