BERT & Family Eat Word Salad: Experiments with Text Understanding

Ashim Gupta, Giorgi Kvernadze, Vivek Srikumar
University of Utah
{ashim, giorgi, svivek}@cs.utah.edu

Abstract

In this paper, we study the response of large models from the BERT family to incoherent inputs that should confuse any model that claims to understand natural language. We define simple heuristics to construct such examples. Our experiments show that state-of-the-art models consistently fail to recognize them as ill-formed, and instead produce high confidence predictions on them. As a consequence of this phenomenon, models trained on sentences with randomly permuted word order perform close to state-of-the-art models. To alleviate these issues, we show that if models are explicitly trained to recognize invalid inputs, they can be robust to such attacks without a drop in performance.

Introduction

The BERT family of models (Devlin et al. 2019; Liu et al. 2019, and others) form the backbone of today's NLP systems. At the time of writing, all eleven systems deemed to outperform humans in the GLUE benchmark suite (Wang et al. 2018) belong to this family. Do these models understand language? Recent work suggests otherwise. For example, Bender and Koller (2020) point out that models trained to mimic linguistic form (i.e., language models) may be deficient in understanding the meaning conveyed by language.

In this paper, we show that such models struggle even with the form of language by demonstrating that they force meaning onto token sequences devoid of any. For instance, consider the natural language inference (NLI) example in fig. 1. A RoBERTa-based model that scores $\sim 89\%$ on the Multi-NLI dataset (Williams, Nangia, and Bowman 2018) identifies that the premise entails the hypothesis. However, when the words in the hypothesis are sorted alphabetically (thereby rendering the sequence meaningless), the model still makes the same prediction with high confidence. Indeed, across Multi-NLI, when the hypotheses are sorted alphabetically, the model retains the same prediction in 79% of the cases, with a surprisingly high average confidence of $\sim 95\%$! We argue that a reliable model should not be insensitive to such a drastic change in word order.

We study the response of large neural models to *destructive transformations*: perturbations of inputs that render them meaningless. Figure 1 shows an example. We define several such transformations, all of which erase meaning from the input text and produce token sequences that are not natural language (i.e., word salad).

Premise	In reviewing this history, it's important to make some crucial distinctions.
Original Hypothesis	Making certain distinctions is imperative in looking back on the past. ENTAILMENT *with probability 0.99*
Sorted Hypothesis	back certain distinctions imperative in is looking making on past the . ENTAILMENT *with probability 0.97*

Figure 1: An Example for Natural Language Inference from the MNLI (-m) validation set. For the original premise and hypothesis, RoBERTa fine-tuned model makes the correct prediction that the premise entails the hypothesis. Alphabetically sorting the hypothesis makes it meaningless, but the model still retains the prediction with high confidence.

We characterize the response of models to such transformations using two metrics: its ability to predict valid labels for invalid inputs, and its confidence on these predictions. Via experiments on three tasks from the GLUE benchmark, we show that the labels predicted by state-of-the-art models for destructively transformed inputs bear high agreement with the original ones. Moreover, the models are highly confident in these predictions. We also find that models trained on meaningless examples perform comparably to the original model on unperturbed examples, despite never having encountered any well-formed training examples. Specifically, models trained on meaningless sentences constructed by permuting the word order perform almost as well as the state-of-the-art models. These observations suggest that, far from actually understanding natural language, today's state-of-the-art models have trouble even *recognizing* it.

Finally, we evaluate strategies to mitigate these weaknesses using regularization that makes models less confident in their predictions, or by allowing models to reject inputs.

In summary, our contributions are[1]:

[1]Our code is available at https://github.com/utahnlp/word-salad

Dataset	Transform	Input	Prediction
Natural Language Inference MNLI		P: As with other types of internal controls, this is a cycle of activity, not an exercise with a defined beginning and end.	
	Original	H: There is no clear beginning and end, it's a continuous cycle.	Ent (99.48%)
	`Shuffled`	H'_1: , beginning end no there clear 's continuous is a it and cycle .	Ent (99.60%)
	`PBSMT-E`	H'_2: The relationship of this is not a thing in the beginning .	Ent (94.82%)
Paraphrase Detection QQP		Q1: How do I find out what operating system I have on my Macbook?	
	Original	Q2: How do I find out what operating system I have?	Yes (99.53%)
	`Repeat`	Q2: out out i find out what out out i find?	Yes (99.98%)
	`CopySort`	Q2: ? do find have how i i macbook my on operating out system what	Yes (98.52%)
Sentiment Analysis SST-2	Original	A by-the-numbers effort that won't do much to enhance the franchise.	–ve (99.96%)
	`Sort`	a by-the-numbers do effort enhance franchise much n't that the to wo.	–ve (99.92%)
	`Drop`	a–n won do to franchise.	–ve (99.96%)

Table 1: We generate invalid token sequences using destructive transformations that render the inputs meaningless. A fine-tuned RoBERTa model assigns a high probability (in parenthesis) to the same label as the original example. For NLI, the model chooses between *entail*, *contradict*, and *neutral*. For sentiment analysis, possible labels are –ve or +ve. For paraphrase detection, model answers if the two texts are paraphrases of each other (Yes or No). The appendix contains more such examples.

