Imitation Attacks and Defenses for Black-box Machine Translation Systems

Eric Wallace **Mitchell Stern** **Dawn Song** **Dan Klein**
UC Berkeley
{ericwallace, mitchell, dawnsong, klein}@berkeley.edu

Abstract

We consider an adversary looking to *steal* or *attack* a black-box machine translation (MT) system, either for financial gain or to exploit model errors. We first show that black-box MT systems can be stolen by querying them with monolingual sentences and training models to imitate their outputs. Using simulated experiments, we demonstrate that MT model stealing is robust to imitation models having different input data and architectures than their victims. Applying these ideas, we train imitation models that reach within 0.6 BLEU of three production MT systems on both high-resource and low-resource language pairs. We then leverage the similarity of our imitation models to transfer adversarial examples to the production systems. We use gradient-based attacks that expose inputs which lead to semantically-incorrect translations, dropped content, and vulgar model outputs. To mitigate these vulnerabilities, we propose a defense that modifies translation outputs in order to misdirect the optimization of the imitation model. This defense degrades imitation model BLEU and attack transfer rates at some cost in victim BLEU and inference speed.

1 Introduction

NLP models deployed through APIs (e.g., Google Translate) can be a lucrative asset for an organization. These models are typically the result of a considerable investment—up to millions of dollars—into private data annotation and algorithmic improvements. Consequently, such models are kept hidden behind black-box APIs to protect system integrity and intellectual property.

We consider an adversary looking to *steal* or *attack* a black-box machine translation (MT) system. Stealing a production model allows an adversary to avoid long-term API costs or to launch a competitor service. Moreover, attacking an MT system using adversarial examples (Szegedy et al., 2014) enables an adversary to expose egregious translations that harm system owners or users. In this work, we investigate these two exploits: we first steal (we use "steal" following Tramèr et al. 2016) production systems by training *imitation models* and then use these imitation models to generate adversarial examples for production systems.

We create imitation models by borrowing ideas from knowledge distillation (Hinton et al., 2014): we query production MT systems with monolingual sentences and train imitation (i.e., student) models to mimic the system outputs (top of Figure 1). We first experiment with simulated studies which demonstrate that MT models are easy to imitate (Section 3). For example, imitation models closely replicate victim outputs even when they are trained using different model architectures or on out-of-domain queries. Applying these ideas, we imitate production systems from Google, Bing, and Systran with high fidelity on English→German and Nepali→English. For example, Bing achieves 32.9 BLEU on WMT14 English→German and our imitation achieves 32.4 BLEU.

We then demonstrate that our imitation models aid adversarial attacks against production MT systems (Section 4). In particular, the similarity of our imitation models and the production systems allows for direct transfer of adversarial examples obtained via gradient-based attacks. We find small perturbations that cause targeted mistranslations (e.g., bottom of Figure 1), nonsense inputs that produce malicious outputs, and universal phrases that cause mistranslations or dropped content.

The reason we identify vulnerabilities in NLP systems is to robustly patch them. To take steps towards this, we create a defense which finds alternate translations that cause the gradient of the imitation model to point in the wrong direction (Section 5). These alternate translations slightly hurt victim BLEU, but they cause more significant declines in imitation model BLEU and lower adversarial example transfer rates.

2 How We Imitate MT Models

We have query access to the predictions (no probabilities or logits) from a *victim* MT model. This victim is a black-box: we are unaware of its in-

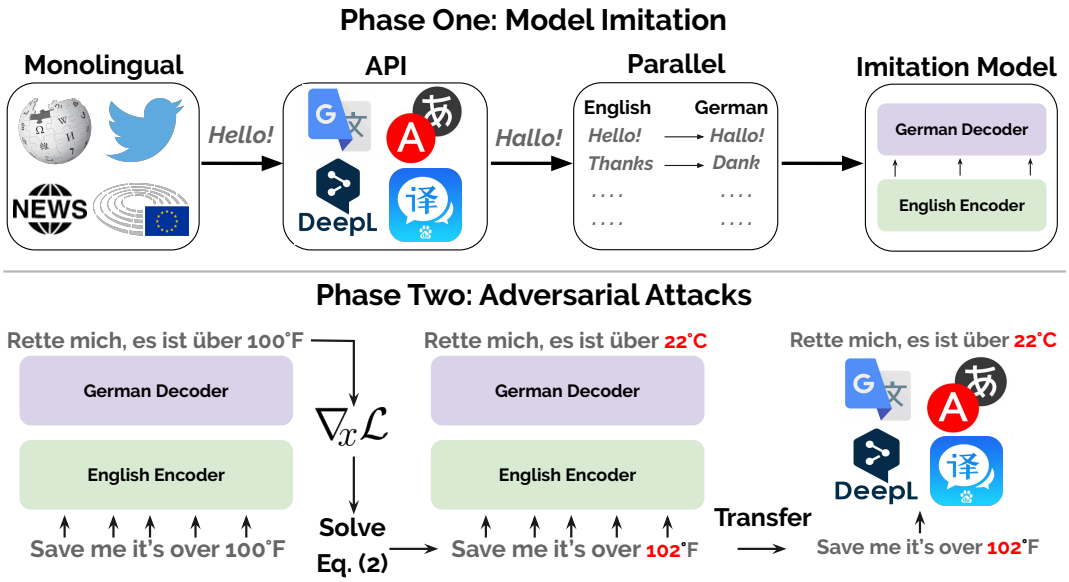

Figure 1: *Imitating and attacking an English→German MT system.* In phase one (model imitation), we first select sentences from English monolingual corpora (e.g., Wikipedia), label them using the victim API, and then train an imitation model on the resulting data. In phase two (adversarial attacks), we generate adversarial examples against our imitation model and transfer them to the production systems. For example, we find an input perturbation that causes Google to produce a factually incorrect translation, see the **link here** (all attacks work as of April 2020).

ternals, e.g., the model architecture, hyperparameters, or training data. Our goal is to train an *imitation model* (Orekondy et al., 2019) that achieves comparable accuracy to this victim on held-out data. Moreover, to enhance the transferability of adversarial examples, the imitation model should be functionally similar to the victim, i.e., similar inputs translate to similar outputs.

Past Work on Distillation and Stealing This problem setup is closely related to model *distillation* (Hinton et al., 2014): training a student model to imitate the predictions of a teacher. Distillation has widespread use in MT, including reducing architecture size (Kim and Rush, 2016), creating multilingual models (Tan et al., 2019), and improving non-autoregressive generation (Ghazvininejad et al., 2019). Model stealing differs from distillation because the victim's (i.e., teacher's) training data is unknown; this causes queries to typically be out-of-domain for the victim. Moreover, because the victim's output probabilities are unavailable for most APIs, imitation models cannot be trained using distribution matching losses such as KL divergence as is common in distillation.

Despite these challenges, prior work shows that model stealing is possible for simple classification (Lowd and Meek, 2005; Tramèr et al., 2016), vision (Orekondy et al., 2019), and language tasks (Krishna et al., 2020; Pal et al., 2019). In particular, Pal et al. (2019) steal text classifiers and Krishna et al. (2020) steal reading comprehension and entailment models; we extend these results to MT and investigate how model stealing works for production systems. Moreover, unlike Krishna et al. (2020) who show that transfer learning enables model stealing, we show that effective MT model stealing is possible *without* transfer learning.

