On the Transferability of Adversarial Attacks against Neural Text Classifier

Liping Yuan[1,2], **Xiaoqing Zheng***[1,2], **Yi Zhou**[1,2]
Cho-Jui Hsieh[3], **Kai-Wei Chang**[3]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Shanghai Key Laboratory of Intelligent Information Processing
[3]Department of Computer Science, University of California, Los Angeles, USA
{lpyuan19,zhengxq,yizhou17}@fudan.edu.cn
{chohsieh,kwchang}@cs.ucla.edu

Abstract

Deep neural networks are vulnerable to adversarial attacks, where a small perturbation to an input alters the model prediction. In many cases, malicious inputs intentionally crafted for one model can fool another model. In this paper, we present the first study to systematically investigate the transferability of adversarial examples for text classification models and explore how various factors, including network architecture, tokenization scheme, word embedding, and model capacity, affect the transferability of adversarial examples. Based on these studies, we propose a genetic algorithm to find an ensemble of models that can be used to induce adversarial examples to fool almost all existing models. Such adversarial examples reflect the defects of the learning process and the data bias in the training set. Finally, we derive word replacement rules that can be used for model diagnostics from these adversarial examples.

1 Introduction

Recent studies demonstrate that deep neural networks are vulnerable to adversarial examples, intentionally crafted to fool the models. Although generating adversarial examples for texts has shown to be more challenging than for images due to their discrete nature, many methods have been proposed to generate adversarial text examples and reveal the vulnerability of deep neural networks in natural language processing (NLP) tasks, including reading comprehension (Jia and Liang, 2017), text classification (Samanta and Mehta, 2017; Wong, 2017; Liang et al., 2018; Alzantot et al., 2018; Yang et al., 2020), machine translation (Zhao et al., 2018; Ebrahimi et al., 2018; Cheng et al., 2020), dialogue systems (Cheng et al., 2019), and dependency parsing (Zheng et al., 2020). These methods perturb text examples by replacing, scrambling, and erasing characters, words or other language units to fool an NLP model.

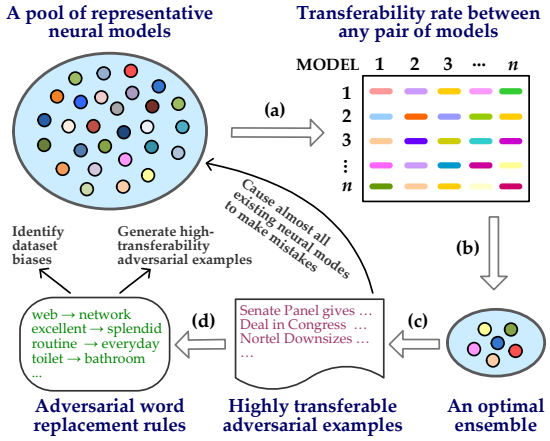

Figure 1: Overview of the study. (a) Given a pool of representative neural models, we compute the adversarial transferability rate between any pair of models; (b) A genetic algorithm is used to find an optimal ensemble with the minimum number of members so that the adversarial examples crafted by attacking the ensemble can strongly transfer to other models; (c) Highly-transferable adversarial examples can be crafted by attacking the ensemble model; (d) We derive adversarial word replacement rules from the adversarial examples constructed by the ensemble such that these rules can be used to identify data biases and to diagnose a model under the black-box setting.

Most existing studies focus on developing effective algorithms for attacking a specific model. The successful attacks demonstrate the instability of model predictions. However, the vulnerability of a model may correlate with different factors, such as network architecture, tokenization scheme, word embedding type, model capacity, and the spurious predictive patterns in the training data.

In this study, we aim to *understand the attack algorithms through the lens of analyzing transferability of adversarial examples.* We first systematically investigate which factors of neural models impact the black-box transferability (i.e., how adversarial examples generated against one model can fool another one (Szegedy et al., 2013)) of adversarial

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 1612–1625
November 7–11, 2021.

Table 1: Three adversarial examples that successfully fool all 63 models with various configurations (see detailed in Section 2.1), crafted by using adversarial word replacement rules discovered by our algorithm on AGNEWS.

Senate Panel Gives NASA Extra Money (AP) AP - NASA would get #36;16.4 billion next year under a bill a Senate committee approved Tuesday, reversing a decision by House lawmakers to ~~cut~~ contract the space agency's budget below this year's levels.
Deal in Congress to ~~keep~~ preserve tax cuts, Widening Deficit Republican and Democratic leaders agreed to extend $5 billion worth of tax cuts sought by ~~President~~ Chairman Bush without trying to pay for them.
Nortel Downsizes Again Aug. 23, 2004 (TheDeal.com) Problem-plagued Nortel ~~Networks~~ Web Corp. announced plans Thursday, Aug. 19, to eliminate an additional 3,500 jobs and fire seven more senior ~~executives~~ administrators as the company labors to reinvent.

examples through extensive experiments on two text classification datasets, Sentiment Movie Reviews (MR) (Pang and Lee, 2005) and AG News corpus (AGNEWS) (Zhang et al., 2015). These factors include network architectures (LSTM, CNN, or Transformer), tokenization schemes (character, sub-word, or word), embedding types (GloVe, word2vec, or fastText), and model capacities (different network depths). We vary one factor at a time while fixing the others to see which factor is the more significant one, and found that the tokenization scheme has the greatest influence on the adversarial transferability, following by network architecture, embedding type, and model capacity.

Based on the analysis, we study whether it is possible to *craft highly-transferable text adversarial examples for many neural models by ensembling a small number of models*. Specifically, these highly-transferable adversarial examples provide the following insights. First, the adversaries do not need white-box access to victim models. They launch the attacks by their own models trained on similar data, which can transfer across models (Moosavi-Dezfooli et al., 2017). Second, as stated in Wallace et al. (2019), such adversarial examples are a useful analysis tool and reveal general input-output patterns learned by models, which can be leveraged to study the influence of dataset biases and to identify those biases learned by models.

We also found that the adversarial examples obtained by an ensemble model are more transferable and propose a genetic algorithm to find an optimal ensemble based on the empirical transferability between different models. The adversarial examples generated by attacking the founded ensemble are strongly transferable to other models. For some models, they even exhibit better transferability than those generated by attacking the same model but with different random initialization.

Finally, inspired by Ribeiro et al. (2018), we generalize the adversarial examples constructed by our ensemble into semantics-preserving adversarial word replacement rules that can induce adversaries on any text input strongly transferring to other neural network-based models (see Table 1). Since those rules are model-agnostic, they provide an analysis of global model behavior and help us to identify dataset biases and to diagnose heuristics learned by the models (See Figure 1 for an illustration of the process).

2 Adversarial Transferability Among Neural Models

In the following, we first want to investigate how network architectures, tokenization schemes, embedding types, and model capacities affect the attack transferability. We conduct an empirical study by varying one factor at a time while fixing the rest to see the differences in their attack transferability. Technically, we generate the adversarial examples by attacking a source model and pass the generated adversarial examples through other models for comparison.

2.1 Experimental Design

We use convolutional neural network (CNN), long short-term memory (LSTM), and bidirectional LSTM as base models with 1, 2, and 4 layers (an additional 6-layer one for CNN). Those networks can take three forms as input: word, character, and word + character. If word-based models are used, their word embeddings are randomly initialized or initialized with GloVe (Pennington et al., 2014), word2vec (Mikolov et al., 2013), or fastText (Joulin et al., 2016). When taking word + character as input, the models are initialized with the embeddings pre-trained by ELMo (Peters et al., 2018). We also include BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), and ALBERT (Lan et al., 2019) into the model pool for analysis, and the total number of models under investigation is 63 (see Appendix A.1 for details), which cover popular neural networks that are used in NLP literature.