1. We define the notion of destructive input transformations to test the ability of text understanding models at processing word salad. We introduce nine such transformation functions that can be used by practitioners for diagnostic purposes without requiring additional annotated data.
2. We show via experiments that today's best models force meaning upon invalid inputs; i.e., they are not using the right kind of information to arrive at their predictions.
3. We show that simple mitigation strategies can teach models to recognize and reject invalid inputs.

Tasks and Datasets

Our goal is to demonstrate that state-of-the-art models based on the BERT family do not differentiate between valid and invalid inputs, and that this phenomenon is ubiquitous. To illustrate this, we focus on three tasks (table 1) , which also serve as running examples.

Natural language inference (NLI) is the task of determining if a premise entails, contradicts, or is unrelated to a hypothesis. We use the MNLI (Williams, Nangia, and Bowman 2018) and SNLI (Bowman et al. 2015) datasets.

Paraphrase detection involves deciding if two sentences are paraphrases of each other. For this task, we use the Microsoft Research Paraphrase Corpus (MRPC, Dolan and Brockett 2005), and Quora Question Pair (QQP) dataset[2].

Sentiment classification requires predicting whether a sentence has a positive or negative sentiment. We use the Stanford Sentiment Treebank (SST-2, Socher et al. 2013).

Destructive Transformations

There has been a growing interest in studying input perturbations (e.g., Ebrahimi et al. 2018; Alzantot et al. 2018; Wallace et al. 2019; Jin et al. 2020; Ren et al. 2019; Garg et al. 2020). Given a model for a task, some input perturbations preserve labels. For example, a true paraphrase of a sentence should not change its sentiment. Certain other perturbations force labels to change in controlled ways. For example, a negated hypothesis should change the label in the NLI task.

In this paper, we focus on a new class of perturbations—*destructive transformations*—which render inputs invalid. Because any informative signal in the input is erased, the transformed examples should not have *any* correct label.[3]

For example, in the NLI task, a hypothesis whose words are shuffled is (with high probability) not a valid English sentence. The transformed hypothesis cannot contain information to support an *entail* or a *contradict* decision. Moreover, it is not a sentence that is unrelated to the premise—it is not a sentence at all! Indeed, the transformation creates an example that lies outside the scope of the NLI task.

Yet, the premise and transformed hypothesis fit the interface of the problem. This gives us our key observation: While NLP models are typically trained to work on well-formed sentential inputs, they accept any sequence of strings as inputs. Of course, not all token sequences are meaningful sentences; we argue that we can gain insights about models by studying their response to meaningless input sequences (i.e., *invalid inputs*) that are carefully crafted from valid inputs. One advantage of this protocol of using transformations is that we do not need to collect any new data and can use original training and validation sets.

Let us formally define destructive transformations. Consider a task with input $x \in X$ and an oracle function f that maps inputs to labels $y \in Y$. A destructive transformation $\pi : X \to X$ is a function that operates on x to produce transformed inputs $x' = \pi(x)$ such that $f(x')$ is *undefined*. That is, none of the labels (i.e., the set Y) can apply to x'.

[2]https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

[3]We refer to such changes to inputs as transformations, instead of perturbations (as in adversarial perturbations), to highlight the fact that change in the inputs need not be small.

Name	Description
`Sort`	Sort the input tokens
`Reverse`	Reverse the token sequence
`Shuffle`	Randomly shuffle tokens
`CopySort`	Copy one of the input texts and then sort it to create the second text. (Only applicable when the input is a pair of texts)

Table 2: Lexical-overlap based transformations

Destructive transformations can be chained: if π_1 and π_2 are destructive transformations for a given input x, then $\pi_1(\pi_2(x))$ is also a destructive transformation. We can think of such chaining as combining different diagnostic heuristics. For tasks whose input is a pair of texts, $x = (x_1, x_2)$, transforming either or both the components should destroy any meaningful signal in the input x. For example, in the NLI task, given an input premise and hypothesis, destroying either of them renders the example invalid. For tasks with a pair of texts as input, for our analyses, we only transform one of the inputs, although there is no such requirement.

Next, let us look at different classes of transformations.