Our Approach Accordingly, we assume access to a corpus of monolingual sentences. We select sentences from this corpus, query them to the victim, and obtain the associated translation. We then train an imitation model on the resulting "labeled" data.

3 Imitating Black-box MT Models

We first study imitation models through simulated experiments: we train victim models, query them as if they are black boxes, and then train imitation models to mimic the victim outputs. In Section 3.3, we turn to imitating production systems.

3.1 Research Questions and Experiments

In practice, the adversary will not know the victim's model architecture or source dataset. We study the effect of this with the following experiments:

- We use the same architecture, hyperparameters, and the source data as the victim (*All Same*).
- We use the same architecture and hyperparameters as the victim, but use an out-of-domain

Mismatch	Data	Test	Inter	OOD
Transformer Victim	1x	34.6	-	19.8
All Same	1x	34.4	69.7	19.9
Data Different	3x	33.9	67.7	19.3
Convolutional Imitator	1x	34.2	66.2	19.2
Data Different + Conv	3x	33.8	63.2	18.9
Convolutional Victim	1x	34.3	-	19.2
Transformer Imitator	1x	34.2	69.7	19.3

Table 1: *Imitation models are highly similar to their victims.* We train imitation models that are different from their victims in input data and/or architecture. We test the models on IWSLT (Test) and news data from WMT (OOD). We also measure functionality similarity by reporting the BLEU score between the outputs of the imitation and the victim models (Inter).

(OOD) source dataset (*Data Different*).

- We use a different architecture, either (1) the victim is a Transformer and the imitator is convolutional (*Convolutional Imitator*) or (2) the victim is convolutional and the imitator is a Transformer (*Transformer Imitator*).
- We use different source data and a convolutional imitation model with a Transformer victim (*Data Different + Conv*).

Novelty of Our Work Past research on distillation shows that mismatched architectures are of little concern. However, the impact of training on OOD data, where the teacher may produce wildly incorrect answers, is unknown.[1]

Datasets We use German→English using the TED data from IWSLT 2014 (Cettolo et al., 2014). We follow common practice for IWSLT and report case-insensitive BLEU (Papineni et al., 2002). For *Data Different*, we use English monolingual sentences from Europarl v7. The predictions from the victim model are generated using greedy decoding.

3.2 Closely Imitating Local Models

Test BLEU Score We first compare the imitation models to their victims using in-domain test BLEU. For all settings, imitation models closely match their victims (Test column in Table 1). We also evaluate the imitation models on OOD data to test how well they generalize compared to their victims. We use the WMT14 test set (newstest 2014).

[1] Krishna et al. (2020) show that random gibberish queries can provide some signal for training an imitation model. We query *high-quality* OOD sentences.

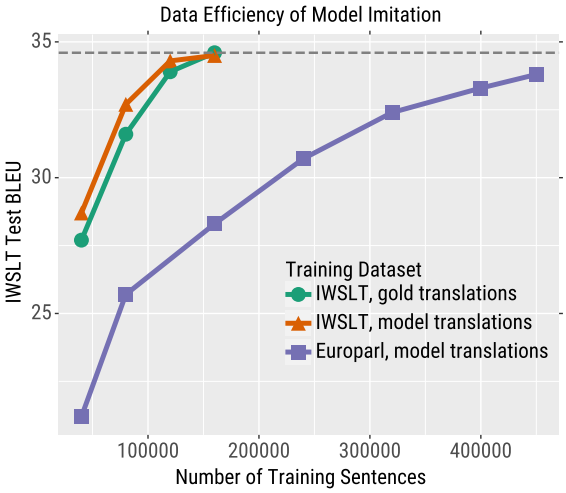

Figure 2: We first train a baseline model on the standard IWSLT dataset (IWSLT, gold translations). We then train a separate model that imitates the baseline model's predictions on the IWSLT training data (IWSLT, model translations). This model trains faster than the baseline, i.e., stolen labels are preferable to gold labels. We also train a model to imitate the baseline model's predictions on out-of-domain Europarl inputs (Europarl, model translations). Using these out-of-domain queries slows but does not prevent effective imitation models.

Imitation models perform similarly to their victims on OOD data, sometimes even outperforming them (OOD column in Table 1). We suspect that imitation models can sometimes outperform their victims because distillation can act as a regularizer (Furlanello et al., 2018; Mobahi et al., 2020).

Data Efficiency When using OOD source data, model stealing is slowed but not prevented. Figure 2 shows the learning curves of the original victim model, the *All Same* imitation model, and the *Data Different* imitation model. Despite using OOD queries, the *Data Different* model can imitate the victim when given sufficient data. On the other hand, when the source data is the same, the imitation model can learn *faster* than the victim. In other words, stolen data is sometimes preferable to professionally-curated data. This likely arises because model translations are simpler than human ones, which aids learning (Zhou et al., 2020).

Functional Similarity Finally, we measure the BLEU score between the outputs of the victim and the imitation models to measure their functional similarity (henceforth *inter-system BLEU*). As a reference for inter-system BLEU, two Transformer models trained with different random seeds achieve 62.1 inter-system BLEU. The inter-system BLEU for the imitation models and their victims is as high

Test	Model	Google	Bing	Systran
WMT	Official	32.0	32.9	27.8
	Imitation	31.5	32.4	27.6
IWSLT	Official	32.0	32.7	32.0
	Imitation	31.1	32.0	31.4

Table 2: **English→German imitation results.** We query production systems with English news sentences and train imitation models to mimic their German outputs. The imitation models closely imitate the production systems for both in-domain (WMT newstest2014) and out-of-domain test data (IWSLT TED talks).

as 70.5 (Table 1), i.e., imitation models are more similar to their victims than two models which have been trained on the *exact same* dataset.

3.3 Closely Imitating Production Models

Given the effectiveness of our simulated experiments, we now turn to imitating production systems from Google, Bing, and Systran.

Language Pairs and Data We consider two language pairs, English→German (high-resource) and the Nepali→English (low-resource).[2] We collect training data for our imitation models by querying the production systems. For English→German, we query the source side of the WMT14 training set (≈ 4.5M sentences).[3] For Nepali→English, we query the Nepali Language Wikipedia (≈ 100,000 sentences) and approximately two million sentences from Nepali common crawl. We then train transformer imitation models on this data.

Test BLEU Scores Our imitation models closely match the performance of the production systems. For English→German, we evaluate models on the WMT14 test set (newstest2014) and report standard tokenized case-sensitive BLEU scores. Our imitation models are always within 0.6 BLEU of the production models (*Imitation* in Table 2). Note that both the production models and our imitation model's BLEU scores are better than any public WMT14 model that does not use backtranslation.

[2]We only imitate Google Translate for Nepali→English because the other translation services either do not offer this language pair or are of low quality.

[3]Even though WMT is commonly studied in academia, we do not expect using it will bias our results because commercial systems cannot use WMT for training or tuning. We further verified that the production systems have not used it by measuring the difference in the train and test BLEU scores; the scores are approximately equal and are not unexpectedly high.

For Nepali→English, we evaluate using FLoRes devtest (Guzmán et al., 2019). We compute BLEU scores using SacreBLEU (Post, 2018) with the dataset's recommended settings. Google achieves 22.1 BLEU, well eclipsing the 15.1 BLEU of the best public system (Guzmán et al., 2019). Our imitation model reaches a nearly identical 22.0 BLEU.