All the models are investigated under two recently proposed attack algorithms, PWWS (Ren et al., 2019) and GA (Alzantot et al., 2018). The sets of allowed word substitutions are based on the synonyms created in WordNet (Miller, 1995), and for any word in a text, the word to replace must have the same part-of-speech (POS) as the original one[1]. Alzantot et al. (2018) also used a language model (LM) to rule out candidate substitute words that do not fit within the context. However, unlike PWWS, ruling out some candidates by an LM will greatly reduce the number of candidate substitute words (65% off on average). For consistency, we report the robust accuracy under GA attack without using an LM. Zang et al. (2020) suggested that existing textual attack algorithms can roughly be divided into two categories: greedy and population-based algorithms. PWWS and TextFooler (Jin et al., 2020) fall into the first category while GA and PSO (Zang et al., 2020) belong to the second one. We chose one attack algorithm in each category when investigating the transferability among neural models and use TextFooler to evaluate the generalizability of the proposed method in Section 3.3.

We conducted experiments on two text classification datasets: Sentiment Movie Reviews (MR) (Pang and Lee, 2005) and AG News corpus (AGNEWS) (Zhang et al., 2015). All models are trained on the standard training set with the cross-entropy loss. For each dataset, we attack $1,000$ randomly selected test examples. For evaluating their transferability on other models, we randomly choose 500 adversarial examples that successfully cause the source model to make incorrect predictions. The transferability between each possible pair of models is shown in Appendix A.2.

2.2 Significance of Various Factors

To find out which factor affects the transferability of adversarial examples the most, we vary one factor at a time while fixing all the others for each model in the pool, and compare the transferability rates between them. For example, we take a 2-layer word-based LSTM model randomly initialized, denoted as "LSTM-Word-Random-2", as a target model. If we want to know the impact of network architecture, we generate $1,000$ adversarial examples each by attacking BiLSTM-Word-Random-2, and CNN-Word-Random-2, and use randomly selected 500 examples each of successful attack to evaluate the robustness of the target model. If we want to understand the impact of word embedding, the adversarial examples will be crafted by LSTM-Word-GloVe-2, LSTM-Word-word2vec-2, and LSTM-Word-fastText-2 models.

Table 2: Relative adversarial transferability rate to the base transferability rate on AGNEWS and MR datasets under PWWS and GA attacks.

Transferability	AGNEWS		MR	
	PWWS	GA	PWWS	GA
Architecture	0.197	0.018	0.145	0.021
Tokenization	0.286	0.030	0.285	0.049
Embedding	0.085	0.015	0.114	0.015
Capacity	0.066	0.013	0.045	0.011

Table 3: Adversarial transferability rate among various neural network architectures on AGNEWS dataset under PWWS attack. The architectures listed in the rows are source models, those in the columns are target ones.

Model	LSTM	BiLSTM	CNN	BERT
LSTM	0.448	0.394	0.353	0.190
BiLSTM	0.387	0.420	0.337	0.183
CNN	0.343	0.334	0.442	0.169
BERT	0.357	0.346	0.348	0.396

Since some models may be *inherently* more vulnerable than others, we need to evaluate the *base transferability rate* to remove the effects that are not caused by the factors we consider. For each target model, we train two instances with the same setting but different random initialization to obtain its base transferability rate by generating adversarial examples against one model and testing them on another. This base transferability rate of a model will be subtracted from all the actual adversarial transferability rates obtained when taking the model as the test model[2]. We report the average of (subtracted) adversarial transferability rates in Table 2, these rates are averaged over the configurations (all possible pairs) described in Section 2.1. Note that the smaller the values are, the more the adversarial

[1]We did not use two recently proposed attack algorithms of BERT-Attack (Li et al., 2020) and BAE (Garg and Ramakrishnan, 2020) because they cannot guarantee that any substitute word is always synonymous with the original word.

[2]For example, the base transferability rate of Model A is 90% and that of Model B is 60%. If we also know that 70% of adversarial examples produced using Model C can cause Model A to make mistakes, and 50% of those are misclassified by Model B, we may conclude that the transferability rate of C-to-A (70%) is higher than that of C-to-B (50%), which is obviously wrong because the difference between the transferability rate of C-to-B (50%) and B's base transferability rate (60%) is much smaller than that between the rate of C-to-A (70%) and A's base rate (90%). We report such (subtracted) adversarial transferability rates in Table 2 only.

transferability rate is close to its base (intra-model) transferability rate. From these rates, we found the tokenization scheme has the greatest influence on the adversarial transferability, followed by network architecture, embedding type, and model capacity no matter what attack algorithm or dataset is used.

2.3 Intra-Factor Transferability

In the following, we drill down into each specific factor. Table 3 shows adversarial transferability among different network architectures and configurations. For example, the architecture of "BERT" includes three variants: vanilla BERT, RoBERTa, and ALBERT. Each cell (i, j) in the table reports the transferability between two classes of models i and j. The value of each cell is computed as follows: for each possible pair of models (s, t) where model s belongs to class i and model t belongs to class j, we first calculate the transferability rate between models s and j, i.e. the percentage of adversarial examples produced using model s misclassified by model t; we then average these transferability rates over all the possible pairs.

As shown in Table 3, the adversarial transferability is not symmetric, i.e. the transferability of the transfer pair (i, j) might be different from the pair (j, i). As expected, intra-model adversarial example transferability rates are consistently higher than inter-model transferability ones. The adversarial examples generated using BERTs transfer slightly worse than other models whereas BERTs show much more robust to adversarial samples produced using the models from other classes. It is probably because BERTs were pre-trained with large-scale data and take different tokenization scheme (i.e. sub-words). We found the models from BERT family tend to distribute their "attention" over more words of an input text than others, which makes it harder to change their predictions by perturbing just few words. In contrast, other models often "focus" on certain keywords when making predictions, which makes them more vulnerable to black-box transfer attacks (see Appendix A.3).

In Table 4, we report the impact of tokenization schemes and embedding types on the adversarial transferability. Each cell is obtained by the method as the values reported in Table 3. The pre-trained models show to be more robust against black-box transfer attacks no matter their word embeddings or other parameters or both are pre-trained with large-scale text data. Character-based models are more robust to transfer attacks than those taking words or sub-words as input, and their adversarial examples also transfer much worse than others.

2.4 Summary of Findings

Some findings on the adversarial transferability among models are summarized below:

- No matter what attack algorithm or dataset is used, the tokenization scheme has the greatest impact on the adversarial transferability, followed by the netowrk architecture, embedding type, and model capacity in the order of importance.
- The adversarial transfer is not symmetric, and the transferability rates of intra-model adversarial examples are consistently higher than those of inter-model ones.
- Pre-trained neural models show to be more robust against black-box transfer attacks no matter their word embeddings or other parameters or both are pre-trained with large-scale text data.
- The adversarial examples produced by attacking BERTs transfer slightly worse than others, but BERTs show much more robust to transfer adversarial attacks. We found that BERTs tend to distribute their "attention" over more words than others, which makes it harder to change their predictions by perturbing just few words. Similar observation has been observed in (Hsieh et al., 2019).
- Character-based models are more robust to transfer attacks than those taking words or sub-words as input, but their adversarial examples also transfer much worse than others.
- Among the models from BERT family, the models pre-trained with more data show to be more robust against black-box transfer attacks using the models pre-trained with less data, while the adversarial examples produced by attacking the former transfer slightly better than the latter.

We also found that the adversarial examples produced by using an ensemble with a small number of models are much more transferable than those by a single model. A small ensemble greatly speeds up the adversarial example generation process, which is useful to perform the test-time attacks when evaluating the robustness of local models or launch the online attack in a real-world simulated environment because attacking an ensemble consisting of all possible models is time-consuming and not cost-efficient. The next two questions are how to

Table 4: Adversarial transferability rates with different tokenization schemes and embedding types on AGNEWS dataset under PWWS attack. The models listed in the rows are source models, those in the columns are target ones.