Lexical Overlap-based Transformations

These transformation operators preserve the bag-of-words representation of the original input but change the word order. They are designed to diagnose the sensitivity of models to the order of tokens in inputs. Table 2 shows the four lexical-overlap based transformations we define here.

We ensure that `Shuffle` sufficiently changes the input by repeatedly shuffling till no bigram from original input is retained. The `CopySort` operation only applies to tasks that have multiple input texts such as NLI and paraphrase detection. As an example, for the NLI task, given a premise-hypothesis pair, it creates a transformed pair whose hypothesis is the alphabetically sorted premise.

Gradient-based Transformations

These transformations seek to study the impact of removing, repeating, and replacing tokens. To decide which tokens to replace, they score input tokens in proportion to their relative contribution to the output. One (admittedly inefficient) way to compute token importance is to calculate the change in output probability when it is removed. Recent work (e.g., Ebrahimi et al. 2018; Feng et al. 2018) suggests that a gradient-based method is a good enough approximation and is much more efficient. We adopt this strategy here.

Given a trained neural model $\mathcal{M}$, and the task loss function $\mathcal{L}$, the change in the loss for the i^{th} input token is approximated by the dot product of its token embedding $\mathbf{t}_i$ and the gradient of the loss propagated back to the input layer $\nabla_{\mathbf{t}_i,\mathcal{M}}\mathcal{L}$. That is, the i^{th} token is scored by $\mathbf{t}_i^{\intercal}\nabla_{\mathbf{t}_i,\mathcal{M}}\mathcal{L}$.

These token scores approximate the relative importance of a token; a higher score denotes a more important token. We use the tokens in the *bottom* $r\%$ as per their score—the least important tokens—to define our gradient-based transformations. We use $r = 50\%$. Table 3 summarizes the transformations that use importance ranking of the tokens.

Name	Description
`Drop`	Drop the least important tokens.
`Repeat`	Replace the least important tokens with one of the most important ones.
`Replace`	Replace the least important tokens with random tokens from the vocabulary
`CopyOne`	Copy the most important token from one text as the sole token in the other. (Only applicable when the input is a pair of texts)

Table 3: Gradient-based transformations

Statistical transformation: **`PBSMT`**

Recent analyses on the NLI task have shown that neural models rely excessively on shallow heuristics (Gururangan et al. 2018; Poliak et al. 2018; McCoy, Pavlick, and Linzen 2019). In particular, Gururangan et al. (2018) showed that annotation artifacts lead to certain words being highly correlated with certain inference classes. For example, in the SNLI data, words such as *animal*, *outdoors* are spuriously correlated with the *entail* label.

Inspired by this observation, we design a transformation scheme that creates invalid examples, and yet exhibit such statistical correlations. We employ a traditional phrase-based statistical machine translation (PBSMT) system to generate examples that use phrasal co-occurrence statistics.

For each label in the task, we train a separate sequence generator that uses co-occurrence statistics for that label. For example, for the NLI task, we have three separate generators, one for each label. Suppose we have a premise-hypothesis pair that is labeled as *entail*. We destroy it using the premise as input to a PBSMT system that is trained only on the entailment pairs in the training set. We use the Moses SMT toolkit (Koehn et al. 2007) for our experiments.

Why should a system trained to generate a sentence that has a certain label (e.g., an entailment) be a destructive transformer? To see this, note that unlike standard machine translation, we use very limited data for training. Moreover, the language models employed (Heafield 2011) are also trained only on examples of one class. As a result, we found that the produced examples are non-grammatical, and often, out of context. The hypothesis H_2' in table 1, generated using `PBSMT-E` (i.e., PBSMT for entailments), is one such sequence. We refer to this transformation as `PBSMT`.

Are the Transformations Destructive?

To ascertain whether our nine transformations render sentences invalid, we asked crowd workers on Amazon's Mechanical Turk to classify each instance as *valid* or *invalid*, the latter category is defined as an example that is incomprehensible, and is therefore meaningless.

We sampled 100 invalid sentences generated by each transformation (900 in total) and an equal number of sentences from the original (*un-transformed*) validation sets.

Transformation	% Invalid
Un-transformed	7.83
`Sort`	94.07
`Reverse`	95.59
`Shuffle`	94.20
`CopySort`	95.42
Avg. Lexical	94.82
`Replace`	91.21
`Repeat`	100.00
`Drop`	85.79
`CopyOne`	100.00
Avg. Gradient	94.25
`PBSMT`	79.92

Table 4: Results of crowdsourcing experiments where annotators are asked to labels sentences as meaningful or not. We aggregate sentence labels from three workers.

For each sentence, we collect validity judgments from three crowd workers and use the majority label. Table 4 shows the percent of sentences marked as invalid; we see that all the transformations make their inputs incomprehensible.