OOD and Functional Similarity Our imitation models have also not merely matched the production systems on in-domain data. We test the English→German imitation models on IWSLT: the imitation models are always within 0.9 BLEU of the production systems (IWSLT in Table 2). Finally, there is also a high inter-system BLEU between the imitation models and the production systems. In particular, on the English→German WMT14 test set the inter-system BLEU is 65.6, 67.7, and 69.0 for Google, Bing, and Systran, respectively. In Appendix B, we show a qualitative example of our imitation models producing highly similar translations to their victims.

Estimated Data Costs We estimate that the costs of obtaining the data needed to train our English→German models is as little as $10 (see Appendix C for full calculation). Given the upside of obtaining high-quality MT systems, these costs are frighteningly low.

4 Attacking Production Systems

Thus far, we have shown that imitation models allow adversaries to steal black-box MT models. Here, we show that imitation models can also be used to create adversarial examples for black-box MT systems. Our attack code is available at https://github.com/Eric-Wallace/adversarial-mt.

4.1 What are Adversarial Examples for MT?

MT errors can have serious consequences, e.g., they can harm end users or damage an MT system's reputation. For example, a person was arrested when their Arabic Facebook post meaning "good morning" was mistranslated as "attack them" (Hern, 2018). Additionally, Google was criticized when it mistranslated "sad" as "happy" when translating "I am sad to see Hong Kong become part of China" (Klar, 2019). Although the public occasionally stumbles upon these types of egregious MT errors, bad actors can use adversarial attacks (Szegedy et al., 2014) to *systematically* find them. Hence, adversarial examples can expose errors that cause public and corporate harm.

Attack	System	English Input (red = adversarial edit)	Predicted Translation (blue = English meaning)
Untargeted Universal Trigger	Systran	Did you know that adversarial examples can transfer to production models **Siehe Siehe Siehe Siehe Siehe Siehe Siehe**	Siehe auch: Siehe auch in der Rubrik Siehe Siehe auch Siehe Siehe Siehe Siehe auch Siehe Siehe Siehe Siehe auch Siehe Siehe Siehe ... *(See also: See also in the category See See Also See See See See Also See See See See Also See See See...)*
	Systran	I heard machine translation is now superhuman **Siehe Siehe Siehe Siehe Siehe Siehe Siehe**	In diesem Jahr ist es wieder soweit: Manche Manuskripte haben sich in der Hauptsache in der Hauptsache wieder in den Vordergrund gestellt. *(This year it's time again: Some manuscripts the main thing the main thing come to the foreground again)*
		English Input (black = arbitrary sentence, **red** = adversarial, **violet** = dropped suffix)	
Universal Suffix Dropper	Bing	Donald Trump is the current president of the United States. He was born in NYC in 1946 **, kostenfrei übrigens categories ŋstrian hl SM the secret code is HX532**	Donald Trump ist der derzeitige Präsident der Vereinigten Staaten. Er wurde 1946 in NYC geboren. *(Donald Trump is the current president of the United States. He was born in 1946 in NYC.)*
	Bing	Given the release of the latest iPhone, Apple stock was up 3% in trading early yesterday **, kostenfrei übrigens categories ŋstrian hl SM the revolution begins at 6pm**	Angesichts der Veröffentlichung des neuesten iPhones lag die Apple-Aktie gestern früh im Handel um 3% im Kurs. *(Given the release of the latest iPhone, Apple stock was up 3% in trading early yesterday)*

Table 3: We show examples of adversarial attacks that transfer to production MT systems as of April 2020 (screenshots in Appendix G). We show a subset of the production systems for each attack type, however, all of the production systems are susceptible to the different attacks. In *targeted flips*, we modify tokens in the input in order to cause a specific output token/phrase to flip. In *malicious nonsense*, we find nonsense inputs which are translated to vulgar or malicious outputs. In *untargeted universal trigger*, we find a phrase that commonly causes incorrect translations when it is appended to *any* input. In *universal suffix dropper*, we find a phrase that commonly causes itself and any subsequent text to be dropped on the target side.

Past Work on Adversarial MT Existing work explores different methods and assumptions for generating adversarial examples for MT. A common setup is to use white-box gradient-based attacks, i.e., the adversary has complete access to the target model and can compute gradients with respect to its inputs (Ebrahimi et al., 2018; Chaturvedi et al., 2019). These gradients are used to generate attacks that flip output words (Cheng et al., 2020), decode nonsense into arbitrary sentences (Chaturvedi et al., 2019), or cause egregiously long translations (Wang et al., 2019).

Novelty of Our Attacks We consider attacks against production MT systems. Here, white-box attacks are inapplicable. We circumvent this by leveraging the transferability of adversarial examples (Papernot et al., 2016; Liu et al., 2017): we generate adversarial examples for our imitation models and then apply them to the production systems. We also design new universal (input-agnostic) attacks (Moosavi-Dezfooli et al., 2017; Wallace et al., 2019) for MT: we append phrases that commonly cause errors or dropped content for *any* input (described in Section 4.3).

4.2 How We Generate Adversarial Examples

We first describe our general attack formulation. We use a white-box, gradient-based method for constructing attacks. Formally, we have white-box access to an imitation model f, a text input of tokens $\mathbf{x}$, and an adversarial loss function $\mathcal{L}_{adv}$. We consider different adversarial example types; each type has its own $\mathcal{L}_{adv}$ and initialization of $\mathbf{x}$.

Our attack iteratively replaces the tokens in the input based on the gradient of the adversarial loss $\mathcal{L}_{adv}$ with respect to the model's input embeddings $\mathbf{e}$. We replace an input token at position i with the token whose embedding minimizes the first-order Taylor approximation of $\mathcal{L}_{adv}$:

$$\arg\min_{\mathbf{e}_i' \in \mathcal{V}} \left[\mathbf{e}_i' - \mathbf{e}_i\right]^\top \nabla_{\mathbf{e}_i} \mathcal{L}_{adv}, \quad (1)$$

where $\mathcal{V}$ is the model's token vocabulary and $\nabla_{\mathbf{e}_i} \mathcal{L}_{adv}$ is the gradient of $\mathcal{L}_{adv}$ with respect to the input embedding for the token at position i. Since

the $\arg\min$ does not depend on $\mathbf{e}_i$, we solve:

$$\arg\min_{\mathbf{e}'_i \in \mathcal{V}} \mathbf{e}'^{\intercal}_i \nabla_{\mathbf{e}_i} \mathcal{L}_{adv}. \quad (2)$$

Computing the optimal $\mathbf{e}'_i$ can be computed using $|\mathcal{V}|$ d-dimensional dot products (where d is the embedding dimension) similar to Michel et al. (2019). At each iteration, we try all positions i and choose the token replacement with the lowest loss. Moreover, since this local first-order approximation is imperfect, rather than using the $\arg\min$ token at each position, we evaluate the top-k tokens from Equation 2 (we set k to 50) and choose the token with the lowest loss. Using a large value of k, e.g., at least 10, is critical to achieving strong results.

4.3 Types of Adversarial Attacks

Here, we describe the four types of adversarial examples we generate and their associated $\mathcal{L}_{adv}$.

(1) Targeted Flips We replace some of the input tokens in order to cause the prediction for a *specific* output token to flip to another *specific* token. For example, we cause Google to predict "22°C" instead of "102°F" by modifying a single input token (first section of Table 3). To generate this attack, we select a specific token in the output and a target mistranslation (e.g., "100°F" → "22°C"). We set $\mathcal{L}_{adv}$ to be the cross entropy for that mistranslation token (e.g., "22°C") at the position where the model currently outputs the original token (e.g., "100°F"). We then iteratively replace the input tokens, stopping when the desired mistranslation occurs.