Input	Random	GloVe	word2vec	fastText	Character	ELMo	BERT	Average
Random	0.457	0.389	0.445	0.434	0.214	0.315	0.166	0.346
GloVe	0.481	0.503	0.489	0.493	0.219	0.336	0.174	0.385
word2vec	0.473	0.413	0.472	0.461	0.216	0.316	0.165	0.360
fastText	0.481	0.442	0.482	0.488	0.222	0.330	0.169	0.373
Character	0.261	0.233	0.256	0.256	0.386	0.300	0.186	0.268
ELMo	0.406	0.379	0.401	0.405	0.256	0.679	0.216	0.392
BERT	0.348	0.329	0.343	0.348	0.328	0.408	0.396	0.357
Average	0.415	0.384	0.413	0.412	0.263	0.383	0.210	0.354

select few models from a pool of models to create an ensemble that has good coverage to attack all other models and whether adversarial examples produced by the ensemble can strongly transfer to other unseen models (not listed in the model pool). We will answer these two questions in Section 3.

The above findings can guide us to choose a pool of representative neural models, from which we select a small number of them to form an ensemble. For example, such a pool of models should include at least one neural network for each type of tokenization scheme, but it does not need to include too many networks of different depths. The smaller the number of models in a pool, the less the computational cost will be for estimating the transferability rate between any pair of them.

3 Highly-Transferable Examples

Next, we discuss how to find an optimal ensemble model that can be used to craft adversarial examples that strongly transfer across other models. We then distill the ensemble attack into adversarial word replacement rules that can be used to generate adversarial examples with high transferability. These rules can also help us to identify dataset biases and analyze global model behaviors.

3.1 Ensemble Method

Consider an ensemble model that outputs the prediction score for a class label by averaging over the scores of individual models, we can generate adversarial examples to fool the ensemble model by applying word substitution-based perturbations to input texts. We take the average of the logits produced by all the member models as the final prediction. Observing that the transferability is affected by various factors, and many factors need to be carefully considered when forming an ensemble, we propose a population-based genetic algorithm to find an optimal ensemble.

In the proposed algorithm, a candidate solution is a set of models $S = (s_1, s_2, \ldots, s_m)$, where m is a pre-defined size of ensemble. A fitness function evaluates each solution to decide whether it will contribute to the next generation of solutions. We define a function $r(s, t, a)$ as the percentage of adversarial samples produced using model s misclassified by model t under attack algorithm a. For a solution S, the fitness function $f(S)$ that returns a measure of the candidate's fitness which we want to maximize is defined as follows:

$$f(S) = \sum_{t_j \in T} \left\{ \max_{s_i \in S, s_i \neq t_j} \left[\min_{a_k \in A} r(s_i, t_j, a_k) \right] \right\} / |T|, \quad (1)$$

where T is a pool of representative models under investigation, $|T|$ is the cardinality of T, and A is a set of attack algorithms. Let $P(n)$ define a population of candidate solutions at the n-th generation : $P(n) = \{s_1^n, s_2^n, \ldots, s_m^n\}$. Initial populations $P(0)$ are selected randomly. After evaluating each candidate by the fintness function, the algorithm takes two candidate solutions based on fitness, merges their sets, and then randomly selects m models from the set to produce new candidates. The mutation is another important genetic operator that takes a single candidate and randomly replaces at most one of its models with another one from T. The algorithm continues until the number of generations reaches the maximum value.

In order to evaluate the ensembles found by the population-based algorithm, we ask a senior researcher to select the ensembles as a baseline. This researcher uses a simple strategy to make selection: first choose the model whose adversarial examples yield the highest transferability, and gradually add complementary models which are different from those already in the ensemble in the aspects of tokenization scheme, architecture, and embedding type. We list the ensembles selected by the algorithm and the human expert in Appendix A.4.

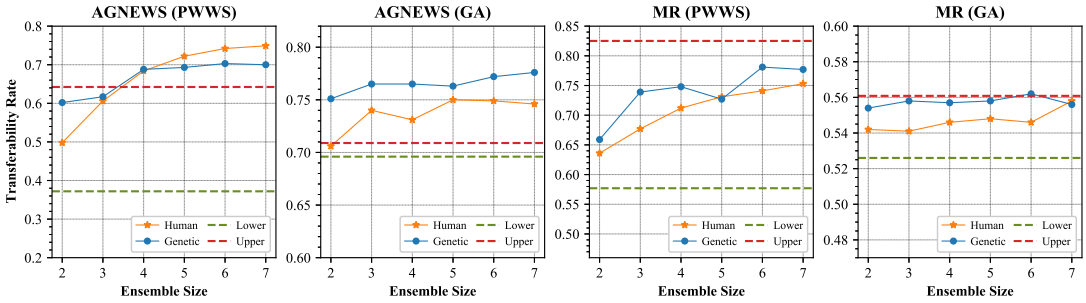

Figure 2: Transferability rates of the adversarial examples generated using different ensembles with various sizes on both AGNEWS and MR datasets under two attack algorithms (PWWS and GA). The upper red dotted line represents the average of the *base transferability rates* (defined in Section 2.2) that theoretically are highest rates that can be achieved using a single local model, and the lower green line shows the average of transferability rates over all the possible pairs of models. The ensemble method clearly outperforms the single model-based transfer method, and in most cases the adversarial examples produced using the ensembles founded by the proposed genetic algorithm transfer better across different models than those selected by a human expert.

In Figure 2, we show the transferability rates of the adversarial examples produced using the ensembles with various sizes on both AGNEWS and MR datasets under two attack algorithms (PWWS and GA). The reported transferability rates are averaged over all the remaining models except those used to produce the adversarial examples. We found that in most cases the adversarial examples produced using the ensemble founded by the genetic algorithm transfer better across different models than those selected by a human expert, especially when the ensemble size is small. The ensemble method performs superior to a single model-based transfer method, and in some cases the transferability rates achieved by the ensemble method even go beyond the upper red dotted line (i.e. the highest rates that can be achieved by using a single local model). When the ensemble size is greater than 6, the marginal gains in average transferability rate decrease no matter what attack algorithm or dataset is used in our experimental setting.

3.2 Mining Word Replacement Rules

We have shown in Section 3.1 that the adversarial examples generated by the ensemble whose members are carefully selected can strongly transfer to other models. We hypothesize that if we can distill the ensemble attack into some word replacement rules, the adversarial examples crafted by applying the distilled rules to perturb input texts also can transfer well across different models. In this section, we want to discover such word replacement rules using an ensemble model, and those rules are expected to be used to generate the model-agnostic examples of transferable hostility. Besides, such rules (if any) also can help us to understand and identify dataset biases "unknowingly" exploited by the models for prediction.

Inputs:
$\mathcal{D}$: a set of training examples.
$\mathcal{Z}$: a set of class labels.
g: an ensemble model that outputs a logit for each class $z \in \mathcal{Z}$.
Output: a set of word replacement rules as well as their salience.
Algorithm:
for each training instance (x, y) in $\mathcal{D}$
  for each word w_i in the input text x
    for each word $\hat{w}_i$ that can be used to replace w_i
      $\hat{x_i}$ = replace w_i with $\hat{w}_i$ in x.
      $c(y, w_i \rightarrow \hat{w}_i) = c(y, w_i \rightarrow \hat{w}_i) + 1$.
      for each label $z \in \mathcal{Z}$
        if $z = y$ **then**
          $h(y, w_i \rightarrow \hat{w}_i) = h(y, w_i \rightarrow \hat{w}_i) + g(x_i; z) - g(\hat{x_i}; z)$.
        else
          $h(y, w_i \rightarrow \hat{w}_i) = h(y, w_i \rightarrow \hat{w}_i) + g(\hat{x_i}; z) - g(x_i; z)$.
for each word replacement rule
  $h(z, w \rightarrow \hat{w}) = h(z, w \rightarrow \hat{w}) / c(z, w_i \rightarrow \hat{w}_i)$.

Figure 3: An algorithm to discover highly-transferable adversarial word replacement rules.