Measuring Responses to Invalid Inputs

We will now define two metrics to quantify model behavior for invalid inputs. Invalid inputs, by definition, are devoid of information about the label. Consequently, they *do not* have a correct label. If the transformations are truly destructive, a *reliable* model will pick one of the labels at random and would do so with low confidence. That is, a reliable model should exhibit the following behavior: a) the agreement between original predictions and predictions on their transformed invalid variants should be random, and, b) predictions for invalid examples should be uncertain. These expected behaviors motivate the following two metrics.

Agreement is the % of examples whose prediction remains same after applying a destructive transformation. A model with agreement closer to random handles invalid examples better. For the operators designed for tasks with a pair of inputs, namely `CopySort` and `CopyOne`, tokens from one of the inputs are copied into another. In such cases, measuring agreement with original input is not useful. Instead, we measure agreement with a default label. For the NLI task, the default label is *entail*, because neural models tend to predict entailment when there is a lexical overlap between the premise and hypothesis (McCoy, Pavlick, and Linzen 2019). Following the same intuition, for paraphrase detection, the default is to predict that the pair is a paraphrase.

Confidence is defined as the average probability of the predicted label. We want this number to be closer to $\frac{1}{N}$, where N is the number of classes.[4]

[4]We could alternatively define confidence using the entropy of the output distribution. In our experiments, we found that confidence, as defined here, and entropy reveal the same insights.

Dataset	Accuracy	Confidence
SNLI	90.87	98.38
MNLI	87.31	98.27
QQP	90.70	98.89
MRPC	89.46	98.40
SST-2	94.04	99.75

Table 5: Baseline Performance, Accuracy and Average Confidence for RoBERTa-*base* on validation sets. For MNLI, we used MNLI-matched for experiments.

Transform	MNLI	SNLI	QQP	MRPC	SST2
`Sort`	79.1	82.6	88.3	81.1	83.3
`Reverse`	76.9	75.1	86.8	77.9	82.5
`Shuffle`	79.4	81.1	88.4	80.4	84.8
`CopySort`	90.5	81.3	93.5	96.8	–
Avg. Lex.	82.4	80.1	89.3	84.1	83.5
`Replace`	63.0	51.9	69.9	56.6	78.1
`Repeat`	49.7	68.5	77.1	68.1	81.3
`Drop`	69.4	72.7	80.4	76.7	82.5
`CopyOne`	80.4	83.7	98.9	100	–
Avg. Grad.	65.6	69.2	81.6	75.4	80.6
`PBSMT`	57.0	65.6	72.5	–	75.2
Random	33.3	33.3	50.0	50.0	50.0

Table 6: Agreement scores between predictions from transformed validation set and original validation set. The closer the numbers are to random better the model behavior is.'–' means the transformation is not defined for that dataset. We do not use PBSMT for MRPC as it is a much smaller dataset.

Experiments

For our primary set of experiments, we use RoBERTa (Liu et al. 2019) as a representative of the BERT family, whose fine-tuned versions have tended to outperform their BERT counterparts on the GLUE tasks (Dodge et al. 2020). We use the *base* variant of RoBERTa that is fine-tuned for three epochs across all our experiments, using hyperparameters suggested by the original paper. These models constitute our baseline. Table 5 shows the accuracy and average confidence of the baseline on the original validation sets.

Results and Observations

We apply the destructive transformation functions described earlier to each task's validation set. To account for the randomness in the `Shuffle` transformation, its results are averaged across five random seeds. For `PBSMT` on SST-2, we use the first half sentence as input and train to predict the second half of the sentence. Table 6 shows the agreement results, and table 7 shows average confidence scores.

High Agreement Scores The high agreement scores show that models retain their original predictions even when label-

	MNLI	SNLI	QQP	MRPC	SST-2
Baseline	94.63	92.46	98.78	97.77	99.13
Random	33.33	33.33	50.00	50.00	50.00

Table 7: Average Confidence over predictions from transformed validation set. We want the the numbers to be closer to random (last row). Refer appendix for full results.

bearing information is removed from examples. This is a puzzling result: the transformations render sentences meaningless to humans, but the model knows the label. How can the model make sense of these nonsensical inputs?

We argue that this behavior is not only undesirable but also brings into question the extent to which these models understand text. It is possible that, rather than understanding text, they merely learn spurious correlations in the training data. That is, models use the wrong information to arrive at the right answer.

High Confidence Predictions. Not only do models retain a large fraction of their predictions, they do so with high confidence (table 7). This behavior is also undesirable: a *reliable* model should know what it does not know, and should not fail silently. It should, therefore, exhibit be uncertain on examples that are uninformative about the label.