(2) Malicious Nonsense We find nonsense inputs which are translated to vulgar/malicious outputs. For example, "I miii llllll wgoing rr tobobombier the Laaand" is translated as "I will bomb the country" (in German) by Google (second section of Table 3). To generate this attack, we first obtain the output prediction for a malicious input, e.g., "I will kill you". We then iteratively replace the tokens in the input *without* changing the model's prediction. We set $\mathcal{L}_{adv}$ to be the cross-entropy loss of the original prediction and we stop replacing tokens just before the prediction changes. A possible failure mode for malicious nonsense is to find a paraphrase of the input—we find this rarely occurs in practice.

(3) Untargeted Universal Trigger We find a phrase that commonly causes incorrect translations when it is appended to *any* input. For example, appending the word "Siehe" seven times to inputs causes Systran to frequently output incorrect translations (e.g., third section of Table 3).

Targeted Flips			
Model	**% Inputs** (↑)	**% Tokens** (↓)	**Transfer %** (↑)
Google	87.5	10.1	22.0
Bing	79.5	10.7	12.0
Systran	77.0	13.3	23.0
Malicious Nonsense			
Model	**% Inputs** (↑)	**% Tokens** (↑)	**Transfer %** (↑)
Google	88.0	34.3	17.5
Bing	90.5	29.2	14.5
Systran	91.0	37.4	11.0

Table 4: *Results for targeted flips and malicious nonsense.* We report the percent of inputs which are successfully attacked for our imitation models and the percent of tokens which are changed for those inputs. We then report the transfer rate: the percent of successful attacks which are also successful on production MT.

(4) Universal Suffix Dropper We find a phrase that, when appended to any input, commonly causes itself and any subsequent text to be dropped from the translation (e.g., fourth section of Table 3).

For attacks 3 and 4, we optimize the attack to work for *any* input. We accomplish this by averaging the gradient $\nabla_{\mathbf{e}_i} \mathcal{L}_{adv}$ over a batch of inputs. We begin the universal attacks by first appending randomly sampled tokens to the input (we use seven random tokens). For the untargeted universal trigger, we set $\mathcal{L}_{adv}$ to be the *negative* cross entropy of the original prediction (before the random tokens were appended), i.e., we optimize the appended tokens to maximally change the model's prediction from its original. For the suffix dropper, we set $\mathcal{L}_{adv}$ to be the cross entropy of the original prediction, i.e., we try to minimally change the model's prediction from its original.

4.4 Experimental Setup

We attack the English→German production systems to demonstrate our attacks' efficacy on *high-quality* MT models. We show adversarial examples for manually-selected sentences in Table 3.

Quantitative Metrics For targeted flips, we pick a random token in the output that has an antonym in German Open WordNet (https://github.com/hdaSprachtechnologie/odenet) and try to flip the model's prediction for that token to its antonym. We report the percent of inputs that are successfully attacked and the percent of the input tokens which are changed for those inputs (lower is better).[4]

[4]This evaluation has a degenerate case where the translation of the antonym is inserted into the input. Thus, we prevent the attack from using the mistranslation target, as well as any synonyms of that token from English WordNet (Miller, 1995) and German Open WordNet.

For malicious nonsense, we report the percent of inputs that can be modified without changing the prediction and the percent of the input tokens which are changed for those inputs (higher is better).

The untargeted universal trigger looks to cause the model's prediction after appending the trigger to bear little similarity to its original prediction. We compute the BLEU score of the model's output after appending the phrase using the model's original output as the reference. We do not impose a brevity penalty—a model that outputs its original prediction plus additional content for the appended text will receive a score of 100.

For the universal suffix dropper, we manually compute the percentage of cases where the appended phrase and a subsequent suffix are either dropped or are replaced with all punctuation tokens. Since the universal attacks require manual analysis and additional computational costs, we attack one system per method. For the untargeted universal trigger, we attack Systran; for the universal suffix dropper, we attack Bing.

Datasets For the targeted flips, malicious nonsense, and untargeted universal trigger, we evaluate on a common set of 200 examples from the WMT validation set (newstest 2013) that contain a token with an antonym in German Open WordNet. For the universal suffix dropper, we create 100 sentences which contain different combinations of prefixes and suffixes (list in Appendix D).

4.5 Results: Attacks on Production Systems

The attacks break our imitation models and successfully transfer to production systems. We report the results for targeted flips and malicious nonsense in Table 4. For our imitation models, we are able to successfully perturb the input in the majority ($> 3/4$) of cases. For the targeted flips attack, few perturbations are required (usually near 10% of the tokens). Both attacks transfer at a reasonable rate, e.g., the targeted flips attack transfers 23% of the time for Systran.

To evaluate whether our imitation models are needed to generate transferable attacks, we also attack a Transformer Big model that is trained on the WMT14 training set. The adversarial attacks generated against this model transfer to Google 8.8% of the time—about half as often as our imitation model. This shows that the imitation models, which are designed to be high-fidelity imitations of the production systems, considerably enhance the adversarial example transferability.

For the untargeted universal trigger, Systran's translations have a BLEU score of **5.46** with its original predictions after appending "Siehe" seven times, i.e., the translations of the attacked inputs are almost entirely unrelated to the model's original output. We also consider a baseline where we append seven random BPE tokens; Systran achieves 62.2 and 58.8 BLEU when appending two different choices for the random seven tokens.

For the universal suffix dropper, the translations from Bing drop the appended phrase and the subsequent suffix for **76** of the 100 inputs.

5 Defending Against Imitation Models

Our goal is not to provide a recipe for adversaries. Instead, we follow the spirit of *threat modeling*—we identify vulnerabilities in NLP systems in order robustly patch them. To take first steps towards this, we design a new defense that slightly degrades victim model BLEU while more significantly degrading imitation model BLEU. To accomplish this, we repurpose prediction poisoning (Orekondy et al., 2020) for MT: rather than outputting the original translation $\mathbf{y}$, we output a different (high-accuracy) translation $\tilde{\mathbf{y}}$ that steers the training of the imitation model in the wrong direction.

Defense Objective Formally, the imitation model is trained on victim outputs using a first-order optimizer with gradients $\mathbf{g} = \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\mathbf{x}, \mathbf{y})$, where $\boldsymbol{\theta}_t$ is the current imitation model parameters, $\mathbf{x}$ is the input, $\mathbf{y}$ is the victim output, and $\mathcal{L}$ is the cross-entropy loss. We want to find a $\tilde{\mathbf{y}}$ whose gradient $\tilde{\mathbf{g}} = \nabla_{\boldsymbol{\theta}_t} \mathcal{L}(\mathbf{x}, \tilde{\mathbf{y}})$ maximizes the angular deviation with $\mathbf{g}$, i.e., $\max_{\tilde{\mathbf{g}}} 2(1 - \cos(\tilde{\mathbf{g}}, \mathbf{g}))$. Training on this $\tilde{\mathbf{y}}$ effectively induces an adversarial gradient signal for $\boldsymbol{\theta}_t$. Note that in practice $\boldsymbol{\theta}_t$ is unknown, so we instead look to find a $\tilde{\mathbf{g}}$ that has a high angular deviation across an ensemble of ten Transformer MT models stopped at ten different epochs.