A word replacement rule is defined as a pair $(z, w \rightarrow \hat{w})$, where z is a class label, and $w \rightarrow \hat{w}$ means to replace the original word w with $\hat{w}$ when the gold label is z. Each rule is associated with a salience $h(z, w \rightarrow \hat{w})$ specifying the priority of the rule, and a higher number denotes a higher priority. We propose an algorithm to discover Highly-transferable Adversarial Word Replacement (HAWR) rules (see Figure 3). The idea behind this algorithm is to estimate the changes in

log-likelihood caused by the word replacements. Once such rules are obtained, they can be used to generate adversarial examples as follows: given an input sentence x and its label y, we find a word w_i in x which has the highest value of $h(y, w_i, \hat{w}_i)$ and replace w_i with $\hat{w}_i$ in x; for all the remaining words in x we repeat the above step until the percentage of words that can be altered reach a given threshold. Note that such adversarial examples can be generated without access to target models.

We report the attack success rates of the adversarial examples generated by applying HAWR rules in Table 5. The attacks based on HAWR rules are comparable to PWWS and GA algorithms that require a large number of queries to the victim model, while the attackers using HAWR do not need to access the victim models. We use the ensemble consisting of six models founded by our genetic algorithm to discover these HAWR rules in this experiment. We list five adversarial word replacement rules each for the positive and negative categories discovered from MR dataset in Table 6.

To understand these adversarial word replacement rules, we analyze their pointwise mutual information (PMI) between words and class label before and after the replacements. The PMI of a pair of discrete random variables quantifies the discrepancy between the probability of their coincidence given their joint distribution and their individual distributions. In this case, it is used to find collocations and associations between words and labels, and the PMI of a word w and a label $z \in \mathcal{Z}$ can be computed as $\text{PMI}(w, z) = p(w, z)/p(w)p(z)$, where $p(\cdot)$ assigns a probability to each possible value. The results show that the PMI are significantly different for the word and its replacement even though they are synonyms. This demonstrates the data bias in the training data.

We obtain similar word replacement rules by ranking all the hypothesis words according to their PMI with each label from a training set. For each label $z \in \mathcal{Z}$ and a possible word replacement $w \rightarrow \hat{w}$, the similar salience of $h(z, w \rightarrow \hat{w})$ can be computed as follows:

$$h(z, w \rightarrow \hat{w}) = [\text{PMI}(w, z) - \text{PMI}(\hat{w}, z)] + \sum_{z' \in \mathcal{Z}, z' \neq z} [\text{PMI}(\hat{w}, z') - \text{PMI}(w, z')] . \quad (2)$$

We also report the attack success rates of the adversarial examples generated by the word replacement rules obtained using PMI only in Table 5. The adversarial examples produced by HAWR rules achieved stronger transferability than those by PMI rules. We believe that it is because HAWR rules are distilled using the logits predicted by models, and the changes in the logits reflect both the characteristics of neural networks and the contexts in which those word replacements are applied.

3.3 Case Study: Natural Language Inference

To evaluate the generalizability of the proposed method, we redo the entire process (illustrated in Figure 1) on a new task of natural language inference (NLI) as a case study. We conducted the experiments of this task on Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) dataset. The HAWR rules generated by our algorithm were tested on three models (ESIM, DecompAtt and XLNet listed in Table 7) that are *unseen* during the process of finding these rules, and compared to a recently proposed attack algorithm, called TextFooler (Jin et al., 2020), which has *not* been used.

We reuse 63 different neural models for text classification (see Appendix A.1) to create a model pool for this task. To perform NLI task, each model in the pool encodes the premise and hypothesis separately and then feeds the concatenation of these encodings to a two-layer feedforward network. We use the ensemble with six models (see Appendix A.5) identified by our generic algorithm to discover HAWR rules based on the adversarial transferability rates between any pair of models in the pool. As shown in Table 7, the attacks based on HAWR rules are comparable to TextFooler that requires many queries to the victim models.

4 Related Work

Transfer-based Attacks Observing that adversarial examples often transfer across different models (Szegedy et al., 2013), the attackers run standard white-box attacks on local surrogate models to find adversarial examples that are expected to transfer to the target models. Unfortunately, such a straightforward strategy often suffers from overfitting to specific weaknesses of local models and transfer-based attacks typically have much lower success rates than the attacks directly launched on the target models. To resolve this problem, many methods have been proposed to improve the transfer success rate of adversarial examples on the target models by perturbing mid-layer activations (Zhou et al., 2018; Huang et al., 2019; Inkawhich et al., 2020; Wu et al., 2020), adding regularization terms to the example generation process (Dong et al., 2018;

Table 5: Attack success rates of the adversarial examples generated by applying the word replacement rules found by our algorithm (HAWR) and pointwise mutual information (PMI) on all 63 and three representative models with AGNEWS and MR datasets, comparing to PWWS and GA attack algorithms. "Succ%" denotes the attack success rate in terms of the number of sentences, and "Qry#" the average number of queries to the victim model required by the attack algorithms. The maximum percentage of words that are allowed to be perturbed was set to 30%.

Success Rate	AGNEWS								MR							
	HAWR		PMI		PWWS		GA		HAWR		PMI		PWWS		GA	
	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#	Succ%	Qry#
ALL	69.0	0.0	39.6	0.0	69.1	175.6	82.4	380.6	87.7	0.0	79.0	0.0	92.0	97.6	95.8	130.5
LSTM	75.9	0.0	45.1	0.0	64.1	175.7	80.7	368.3	95.3	0.0	87.3	0.0	95.2	97.2	97.4	104.3
CNN	75.1	0.0	39.7	0.0	76.1	174.7	88.7	316.6	89.1	0.0	84.3	0.0	93.5	97.2	96.6	100.2
BERT	33.0	0.0	19.7	0.0	33.2	178.2	65.5	498.2	58.3	0.0	45.1	0.0	78.5	98.2	92.8	167.4

Table 6: Five adversarial word replacement rules discovered from MR dataset each for the positive and negative categories as well as their changes in the pointwise mutual information (PMI).

Class	Word Substitution			PMI(Word; Positive)	PMI(Word; Negative)
Positive	flaws	→	flaw	6.408 → 2.086 (**4.392** ↓)	0.000 → 6.478 (**6.408** ↑)
	average	→	mediocre	5.741 → 2.086 (**3.655** ↓)	5.004 → 6.408 (**1.404** ↑)
	glorious	→	splendiferous	6.478 → 0.000 (**6.478** ↓)	0.000 → 0.000 (**0.000** ↑)
	web	→	network	6.478 → 0.000 (**6.478** ↓)	0.000 → 6.478 (**6.478** ↑)
	brilliant	→	brainy	6.168 → 0.000 (**6.168** ↓)	4.109 → 6.478 (**2.369** ↑)
Negative	toilet	→	bathroom	0.000 → 6.478 (**6.478** ↑)	6.478 → 0.000 (**6.478** ↓)
	excellent	→	splendid	6.013 → 6.478 (**0.466** ↑)	4.620 → 0.000 (**4.620** ↓)
	bizarre	→	outlandish	5.478 → 6.478 (**1.000** ↑)	5.478 → 0.000 (**5.478** ↓)
	excruciating	→	harrowing	0.000 → 6.256 (**6.256** ↑)	6.478 → 3.671 (**2.807** ↓)
	routine	→	everyday	2.230 → 6.478 (**4.248** ↑)	6.400 → 0.000 (**6.400** ↓)

Table 7: Attack success rates on SNLI. "Succ%" and "Qry#" have the same meaning as Tabel 5. The maximum percentage of words perturbed was set to 30%.

Success Rate	HAWR		TextFooler	
	Succ%	Qry#	Succ%	Qry#
ESIM (Chen et al., 2017)	72.4	0.0	72.9	24.8
DecomAtt (Parikh et al., 2016)	73.8	0.0	75.8	23.7
XLNet (Jin et al., 2020)	68.4	0.0	67.7	24.4

Huang et al., 2019), or ensembling multiple local models (Wu et al., 2018; Tramèr et al., 2018; Liu et al., 2017; Wallace et al., 2019).