Research on reliability of predictions suggests that these models are poorly calibrated (Guo et al. 2017).

Specific transformations. Invalid examples constructed by lexical transformations are more effective than others, with all agreements over 80%. Examples from such transformations have high lexical overlap with the original input. Our results suggest that models do not use input token positions effectively. We need models that are more sensitive to word order; lexical transformations can be used as a guide without the need for new test sets.

We find that SNLI models have higher agreement scores than MNLI ones for both gradient and statistical correlation based invalid examples. This could mean that the former are more susceptible to gradient-based adversarial attacks. Moreover, the lower scores for `PBSMT` on the MNLI model shows that it relies less on these statistical clues than SNLI—corroborating an observation by Gururangan et al. (2018).

Human response to transformed inputs. Results in table 4 show that transformed sentences are invalid. We now perform another set of human experiments to determine if the invalid examples generated (by transformations) make it difficult to perform the classification task . This mimics the exact setting that all models are evaluated on by asking humans to perform classification tasks on invalid inputs. Concretely, we ask turkers to perform the NLI task on 450 destructively transformed inputs (50 for each transformation) by "*reconstructing the inputs to the best of their abilities*". We found that turkers can only 'predict' the correct label for invalid examples in 35% of the cases as opposed to 77% for original un-transformed examples. These results reinforce the message that large transformer models can make sense of meaningless examples, whereas humans are near-random.

Calibration Method	Accuracy	ECE
Baseline	87.31	0.11
Label Smoothing	86.89	0.06
Focal Loss	86.98	0.05
Temperature Scaling	87.31	0.09

Table 8: Accuracy on the original validation set and the Expected Calibration Error (ECE) on the validation set for MNLI. Accuracy with temperature scaling is the same as baseline since it is a post-training method for calibration.

	B	LS	FL	B + TS
Lexical	82.35	81.85	80.39	81.49
Gradient	60.65	59.51	60.32	59.18
PBSMT	57.02	56.30	56.49	57.04

Table 9: Average agreement for three calibration methods on MNLI. Calibration does not improve model's response to invalid inputs. B: Baseline, LS: Label Smoothing, FL: Focal Loss, B + TS: Temperature Scaling on baseline.

Analysis & Discussion

Are calibrated models more reliable? Neural networks have been shown to produce poorly calibrated probabilities, resulting in high confidence even on incorrect predictions (Guo et al. 2017). Research in computer vision has shown that improving model calibration improves adversarial robustness as well as out-of-distribution detection (Hendrycks and Gimpel 2017; Thulasidasan et al. 2019; Hendrycks, Lee, and Mazeika 2019). Given the confidence scores in table 7, a natural question is: *Does improving the calibration of BERT models improve their response to invalid examples?* We answer this question by training confidence calibrated classifiers using three standard methods.

First we use *label smoothing*, in which training is done on soft labels, with loss function being a weighted average of labels and uniform probability distribution (Pereyra et al. 2017). *Focal loss* prevents the model from becoming overconfident on examples where it is already correct. Mukhoti et al. (2020) showed that focal loss improves calibration of neural models. *Temperature scaling* is a simple calibration method that scales the network's logit values before applying the softmax (Guo et al. 2017; Desai and Durrett 2020).

We use *Expected Calibration Error* (ECE, Naeini, Cooper, and Hauskrecht 2015) to measure a model's calibration error. Due to space constraints, we refer the reader to the original work for a formal definition. Better calibrated models have lower ECE. All three methods improve calibration of the original model; table 8 shows results on the MNLI validation data. However, table 9 shows that none of them improve model response to invalid examples.

Impact of pretraining tasks. We now investigate the impact of pre-training tasks on a model's response to invalid examples. Both BERT and RoBERTa use a word-based masked language modeling (W-MLM) as the auto-encoding objective. BERT uses Next Sentence Prediction (NSP) as an additional pre-training task. We experiment with other BERT variants pre-trained with different tasks: ALBERT (Lan et al. 2019) uses Sentence Order Prediction (SOP), SpanBERT (Joshi et al. 2020) and BART (Lewis et al. 2020) use Span-based MLM (S-MLM) instead of W-MLM. SpanBERT additionally uses NSP, while BART uses a Sentence Shuffling (SS) pretraining objective. ELECTRA (Clark et al. 2019) uses a Replaced Token Detection (RTD) instead of an MLM objective.

These models are trained on different corpora, and use different pre-training tasks. Despite their differences, the results presented in table 10 suggest that all of these models are similar in their responses to invalid examples. These results highlight a potential weakness in our best text understanding systems.