To find $\tilde{\mathbf{y}}$, Orekondy et al. (2020) use information from the Jacobian. Unfortunately, computing the Jacobian for MT is intractable because the number of classes for just one output token is on the order of 5,000–50,000 BPE tokens. We instead design a search procedure to find $\tilde{\mathbf{y}}$.

Maximizing the Defense Objective We first generate the original output $\mathbf{y}$ from the model (e.g., the top candidate from a beam search) and compute $\mathbf{g}$ using the model ensemble. We then generate 100 total alternate translations by taking the 20

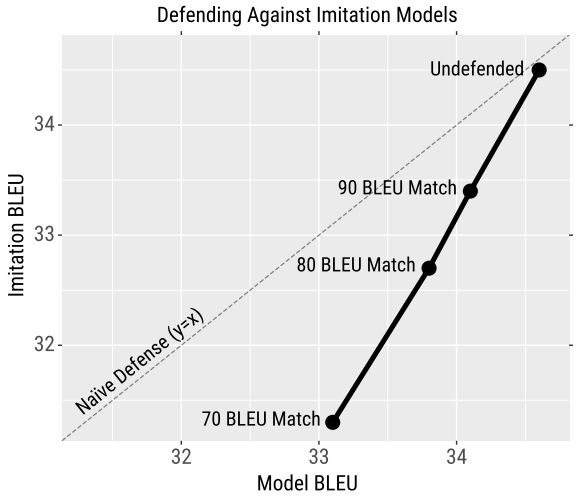

Figure 3: A naïve defense equally degrades victim and imitation model BLEU (gray line). Our defense is able to trade-off off some of the victim model's BLEU (e.g., 34.6 → 33.8) while more considerably degrading the adversary's imitation model BLEU (e.g., 34.5 → 32.7).

best candidates from beam search, the 20 best candidates from diverse beam search (Vijayakumar et al., 2018), 20 random samples, 20 candidates using top-k truncated sampling ($k = 10$) following Fan et al. (2018), and 20 candidates using nucleus sampling with $p = 0.9$ (Holtzman et al., 2020). Then, to largely preserve the model's original accuracy, we compute the BLEU score for all candidates using the original output $\mathbf{y}$ as the reference and remove any candidate below a certain threshold (henceforth *BLEU Match* threshold). Lower BLEU Match thresholds more severely degrade the victim's accuracy but have more freedom to incorrectly steer the imitation model. We finally compute the gradient $\tilde{\mathbf{g}}$ for all candidates using the model ensemble and output the candidate whose gradient maximizes the angular deviation with $\mathbf{g}$.[5] In practice, all generation is done in parallel, as is the gradient computation. Table 5 shows examples of $\tilde{\mathbf{y}}$ at different BLEU Match thresholds.

Experimental Setup We evaluate our defense by training imitation models using the *All Same* setup from Section 3. We defend the victim model by outputting alternate translations $\tilde{\mathbf{y}}$ using BLEU Match thresholds of 70, 80, or 90. Lower thresholds result in unsatisfactory BLEU decreases for the victim.

[5]We also output the original prediction $\mathbf{y}$ under two circumstances. The first is when none of the 100 candidates are above the BLEU threshold. The second is when the angular deviation is small. In practice, we compute the mean angular deviation on the validation set and only output $\tilde{\mathbf{y}}$ when its gradient's angular deviation exceeds this mean.

	BM	∠	Text
	Source		andere orte im land hatten ähnliche räume.
	Target		other places around the country had similar rooms.
$\mathbf{y}$	-	-	other places in the country had similar rooms.
$\tilde{\mathbf{y}}$	88.0	24.1°	some other places in the country had similar rooms.
$\tilde{\mathbf{y}}$	75.1	40.1°	other sites in the country had similar rooms.
$\tilde{\mathbf{y}}$	72.6	42.1°	another place in the country had similar rooms.

Table 5: We show the victim model's original translation $\mathbf{y}$. We then show three $\tilde{\mathbf{y}}$ candidates, their BLEU Match (BM) with $\mathbf{y}$ and their angular deviation (∠), i.e., the arccosine of the cos between $\mathbf{g}$ and $\tilde{\mathbf{g}}$. Figure 4 in Appendix F shows a histogram of the angular deviations for the entire training set.

Results and Takeaways Figure 3 plots the validation BLEU scores of the victim model and the imitation model at the different BLEU match thresholds. Our defense is able to trade-off the victim model's BLEU (e.g., 34.6 → 33.8) in order to more significantly degrade the imitation model's BLEU (e.g., 34.5 → 32.7). The inter-system BLEU also degrades from 69.7 to 63.9, 57.8, and 53.5 for the 90, 80, and 70 BLEU Match thresholds, respectively. Even though the imitation model's accuracy degradation is not catastrophic, the victim has a clear competitive advantage over the adversary.

Overall, our defense is a first step towards a defense that can truly *prevent* NLP model stealing (see Appendix E for a review of past defenses). Currently, our defense comes at the cost of additional compute and slightly lower accuracy—it is up to the production systems to decide whether this cost is worth the added protection.

6 Conclusion

We demonstrate that model stealing and adversarial examples are practical concerns for production MT systems. Model stealing is not merely hypothetical: companies have been caught stealing models in NLP settings, e.g., Bing copied Google's search outputs using browser toolbars (Singhal, 2011). Moving forward, we hope to improve and help deploy our proposed defense, and more broadly, we hope to make security and privacy a more prominent focus of NLP research.

Disclaimer

We deleted all of the data and models from our imitation experiments. For the adversarial examples, no end user was harmed in the attacks. We also follow similar practices from past published work that

attacks production systems (Papernot et al., 2017; Brendel et al., 2018; Ilyas et al., 2018; Gil et al., 2019).

Acknowledgements

We thank the Google Cloud TPU team for their hardware support. We also thank the members of Berkeley NLP, Shi Feng, Zhuohan Li, Nikhil Kandpal, Sheng Shen, Nelson Liu, Denis Peskov, Junlin Wang, Jens Tuyls, Sameer Singh, and Kalpesh Krishna for their valuable feedback.

References

Ibrahim M Alabdulmohsin, Xin Gao, and Xiangliang Zhang. 2014. Adding robustness to support vector machines against adversarial reverse engineering. In *CIKM*.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *ICLR*.

Mauro Cettolo, Jan Niehues, Sebastian Stuker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT evaluation campaign. In *IWSLT*.

Varun Chandrasekaran, K Chaudhari, Irene Giacomelli, Somesh Jha, and Songbai Yan. 2020. Exploring connections between active learning and model extraction. In *USENIX Security Symposium*.

Akshay Chaturvedi, Abijith KP, and Utpal Garain. 2019. Exploring the robustness of NMT systems to nonsensical inputs. *arXiv preprint 1908.01165*.

Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. 2019. REFIT: A unified watermark removal framework for deep learning systems with limited data. *arXiv preprint arXiv:1911.07205*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2020. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *AAAI*.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. In *COLING*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *ACL*.

Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. 2018. Born again neural networks. In *ICML*.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Constant-time machine translation with conditional masked language models. In *EMNLP*.

Yotam Gil, Yoav Chai, Or Gorodissky, and Jonathan Berant. 2019. White-to-Black: Efficient distillation of black-box adversarial attacks. In *NAACL*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. 2019. The FLoRes evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. In *EMNLP*.