The proposed ensemble-based methods resemble to Liu et al. (2017), which hypothesized that if an adversarial example remains adversarial for multiple models, then it is more likely to transfer to other models as well. Wu et al. (2018) found that the local non-smoothness of loss surface harms the transferability of adversarial examples, and proposed a variance-reduced attack to enhance the transferability by applying the locally averaged gradient to reduce the local oscillation of loss surface. The existing studies on the ensemble-based transfer attacks are mainly conducted in the image domain (Wu et al., 2018; Tramèr et al., 2018; Liu et al., 2017), transfer-based attacks for NLP models are relatively much underexplored.

Discovering Adversarial Word Replacement Rules Ribeiro et al. (2018) presented semantic-preserving perturbations that cause models to change their predictions by the paraphrases generated using back-translation, and generalized these perturbations into universal replacement rules that induce adversaries on many text instances. They use the word "universal" to mean that their replacement rules can be used to any input text if the rules are matched with the input and these rules were generalized across some specific models. With a different goal, we aim to find the highly-transferable adversarial replacement rules by which the crafted adversarial examples can fool almost all models. Besides, the number of their replacement rules is relatively small compared to ours.

5 Conclusion

We investigated four critical factors of NLP neural models, including network architectures, tokenization schemes, embedding types, and model capacities and how they impact the transferability of text adversarial examples with more than sixty different models. We also proposed a genetic algorithm to find an optimal ensemble of very few models that can be used to generate adversarial examples that transfer well to all the other mod-

els. Then, we described a algorithm to discover highly-transferable adversarial word replacement rules that can be applied to craft adversarial examples with strong transferability across various neural models without access to any of them. Finally, since those adversarial examples are model-agnostic, they provide an analysis of global model behavior and help to identify dataset biases.

Acknowledgements

This work was partly supported by Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103) and National Science Foundation of China (No. 62076068). CJH is supported by NSF IIS-1901527, IIS-2008173 and IIS-2048280. KWC is supported by an Amazon Research Award.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. 2020. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. 2018. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *the Annual Meeting of the Association for Computational Linguistics*.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge belongie, and Ser-Nam Lim. 2019. Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE International Conference on Computer Vision*.

Nathan Inkawhich, Kevin J. Liang, Lawrence Carin, and Yiran Chen. 2020. Transferable perturbations of deep feature distributions-feature space. In *Proceedings of the Conference of the International Conference on Learning Representations*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? natural language attack on text classification and entailment. In *the AAAI Conference on Artificial Intelligence*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. FastText.zip: Compressing text classification models. *arXiv*, 1612.03651.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

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

the Conference of the International Conference on Learning Representations.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv*, 1907.11692.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv*, 1301.3781.

George A. Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the conference on empirical methods in natural language processing*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv*, 1707.02812.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2013. Intriguing properties of neural networks. *arXiv*, 1312.6199.

Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proceedings of the Conference of International Conference on Learning Representations*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing*.

Catherine Wong. 2017. DANCin SEQ2SEQ: Fooling text classifiers with adversarial text example generation. *arXiv*, 1712.05419.

Lei Wu, Zhanxing Zhu, Cheng Tai, and Weinan E. 2018. Understanding and enhancing the transferability of adversarial examples. In *Proceedings of the Annual Conference on Neural Information Processing Systems*.

Weibin Wu, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R. Lyu, and Yu-Wing Tai. 2020. Boosting the transferability of adversarial samples via attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. 2020. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *Journal of Machine Learning Research*, 21(43):1–36.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of Annual Meeting of Association for Computational Linguistics*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the Conference on Neural Information Processing Systems*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Wen Zhou, Xin Hou, Yongjun Chen, Mengyun Tang, Xiangqi Huang, Xiang Gan, and Yong Yang. 2018. Transferable adversarial perturbations. In *Proceedings of European Conference on Computer Vision*.

Appendix

A.1 All Neural Models under Investigation

We systematically investigated many popular architectures of neural models with different configurations. Specifically, we consider various network architectures (LSTM, BiLSTM, CNN, or BERT), tokenization schemes (Word, character, or word + character, denoted by "W", "C", "WC" respectively), word embeddings (randomly-initialized, GloVe, word2vec, or fastText), and model capacities (various numbers of layers). All models under investigation are listed in Table 8, and we believe that they cover the popular neural networks that have been used for text classification tasks in NLP literature.

Table 8: All neural models under investigation.

ID	Model	ID	Model
[1]	LSTM-W-Random-1	[2]	LSTM-W-GloVe-1
[3]	LSTM-W-word2vec-1	[4]	LSTM-W-fastText-1
[5]	LSTM-C-Random-1	[6]	LSTM-WC-ELMo-1
[7]	LSTM-W-Random-2	[8]	LSTM-W-GloVe-2
[9]	LSTM-W-word2vec-2	[10]	LSTM-W-fastText-2
[11]	LSTM-C-Random-2	[12]	LSTM-WC-ELMo-2
[13]	LSTM-W-Random-4	[14]	LSTM-W-GloVe-4
[15]	LSTM-W-word2vec-4	[16]	LSTM-W-fastText-4
[17]	LSTM-C-Random-4	[18]	LSTM-WC-ELMo-4
[19]	BiLSTM-W-Random-1	[20]	BiLSTM-W-GloVe-1
[21]	BiLSTM-W-word2vec-1	[22]	BiLSTM-W-fastText-1
[23]	BiLSTM-C-Random-1	[24]	BiLSTM-WC-ELMo-1
[25]	BiLSTM-W-Random-2	[26]	BiLSTM-W-GloVe-2
[27]	BiLSTM-W-word2vec-2	[28]	BiLSTM-W-fastText-2
[29]	BiLSTM-C-Random-2	[30]	BiLSTM-WC-ELMo-2
[31]	BiLSTM-W-Random-4	[32]	BiLSTM-W-GloVe-4
[33]	BiLSTM-W-word2vec-4	[34]	BiLSTM-W-fastText-4
[35]	BiLSTM-C-Random-4	[36]	BiLSTM-WC-ELMo-4
[37]	CNN-W-Random-1	[38]	CNN-W-GloVe-1
[39]	CNN-W-word2vec-1	[40]	CNN-W-fastText-1
[41]	CNN-C-Random-1	[42]	CNN-WC-ELMo-1
[43]	CNN-W-Random-2	[44]	CNN-W-GloVe-2
[45]	CNN-W-word2vec-2	[46]	CNN-W-fastText-2
[47]	CNN-C-Random-2	[48]	CNN-WC-ELMo-2
[49]	CNN-W-Random-4	[50]	CNN-W-GloVe-4
[51]	CNN-W-word2vec-4	[52]	CNN-W-fastText-4
[53]	CNN-C-Random-4	[54]	CNN-WC-ELMo-4
[55]	CNN-W-Random-6	[56]	CNN-W-GloVe-6
[57]	CNN-W-word2vec-6	[58]	CNN-W-fastText-6
[59]	CNN-C-Random-6	[60]	CNN-WC-ELMo-6
[61]	BERT	[62]	RoBERTa
[63]	ALBERT		