Different Inductive Bias. All variants of BERT considered thus far are trained with one of the auto-encoding (AE) objectives and perform rather poorly. This raises a question: *Would models that explicitly inject a word order based inductive bias into the model perform better?*

To answer this question, we consider three auto-regressive (AR) models with a recurrent inductive bias, namely, ESIM-Glove (Chen et al. 2017), ESIM-ELMo, and XLNet (Yang et al. 2019). Both ESIM models are LSTM based models, while XLNet is a transformer-based model that is trained using an auto-regressive language modeling objective along with Permutation LM (P-LM). ESIM-Glove does not use any other pre-training task, while ESIM-ELMo is based on ELMo (Peters et al. 2018) which is pre-trained as a traditional auto-regressive LM.

The results are shown in table 10. Again, the results are similar to models trained with auto-encoding objective. Surprisingly, even a strong recurrent inductive bias is unable to make the models sensitive to the order of words in their inputs: all the AR models have high agreement scores (over 75%) on lexical overlap-based transformations. We refer the reader to the appendix for more results.

Bigger is not always better. While larger BERT-like models show better performance (Devlin et al. 2019; Raffel et al. 2020), we find that same does *not* hold for their response on invalid examples. Table 10 shows that larger BERT models (Large vs Base) do not improve response to invalid examples (recall that smaller agreement scores are better). We see that both BERT variants outperform the RoBERTa counterparts; BERT-*base* provides over 4.5% improvement over RoBERTa-*base* in terms of agreement on invalid examples.

Small vs. large perturbations. Previous work on adversarial robustness (Alzantot et al. 2018; Jin et al. 2020; Ebrahimi et al. 2018) suggests that robustness of the model to small input perturbations is desirable, meaning that a model's prediction should not change for small perturbations in the input. However, excessive invariance to large input perturbations is undesirable (Jacobsen et al. 2019). Our focus is not on small input changes, rather large ones that destroy useful signals (i.e., destructive transformations). The three types of transformations we discuss in this work achieve this in different ways. We argue that language understanding systems should not only provide robustness against small perturbations (adversarial robustness) but also recognize and reject large perturbations (studied in this work).

Class	**Pretraining**	**Model**	**Agreement**
AE	W-MLM + NSP	BERT-B	67.1
		BERT-L	69.0
	W-MLM	RoBERTa-B	71.7
		RoBERTa-L	73.5
	W-MLM + SOP	ALBERT-B	67.6
AE	S-MLM + NSP	SpanBERT-B	67.6
	S/W-MLM + SS	BART-B	70.0
	RTD	ELECTRA-B	68.8
AR	P-LM	XLNet-B	70.1
	LM	ESIM- ELMo	75.8
	-	ESIM- Glove	73.5

Table 10: Agreement score of different models on MNLI. B refers to the base variant, L refers to the large one. AE refers to models pretrained with auto-encoding objective, AR refers to auto-regressive models. Refer text for full key.

	MNLI	**SNLI**	**QQP**	**MRPC**	**SST-2**
Shuffled	84.56	89.44	92.15	84.80	91.97
Original	87.31	90.70	94.04	89.46	94.04

Table 11: Training on only invalid examples generated from `Shuffle`, evaluation is on original validation data.

Are models learning spurious correlations? The results presented in this work raise an important question: *Why does this undesirable model behavior occur in all models, irrespective of the pretraining tasks, and is even seen in models with a recurrent inductive bias?* We hypothesize that this behavior occurs because these large models learn spurious correlations present in the training datasets, studied previously by Gururangan et al. (2018); Min et al. (2020). A simple experiment substantiates this claim. So far, we trained models on valid data and evaluated them on both valid and invalid examples. We now flip this setting: we train on *invalid* inputs generated by a transformation and evaluate on well-formed examples from the validation sets.

Table 11 presents accuracies on the original validation examples for five datasets. We observe that models trained only on shuffled examples perform nearly as well (within 97% for MNLI) as the ones trained on valid examples (second row)! These observations demonstrate that our models do not use the right kind of evidence for their predictions, or at least, the

	B + Th	Ent + Th	B + Invalid
MNLI	83.40/ 57.95	**86.11**/ 89.22	85.44/ **97.10**
SNLI	88.01/ 54.68	89.65/ 93.54	**90.88/ 98.41**
QQP	90.25 / 29.20	**90.29** / 88.72	90.08 / **95.24**
MRPC	89.46 / 36.82	88.24 / 99.43	**88.73 / 99.78**
SST-2	90.37 / 35.79	92.66 / 95.41	**92.78 / 96.35**

Table 12: Comparison of mitigation strategies. First number in each cell is accuracy on original validation set. Second number is the % of examples correctly classified as invalid. Test set for invalid contains examples generated with all nine transformation functions. B refers to the baseline model.

kind of evidence a human would use. This result should raise further concerns about whether we have made real progress on language understanding.