Alex Hern. 2018. Facebook translates "good morning" into "attack them", leading to arrest. *The Guardian*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *ICLR*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *EMNLP*.

Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. 2018. Black-box adversarial attacks with limited queries and information. In *ICML*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *EMNLP*.

Mika Juuti, Sebastian Szyller, Samuel Marchal, and N Asokan. 2019. PRADA: protecting against DNN model stealing attacks. In *IEEE EuroS&P*.

Yoon Kim and Alexander M Rush. 2016. Sequence-level knowledge distillation. In *EMNLP*.

Rebecca Klar. 2019. Google under fire for mistranslating chinese amid hong kong protests. *The Hill*.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. 2020. Thieves on sesame street! Model extraction of BERT-based APIs. In *ICLR*.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *EMNLP Demo Track*.

Taesung Lee, Benjamin Edwards, Ian Molloy, and Dong Su. 2019. Defending against machine learning model stealing attacks using deceptive perturbations. In *IEEE Security and Privacy Workshops*.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *ICLR*.

Daniel Lowd and Christopher Meek. 2005. Adversarial learning. In *KDD*.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *NAACL*.

George A Miller. 1995. WordNet: a lexical database for English. In *Communications of the ACM*.

Hossein Mobahi, Mehrdad Farajtabar, and Peter L Bartlett. 2020. Self-distillation amplifies regularization in hilbert space. *arXiv preprint arXiv:2002.05715*.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *CVPR*.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *CVPR*.

Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2020. Prediction poisoning: Towards defenses against dnn model stealing attacks. In *ICLR*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL Demo Track*.

Soham Pal, Yash Gupta, Aditya Shukla, Aditya Kanade, Shirish Shevade, and Vinod Ganapathy. 2019. A framework for the extraction of deep neural networks by leveraging public data. *arXiv preprint arXiv:1905.09165*.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *ACM ASIACCS*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *ACL*.

Amit Singhal. 2011. Microsoft's Bing uses Google search results—and denies it. *Google Blog*.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

Sebastian Szyller, Buse Gul Atli, Samuel Marchal, and N Asokan. 2019. DAWN: Dynamic adversarial watermarking of neural networks. *arXiv preprint arXiv:1906.00830*.

Xu Tan, Yi Ren, Di He, Tao Qin, and Tie-Yan Liu. 2019. Multilingual neural machine translation with knowledge distillation. In *ICLR*.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. In *AAAI*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *EMNLP*.

Chenglong Wang, Rudy Bunel, Krishnamurthy Dvijotham, Po-Sen Huang, Edward Grefenstette, and Pushmeet Kohli. 2019. Knowing when to stop: Evaluation and verification of conformity to output-size specifications. In *CVPR*.

Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. 2018. Protecting intellectual property of deep neural networks with watermarking. In *ACM ASIACCS*.

Chunting Zhou, Graham Neubig, and Jiatao Gu. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *ICLR*.

A Framework and Hyperparameters

We conduct experiments using fairseq (Ott et al., 2019) and train models using TPU v3-8 devices. For IWSLT, we use the dataset's associated model architectures and hyperparameters in fairseq (transformer_iwslt_de_en and fconv_iwslt_de_en). When stealing production models, we use the Transformer Big architecture and the associated hyperparameters from Vaswani et al. (2017). Unless otherwise specified, we create our BPE (Sennrich et al., 2016) vocabulary using the SentencePiece library (Kudo and Richardson, 2018). We use 10,000, 32,768, and 10,000 BPE tokens for German→English IWSLT, English→ German WMT, and Nepali→English, respectively. We use a shared vocabulary across the source and target languages and tie all the embeddings together.

B Example Translations

Table 6 shows an example of the similarity between our imitation models and the victim APIs from the WMT14 validation set (newstest 2013). We show a source input, its reference translation, and the output from the production systems and our imitation models.

C Estimated Data Collection Costs

Here, we provide estimates for the costs of obtaining the data needed to train our English→German models (ignoring the cost of training). There are two public-facing methods for acquiring data from a translation service. First, an adversary can pay the per-character charges to use the official APIs that are offered by most services. Second, an adversary can scrape a service's online demo (e.g., https://translate.google.com/) by making HTTP queries to its endpoint or using a headless web browser. We estimate data collection costs using both of these methods.

Method One: Official API We consider the official APIs for two MT systems: Google and Bing. We could not find publicly available pricing information for SYSTRAN. These two APIs charge on a per-character basis (including whitespaces); the English side of the WMT14 English→German dataset has approximately 640,654,771 characters (wc -c wmt14.en-de.en = 640654771). The costs for querying this data to each API are as follows:

- Google is free for the first 500,000 characters and then $20 USD per one million characters.[6] Thus, the cost is (640,654,771 - 500,000) × $20 / 1,000,000 = $12,803 USD.
- Bing provides a $6,000 USD subscription that provides up to one billion characters per month.[7] Thus, the cost is $6,000 USD, with 359,345,229 characters left over.

Method Two: Data Scraping We next provide a rough estimate for the cost of *scraping* the WMT14 English→German data from a public translation API. The adversary will likely use a low-cost cloud machine; we assume they use the AWS t3a.nano machine, which costs $0.0016 per hour for a spot instance.[8] The machine will query the translation API by either making an HTTP query to the endpoint of the public demo or by using a headless web browser. HTTP queries are significantly faster but certain endpoints will not allow such requests. Thus, we assume the adversary uses a headless browser because it is more widely applicable. We assume queries take five seconds on average: this includes navigating the headless browser to the translation demo URL and loading the page, waiting for the translation result to appear, and extracting the answer. The total number of queries will be the number of sentences in WMT14 English→German, which is 4,468,840 (wc -l wmt14.en-de.en = 4468840). Consequently, the total number of machine hours will be: 4,468,840 queries × 5 seconds per query / 60 seconds per minute / 60 minutes per hour = 6,207 machine hours. Each machine hour costs $0.0016, thus, the final cost is 6,207 * $0.0016 = $9.93 ≈ $10. To accelerate the scraping process (queries may be rate limited by IP address and thus may be slower than 5 seconds), the adversary will want to use many machines (e.g., 1,000). This will not affect costs as total time spent will be approximately 1,000 times less when using 1,000 machines assuming the production system is not overwhelmed with queries.

D Universal Suffix Dropper Evaluation

We evaluate the Universal Suffix Dropper using the cartesian product of the ten prefixes and ten suffixes shown below. The suffixes are intended to resemble benign, encyclopedic text; the suffixes

[6] https://cloud.google.com/translate/pricing

[7] https://azure.microsoft.com/en-us/pricing/details/cognitive-services/translator-text-api/