A.2 Transferability among Different Neural Models

We show in Figure 4 the transferability rate among all neural models in the model pool. The column and row headers indicate the IDs of source and target models respectively. The mapping of IDs and the corresponding models is shown in Figure 8. We generate adversarial examples by attacking a source model, and report the transferability rates on a target (or victim) model.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	.71	.50	.68	.66	.21	.30	.50	.39	.48	.48	.22	.27	.51	.49	.57	.50	.21	.29	.44	.38
2	.55	.63	.58	.59	.21	.33	.46	.47	.46	.48	.20	.31	.49	.59	.52	.50	.18	.30	.45	.47
3	.68	.53	.70	.70	.22	.28	.49	.41	.50	.48	.21	.26	.52	.50	.58	.55	.20	.27	.44	.41
4	.66	.60	.68	.73	.24	.31	.49	.45	.51	.49	.23	.28	.58	.56	.63	.59	.21	.29	.48	.46
5	.27	.23	.28	.29	.43	.29	.27	.23	.28	.27	.40	.28	.27	.25	.28	.27	.41	.28	.26	.23
6	.44	.39	.42	.44	.28	.89	.42	.37	.38	.41	.26	.74	.43	.40	.45	.44	.29	.82	.39	.38
7	.47	.39	.47	.46	.21	.30	.62	.44	.51	.56	.23	.30	.42	.39	.43	.43	.20	.29	.48	.42
8	.48	.56	.51	.52	.23	.31	.52	.63	.56	.55	.21	.30	.48	.49	.52	.47	.21	.27	.53	.55
9	.46	.44	.48	.48	.21	.28	.57	.47	.56	.63	.22	.29	.46	.45	.49	.45	.20	.27	.50	.44
10	.49	.46	.49	.50	.22	.28	.57	.49	.55	.65	.23	.30	.44	.44	.46	.45	.21	.27	.47	.48
11	.25	.25	.28	.28	.40	.30	.28	.25	.28	.28	.44	.29	.26	.24	.29	.28	.38	.26	.30	.24
12	.44	.43	.44	.46	.28	.71	.44	.39	.43	.42	.28	.82	.43	.39	.43	.43	.27	.63	.44	.39
13	.57	.50	.57	.61	.23	.36	.49	.45	.50	.47	.22	.35	.63	.56	.63	.63	.21	.33	.50	.45
14	.58	.66	.63	.67	.24	.34	.50	.53	.54	.56	.24	.31	.57	.67	.63	.60	.23	.30	.51	.56
15	.56	.45	.56	.58	.24	.30	.45	.41	.46	.46	.21	.26	.63	.50	.67	.59	.21	.26	.44	.39
16	.59	.52	.57	.63	.25	.31	.50	.43	.47	.47	.22	.31	.65	.54	.73	.65	.21	.28	.47	.45
17	.27	.25	.30	.29	.45	.31	.28	.21	.26	.28	.43	.31	.29	.26	.29	.29	.48	.30	.28	.23
18	.43	.42	.42	.43	.28	.86	.40	.40	.40	.42	.27	.75	.43	.41	.47	.45	.27	.83	.41	.40
19	.47	.42	.46	.46	.25	.30	.53	.41	.51	.49	.24	.31	.46	.40	.48	.47	.23	.31	.56	.45
20	.53	.60	.57	.59	.26	.33	.57	.65	.61	.61	.26	.33	.54	.57	.58	.55	.25	.31	.58	.66
21	.48	.43	.46	.47	.23	.30	.57	.46	.57	.54	.22	.30	.45	.42	.45	.45	.21	.27	.66	.46
22	.50	.46	.52	.52	.25	.29	.58	.53	.61	.60	.24	.29	.48	.46	.52	.51	.23	.27	.57	.51
23	.28	.26	.28	.28	.42	.30	.29	.25	.28	.30	.47	.28	.27	.24	.28	.30	.42	.26	.29	.26
24	.42	.37	.40	.41	.26	.74	.40	.38	.39	.41	.24	.79	.40	.37	.40	.40	.24	.67	.40	.36
25	.61	.54	.58	.64	.29	.35	.53	.49	.53	.54	.24	.33	.68	.55	.71	.64	.25	.33	.53	.48
26	.60	.64	.61	.68	.24	.36	.52	.49	.53	.55	.19	.32	.60	.66	.68	.66	.22	.32	.52	.49
27	.57	.51	.57	.60	.27	.32	.48	.47	.48	.48	.23	.31	.69	.56	.69	.66	.23	.33	.49	.46
28	.60	.52	.57	.62	.26	.36	.52	.48	.50	.51	.24	.34	.66	.57	.67	.67	.24	.35	.49	.50
29	.29	.26	.28	.30	.49	.33	.29	.25	.29	.28	.47	.31	.27	.26	.29	.30	.49	.28	.28	.25
30	.45	.40	.44	.45	.31	.79	.40	.38	.40	.41	.29	.72	.44	.41	.45	.45	.27	.77	.39	.39
31	.42	.39	.44	.45	.23	.30	.53	.42	.50	.46	.23	.31	.46	.40	.48	.46	.21	.28	.54	.41
32	.53	.53	.54	.58	.24	.32	.53	.59	.57	.56	.24	.32	.52	.54	.56	.53	.24	.31	.54	.62
33	.44	.41	.47	.47	.24	.31	.56	.45	.51	.52	.24	.31	.50	.43	.52	.50	.23	.30	.58	.45
34	.48	.48	.51	.53	.27	.34	.56	.53	.59	.55	.25	.36	.51	.47	.55	.50	.24	.32	.63	.50
35	.28	.23	.30	.29	.48	.32	.29	.22	.28	.29	.50	.30	.29	.24	.30	.29	.45	.30	.30	.25
36	.40	.39	.41	.39	.29	.70	.38	.38	.38	.40	.26	.78	.41	.37	.42	.42	.26	.65	.39	.35
37	.32	.28	.32	.32	.18	.27	.30	.27	.28	.31	.16	.24	.31	.29	.36	.34	.16	.24	.29	.28
38	.44	.42	.43	.47	.21	.32	.41	.41	.42	.44	.21	.29	.44	.44	.45	.44	.19	.29	.39	.38
39	.34	.30	.32	.35	.17	.27	.34	.29	.34	.34	.19	.25	.34	.31	.35	.36	.15	.25	.32	.30
40	.39	.34	.37	.39	.19	.30	.37	.34	.37	.36	.18	.27	.39	.36	.39	.40	.17	.27	.36	.33
41	.27	.23	.27	.28	.37	.31	.28	.23	.25	.27	.36	.28	.30	.25	.29	.29	.37	.28	.27	.25
42	.41	.36	.40	.41	.24	.57	.38	.36	.38	.41	.24	.50	.40	.38	.41	.43	.24	.49	.37	.35
43	.39	.35	.40	.41	.17	.30	.37	.33	.37	.39	.19	.26	.40	.36	.42	.39	.19	.26	.36	.33
44	.44	.43	.45	.46	.21	.31	.46	.42	.44	.47	.19	.29	.48	.46	.51	.49	.19	.29	.41	.41
45	.43	.39	.41	.45	.19	.30	.43	.37	.41	.44	.20	.27	.43	.40	.46	.47	.17	.27	.42	.39
46	.43	.39	.43	.45	.18	.33	.43	.38	.40	.43	.21	.28	.43	.42	.42	.44	.19	.29	.40	.40
47	.24	.19	.24	.22	.32	.26	.23	.21	.24	.23	.30	.26	.25	.20	.25	.25	.31	.25	.24	.20
48	.42	.39	.43	.43	.24	.58	.42	.37	.38	.41	.24	.53	.42	.40	.43	.43	.22	.52	.39	.35
49	.37	.36	.38	.37	.18	.27	.36	.33	.34	.39	.20	.24	.37	.33	.38	.39	.18	.25	.36	.33
50	.43	.41	.43	.46	.20	.33	.43	.40	.44	.48	.22	.30	.44	.45	.46	.46	.18	.29	.41	.40
51	.44	.42	.46	.44	.21	.31	.42	.36	.42	.44	.21	.30	.44	.41	.46	.43	.19	.29	.44	.39
52	.44	.40	.45	.47	.20	.32	.45	.42	.43	.49	.21	.30	.47	.43	.48	.48	.20	.30	.42	.41
53	.25	.22	.27	.27	.31	.27	.28	.22	.25	.24	.33	.26	.25	.23	.25	.25	.32	.27	.26	.23
54	.45	.39	.43	.45	.23	.59	.41	.39	.41	.42	.24	.56	.44	.40	.45	.44	.25	.51	.38	.38
55	.39	.38	.40	.40	.19	.29	.38	.35	.37	.43	.20	.25	.41	.37	.40	.43	.17	.25	.40	.36
56	.44	.45	.45	.47	.22	.32	.44	.45	.48	.47	.22	.32	.48	.47	.49	.48	.19	.30	.45	.43
57	.43	.39	.42	.43	.22	.29	.42	.38	.38	.43	.22	.28	.44	.39	.44	.44	.19	.28	.39	.38
58	.41	.41	.43	.42	.21	.31	.41	.40	.39	.45	.21	.29	.44	.40	.44	.44	.17	.27	.41	.39
59	.26	.22	.27	.27	.32	.27	.26	.23	.24	.28	.36	.27	.28	.22	.29	.28	.33	.27	.25	.22
60	.44	.39	.42	.44	.25	.57	.41	.37	.40	.43	.23	.53	.43	.41	.45	.45	.25	.50	.39	.37
61	.36	.36	.37	.37	.32	.40	.35	.32	.36	.37	.32	.38	.35	.33	.31	.35	.32	.37	.34	.33
62	.41	.38	.42	.42	.41	.51	.41	.37	.41	.42	.40	.48	.44	.41	.40	.43	.41	.49	.39	.40
63	.29	.28	.30	.31	.28	.34	.27	.26	.28	.29	.25	.32	.31	.29	.31	.30	.27	.33	.29	.27