Mitigation Strategies

We evaluate three mitigation strategies to alleviate the problem of high certainty on invalid inputs. The goal is to give models the capability to recognize invalid inputs. Two strategies augment the training data with invalid examples. All three introduce new hyperparameters, which are tuned on a validation set constructed by sampling 10% of the training set. The final models are then trained on the full training set.

Entropic Regularization The central problem that we have is that the models have high certainty on invalid inputs. To directly alleviate the issue, we can explicitly train them to be less certain on invalid inputs. We do so by augmenting the loss function with a new term. Let D be the original dataset and $D^{'}$ be its complementary invalid dataset. The new training objective is then defined as

$$L(\text{model}) = L_D(\text{model}) + \lambda H_{D'}(\text{model}) \qquad (1)$$

where L_D is the standard cross-entropy loss, and $H_{D'}$ denotes the entropy of model probabilities over invalid examples. The hyperparameter λ weighs the relative impact of the two terms. Feng et al. (2018) used similar entropic regularization to improve interpretability of neural models.

We initialize the model with fine-tuned weights from the baseline model and train for three more epochs with the new training objective. The appendix provides further details.

Thresholding Model Probabilities From our results in table 7, we observe that although models are confident on invalid examples, their confidence is higher on valid ones. Following this, we experiment with a straightforward approach that thresholds output probabilities to tell valid and invalid examples apart. We used temperature scaling to ensure that the classifier probabilities are calibrated.

This approach is parameterized by a threshold θ: if the maximum probability of the classifier's output is below θ, we deem the input invalid. We used grid search in the range $[\frac{1}{N}, 1.0]$ to find the best performing θ on a separate validation set. Here, N represents the number of labels.

Invalid as an extra class (B + Invalid) Since one of the goals is to be able to recognize invalid inputs, we can explicitly introduce a new class, *invalid*, to our classification task. The training objective for this new $N+1$ class classification task remains the same, i.e. cross-entropy loss.

Results

With entropic regularization, we observe a significant drop in agreement scores on invalid examples. Indeed, the agreement scores on invalid examples decrease to an average of 35% after regularization. We also notice a significant increase in uncertainty on invalid examples. However, we see that in some cases, accuracies on the original validation set drop by over 1%, suggesting a trade-off between accuracy on valid examples and reliable response on invalid examples.

A well-behaved model should maintain high accuracy on valid data and also reject invalid inputs. To compare the three mitigation strategies on an equal footing, we measure accuracy on the original validation data and the percentage of invalid examples correctly identified. Since entropic regularization does not explicitly recognize invalid inputs, we apply the thresholding strategy to it (**Ent + Th**).

Table 12 compares the three methods. We see that simple thresholding **(B + Th)** does not work well and having the model learn from invalid examples is beneficial. It appears that, out of the three methods, training with an extra *invalid* label best maintains the balance between accuracy and invalid input detection.

We also studied if mitigating one kind of transformation helps against others. Using (B + Invalid) we train on one transformation and test on the rest. We found that mitigation can be transferable. The appendix provides detailed results.

Final Words

The main message of this paper is that today's state-of-the-art models for text understanding have difficulties in telling the difference between valid and invalid text. This observation is congruent with several other recent lines of work that highlight the deficiencies of today's text understanding systems. For example, Feng et al. (2018) construct irregular examples by successively removing words without affecting a neural model's predictions. Adversarial attacks on NLP models (e.g. Jin et al. 2020) expose their vulnerabilities; for example, Wallace et al. (2019) offer an illustrative example where a model's prediction can be arbitrarily changed.

Statistical models need not always get well-formed inputs. Consequently, when models are deployed, they should be guarded against invalid inputs, not just in NLP, but also beyond (e.g., Liang, Li, and Srikant 2018). Krishna et al. (2020) showed that it is possible to steal BERT-like models by using their predictions on meaningless inputs. Our work can be seen as highlighting why this might be possible: model predictions are largely not affected even if we destructively transform inputs. Our work shows simple mitigation strategies that could become a part of the standard modeling workflow.

Acknowledgments

We thank members of the Utah NLP group for their valuable insights, and reviewers for their helpful feedback. This material is based upon work supported by NSF under grants #1801446 (SATC) and #1822877 (Cyberlearning).

Ethics Statement

Our work points to a major shortcoming of the BERT family of models: they have a hard time recognizing ill-formed inputs. This observation may be used to construct targeted attacks on trained models, especially publicly available ones. We call for a broader awareness of such vulnerabilities among NLP practitioners, and recommend that NLP models should be actively equipped with mitigations against such attacks.