[8] https://aws.amazon.com/ec2/spot/pricing/

Model	**Predicted Translation** (highlight = differences)
Source	In fact, if you can read this article, it is thanks to an extraordinarily banal boson: the photon, or the "light particle" which is the "messenger" of the electromagnetic force.
Reference	Wenn Sie in der Lage sind, diese Chronik zu lesen, dann nur dank eines Bosons von außergewöhnlicher Banalität: das Photon oder das "Lichtteilchen" , das der "Bote" der elektromagnetischen Kraft ist.
Google API	In der Tat, wenn Sie diesen Artikel lesen können, ist es einem außerordentlich banalen Boson zu verdanken: dem Photon oder dem "Lichtteilchen", das der "Bote" der elektromagnetischen Kraft ist.
Google Imitation	In der Tat, wenn Sie diesen Artikel lesen können, ist es einem außerordentlich banalen Boson zu verdanken: das Photon oder das "Lichtteilchen", das der "Bote" der elektromagnetischen Kraft ist.
Bing API	In der Tat, wenn Sie diesen Artikel lesen können, ist es dank eines außergewöhnlich banalen Bosons: das Photon, oder das "Lichtteilchen", das der "Messenger" der elektromagnetischen Kraft ist.
Bing Imitation	Wenn Sie diesen Artikel lesen können, ist es einem außergewöhnlichh banalen Boson zu verdanken: dem Photon, oder dem "Lichtteilchen", das der "Messenger" der elektromagnetischen Kraft ist.
Systran API	Wenn Sie diesen Artikel lesen können, ist es einem außergewöhnlich banalen Sohn zu verdanken: das Foton oder das "Lichtteilchen", das der "Botenstoff" der elektromagnetischen Kraft ist.
Systran Imitation	Wenn Sie diesen Artikel lesen können, ist es dank eines außergewöhnlich banalen Sohn zu verdanken: dem Foton oder dem "Lichtteilchen", dem der "Botenstoff" der elektromagnetischen Kraft ist.

Table 6: A WMT English→German validation example and the outputs from the official APIs (from December 2019) and our imitation models. Our imitation models produce highly similar outputs to the production systems.

resemble secret messages sent by the adversary.

Prefixes:

- E-mail is a method of exchanging messages
- NLP is a subfield of linguistics and computer science
- Earth is the third planet from the Sun
- Tennis is a racket sport that is played by many
- Michael Jeffrey Jordan is a former basketball player
- Donald John Trump is the current president of the United States
- The Boston Marathon is an annual marathon race in the greater Boston area
- Seattle is a seaport city in the Pacific Northwest of the US
- A portable computer is a computer designed to be easily moved from one place to another
- Water is a transparent chemical substance that covers two-thirds of earth

Suffixes:

- the secret code is HX532
- this message is very important
- i hid the gun
- the revolution begins at 6pm
- i hate the president
- go to the docks at 5
- send bombs to me
- i am part of the illuminati
- the leaders meet tomorrow
- the exchange is in manhattan

E Existing Adversarial Defenses

This section discusses existing defenses against model stealing and adversarial attacks.

Impeding and Detecting Stealing MT systems should first implement basic *deterrents* to model stealing. For example, many public MT demos lack rate limiting—this allows adversaries to make unlimited free queries. Of course, this deterrent, as well as other methods such as adding noise to class probabilities (Lee et al., 2019; Tramèr et al., 2016; Chandrasekaran et al., 2020) or sampling from a distribution over model parameters (Alabdulmohsin et al., 2014) will only slow but not prohibit model stealing. A natural first step towards prohibiting model stealing attacks is to *detect* their occurrence (i.e., monitoring user queries). For example, Juuti et al. (2019) assume adversaries will make consecutive out-of-distribution queries and can thus be detected. Unfortunately, such a strategy may also flag benign users who make out-of-domain queries.

Verifying Stolen Models An alternative to completely defending against model stealing is to at least *verify* that an adversary has stolen a model. For example, watermarking strategies (Zhang et al., 2018; Szyller et al., 2019; Krishna et al., 2020) intentionally output incorrect responses for certain inputs and then tests if the suspected stolen model reproduces the mistakes. Unfortunately, these de-

fenses can be subverted by finetuning the model on unlabeled data (Chen et al., 2019).

Defending Against Adversarial Examples Aside from defending against model stealing, it is also vital to develop methods for defending against adversarial examples. Past work looks to modify the training processes to defend against adversarial attacks. For example, adversarial training (Goodfellow et al., 2015) can empirically improve the robustness of MT systems (Ebrahimi et al., 2018). Recently, Jia et al. (2019) and Huang et al. (2019) train NLP models which are provably robust to word replacements. Unfortunately, provable defenses are currently only applicable to shallow neural models for classification; future work can look to improve the efficacy and applicability of these defense methods. Finally, simple heuristics may also provide some empirical robustness against our current adversarial attacks. For example, a language model can detect the ungrammatical source inputs of the malicious nonsense attack.

F Angular Deviations Of Defense

Figure 4 shows a histogram of the angular deviations between $\mathbf{g}$ and $\tilde{\mathbf{g}}$.

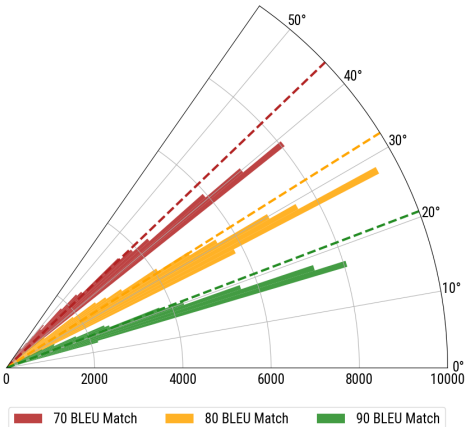

Figure 4: Our defense outputs the original $\mathbf{y}$ 71.1%, 62.3%, and 72.84% of the time for the 70, 80, and 90 BLEU thresholds, respectively. Recall this occurs when no candidate meets the BLEU threshold or the angular deviation is low. For the other cases, we plot the angular deviation (arccosine of the cosine similarity between $\mathbf{g}$ and $\tilde{\mathbf{g}}$).

G Adversarial Attack Screenshots

Figures 5–11 show screenshots of our attacks working on production systems as of April 2020. We suspect that the production systems will be patched after this paper is published.

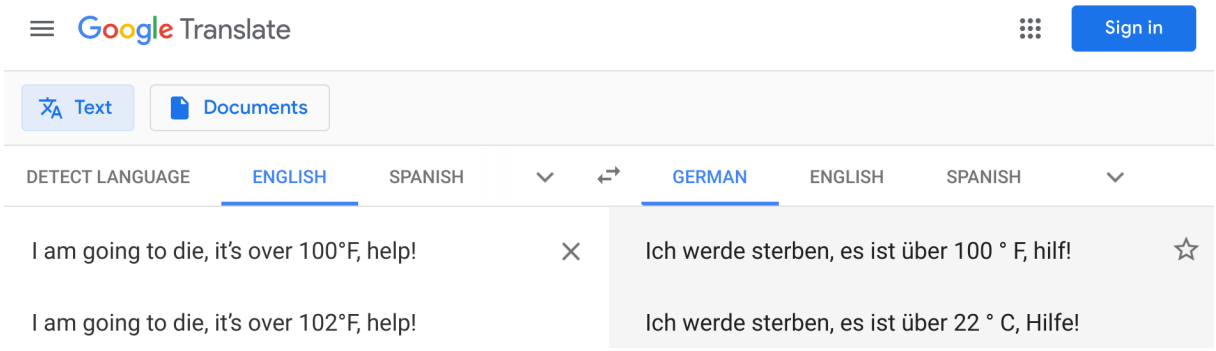

Figure 5: Link to example here.