Figure 4: Transferability among neural models (part 1).

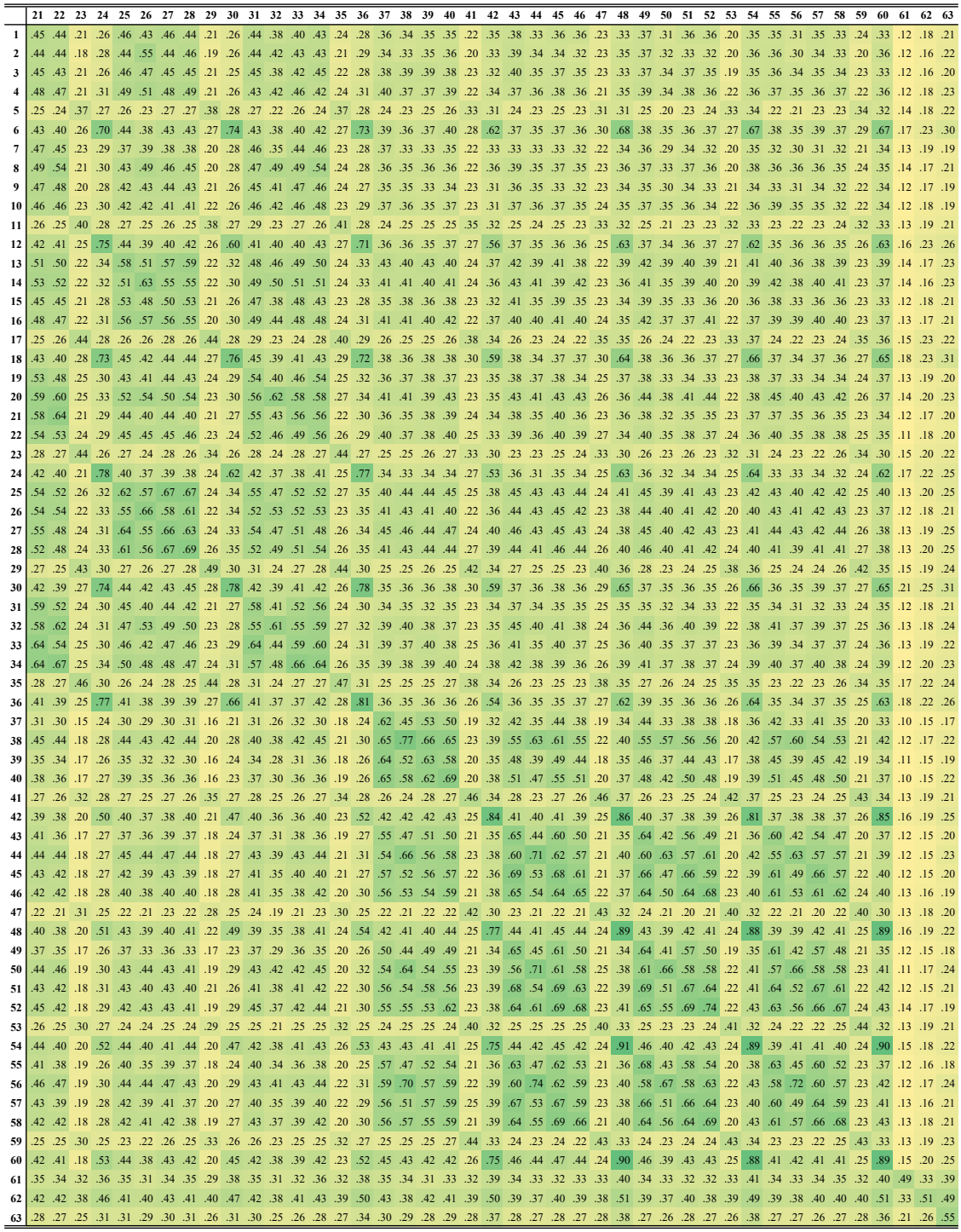

Figure 4: Transferability among neural models (part 2).

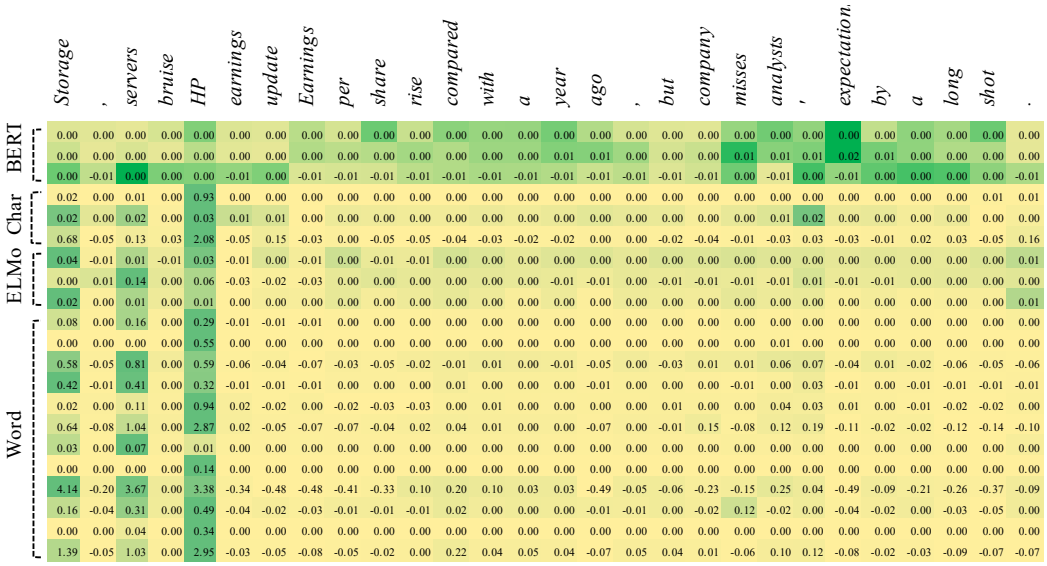

Figure 6: Importance of original words.

A.3 Heatmap of Word Importance

To study how each word in the sentence impacts the prediction of the model, we define word importance as follows:

- For an original word, its importance is calculated as the difference between the log likelihood of a gold label before and after the original word is replaced with a special "unknown" symbol (<unk>).
- For a substitute word, its importance is estimated as the difference between the log likelihood of a gold label predicted by the model before and after the original word is replaced with the substitute one.