References

Alzantot, M.; Sharma, Y.; Elgohary, A.; Ho, B.-J.; Srivastava, M.; and Chang, K.-W. 2018. Generating Natural Language Adversarial Examples. In *EMNLP*.

Bender, E. M.; and Koller, A. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *ACL*.

Bowman, S.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A Large Annotated Corpus for Learning Natural Language Inference. In *EMNLP*.

Chen, Q.; Zhu, X.; Ling, Z.-H.; Wei, S.; Jiang, H.; and Inkpen, D. 2017. Enhanced LSTM for Natural Language Inference. In *ACL*.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2019. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*.

Desai, S.; and Durrett, G. 2020. Calibration of Pre-trained Transformers. In *EMNLP (1)*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. 2020. Fine-tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv preprint arXiv:2002.06305* .

Dolan, W. B.; and Brockett, C. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *IWP@IJCNLP*.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *ACL*.

Feng, S.; Wallace, E.; Grissom II, A.; Iyyer, M.; Rodriguez, P.; and Boyd-Graber, J. 2018. Pathologies of Neural Models Make Interpretations Difficult. In *EMNLP*.

Garg, S.; Kumar, A.; Goel, V.; and Liang, Y. 2020. Can Adversarial Weight Perturbations Inject Neural Backdoors. In *CIKM*.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On Calibration of Modern Neural Networks. In *ICML*.

Gururangan, S.; Swayamdipta, S.; Levy, O.; Schwartz, R.; Bowman, S.; and Smith, N. A. 2018. Annotation Artifacts in Natural Language Inference Data. In *NAACL*.

Heafield, K. 2011. KenLM: Faster and smaller language model queries. In *WMT@EMNLP*.

Hendrycks, D.; and Gimpel, K. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *ICLR*.

Hendrycks, D.; Lee, K.; and Mazeika, M. 2019. Using Pre-Training Can Improve Model Robustness and Uncertainty. In *ICML*.

Jacobsen, J.; Behrmann, J.; Zemel, R. S.; and Bethge, M. 2019. Excessive Invariance Causes Adversarial Vulnerability. In *ICLR*.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *AAAI*.

Joshi, M.; Chen, D.; Liu, Y.; Weld, D. S.; Zettlemoyer, L.; and Levy, O. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *TACL* 8.

Koehn, P.; Hoang, H.; Birch, A.; Callison-Burch, C.; Federico, M.; Bertoldi, N.; Cowan, B.; Shen, W.; Moran, C.; Zens, R.; et al. 2007. Moses: Open source Toolkit for Statistical Machine Translation. In *ACL (Demos)*.

Krishna, K.; Tomar, G. S.; Parikh, A. P.; Papernot, N.; and Iyyer, M. 2020. Thieves on Sesame Street! Model Extraction of BERT-based APIs. In *ICLR*.

Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; and Soricut, R. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *ACL*.

Liang, S.; Li, Y.; and Srikant, R. 2018. Enhancing the Reliability of Out-of-distribution Image Detection in Neural Networks. In *ICLR*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692* .

McCoy, T.; Pavlick, E.; and Linzen, T. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *ACL*.

Min, J.; McCoy, R. T.; Das, D.; Pitler, E.; and Linzen, T. 2020. Syntactic Data Augmentation Increases Robustness to Inference Heuristics. In *ACL*.

Mukhoti, J.; Kulharia, V.; Sanyal, A.; Golodetz, S.; Torr, P. H. S.; and Dokania, P. K. 2020. Calibrating Deep Neural Networks using Focal Loss. In *NeurIPS*.

Naeini, M. P.; Cooper, G.; and Hauskrecht, M. 2015. Obtaining well Calibrated Probabilities using Bayesian Binning. In *AAAI*.

Pereyra, G.; Tucker, G.; Chorowski, J.; Kaiser, L.; and Hinton, G. E. 2017. Regularizing Neural Networks by Penalizing Confident Output Distributions. In *ICLR (Workshop)*.

Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep Contextualized Word Representations. In *NAACL*.

Poliak, A.; Naradowsky, J.; Haldar, A.; Rudinger, R.; and Van Durme, B. 2018. Hypothesis Only Baselines in Natural Language Inference. In *NAACL*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research* 21(140): 1–67.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A.; and Potts, C. 2013. Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank. In *EMNLP*.

Thulasidasan, S.; Chennupati, G.; Bilmes, J. A.; Bhattacharya, T.; and Michalak, S. 2019. On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks. In *NeurIPS*.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019. Universal Adversarial Triggers for Attacking and Analyzing NLP. In *EMNLP-IJCNLP (1)*.

Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; and Bowman, S. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *BlackboxNLP@EMNLP*.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL*.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.