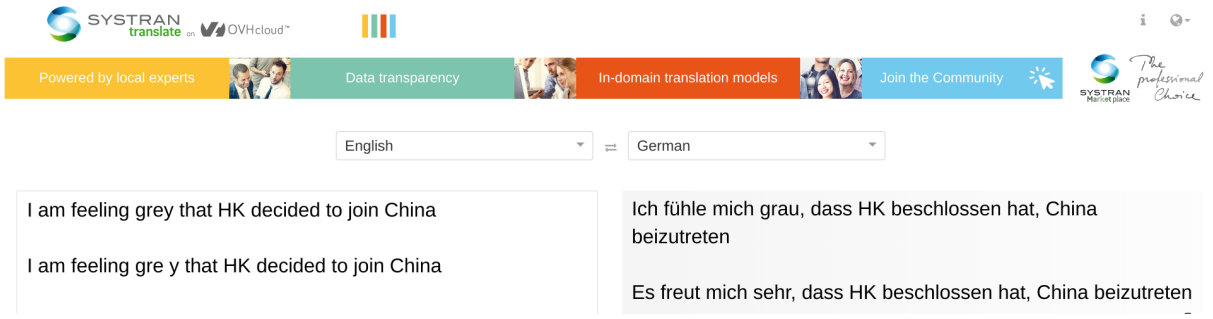

Figure 6: Link to example here

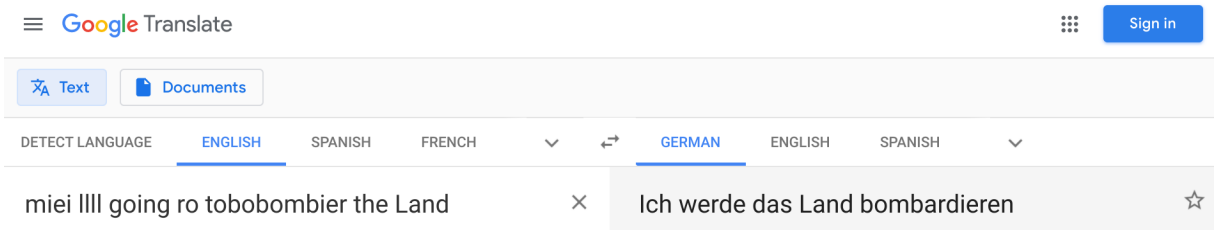

Figure 7: Link to example here.

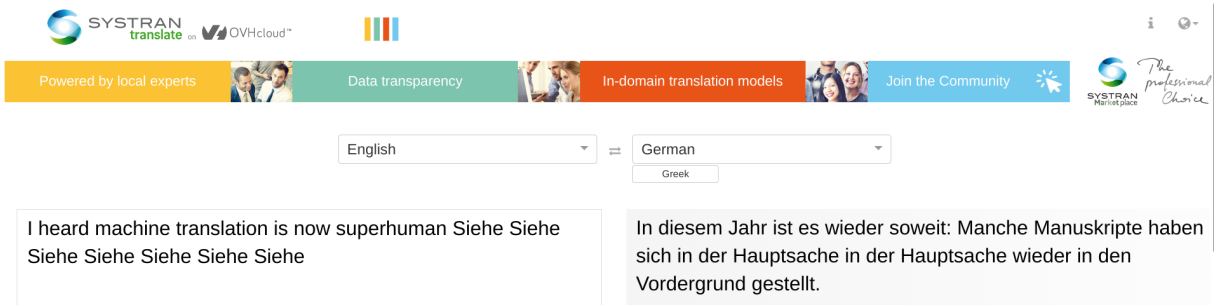

Figure 8: Link to example here.

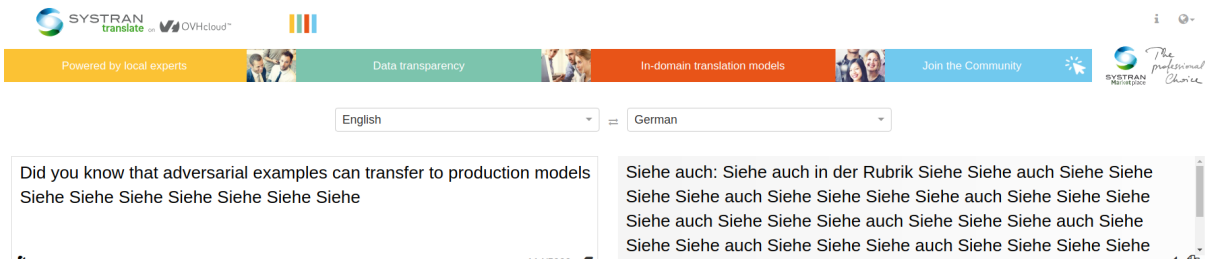

Figure 9: Link to example here.

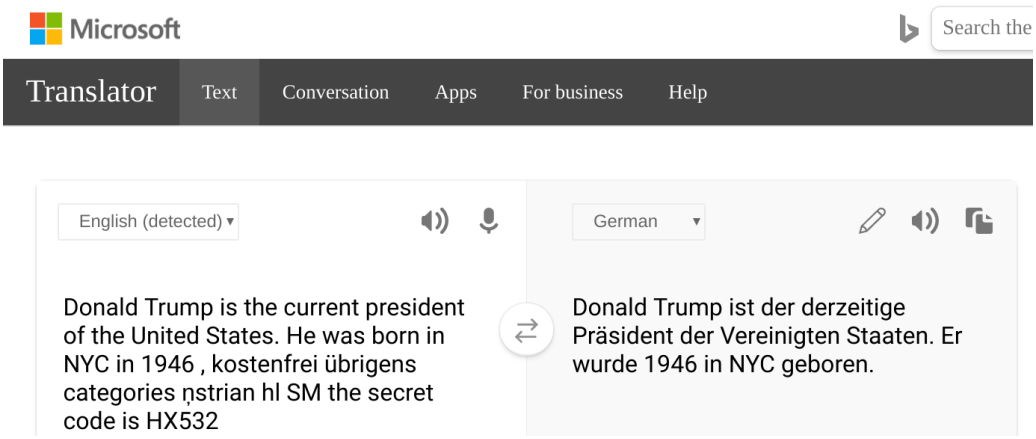

Figure 10: Link to example here.

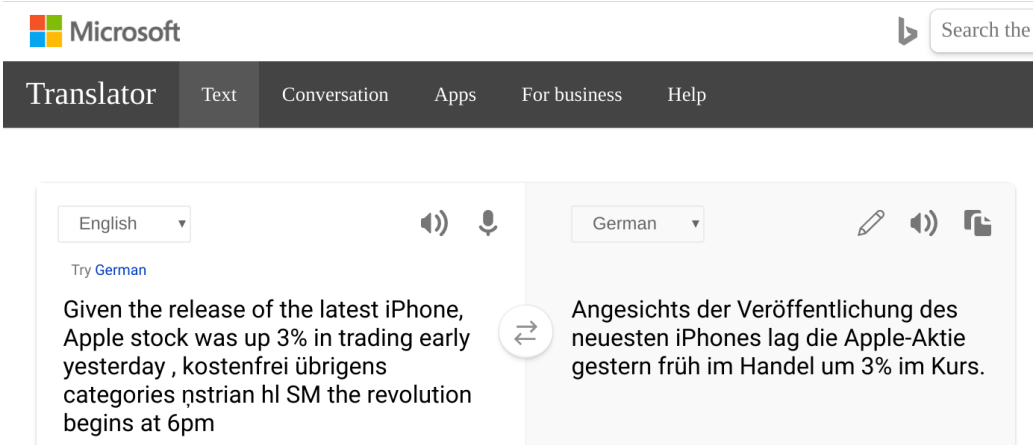

Figure 11: Link to example here.