Figure 6 and 7 show the importance of original and substitute words for different models. We here only consider the models (with one hidden layer) listed in Figure 8 and take the following sentence as an example input:

Storage, servers bruise HP earnings update Earnings per share rise compared with a year ago, but company misses analysts' expectations by a long shot.

We observed that different models generally show similar behavior: for the original words, most of the models mainly focus on three words, namely "Storage", "servers" and "HP"; for the substitute words, the attentions have been given to the word "depot" for most of the models. Thanks to such a similarity, it is possible to generate the adversarial

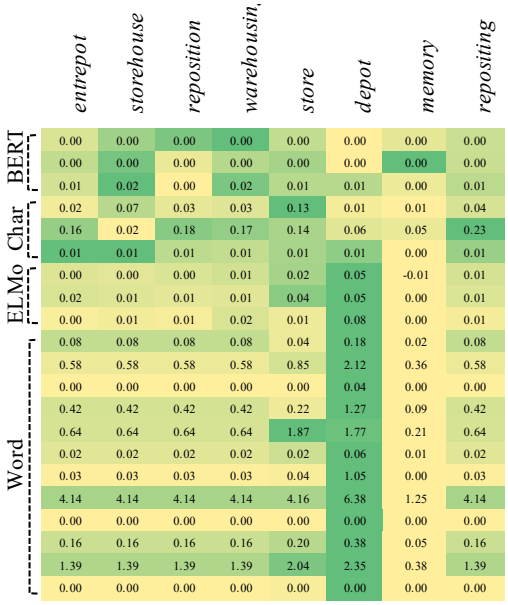

Figure 7: Importance of the words used to replace an original word "Storage" (the first word in the sentence).

examples using one model, which strongly transfer to the others.

However, the models from BERT family show much more robust to transfer adversarial attacks. They tend to distribute their "attention" over more words both for original words and substitute words. As to character-based models, they also distribute their attention in a way that is clearly different from the other models. These differences can explain the lower transferability rates achieved by the adversarial examples generated by using BERTs and character-based models.

Table 9: Different ensembles selected by human expert and algorithm on text classification task.

Human	2	LSTM-W-Random-1, CNN-C-Random-1
	3	LSTM-W-Random-1, CNN-C-Random-1, LSTM-WC-ELMo-1
	4	LSTM-W-Random-1, CNN-C-Random-1, LSTM-WC-ELMo-1, BERT
	5	LSTM-W-Random-1, CNN-C-Random-1, LSTM-WC-ELMo-1, BERT, CNN-W-Random-1
	6	LSTM-W-Random-1, CNN-C-Random-1, LSTM-WC-ELMo-1, BERT, CNN-W-Random-1, LSTM-C-Random-1
	7	LSTM-W-Random-1, CNN-C-Random-1, LSTM-WC-ELMo-1, BERT, CNN-W-Random-1, LSTM-C-Random-1, CNN-WC-ELMo-1
Algorithm		**AGNEWS (PWWS)**
	2	LSTM-WC-ELMo-4, CNN-W-GloVe-6
	3	BiLSTM-W-GloVe-2, LSTM-WC-ELMo-4, CNN-W-fastText-4
	4	LSTM-WC-ELMo-1, BiLSTM-W-GloVe-2, CNN-W-fastText-4, RoBERTa
	5	LSTM-WC-ELMo-1, BiLSTM-W-GloVe-2, LSTM-W-Random-4, CNN-W-fastText-4, RoBERTa
	6	LSTM-WC-ELMo-2, BiLSTM-W-GloVe-2, LSTM-W-Random-4, CNN-W-fastText-4, CNN-WC-ELMo-4, RoBERTa
	7	LSTM-WC-ELMo-2, BiLSTM-W-GloVe-2, LSTM-W-Random-4, CNN-W-fastText-4, CNN-WC-ELMo-4, CNN-W-GloVe-6, RoBERTa
		AGNEWS (GA)
	2	LSTM-WC-ELMo-1, CNN-WC-ELMo-1
	3	LSTM-WC-ELMo-1, CNN-C-Random-1, CNN-WC-ELMo-1
	4	BiLSTM-WC-ELMo-4, CNN-C-Random-1, CNN-WC-ELMo-1, CNN-WC-ELMo-6
	5	BiLSTM-WC-ELMo-4, CNN-C-Random-1, CNN-WC-ELMo-1, CNN-WC-ELMo-6, RoBERTa
	6	LSTM-WC-ELMo-4, BiLSTM-WC-ELMo-4, CNN-C-Random-1, CNN-WC-ELMo-1, CNN-WC-ELMo-6, RoBERTa
	7	LSTM-WC-ELMo-4, BiLSTM-WC-ELMo-4, CNN-C-Random-1, CNN-WC-ELMo-1, CNN-WC-ELMo-2, CNN-WC-ELMo-6, RoBERTa
		MR (PWWS)
	2	LSTM-C-Random-4, CNN-WC-ELMo-6
	3	LSTM-C-Random-4, BiLSTM-W-GloVe-4, CNN-WC-ELMo-2
	4	LSTM-C-Random-4, BiLSTM-W-GloVe-4, CNN-W-fastText-2, CNN-WC-ELMo-2
	5	LSTM-W-word2vec-1, LSTM-C-Random-4, CNN-W-fastText-2, CNN-WC-ELMo-2, CNN-W-GloVe-6
	6	LSTM-W-word2vec-1, LSTM-C-Random-4, CNN-W-fastText-2, CNN-WC-ELMo-2, CNN-W-GloVe-6, RoBERTa
	7	LSTM-W-word2vec-1, LSTM-C-Random-4, BiLSTM-W-GloVe-4, CNN-WC-ELMo-2, CNN-W-GloVe-6, CNN-W-word2vec-6, RoBERTa
		MR (GA)
	2	BiLSTM-W-GloVe-4, RoBERTa
	3	LSTM-C-Random-4, BiLSTM-W-GloVe-4, RoBERTa
	4	LSTM-W-Random-1, LSTM-C-Random-4, CNN-W-GloVe-1, RoBERTa
	5	LSTM-W-Random-1, LSTM-C-Random-4, BiLSTM-W-GloVe-4, CNN-W-GloVe-1, RoBERTa
	6	LSTM-W-Random-1, LSTM-C-Random-4, BiLSTM-W-GloVe-4, BiLSTM-C-Random-4, CNN-W-GloVe-1, RoBERTa
	7	LSTM-W-Random-1, LSTM-W-word2vec-1, LSTM-C-Random-4, BiLSTM-W-GloVe-4, BiLSTM-C-Random-4, CNN-W-GloVe-1, RoBERTa

Table 10: Different ensembles selected by genetic algorithm on natural language inference task.

Ensemble Size	2	BiLSTM-W-GloVe-2, BiLSTM-WC-ELMo-2
	3	RoBERTa, BiLSTM-W-GloVe-2, BiLSTM-WC-ELMo-2
	4	RoBERTa, LSTM-W-GloVe-2, CNN-W-fastText-1, BiLSTM-WC-ELMo-2
	5	RoBERTa, LSTM-W-GloVe-2, CNN-W-fastText-1, BiLSTM-WC-ELMo-2, BiLSTM-W-GloVe-2
	6	RoBERTa, LSTM-W-GloVe-4, CNN-W-fastText-1, BiLSTM-WC-ELMo-2, BiLSTM-W-GloVe-2, LSTM-W-fastText-2
	7	RoBERTa, BiLSTM-W-GloVe-2, CNN-W-fastText-1, BiLSTM-WC-ELMo-2, BiLSTM-C-Random-2, LSTM-W-fastText-2, CNN-WC-ELMo-1

A.4 The Ensembles of Text Classification Task

Table 9 shows the ensemble models selected by the proposed genetic algorithm and human expert of AGNEWS and MR datasets.

A.5 The Ensembles of Natural Language Inference Task

Table 10 shows the ensemble models selected by the proposed genetic algorithm on SNLI dataset.